

THEOS[®] 32

Operating System *Reference*



Second Edition, revised: December, 1998

Copyright © 1995, 1996, 1998 by THEOS Software Corporation.

All rights reserved.

The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used only in accordance with the terms of the agreement. Information in this document is subject to change. No part of this manual may be reproduced, transmitted, transcribed or translated into any language in any form by any means without the written permission of THEOS Software Corporation. Printed in the United States of America.

THEOS Software Corporation
1801 Oakland Boulevard, Suite 315
Walnut Creek, CA 94596-7000

Telephone: 925-935-1118
Fax: 925-935-1177
E-mail: sales@theos-software.com
WWW: <http://www.theos-software.com>

THEOS® and the THEOS logo are registered trademarks of THEOS Software Corporation. Microsoft®, Windows®, Windows 95® and Windows NT® are registered trademarks of Microsoft Corporation.

Documentation by Computer Printed Words.

Table of Contents

Introduction	15
What's New in Version 4.1	15
New Date Format Support	17
ATAPI and CD-ROM Drive Support	17
Removable Hard Drive Support	17
64 Printer Support	18
64 Form and Queue Support in Spooler	18
Support for FAT32 and NTFS Partitions	18
New Show Command Displays	18
New TBackup Utility	18
New Indexed File Diagnostic Utility	19
New Function in Spooler Command	19
Other Changes	19
 Documentation Conventions	20

Part I: Fundamentals

1 Starting THEOS 32	23
THEOS MultiBoot	23
Booting and Rebooting the System	24
Maintenance Mode	27
Automatic File Allocation Test	28
Booting with the Emergency Boot Diskettes	29
Returning to Single-User Operation	30
THEOS Serial Number and Software Portal	31
THEOS PowerStep	32
 2 Entering Commands	33
The Command String Interpreter	34
Entering Command Lines	34
Command-Line Parsing	35
The CSI and the Help Key (F1)	38
Editing Command Lines	38
Command Name Abbreviations and Synonyms	40
Command-Line Recall and History	41
Last Command-Line Recall	42
Last Command-Line Token Recall	42
Prior Command-Line Recall	43
Command-Line History Window	45
CSI Calculator	46

Command Search Sequence	48
Program Name Search Rules	49
Standard Input, Standard Output and I/O Redirection	51
I/O Redirection	51
Pipes	53
3 Multiuser and Multisession	55
Multiuser vs. Multitasking vs. Multisession	55
Multiuser	55
Multitasking	56
Multisession	56
Starting Users	56
Foreground Users	57
Background Users	58
Stopping Users	59
Session Management	60
Starting Sessions	62
Multiple-Session, Keyboard Status	64
Switching Between Sessions	64
Determining Session Number	65
Program Control of Sessions	65
Number of Users, Tasks, Sessions	66
4 Consoles, Keyboards and Mice	67
Consoles	67
Console Class Codes	67
Console Screen Sizes	70
The Console Keyboard	71
Special Keys	71
Display Attributes	73
Using Menus	75
Menu Hot-Keys	76
Multiple-page Display Browsing	77
Mice	78
Mouse Setup	78
Using a Mouse	78
Using a Mouse with Menus	79
5 Printers	81
Printer Class Codes	81
Private Printers	82
Local Slave Printers	82
Public Printers	84
Printer Bypass	84
Single-Character Bypass	85
Multiple-Character Bypass	85

File Bypass	86
Print Spooler	87
Spooler Terms	87
Starting the Print Spooler	88
Specifying Spooler Forms & Queues and Compatibility.	89
Default Queue Assignment.	90
Overriding Queue Defaults	92
Spooled Printer Forms	93
Spooler Secure Mode.	94
6 User Account Environments	95
Account Name and Number	95
Passwords	96
Privilege Level.	97
Account Environment	98
Environment Switches	98
Environment Variables.	102
Default Libraries.	110
7 Disks and Tapes	113
Disk Drives	113
Hard Disk Drives	113
Hard Drive Partitions.	114
Formatting Hard Disks	116
Removable Hard Disk Drives	117
CD-ROM Drives	118
Floppy Disk Drives	119
Formatting Floppy Diskettes.	119
RAM Disk Drives	120
Using RAM Disk Drives.	120
Image Disk Drives.	122
Creating Image Disk Drives	122
Using Image Disk Drives	123
Disk Caching	124
Using Disk Drives	125
Drive Search Sequence.	126
Tape Drives	127
Using Tape Drives.	127
8 Directories and Files	129
Directories	129
The Main Directory of a Disk	129
Subdirectories	130
Libraries	131
Flat Files	131
Using Subdirectories and Libraries.	132

Current Working Directory	132
Default Library	134
Files	135
File Names	135
Libraries and Library Members	135
File Path Names	136
Wild Card Specifications	136
File Search Sequence	140
File Organization Types	141
File Attributes	142
File Ownership	142
File Access Protection	143
Growth Factor	144
9 Windows	145
What is a Window?	145
Window Attributes	145

Part II: Command Reference

10 Commands	155
Account	156
Archive	168
Attach	178
Backup	194
Boot32	198
Cache	200
Calendar	206
Cat	209
CDPlayer	212
Change	216
ChDir	222
ClassGen	225
Clear	240
Color	241
Compare	244
Compress	248
CopyFile	252
Create	269
CRLF	274
CRT	276
Date	291
Dial	294
Disk	296

Echo	312
Eject	314
Erase	315
Exit	318
Expand	319
File	324
FileList	326
FileType	342
Find	344
Force	347
GetFile	349
Head	354
Help	356
Ident	359
Install	360
IXDiag	362
Keyword	367
Line	370
LineEdit	371
List	375
Load	383
LogName	385
Logoff	386
Logon	386
Look	390
Lowcase	397
Mailbox	398
MakeBoot	401
Message	406
MkDir	410
More	412
Mount	414
Msg	415
Number	418
Password	420
Patch	421
Peek	439
Printer	440
PutFile	445
PWD	448
Reboot	449
Receive	450
Reminder	452
Rename	454
Repeat	457
Restore	458

Rmdir	467
See	469
Send	471
Set	474
Setup	482
Setup COLOR	484
Setup CRT	485
Setup DIGIXE	488
Setup DISK	490
Setup FLOPPY	494
Setup SIO	496
Shell	499
Show	501
Sleep	512
Sort	513
Split	517
Spooler	518
Start	527
Stop	527
Sysgen	531
System	545
Tail	547
Tape	549
TBackup	552
Tee	569
THEO+COM	570
Touch	574
Tree	576
Unique	579
Unload	582
Unnumber	583
Uppcase	585
WhereIs	586
Who	588
WhoAmI	588
Window Management Commands	590
WinWrite	607
WordCount	609
11 EXEC Job Control Language	613
Introduction	613
EXEC Language Concepts	614
EXEC Program Lines	614
EXEC Statements	614
EXEC Remarks	614
CSI Commands	615

Statement Labels	615
Constants, Variables, Functions and Keywords	616
Constants	616
Variable Names	616
Functions.....	617
Keywords.....	617
Command-Line Arguments	618
Expressions and Operators.....	619
String Expressions	619
Numeric Expressions	621
Relational Expressions.....	622
Logical Expressions	624
Expression Evaluation and Operator Precedence.....	625
EXEC Input Stack.....	627
Creating an EXEC Program	628
EXEC Statements	629
EXEC Functions	663

Appendices

A Contacting THEOS	677
B LineEdit Text File Editor	679
Introduction.....	679
LineEdit Command-lines	680
LineEdit Commands	683
C EXEC Language Summary	717
EXEC Statements.....	717
EXEC Functions	720
D System Files.....	721
<i>account</i> .COMMAND and <i>account</i> .EXEC	721
<i>account</i> .LOGON.....	721
<i>account</i> .REMINDER	721
SYSTEM.BOOTLOG.....	721
CDROM.CATALOG.....	721
SYSTEM.CHIST <i>nnn:work-drive</i>	722
SYSTEM.CMD32 Library	722
SYSTEM.HELP32 Library	722
SYSTEM.HISTORY.....	722
SYSTEM.LOGON	723
SYSTEM.MAILBOX	723
SYSTEM.MENU32 Library	724
LOGON	724

MSG	724
SHOW_VER	724
SYSTEM.MODEM Library	725
SYSTEM.REMINDER	726
SYSTEM.SPOOLER Library	726
SYSTEM.TEOS32 Library	726
ACCOUNT	726
B32RT*, B3220* and B386*	727
BOOTER1	727
BOOTER2	727
BOOTER3	727
BOOTMSG	727
CLASS _{nnn} (Class Codes)	728
CLOCK	729
COLORCFG	729
CONFIG	729
CRTCFCG	729
CSI	729
DESPOOL	730
DEV _{nnn}	730
DEVNAMES	731
DTIME	732
FIXDEV	732
FORM _{nnn}	732
HOSTS	732
KEYWORD	733
MESSAGE	733
MODEM	733
NUCLEUS	733
PATCHDOS	733
SESSMAN	734
SET _{aaaaa}	734
SMCFG	734
STIME	734
SYNONYM	734
VDI _{nnn}	735
System Files That May be Deleted	735
E System Log Files	737
Log File Location	737
Log File Names	737
System Boot Log File	738
System Command History Log File	739
F Command Privilege Levels	741

G Command Return Codes	747
Displaying Return Codes	748
Commands Using Wildcard Arguments	748
Setting and Testing the Return Code	749
Return Codes in EXEC Procedures	749
Return Codes in MultiUser BASIC Programs	749
H File Systems	751
THEOS/4G File System	751
Disk Layout	751
Main Directory	752
Library and Subdirectory Directories	752
THEOS/4G Files	753
I System Date and Time	755
Date and Time Display Formats	755
What is UTC?	756
Time Zones	756
Daylight Savings Time and Automatic Adjustment	757
21st Century	758
The Year 2000 Problem	759
J THEOS Character Sets	761

List of Tables

1	THEOS PowerSteps	32
2	CSI Command-Line Edit Keys	39
3	CSI Last Command-Line Recall.	42
4	CSI Calculator Operators	47
5	I/O Redirection Symbols	52
6	System Control Keys.	71
7	Console Display Attributes.	73
8	Menu Selection Keys.	75
9	Privilege Level Categories	97
10	Recommended Language Codes	100
11	CSI Prompt Variables.	108
12	CSI Prompt CRT Attributes	109
13	Floppy Disk Formats Supported	119
14	File Name Wild Cards.	137
15	RAM Disk Names and Sizes.	189
16	Color Codes & Names	242
17	Disk Density Format Codes	305
18	Floppy Disk Formats.	309
19	LINEEDIT Command Summary.	372
20	LINEEDIT Backups	373
21	ANSI Forms Control Codes	377
22	Regular Expression Control Character Specification	392

23 Regular Expression Metacharacters	393
24 More Command Browse Keys	413
25 Patch Video Mode Commands	423
26 Patch Expression Operators	426
27 NOSYSFILES Files	466
28 DATEFORM Codes	476
29 Time Zones	536
30 Logical Operator Truth Tables	624
31 EXEC Operator Precedence	626
32 EXEC &CRT Video-controls	638
33 LineEdit Command-line Editor Functions	680
34 LineEdit Modify Command Editor Functions	700
35 Logon Banner File Escape Characters	723
36 Device Driver Numbers	730
37 THEOS/4G File System, Disk Layout	751
38 File Directory Entry Contents	752

Introduction

THEOS 32 Version 4 is a high-performance 32-bit operating system for Intel and compatible 386, 486 and Pentium-based CPUs. It is fully optimized for Pentium, Pentium II and Pentium Pro machines. It offers integrated support for IDE, Adaptec 1542 bus master SCSI, Adaptec 1505/1510/152x low-cost ISA family, Adaptec 2910/2940 PCI family, Tekram DC-300 SCSI controllers, and the DPT ISA and PCI line of SCSI controllers. It also includes support for files and commands through December 31, 2099.

■ What's New in Version 4.1

If you are upgrading from Version 4, this release has many small and large changes in its features and performance.

- ▶ File and disk system caching, performance and integrity have been improved.
- ▶ Large memory systems are supported...256MB +.
- ▶ Directory sizes have been increased to more than 10,000 files.
- ▶ SHOW now supports the options IRQ, PCI, SCSI, DISK and TAPE displaying an interrupt request (IRQ) usage table, PCI device usage table, SCSI usage table, disk hardware information and tape hardware information. Also a SHOW DEVICE has been added to display information about attached devices. See [Show](#) command on page [501](#).
- ▶ The SHOW USER command has a new option PRIORITY that shows the priorities of all users. See [Show](#) command on page [501](#).
- ▶ The Maxspeed PS-8 and PS-16 PCI multiport serial adapters are now supported.
- ▶ Support for ATAPI devices including CD-ROM. See [Disks and Tapes](#) chapter on page [118](#).
- ▶ Removable drives that do not contain media at boot time are now detected.
- ▶ The parallel printer driver has been rewritten to eliminate character dropouts.
- ▶ ANSI color support now works in a windowed environment.
- ▶ MultiBoot now recognizes Windows 95 FAT32 and Windows NTFS

partitions.

- ▶ Class 180 no longer supports bright white. If your terminal (or ScanTerm) does support bright white while using WindoWriter, and you want this feature, attach with Class 210, rather than Class 180.
- ▶ Support for the Adaptec 2900 series/7800 Family Manager Set of PCI controllers. Up to four cards in a single system are supported. The 2920 controller is not supported.
- ▶ Running an IDE hard disk with an Adaptec 2940 series SCSI controller is now supported.
- ▶ New function in EXEC language: [&SYSTEM FTIME](#). This function formats the date and time according to a user-supplied format. See page [670](#).
- ▶ CRLF command has new options to control translation of national characters in file.
- ▶ Euro symbol supported for both input and display.
- ▶ System history logging enhanced to support a boot log history and new daily, weekly and monthly settings. See [Sysgen](#) command on page [531](#).
- ▶ The [MakeBoot](#) command now creates a two-diskette boot set. This provides a faster diskette boot process and provided more room for adding additional commands. Also the current console attachment is copied to the boot diskettes and may be a serial port connection. See [MakeBoot](#) command on page [401](#).
- ▶ 64 printer support.
- ▶ 64 forms and 64 queues supported by spooler. See [Attach](#) command on page [178](#), [Spooler](#) command on page [518](#) and discussion on [Printers](#) on page [81](#).
- ▶ The Mount command now tests a disk to determine whether or not the disk contains a THEOS file system. See [Mount](#) command on page [414](#).
- ▶ The MSG command, when used to broadcast a message to all users, only sends the message to the active session of consoles with multiple sessions defined. See [Msg](#) command on page [415](#).
- ▶ New spooler integrity check can be performed on demand and when system is booted. See [Spooler](#) command on page [518](#) and [Sysgen](#) command on page [543](#).
- ▶ New backup utility provides an alternate, more modern and more reliable means of backing up data. See [TBackup](#) command on page

552.

- ▶ New indexed file diagnostic utility to check and possibly correct indexed file structure problems. See [IXDiag](#) command on page 362.
- ▶ The session manager (console, session and window display control) has been significantly improved and is much more reliable.
- ▶ New modem server provides improved remote logon support via modems. See [Start](#) command on page 527 and [Sysgen](#) command on page 531.

■ New Date Format Support

With this release, THEOS has moved date format and date input routines into the operating system nucleus. Three different ways of handling dates in the 20th and 21st centuries have been implemented. These changes affect compiled BASIC286, BASIC386 and MultiUser Basic programs. These changes will not affect the interpreter.

These changes allow the developer to control how MultiUser BASIC applications handle two- and four-digit years for input and output. There are two new system-level variables that control this behavior: DATEIN and DATEOUT.

These environment variables are global in scope and can be specified in the [Sysgen](#) configuration, with the [Set](#) command or with the MUB SYS.ENV\$ function. The current setting is displayed with the [Show](#) command or the SYS.ENV\$ function.

■ ATAPI and CD-ROM Drive Support

CD-ROM drives are now supported as input devices. CD-ROM drives are attached like any other removable drive. Files on a CD can be copied, listed, opened and read by utilities and applications.

Audio CDs can be played through speakers connected to the front output jack found on most CD players.

CD-ROM changes are supported (multiple platter CD-ROMs).

■ Removable Hard Drive Support

Removable hard drives, such as Iomega's Zip and Jaz drives, are fully supported by this version of the operating system. This type of drive is primarily used for creating off-line backups but it can also be used as a standard disk drive.

■ 64 Printer Support

The previous number of printers attached to the system has been quadrupled from 16 printers to 64 printers. Each session can have up to 64 printers and the spooler now supports 64 printers as well. Slave printers may also be attached as PRT17 to PRT64.

■ 64 Form and Queue Support in Spooler

The number of forms and queues supported by the printer spooler has been increased from 26 to 64. The new queues are still single-character and in most cases will not affect the operation of programs or procedures designed for the prior set of queues and forms. These new queues and forms are the lowercase letters a–z and the special characters #, \$, %, &, *, +, -, <, =, >, ^ and ~.

■ Support for FAT32 and NTFS Partitions

The multiple partition disk boot program (MultiBoot) provided with THEOS now recognizes and can boot from Windows 95 FAT32 partitions and Windows NT NTFS partitions.

In addition, the [GetFile](#) and [PutFile](#) commands can access Windows 95 FAT32 partition disks and can copy from or to that partition.

■ New Show Command Displays

The [Show](#) command has new functions to display:

- ▶ **DEVICE:** Displays information about attached devices.
- ▶ **IDE:** Shows the IDE devices currently configured on this system.
- ▶ **IRQ:** Displays the current Interrupt Request usage.
- ▶ **PCI:** Shows the PCI devices currently configured on this system.
- ▶ **SCSI:** Shows the SCSI devices currently configured on this system.
- ▶ **DISK:** Shows map of all configured disk drives.
- ▶ **TAPE:** Shows map of all configured tape drives.
- ▶ **USER:** The option **PRIORITY** may now be specified which causes the priority for each of the users to be included in the display.

■ New TBackup Utility

A new backup utility is included with this release. This utility, named **TBackup**, has all of the capabilities of the existing **Archive** and **Restore** utilities but has an improved user interface and data compression. It also uses better error detection and correction algorithms.

■ New Indexed File Diagnostic Utility

A new utility is provided (IXDiag) that analyzes an indexed file's internal structure to verify that it is correct. When errors are found, in most cases they can be repaired by the utility without requiring special programming.

■ New Function in Spooler Command

The Spooler command has a new function (Verify) that tests the integrity of the queued reports. This function compares the queue to the collection of files saved and the collection of files saved to the queue. It also verifies the integrity of the queue itself. This function can be configured in the system configuration file for automatic checking at boot time.

■ Other Changes

There have been many small corrections and enhancements throughout the set of commands included with the operating system and in the operating system itself.

A new command, **Eject**, is provided to eject the disk or tape from a removable hard drive, CD-ROM drive or tape drive, if the device has a motorized ejection mechanism and supports this capability.

Documentation Conventions

Throughout this manual, syntax is used that looks like:

```
MAILBOX user
MAILBOX user text
MAILBOX user file
MAILBOX ( option
LIST file ( FILES option...
```


These symbols are used to show the correct syntax for typing the command.

BOLD Words and characters shown capitalized and in boldface must be entered exactly as shown.

italics Italicized words show parameters supplied by you.

FILELIST Underlined portions of a word indicate the minimum spelling allowed for abbreviations of the word.

... An ellipsis indicates that the preceding term may be repeated several times.

 Specific keys on the keyboard are shown with a “Key Caps” font.

These standards are used in the formal definitions of the syntax of a command or feature and in the descriptive text of the feature. The descriptive text also uses the following typographic conventions to identify the various items described.

File names File names that appear in text are always shown in small caps. For instance, the SYSTEM.TEOS32.DEV NAMES file.

Definitions A significant word or term that is being defined in the text is shown in boldface italics. For instance: The ***command name*** is the name of the program you want to execute.

Literal text Text appearing in an example that is also referred to in the text description of the example is shown with a different typeface. For instance: When a report is aborted, the spooler prints the message “**** A B O R T ****” and performs a form-feed.

Part I ---

Fundamentals

1 Starting THEOS 32

After THEOS is installed on a computer system, it can be started in one of four ways:

1. Powering the computer system on.
2. Pressing the “Reset” button on the computer.
3. Pressing **Ctrl+Alt+Del** on the main console.
4. Executing the [Reboot](#) command described on page [449](#).

Performing any of these actions causes the computer system’s firmware to go through its normal system startup. Refer to your computer system’s documentation on its BIOS setup for information about this process and how to configure it for your particular system configuration.

Warning: THEOS is a multiuser operating system. This means that other, possibly remote, users may be active on your system. Do not reboot the computer unless it is absolutely necessary and you have made sure that all other users will not be adversely affected. All other users should be at a Logon prompt or at least at the CSI prompt. Use the [Who](#) command described on page [588](#) or the [Show USERS](#) command described on page [501](#) to determine what other users are doing before rebooting the system.

If network operation is in use, any client users should disconnect from this network server before you reboot.

■ THEOS MultiBoot

THEOS MultiBoot is automatically installed on a hard disk that has multiple primary partitions with a different operating system on each partition. When you boot the system, MultiBoot asks:

```
OS (THEOS, Win, NT) ?
```

Enter a “T” to boot the primary THEOS partition, a “W” to boot the primary Windows or DOS partition or an “N” to boot the primary Windows NT partition. If you do not respond within five seconds, or if you respond with **Enter** or **Esc**, MultiBoot boots the active partition.

You may suppress the MultiBoot question by responding with **Del**. Entering **Del** causes MultiBoot to boot the current, active partition. Subsequent reboots will not ask which partition you want booted.

MultiBoot recognizes THEOS, DOS (also Windows 3.0 and 3.1), Windows 95/98 FAT32 and Windows NT operating system partitions. It is installed by the SETUP DISK command when the active partition is changed or when a primary partition is added to a drive that already has a primary partition from another operating system.

■ **Booting and Rebooting the System**

After the firmware performs its system startup, it loads the first sectors from the **boot drive** and transfers control to that boot program. (If you have partitioned your hard disk and installed multiple operating systems on the disk, the THEOS Multiboot offers you a choice of operating systems to boot. Refer to “[THEOS MultiBoot](#)” on page 23 for additional information.)

The THEOS boot process involves several programs and many steps. For a description of these programs and steps, refer to “[BOOTER1](#)” on page 727, “[BOOTER2](#)” on page 727 and “[BOOTER3](#)” on page 727.

When THEOS boots, it displays some important information about your computer system and the THEOS operating system.

```
THEOS® 32 Version 4.1
Serial: 102-12345, PowerStep: 8, Patch level 41000.
© 1980-1998 THEOS Software Corporation
All rights reserved.

07:45, Monday, 20 April, 1998

Logon please:
```

The first line displays the operating system version. This information is important when communicating with your distributor if you have prob-

lems or need assistance. This information can also be displayed with the Show VERSION command.

The second line, “Serial: 102-12345, PowerStep: 5,” is the serial number of this copy of the operating system and the **PowerStep** allowed. It also includes the **patch level** if any patches have been applied to the system. This information is also important when communicating with your distributor. The serial number can also be displayed with the Show SERIAL command. Refer to “THEOS Serial Number and Software Portal” on page 15 and “THEOS PowerStep” on page 17 for additional information.

After this information is displayed the boot process looks for the file SYSTEM.TEOS32.BOOTMSG:S and, if it finds it, the contents of that file are displayed on the screen. This file is useful when you want to provide your own, customized bootup screen or reminder messages about device attachments, *etc.*

The time and date displayed after the copyright notice and the optional bootmsg text reflect the current time and date used by the operating system. If this is not correct, after bootup finishes you should use the [Set DATE](#) and [Set TIME](#) commands described on page 474. Or, if networking is installed on this system, use the [Net TIME](#) command to synchronize this system to another computer’s clock.

Before the “Logon please:” prompt displays you may be asked if you want to use “Maintenance Mode”. This question is only displayed if you have enabled the “[Allow Maintenance Mode](#)” switch in the system configuration (see “[Sysgen](#)” on page 531). Refer to the description of this operation in the section “[Maintenance Mode](#)” on page 27.

Whether or not maintenance mode is selected, the disk integrity may be checked. This is done only if you have enabled the “[File Allocation Auto Test](#)” switch in the system configuration (see “[Sysgen](#)” on page 531). Refer to the description of this operation in the section “[Automatic File Allocation Test](#)” on page 28.

Next, disk caching is enabled if specified in the system configuration. Also, if you have configured a printer spooler, a simple verification test may be performed on the integrity of the spooler queue and its spooled reports. This test is only performed if you have enabled the “[Check at Boot](#)” switch in the system configuration of the spooler (see “[Sysgen](#)” on page 531).

After these tests have been performed other tasks specified in the file SYSTEM.TEOS32.STARTCFG:S are started and other users defined in the system configuration are started.

If you have enabled the “[Save History Information](#)” switch in the system configuration all of the operations performed by the boot process are written to the boot log file. For the name and location of this log refer to Appendix E: “[System Log Files](#),” starting on page [737](#).

■ Maintenance Mode

During system startup you may request that you be allowed to boot up in maintenance mode. To enable this option, set the “[Allow Maintenance Mode](#)” field in the first screen of the [Sysgen](#) command to “Y.” When this is done the following prompt appears on the screen during bootup. The message is only displayed on the screen for a few seconds.

```
THEOS® 32 Version 4.1
Serial: 102-12345, PowerStep: 8, Patch level 41000.
© 1980-1998 THEOS Software Corporation
All rights reserved.
```

```
07:45, Monday, 20 April, 1998
```

```
Press ESC now to enter maintenance mode.
```

This maintenance mode is a special form of single-user operation. When **[Esc]** is pressed while this message is on the screen, the system boots with a configuration suitable for maintaining the operating system and components in a single-user environment. Specifically:

- ▶ All disk drives other than “s” are not attached. If a floppy drive (DEV1) is specified in the configuration, drive “F” is attached to it.
- ▶ No COM, TAPE or PRINTER devices are attached.
- ▶ The spooler is not started.
- ▶ No users or sessions are started except for the first session on the main console.
- ▶ No networking services are started.
- ▶ The disk cache is enabled if specified in the configuration but delayed writes to the disk system are disabled.
- ▶ Automatic login is disabled and the main console is started at the “Logon please” prompt.

■ Automatic File Allocation Test

During system startup you may request that the hard disk be tested for proper file allocation. This is an excellent time to do this because no other users are active on the system yet and the file system is not changing. To enable this option, set the “[File Allocation Auto Test](#)” field in the first screen of the [Sysgen](#) command to “Y.”

When this feature is enabled, the boot process automatically analyzes the file structure and allocation for each hard disk attached in the system configuration. The system or “S” disk is tested first, then all other attached disks are tested in alphabetical sequence. Removable hard drives are not tested with this operation.

```
THEOS® 32 Version 4.1
Serial: 102-12345, PowerStep: 8, Patch level 41000.
© 1980-1998 THEOS Software Corporation
All rights reserved.
```

```
07:45, Monday, 20 April, 1998
```

```

File System Check
Checking file system on drive "S".
File system = THEOS/4G.
Capacity = 267,386,880 (255 cylinders, 64 heads, 64 sectors).
Main directory uses 328 out of 1,004 files.
Total files = 8,650.
    64 directories.
    247 libraries.
    6,934 stream files.
    1,063 program files.
    245 indexed files.
    11 keyed files.
    86 relative files.
File allocation appears OK.
```

You may terminate this test early by pressing **Esc**.

If any errors are found, an attempt is made to correct them. However, not all errors can be corrected automatically by this process. The test and result of the test are logged to the boot log file if history is enabled.

■ Booting with the Emergency Boot Diskettes

The Emergency Boot Diskettes are a set of two diskettes created with the [MakeBoot](#) command. These disks are used when the integrity of the hard disk or the operating system on the hard disk is suspect. The disks contain a minimal operating system and sufficient commands to allow you to do maintenance of the system.

To use these diskettes, load the first disk in the floppy drive and boot the system. The first disk will start the boot process, loading the necessary components of the operating system into memory. It will then ask you to load the second disk.

When it finishes booting the system you are in single-user mode and are asked to log onto an account. At this point you have all of the account definitions that were available when the emergency boot diskettes were created and your device attachments are:

F	Hard disk drive
M	2M ram disk
S	Floppy disk drive
CON	

The console is attached as defined in the system configuration file at the time the boot diskettes were created and the F drive is attached to the disk that was assigned as the S drive when the diskettes were created.

Although the S drive (floppy) does not have an operating system on it, all of the necessary components have been loaded into memory from the first diskette of the boot disk set. For a list of the commands available to you with this boot disk, refer to the [MakeBoot](#) command described on page 401.

■ Returning to Single-User Operation

You may desire to return to **single-user** operation after your system is booted. Some programs, such as [Archive](#), [Restore](#), [Backup](#) and [TBackup](#), may require single-user mode, or you may want single-user mode for system or software maintenance and installation.

The system commands that require single-user operation generally only require that no other user be logged onto the system. As long as the other users are at the logon prompt the program will proceed. These programs may allow you to operate by specifying a **MULTIUSER** option that tells the program to ignore the fact that other users are logged onto the system.

In either case, it is potentially dangerous to use a program that requires single-user operation if other users are logged onto the system or are able to log onto the system during the operation of the program. To prevent users from logging onto the system, the system should be returned to single-user operation.

There are two methods of returning to single-user operation. The first method is to enable the “[Allow Maintenance Mode](#)” field in the system configuration, reboot the system and press **[Esc]** when prompted, as described on page [27](#).

The second method is a “manual” method in which you use the **STOP** command to stop all user partitions. To return to single-user operation with this method perform the following steps. You must use the main console, session one, for these operations.

1. Use the [Show USERS](#) command described on page [501](#) to determine which partitions must be stopped.
2. If any of the started users are not at the Logon prompt, ask them to log off.
3. Make sure that all network clients are logged off and disconnected from the network.
4. Use the [Stop](#) command to stop each of the users. For true single-user operation you should also stop the other sessions on your own console.
5. Use the **Net** command to stop each network server that is running.
6. Use the [Spooler](#) command described on page [518](#) to stop each of the spooled printers. It is not necessary to deactivate the spooler.

You are now in single-user mode.

■ THEOS Serial Number and Software Portal

As described in the *THEOS 32 Version 4 Operating System Installation Guide*, THEOS uses a security device called the **THEOS Software Portal**. This device is connected to a parallel port. It is a “pass-through” device because it does not interfere with any parallel device that is also connected to the same parallel port. This software portal is keyed to the serial number of the operating system and must be present at all times for proper operation of the system.

Each major THEOS product (operating system, language, Plus Pak, *etc.*) and some applications are coded with a **serial number**. These serial numbers must all be the same and they must match the serial number of the Software Portal. If they do not, the products cannot be used.

As described on page 24, the serial number of the operating system is displayed when you boot the system. It may be displayed at any time with the [Show SERIAL](#) command described on page 501.

The serial number of a program can be displayed by using the SERIAL option of the [FileList](#) command described on page 326.

■ THEOS PowerStep

The PowerStep for your system defines how many users are allowed on the system at one time. A user is defined as a started, and potentially logged on, console keyboard. Thus, multiple sessions using the same keyboard count as only one user.

PowerStep	Number of Users [†]	Number of Terminals [‡]	Number of Network Logins
1	1	0	0
2	2	1	1
5	5	4	4
8	9	8	8
16	17	16	16
24	25	24	24
32	33	32	32
48	49	48	48
64	65	64	48
80	81	80	48
96	97	96	48
112	113	112	48
128	129	128	48
162	163	162	48
212	213	212	48
[†] The number of users is the total of the main console plus the total terminals and network login users. [‡] All systems are allowed one main console in addition to the number of terminals listed here.			

Table 1: THEOS PowerSteps

The total number of users that a particular PowerStep allows may be a mixture of terminals and network clients or servers. For example, with a PowerStep of 5 you may have the main console plus four terminals, the main console plus two terminals and two network users, the main console plus four network users and no terminals, *etc.*

Each main console and each terminal may have as many as eight sessions. These additional sessions are not counted as users for the purpose of PowerStep restrictions.

2 Entering Commands

THEOS 32 is a command-driven, text-based operating system, as compared to a menu-driven or graphical user interface operating system. As such, the primary method of telling the operating system what to do is accomplished by entering a command line.

Commands are entered when the CSI prompt is displayed:

```
Logon please: SYSTEM
Logon at 10:39am, on Monday, October 7, 1996.
>
```

The *CSI prompt* is displayed when you first log onto an account and when a command finishes execution. In the above screen the CSI prompt is the greater-than character (>).

A command line is composed of three parts:

```
>list this.file that.file another.file ( print olddate
```

Command-Name	Operand	Options

The *command name* is the name of the program that you want to execute. Because commands perform actions, most command names are English language verbs such as List, Erase, Show, *etc.* *Operands* are normally file names but they may be anything that the command operates upon. *Options* specify a feature of the command that is not automatically used.

■ The Command String Interpreter

The THEOS Command String Interpreter, or CSI, is the part of the THEOS operating system that accepts command lines from the user and executes the program requested. The CSI has several features that make it easier to enter commands, especially repetitive commands. Some of these features are:

- ▶ A command-line editor that allows you to enter and make changes as you enter the command.
- ▶ Command-line recall that lets you recall any of the last 16 commands entered, or to recall some of the phrases used in the last command line used.
- ▶ Command history that provides a menu-like method of recalling any of the last 16 commands used.

The CSI is also responsible for checking your mailbox and for adjusting the system clock for daylight savings time:

- ▶ The first time the CSI prompt displays, the `SYSTEM.MAILBOX` file is checked for any undelivered messages to this account. If any are found, the message “You have mail.” is displayed. See “[Message](#)” on page 406 and “[Msg](#)” on page 415.
- ▶ Each time that the CSI prompt displays, the system’s time-of-day clock is checked to see if it needs adjustment due to a change in daylight savings time. Refer to “[Daylight Savings Time and Automatic Adjustment](#)” on page 757.

■ Entering Command Lines

Commands are executed by typing the command when the CSI prompt is displayed and THEOS is waiting for you to enter a command. The syntax for each command is described in the section “[Command Reference](#)” starting on page 153.

For example, to ask THEOS to display the list of devices currently attached and available, you would enter the `ATTACH` command:

```
>attach Enter ↵
```


Commands may be typed using uppercase, lowercase or mixed-case characters. For instance, the preceding command could be entered as:

>ATTACH [Enter ↵]

or

>**Attach** Enter ↵

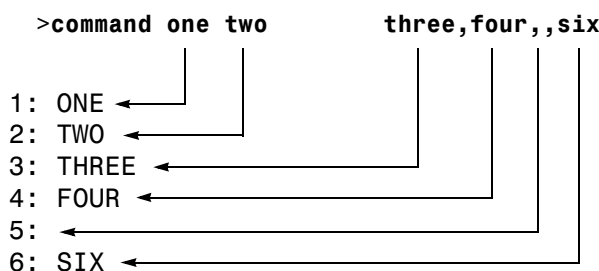
The case mode for entry is defined in the account environment (see “[Account](#)” on page 156). You may change it by pressing **Ctrl**+**C** before entering a command. This “toggles” the entry case mode for this command and all subsequent commands entered. For instance, if your case mode is currently “L,” meaning lowercase text, and you press **Ctrl**+**C** followed by “attach,” the CSI will accept and echo your command line as “ATTACH.”

As indicated, all commands entered are terminated by pressing the  key. Throughout the remainder of this manual, this will not be indicated again unless it is unclear.

- **Command-Line Parsing**

The Command String Interpreter does not have any special knowledge about the components of the command line except to assume that the first “word” is the name of a command to be executed. The remainder of the command line is “parsed” and passed onto the command.

Parsing is a term meaning to break up or divide the words on the command line into separate “tokens” or command-line arguments. These tokens are then given to the command as a list of arguments. A **token** in a command line is a sequence of non-space characters separated from other tokens in the command line by one or more spaces or a single comma. For instance:



Note that the fifth token is blank. The pair of commas in “four,,six” indicate that there is a missing argument. Also note that all of the tokens have been changed to uppercase, even though they were typed in lowercase.

The left or opening parenthesis character is treated as a separate token. This makes it easy for commands to find the start of the specified options.

The CSI converts all characters and tokens to uppercase except tokens enclosed in a pair of double quotation marks, tokens starting with the UNIX-style option character (-) and characters following an equal sign. For instance:

```
>list system.theos32.devnames (title "Device Names File"
```

In the above command the tokens are:

```
1: SYSTEM.THEOS32.DEV NAMES
2: (
3: TITLE
4: Device Names File
```

```
>set myname=JohnHancock
```

```
1: MYNAME=JohnHancock
```

```
>sort -o sorted.data unsorted.data
```

```
1: -o
2: SORTED.DATA
3: UNSORTED.DATA
```

You may embed a double quotation mark in a token to control the case-mode translation of that token. Although the double quotation mark is removed from the token, it toggles the case-mode translation of the remainder of the token. For instance:

```
>example one"two"three"four
```

```
1: ONEtwoTHREEfour
```

This case-mode toggle does not apply if the CSI has already determined that the token is not to be folded to uppercase. However, the embedded double quotation mark is always removed from the token.

```
>example -one"two"three abc=one"two"three "one" two"three
```

```
1: -onetwothree
2: ABC=onetwothree
3: one
4: TWOthree
```

The above capability is useful in some special circumstances. The important item to note is **embedded double quotation marks are always removed from the token**. (Single quotation characters are not treated specially by the CSI and may be used freely.)

You may add a **comment** to a command line. When the CSI encounters a semicolon character (;), it ignores the character and **all characters following**.

The CSI treats [Echo](#) command lines specially. As described on page 312, the [Echo](#) command's purpose is to echo or copy the remainder of the command line to the standard output device. The CSI does not change the case mode for any of the tokens when it recognizes the command name ECHO. For instance:

```
>echo This is a test of the ECHO command.
This is a test of the ECHO command.
```

Other than commas, semicolons and the left parenthesis, there are a few other characters that are treated specially by the CSI. These characters are the greater-than symbol (>), the less-than symbol (<) and the vertical bar symbol (|). The handling of these characters is discussed in the section “Standard Input, Standard Output and I/O Redirection” starting on page 51. The exclamation character also has special meaning to the CSI. Its usage is described in the section “[Command-Line Recall and History](#)” on page 41.

To summarize, the CSI uses the following rules when accepting and interpreting command lines:

1. The command line is broken up into tokens.
2. A token is a string of characters separated from other tokens by:
 - a. A space character (multiple, consecutive spaces are treated as a single space).
 - b. A comma.
3. Each token is changed to uppercase except:

- a. Tokens enclosed in a pair of double quotation marks.
 - b. Tokens starting with a dash.
 - c. The characters in a token following an equal sign.
 - d. The characters following an embedded double quotation mark character. An embedded double quotation mark character in a token toggles the case-mode folding of the token. The embedded quotation mark is always removed from the token.
 - e. The tokens of an [Echo](#) command line.
4. A semicolon marks the beginning of a comment. The semicolon and the remainder of the command line are ignored.

■ The CSI and the Help Key (`F1`)

As described in the “[Help](#)” on page [356](#), pressing the help key instead of entering a command line invokes the [Help](#) command. The CSI allows the `F1` key to be used while entering a command line.

When entering a command you may want to get the help screen for the command. You do not have to cancel entry of the command. Merely press `F1` and the CSI transforms your command line into a help request. For instance:

```
>filelist * ( printF1
```

The CSI transforms this command line into:

```
>help filelist * ( print
```

After viewing the help display, return to the CSI. Use the recall command-line key described in “[Command-Line Recall and History](#)” on page [41](#) to recall the command. Delete the “help ” word from the beginning of the line and then continue entering the command.

■ Editing Command Lines

You may make a mistake or want to change some parameter of the command line as you enter it. It is not necessary to cancel the entire command and start over. Merely use the CSI editing keys to make changes to the command before pressing `Enter ↵`.

The table on the following page lists all of the command-line editing keys that may be used while entering a command.

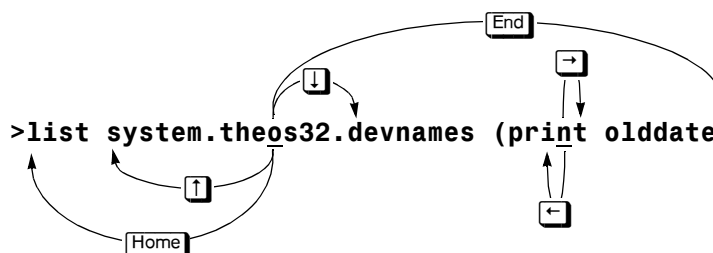
Edit Key†	Control Key‡	Function
Home	Ctrl+G	Move to the beginning of the command line.
End	Ctrl+E	Move to the end of the command line.
Backspace		Delete the character to the left of the cursor.
Del	Ctrl+Z	Delete the character under the cursor.
F5	Ctrl+N	Delete all characters to end-of-line.
←	Ctrl+H	Move the cursor left one character position.
→	Ctrl+L	Move the cursor right one character position.
↑	Ctrl+K	Move the cursor to the start of preceding “word.”
↓	Ctrl+J	Move the cursor to the start of the next “word.”
Insert	Ctrl+R	Toggle between character insert and replace mode. *
⇧ Shift+F7	Ctrl+O	Transpose the current and the next character. Cursor is moved to the position following the next character.
F2,c	Ctrl+W,c	Move the cursor to the next occurrence of the character c.
Esc	Ctrl+I	Erase the entire command line.
Enter ↵	Ctrl+M	Terminate entry and execute the command.
† The keys indicated in this table, and in other tables throughout this manual, refer to the keys normally defined for a “PC Term” style keyboard. The specific keys that you use for your keyboard can be determined by using the CRT command, “Keyboard” function.		
‡ You can use the control keys listed here on almost all keyboards. It is possible, although unlikely, that these keys might have been redefined for your terminal. Use the CRT command to test your keyboard definition.		

Table 2: CSI Command-Line Edit Keys

Edit Key†	Control Key‡	Function
* On most terminal displays, character insert mode is indicated with a blinking block cursor and replace mode is indicated with a blinking underline cursor.		

Table 2: CSI Command-Line Edit Keys

For instance:



■ Command Name Abbreviations and Synonyms

The CSI is responsible for implementing several of the account environment switches described in Chapter 6 “[User Account Environments](#),” starting on page 95. Earlier, the “CSI case mode” switch was mentioned. Two other environment switches are also important when using the CSI. The CSI is responsible for using the “Abbreviation” switch and the “Standard synonym” switch.

The “Abbreviation” switch controls whether or not command names may be abbreviated. Throughout the “[Command Reference](#)” section, the command name abbreviations are indicated by underlining the minimum spelling allowed. For instance:

FILELIST drive

This syntax indicates that the FILELIST command may be executed by entering F, FI, FIL, FILEL, FILELI, FILELIS or FILELIST. This is true only when the “Abbreviation” switch is on. (Although the abbreviation FILE is a valid abbreviation, it is also the full name of another command. When FILE is entered, the command [File](#) is executed, not [FileList](#).)

Some commands have synonym names defined. For instance, the [ChDir](#) command has a command synonym of CD. There is no command whose name is CD and, if the environment switch “Standard synonym” is off, the

command name CD will not be recognized as a synonym to the [ChDir](#) command.

It is the CSI that is responsible for implementing both of these switches. It uses the file SYSTEM.TEOS32.SYNONYM to determine the minimum spelling allowed for a command and to determine whether or not a command name is actually a synonym to another command name.

Programmer's Note: The command-line tokens of the command used to invoke a program are available to the program. The specific program name used is token number zero. This token is the full path and file name of the program after synonyms and abbreviations are interpreted by the CSI. For instance, the following shows all of the tokens that are actually passed to the program:

```
>acc ( type
0: /SYSTEM.CMD32.ACCOUNT:S
1: (
2: TYPE
```

■ Command-Line Recall and History

In many cases, particularly during program development and installation, several commands are entered that are only slightly different from each other or that are repeated in sequence. For instance:

```
>list data/customer.master
>list data/customer.master (print
```

or

```
>winwrite sample.basic      ; edit program
>basic sample.basic         ; compile program
>sample.command             ; test program
>winwrite sample.basic      ; edit program
>basic sample.basic         ; compile program
>sample.command             ; test program
```

To make it easier to duplicate commands and to repeat sequences of commands, the CSI saves the last 16 command lines entered. It provides several methods of recalling the last command line entered, one of the saved command lines entered or portions of the last command line entered.

- **Last Command-Line Recall**

The last command line entered can be quickly recalled by using one of several keys:

Edit Key	Control Key	Function
F3	Ctrl+A	Recalls last command line.
End	Ctrl+E	Recalls last command line.
PageUp	Ctrl+B	Recalls prior command line.

Table 3: CSI Last Command-Line Recall

For instance:

```
>list data/customer.master
>F3
>list data/customer.master
```

When the line is recalled, you can use the editing keys described on page 38 to modify the line as desired. For instance, press **End** and add the print option to change the command line into:

```
>list data/customer.master (print
```

- **Last Command-Line Token Recall**

You can recall unedited tokens of the last command line by using the special exclamation operator (!). When the exclamation character is entered followed by a number, the exclamation and number are replaced with the corresponding token from the last command line entered.

```
>list data/customer.master (print
>copy !1Enter
>copy data/customer.master
```

After the token is recalled, you may continue to enter new text, make changes or recall another token from the last command line.

You may specify several token recall commands before pressing the **Enter** key. If the CSI detects any exclamation operators in the command line, it

replaces them with the appropriate token or tokens, redisplay the line and returns to edit mode. For instance:

```
>list one.file two.file three.file (printEnter ↵)
>copy !1 !3Enter ↵
>copy one.file three.file
```

There are several exclamation operators that recall tokens from the last command line.


Operator	Meaning
! <i>n</i>	Recalls token number <i>n</i> of the last command line entered.
! <i>n</i>	Recalls tokens numbered zero through <i>n</i> of the last command line entered.
!\$	Recalls the last token of the last command line entered.
!^	Recalls the first token (number one) of the last command line entered.
! <i>n-m</i>	Recalls tokens numbered <i>n</i> through <i>m</i> of the last command line entered.
! <i>n</i> *	Recalls tokens numbered <i>n</i> through the end of the line of the last command line entered.
! <i>n</i> -	Recalls tokens numbered <i>n</i> through the end of the last command line entered, omitting the last token of that line.
!*	Recalls all of the tokens except the first (command name) of the last command line entered.

• Prior Command-Line Recall


The exclamation operator can also be used to recall any of the last 16 command lines entered. When the exclamation character is the first character of the command line and it is followed by a number, the prior command line saved with that number is recalled.

Command-line numbers start with number zero when you log onto an account. The number increments each time a new command line is entered. For instance, after entering 20 commands, the last 16 command lines saved are numbered 5–20. You may display the command-line history by entering an exclamation character followed by the ^{Enter ↵} key.


```





>! 
12:      set filetype=c
13:      winwrite sample
14:      make sample
15:      sample
16:      winwrite sample
17:      make sample
18:      sample
19:      list sample (print
20:      set cflag=d2
21:      set lflag=debugsym
22:      make sample
23:      debug sample
24:      winwrite sample
25:      make sample
26:      sample
27:      list sample (print

>!21

>Set lflag=debugsym

```

You may also recall command lines by using the exclamation character followed by text and then pressing the  key. When this is done the CSI recalls the last command line entered that starts with that text. For instance, instead of entering !21 in the example above, you could have entered !set. The last command line entered that starts with the characters “set” is command line number 21.

An easier method of recalling command lines is to use the keys  and  to “browse” through the list of saved command lines. The  key recalls the prior line in the command-line history;  recalls the next line in the command-line history.

• Command-Line History Window

The command-line history can be displayed in a pop-up window. Pressing **F2** at the start of the command line displays this command-line history window and you may then select the command line to recall. You may make changes before the command line is executed.

```

>list sample (print
>
Command History
set filetype=c
winwrite sample
make sample
sample
winwrite sample
make sample
sample
list sample (print
set cflag=d2
set lflag=debugsym
make sample
debug sample
winwrite sample
make sample
sample
list sample (print

```

To select a command line, use the **↑** and **↓** keys to position to the desired command. You may also enter a letter key to position to the next command line starting with that letter.

Once you have positioned to the desired command line, it may be recalled and executed by pressing **Enter**. You may make changes to it before executing it by using the appropriate editing keys described in Table 2: “CSI Command-Line Edit Keys” on page 39. All of these keys may be used except for: **Insert**, **Shift+F7** and **F2**,c.

Esc is used to exit this history window without invoking any of the commands. If you have made changes to a line but have not executed it yet, entry of **Esc** restores the line to its unchanged condition.

■ CSI Calculator

As a convenience, the CSI has a built-in calculator. Although this calculator is principally of use to programmers, it may be used by anybody, especially with its standard arithmetic operators.

Any command line starting with a number, a mathematical operator or a “variable assignment” expression is interpreted by the CSI as a calculator expression.

```
>48 * 15 + 33
753                                0X000002F1
```

The results of a calculation are displayed in decimal and hexadecimal, if possible. Hexadecimal result displays are possible if the result is an integer in the range of $\pm 2,147,483,647$. Floating-point results are displayed in decimal only.

Calculator expressions are composed of constants, variables and operators. A constant may be expressed as a decimal number, a hexadecimal number or as ASCII characters enclosed in single quotation marks.

```
>12345
12,345                                0X00003039

>0x12345
74,565                                0X00012345

>'abcd'
1,633,837,924                        0X61626364

>
```

As indicated, hexadecimal integers are specified with leading “0x” characters.

The operators that the CSI calculator recognizes include:

Operator	Meaning
+	Add two values
-	Subtract one value from another
*	Multiply two values
/	Divide one value by another
%	Remainder of one value divided by another
&	Bitwise AND of two values
	Bitwise OR of two values
^	Bitwise XOR of two values
~	Unary, one's complement of a value
-	Unary, two's complement of a value
+	Unary plus
()	Parenthetical grouping to change precedence

Table 4: CSI Calculator Operators

The calculator supports 26 variables named A through Z. Initially, all variables have the value zero. They are assigned values by using a calculator assignment statement:

```
>a = 123
A = 123                                0X0000007B

>b = 48 * a
B = 5,904                              0X00001710

>?
A = 123                                0X0000007B
B = 5,904                              0X00001710

>256*a+b*325276
1,920,460,992                          0X7277E8C0

>d = b/1.2345
D = 4,782.5030376670716

>
```

■ Command Search Sequence

When a command line is entered that is not a calculator expression, the CSI searches for the command name on disk, loads the program into memory and executes it. The methods and sequence of steps used to locate the program depend upon the account and system environment. Although most users need not be concerned with this process, understanding this sequence of steps can help program developers and system managers in setting up a system that operates efficiently.

The CSI must have a set of rules for locating a program because a command name may be the name of a compiled program or an EXEC language program. It may be a “flat file,” such as `SAMPLE.COMMAND`, or it may be a member of a library, such as `SYSTEM.CMD32.SAMPLE`. The command name may be an abbreviation of the full command name or it may be a synonym name to the full command name. The program might be owned by the current account or be owned by the public, system account. It may be located in a directory other than the current or root directory. To complicate matters further, there may be multiple disk drives attached and the program might be located on a disk other than the system disk.

In general, the rules used to locate a program are designed to allow you to enter a simple program name or a program name abbreviation without requiring you to specify the complete path and file name of the program. However, if the CSI cannot find the program using the command name entered, the easiest solution is to specify the complete path and file name of the desired program. For instance:

```
>sample
Command not found.

>private.cmd32.sample:g
```


■ Program Name Search Rules

The CSI uses the following rules when it attempts to locate and execute a program.

Programmer's Note: The command search sequence described here applies to the MultiUser BASIC language CSI and SYSTEM statements, and to the C language `csi()` and `system()` function calls.

In the following searches, if the drive code is not specified, the default drive search sequence is used, as described on page 140. If a drive code is specified, then only that drive is searched.

1. The command name contains one or more periods. For instance:

>program.filetype

>libname.libtype.program

The specified command is located and executed. If it cannot be found, the message “Command not found.” displays and no further attempt is made to locate or execute the command.

Note that synonym names are not used in this situation.

2. The command name contains a path or account name specification. For instance:

>private\program

>company/business/program

>private\company/business/program

The CSI searches the specified account and/or path for PROGRAM.COMMAND and PROGRAM.EXEC. If it cannot be found, the message “Command not found.” displays and no further attempt is made to locate or execute the command.

Note that synonym names are not used in this situation.

3. Using the account environment switches “Abbreviation” and “Standard synonym,” a search is made of the SYSTEM.TEOS32.SYNONYM file and any user-defined synonym file (environment variable [SYNONYM](#)). If a match is found, the full name is saved for usage in the following steps. This synonym command name is referred to as SYNONYM in the following descriptions.

4. When the command name does not contain a period, account or path specification, the CSI searches for the following files:

```
program.command
synonym.command
program.exec
synonym.exec
```

If a file is found, it is executed.

5. If the environment variable **PATH** is defined, it may have one of two types of definitions. It is either a single, simple name such as **USER**, or it is a list of one or more libraries and/or directory names.
 - a. If **PATH** is defined as a single, simple name, it is assumed to be an “old style” (prior to Version 4.0) command library specification. In this case, a search is made for the following files:

```
path.cmd32.program
path.cmd32.synonym
path.cmd386.program
path.cmd386.synonym
path.cmd286.program
path.cmd286.synonym
```

If a file is found, it is executed.

- b. If **PATH** has more than one specification or it has a single name containing a period, it is a “new style” command library specification. For instance, **PATH** might be defined as “**DIRECTRY,COMMAND.LIBRARY**.” This definition states that commands might be found in the directory named **DIRECTRY** or in the library named **COMMAND.LIBRARY**.

For each item in the **PATH** definition, the item is searched for on disk.

- a. If the item is a library name, a search is made for members in that library named **PROGRAM** and **SYNONYM**.
 - b. If the item is a directory name, a search is made for the following files in that directory:

```
program.command
synonym.command
program.exec
synonym.exec
```

If a file is found, it is executed.

6. If the program has not been found, the system command libraries (SYSTEM.CMD32 and SYS.CMD386) are searched for members named PROGRAM and SYNONYM. If the program is found, it is executed. Otherwise, the message “Command not found.” displays.

■ Standard Input, Standard Output and I/O Redirection

The purpose of many commands is to display information on the console screen. For instance, the [FileList](#) command displays a directory listing of the files on a disk; the [Calendar](#) command displays a month or year calendar. Your console or terminal screen is the **standard output device** and is used by these and other commands for displaying information.

Some commands, such as [List](#), accept input from a file or from the console keyboard. Your console or terminal keyboard is the **standard input device** and is used by many commands for the source of their data.

Almost all commands can detect and report errors during their operation. These error messages are displayed on the **standard error device**, which is normally your console or terminal screen.

It seems obvious that your console is used for displaying information and accepting data. However, it is important to understand these concepts because sometimes it is desirable to display information on a device other than the screen, or to accept information from a source other than the keyboard.

Many THEOS commands allow you to use a command-line option to specify that information is output to one of the printers. The [Echo](#) command (see page 312) can let you copy all information subsequently displayed on your console to another device such as a printer or a disk file. However, in many instances it is more flexible and powerful to merely request that the output or input be redirected to or from a device other than your console.

■ I/O Redirection

By using the concept of **i/o redirection**, it is easy to change where a command displays its information or gets its data. You can change the device used for standard input, standard output and standard error by ending a command line with special characters that tell the CSI to redefine what these devices are mapped to.

For instance, the [WhoAml](#) command displays information about your connection to the system on the standard output device. By ending the com-

mand line with the greater-than symbol (>) followed by a file name, the output of this command can be sent to a file.

```
>whoami > whoami.log
```

There are other characters used to redirect standard input and standard error:

Redirection Symbol	Meaning
>	Redirect standard output
<	Redirect standard input
>&	Redirect standard error
>>	Redirect standard output, append to existing
>>&	Redirect standard error, append to existing

Table 5: I/O Redirection Symbols

These symbols are followed by a device or file name. For instance:

```
>attach > current.attachmnt
>show who > private/personal/current.attachmnt
>show who > :prt
```

Redirecting the standard error device is particularly useful when executing several commands, one after another. For instance:

```
>basic program1 >& error.log
>basic program2 >>& error.log
>basic program3 >>& error.log
```

These commands will output any and all error messages to the file ERROR.LOG. The second and subsequent commands in this series use the append syntax to add their messages to the end of any existing messages in the ERROR.LOG file.

I/O redirection applies to one command at a time. When the command finishes execution and control returns to the CSI, the default devices are re-established.

Some commands always output to the console or always accept input from the keyboard, without using the standard output or standard input devices. Generally, these types of commands are interactive in nature and it would make no sense to use redirected output and input devices with them. An example of this type of command is [Account](#). (The TYPE option of the [Account](#) command does output to the standard output device.)

■ Pipes

Some commands are designed to use both the standard input device and the standard output device. They accept data from the input device, process it in some manner, and then output the results on the standard output device. This type of command is referred to as a *filter* program. These filters are identified as such in the “[Command Reference](#)” section of this manual.

For instance:

```
>number < text.file > numbered.textfile
```

The above command uses TEXT.FILE for input, adds line numbers to each record in the file and outputs the result to the file NUMBERED.TEXTFILE.

Situations frequently arise where you want to use several commands on a single set of data. For instance, you might want to sort a text file and display the result. You want the sorted display to include the line numbers from the original file. You could do this by:

```
>number text.file ( 1000 1 > work.file
>sort work.file +0.6 > sorted.file
>list sorted.file
>erase work.file
>erase sorted.file
```

An easier method for chaining several commands together is to use a pipe. A *pipe* is a command line that specifies multiple commands. Each command passes its output as the input to the next command in the pipe. The character used to indicate that output is piped to the next command is the vertical bar symbol (|).

Using this technique, the previous commands can be performed with one command line.

```
>number text.file (1000 1 | sort +0.6 | list
```

This command line causes number to send its output to the sort command via an intermediate file. The output of the sort command is sent to the list command in another intermediate file. There is no need to erase these intermediate files because they are created as temporary files and the system automatically erases them when each command is finished.

Any command that sends its output to the standard output device can be used for the start of a pipe. Any command that uses the standard input device can be used for the end of the pipe. Only commands that use both the standard input device and the standard output device can be used in the middle of a pipe.

Standard commands using both standard input and standard output include:

Cat	Sort
Head	Split
Line	Tail
List	Tee
Look	Unique
Lowcase	Unnumber
More	Ucase
Number	WordCount
See	

3 Multiuser and Multisession

The THEOS operating system is a true multiuser, multitasking and multisessioning operating system. This means that several people can be using the computer system at the same time, each on their own console and each executing one or more programs (tasks) at the same time.

■ Multiuser vs. Multitasking vs. Multisession

■ Multiuser

A **user** is a term applying to both the person using the computer and to the program or task that is executing. A single-user system allows only one person, using a single console, to use the computer. This single user has exclusive use of the resources of the computer, including the console, memory, disks, and other input/output devices.

The THEOS multiuser operating system provides the capability of using multiple consoles connected to a single computer system. Each console is used by a user executing some program or task. A **console** is composed of a display monitor and a keyboard, with an optional mouse input device. The resources of the computer are shared between all of the users on the system.

Some of the resources of the computer are shared between the users, but only one user at a time can actually use the resource. An example of this type of resource is a serial port. For instance, a modem connected to the computer can only be used by one user at a time. After the user is finished using the device, another user can use it. Resources that can be shared between several users, but only one at a time, include: the console display (see “[Multisession](#)” on page 56), serial devices, non-spooled printers and tape drives.

Other resources of the computer can be shared between users with all users having access to the resource at the same time. An example of this type of resource is the disk system: Any user on the system can use the disk at the same time as other users. Resources that can be shared simultaneously between several users include: memory, disk and the print spooler.

Some resources cannot be shared between users. These include the console keyboard and mouse, as well as printers connected as slave printers.

■ Multitasking

When an application uses two or more programs to perform some operations, with each of the programs executing at the same time, it is referred to as a multitasking application. For instance, an accounts payable application might start a subprogram to print a ledger or check register. While the subprogram is printing, the main program can be accepting other requests from the operator. The application could start several print programs at the same time while still performing some other task for the operator. Each of the print programs is a subtask of the main program.

Another example of multitasking usage is the ARCHIVE command. When archiving a disk, the command starts a subtask to perform the actual tape output. Thus, one task reads the data from disk and prepares it for output, another task writes the compressed data to tape. This design allows for more efficient usage of both resources.

Although multitasking is a feature of the operating system, no commands are provided to control it. Multitasking is controlled solely by the application programs. Both the THEOS MultiUser BASIC language and the THEOS C language provide tools for a programmer to implement a multitasking application, including all of the necessary intertask communication tools.

■ Multisession

The multisession capability of THEOS is a special form of its multiuser capability. This capability allows a person's console to be used as the keyboard and display for multiple users performing different, unrelated tasks. See "[Session Management](#)" on page 60.

■ Starting Users

Users on a THEOS system may be either foreground users with a console and keyboard or background users that have no display or keyboard. These users may be started automatically, when the system boots, or manually, by entering a user start command.

THEOS allocates a user partition for each user, task or session on the system. A ***user partition*** is an area of memory dedicated to the particular user, task or session. Only the information and data unique to that user, task or session is kept in the user partition memory. Information that is not unique to the partition and program code is separate from the partition memory and can be shared with several users, tasks and sessions.

A user partition is referenced by its **user partition number**. This number is defined when the user is started.

■ Foreground Users

A foreground user is the normal type of user with a console for display and input.

Automatic Startup. Foreground users are started automatically on startup if the system configuration file defined by the [Sysgen](#) command defines the users that are to be started. All users defined on the screen “User Console Attachments,” described on page [541](#), are started when the system is booted.

Manual Startup. A user can be started “manually” by using the [Start](#) command described on page [527](#). With this command you specify the user number that you want started, the device used for the console and the parameters defining the console connected to that device. For instance:

```
>start 20 multi5 ( c90 b38400 w8 e5
```

This command starts user number 20 using the fifth port of a multiport serial board. The user is started as a PC Term terminal at 38,400 baud using 8-bit word length and PC-XON/XOFF transmission protocol.

Note: The device names file (SYSTEM.THEOS32.DEVNAMES) may be edited to create a synonym for this device with all of the operating parameters specified. For example, if the following entry is added to the device names file:

```
DEVELOP3    27:16:4    CPSIO C90 B38400 W8 E5
```

The user can then be started with the following simple command:

```
>start 20 develop3
```

Not only is this easier, it is more descriptive. The device names file is described on page [731](#).

When a foreground user is started, whether automatically or manually, the operating system automatically outputs several initialization character strings to the console.

1. First, should a user be connected via a modem, a modem initialization string is sent to the device. This modem initialization string is found in one of two files. If the file SYSTEM.MODEM.nn exists (nn represents the started user number), then the text in that file is sent to the device. If that file cannot be found, the file sys-

TEM.TEOS32.MODEM is used. If neither file exists, no modem initialization string is sent.

Note: When the console is not connected via a modem, this modem initialization string is briefly displayed on the console. It is immediately cleared by step 3.

2. Next, the class code initialization string is sent to the device. This string is defined in the file SYSTEM.TEOS32.CLASS nn (nn is the class code number specified for the console).
3. Finally, a clear screen command is sent to the console.

After the device and console are initialized with the above process, the [Logon](#) command is executed for the user. This command asks the user to log onto an account. The user definition may specify that the user is started and automatically logged onto a specific account. For instance:

```
>start 20 develop3 (account develop
```

When this is done, the user is started in the manner already described and is automatically logged onto the specified account. (You may also specify an account name and password with the automatic user definition.)

■ Background Users

A background or “phantom” user is a user that has no console. Input to programs running in the background may only come from data stacked by an EXEC language program. Background users are used to run programs that do not need to interact with the operator. Automatic daily backups are frequently done by background users.

A background user can execute any program that a normal user can except that it cannot use any program that requires input from the operator. If a program running in the background requests input from the keyboard and there is no stacked data available from an EXEC language program, the program terminates and the background user is stopped.

Background users may only be started with the [Start](#) command described on page 527. However, a background user may be started by a foreground user that is started automatically on startup. If one of the foreground users is started with an account name specified, the *account.EXEC* can start background users.

For instance, user number four is defined with an account name of “BACKGRND” and there is a BACKGRND.EXEC program containing [Start](#) commands:

```
BACKGRND.EXEC

start daily.process
start office.mail

logoff
```

This BACKGRND.EXEC program is automatically executed when a user logs onto the BACKGRND account. It causes two background tasks to be started executing the DAILY.PROCESS and OFFICE.MAIL programs respectively. After these processes are started, the BACKGRND.EXEC program logs off, allowing the console to be used as a normal, foreground user console.

A background task is automatically terminated whenever the program running in the background exits. You can make a background user terminate early by using the [Force](#) command described on page [347](#).

The status of a background task can be checked by peeking at its “console.” Although no physical console is used by the task, THEOS allows the background task to display its output to the *standard output device* which can be viewed by another user with the [Peek](#) command described on page [439](#).

■ Stopping Users

Normally, users do not have to be stopped. However, if it is necessary, the [Stop](#) command (see page [527](#)) can stop any inactive user.

```
>stop 4
```

The above command will stop user number four if it is at the “Logon please” prompt. If the user is not at the logon prompt, a message stating that the user is still active appears. You can either go to that user’s console and execute the [Logoff](#) command or use the [Force](#) command on your console to force the user to log off of the system. Refer to the [Force](#) command described on page [347](#) for cautions about using that command.

■ Session Management

With THEOS 32, all consoles are capable of multiple sessions. A **session** is a special mode of multiple user operation. Instead of using separate consoles to execute separate programs, the console is shared between the programs.

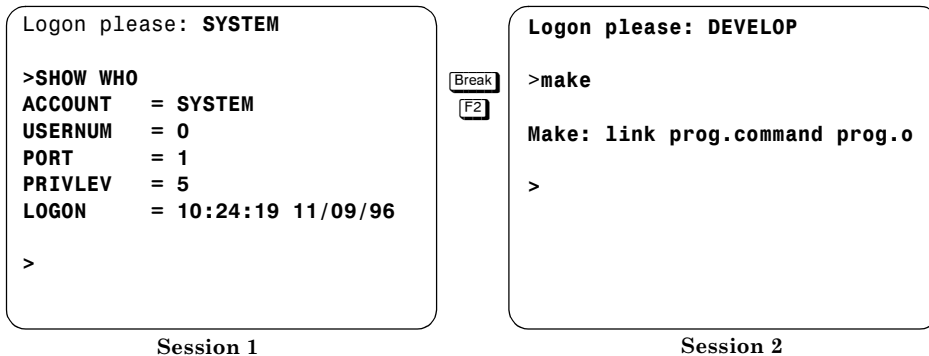
Each console may have as many as eight sessions active at any one time. However, only one session is allowed to use the display and keyboard at any one time.

Each session on a console operates just as if it were a separate console except that it shares the display with the other sessions on that console. Each session executes until it requests input from the keyboard. Since only one session controls the display and keyboard, an unselected session will wait for the input until it is selected.

A session that displays information on the screen will not wait if it is not the selected session. The program will display its information on a “virtual” display that is saved by the system. When the session is selected, the screen is updated to reflect the current information for that session.

A session might be listing a very large file. If it is an unselected session and it is not performing “page waits,” the program displays its information continuously until it reaches the end of the file. When you select that session, all you will see is the last screenful of information.

For instance, if you have two sessions defined for your console, you can perform two separate operations. On one session you might perform a **SHOW** command and on the other session you might update some programs on a development account:



Merely switch between the sessions to see the progress of or to interact with the program running on that session. When you switch to a different session, the screen is updated to reflect that session's display.

Multiple sessions are powerful and very useful for both end users and developers. For example, an end user performing data entry may need to do a database lookup that is unrelated to the data entry. By merely switching to a different session, the lookup is performed and then the user can switch back to the original session and continue data entry. No special programming is needed at the application level to provide this capability.

Developers can use multiple sessions to provide separate sessions for editing a program, compiling the program and testing the program. It is much faster to edit a program on one session, save the file without exiting the editor and then switch to another session to compile the changed program. By using the command-line recall capabilities of the CSI, the session used to compile the program can be made even more efficient by merely pressing **PageUp**, **Enter** and the last command executed on that session is reinvoked. Then just switch to the session used to test the changed program.

■ Starting Sessions

A session is started in one of two ways:

- ▶ [Sysgen](#) and the system configuration file
- ▶ The [Start](#) command

Automatic Session Startup: Screen four of the [Sysgen](#) command defines user consoles that are started when the system is booted.

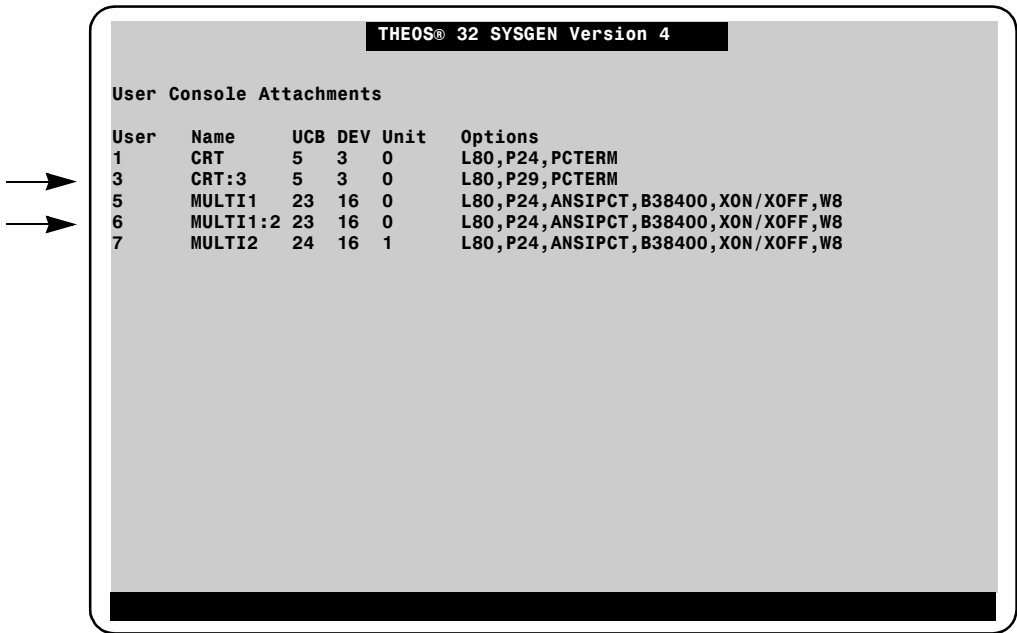
THEOS® 32 SYSGEN Version 4					
User Console Attachments					
User	Name	UCB	DEV	Unit	Options
1	CRT	5	3	0	L80,P24,PCTERM
5	MULTI1	23	16	0	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
6	MULTI2	24	16	1	L80,P24,ANSIPCT,B38400,XON/XOFF,W8

A secondary session is defined for a console by ending the “Name” field with a colon followed by a digit. The digit refers to the session number for the console attached to that device.

If the console supports different screen sizes, each session on a console may be started differently. For instance, the first session might use 24×80, another session could have 29×80.

All sessions started on a particular console must use the same class code.

For instance, the third session of the main console is defined with `CRT:3`. The second session for the console attached to the multiport board, channel one is defined with `MULTI1:2`.



User	Name	UCB	DEV	Unit	Options
1	CRT	5	3	0	L80,P24,PCTERM
3	CRT:3	5	3	0	L80,P29,PCTERM
5	MULTI1	23	16	0	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
6	MULTI1:2	23	16	0	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
7	MULTI2	24	16	1	L80,P24,ANSIPCT,B38400,XON/XOFF,W8

After the multiple sessions are defined in the [Sysgen](#) configuration, the next time that the computer is booted, those sessions are started.

For additional information about the features of the [Sysgen](#) command, refer to page [531](#).

Manual Session Startup: A session is started after system startup by using the [Start](#) command. The syntax is similar to that used in the [Sysgen](#) configuration. For instance, to start the fourth session on the main console you would use the command:

```
>start 8 crt:4
```

The number 8 in the above example, refers to the partition number. Use a number that is currently unused on your system.

The [Start](#) command can be used in an EXEC program or in an application program. For additional information about the features of the [Start](#) command, refer to page [527](#).

When a session is started, it is started just like a new user terminal is started and the “Logon please” prompt displays. Sessions may be started that automatically log onto a specific account and begin executing the *account.EXEC* program. Refer to the [Sysgen](#) and [Start](#) commands for information about this feature.

■ Multiple-Session, Keyboard Status

The settings of the **Caps Lock** key and the **Num Lock** key is unique to each console on the system. However, all of the sessions on one console share the status of these keys. Thus, when the **Num Lock** key is set on one session and you switch to another session, it is also set in that session.

The **Scroll Lock** key is unique to each session. When you switch from one session to another, the status of the **Scroll Lock** is set to the last setting used by the session that you switched to.

■ Switching Between Sessions

To switch from the current session to another session, you use the hot-key **Break**, *digit* or **Break**, *function-key*, or you may use the **Break**, **Tab** hot-key to switch to the next available session.

For instance, to switch to session three of your console, press **Break**, **3** or **Break**, **F3**.

On some terminals, it may take several seconds for the Session Manager to update the screen for the new session. This is particularly true when the display uses lots of line-graphics characters or video attributes.

Note: You cannot switch to a session that has not been started: The request is merely ignored. You cannot switch from the current session until it is finished updating the screen. This should only take a few seconds. Some application programs can disable session switching or even force a session switch to a particular active session.

■ Determining Session Number

The current session number is not displayed on the screen. You can determine the session number by displaying the console device name by using either the [Attach](#) command or the [WhoAml](#) command. For instance:

```
>whoami
Account Name: SYSTEM
Logon Date: 14 February 1998, 09:56 am
Pid: 16
Console: SVGA:2 C90,L80,P24
Network Id: Doc_Server
IP Address: 192.168.100.1
```

Note that the console is attached to the device name **SVGA:2**. This means that this is session number two of the device SVGA.

You can also determine the session number with the [wSwitch](#) command. This command sets the return code to the session number. The return code is viewed if the “ready message” is enabled.

```
>set rdymsg on

RC = 0, 10:02:07, ET = 0:00, CPU = 0.040

>wswitch ?

RC = 2, 10:02:13, ET = 0:00, CPU = 0.020

>
```

The return code of 2 in the above example means that this program ([wSwitch](#)) was executed from session two.

■ Program Control of Sessions

The active session is normally controlled by the operator with the hot-keys just described. However, some situations require the program to take control of the display and keyboard, even if it is not the session selected by the operator. For instance, an unselected session may be running a program that needs a response from the operator now without waiting until the operator selects the session.

EXEC programs can control sessions with the [wSwitch](#) command described on page [605](#). With this command, you can enable or disable session switching, determine the session number, or switch to a specific session.

Application programs written in MultiUser BASIC can use the `SYS.ENV$(27)` function. A THEOS C language program can use the

wGetSess function. Refer to their language reference manuals for additional information about controlling sessions with a program.

■ **Number of Users, Tasks, Sessions**

There are four factors that limit the number of users, tasks and sessions on a system:

- ▶ System PowerStep
- ▶ System limits
- ▶ Amount of available memory
- ▶ Maximum number of tasks defined in system configuration

The system PowerStep is described on page 32. It limits the number of consoles that can be started on a particular system. For instance, a PowerStep 8 system may have eight consoles started at one time, in addition to the main console. The PowerStep limits only the number of foreground users (users with console) and does not limit the number of background users or tasks used by any user.

There are system-defined limitations to the maximum number of users, tasks and sessions that are allowed:

- ▶ Assuming the PowerStep allows it, there can be a maximum of 213 user consoles defined.
- ▶ Each console may have a maximum of eight sessions started.
- ▶ The maximum number of tasks that can be executing at any one time is 250.

The system configuration specifies the maximum number of tasks that may be used at any one time on a system. This limit is defined in the first screen of the [Sysgen](#) command in the field “Maximum Number of Tasks.” When specifying this limit, keep in mind that tasks are required for each of the following uses:

- ▶ One task for each started user (foreground or background)
- ▶ One task for each additional session started
- ▶ One task for the disk cache, if enabled
- ▶ One task for the print spooler and one task for each spooled printer

THEO+Net, THEO+Server, THEO+Fax and other add-on products may use additional tasks.

4 Consoles, Keyboards and Mice

The console is the primary interface between the computer and the user.

■ Consoles

The term **console** refers to both the console display and its keyboard. There are many types of consoles supported by THEOS including the main console, video displays and keyboards connected via special cabling (i.e., TG/MAX), PC Term and ASCII terminals connected via RS-232 serial cabling, and Windows® client workstations and THEOS NETTERM client workstations connected via Ethernet cabling.

The **main console** refers to the video display and keyboard connected directly to PC-compatible computers using a special circuit board and cabling. The video display is typically a VGA or Super VGA color monitor and is a very high-speed display because of its direct connection to the computer.

Video displays and keyboards using the MaxStation MultiVGA Solution have capabilities similar to the main console and are nearly as fast.

PC Term and ASCII terminals are consoles providing multiuser access to the computer system.

Network workstations such as a Windows client or a THEOS NETTERM client are consoles when they are used as a client to the THEO+Net Login Server. This is an add-on product to the basic operating system.

■ Console Class Codes

Although all terminals and keyboard/displays have similar capabilities, the specific capabilities and the commands used to invoke those capabilities vary from manufacturer to manufacturer and even from model to model from the same manufacturer. For instance, on one type of terminal the cursor might be moved to the top, left corner of the screen by sending an 0x1b, 'H' sequence. Another terminal might use 0x1e.

THEOS provides a uniform interface so programs can perform various display functions without using different codes for each type of terminal on the system. A **class code** is a table definition for a terminal that informs the operating system how to perform various functions on that type of dis-

play. It also defines the value transmitted by the keys on the keyboard because these also vary from manufacturer to manufacturer.

There are many console class code definitions provided with THEOS. These are listed in the `SYSTEM.TEOS32.DEVNAMES` file and on page 731 of this manual. A new terminal class code can be created with the [ClassGen](#) command described on page 225.

A console class code is specified when the console is attached, either with the [Sysgen](#) or [Start](#) commands or with the [Attach](#) command. The class code is identified by its number or its name, as defined in the device names file (`SYSTEM.TEOS32.DEVNAMES`). For instance:

```
>attach con sio1 (c90
>attach con sio2 (pcterm
```

Both of the above commands attach the console as a PC Term compatible display and keyboard.

Console Bypass: Sometimes it is desirable or necessary to transmit a character to the console display that would normally be interpreted and translated by THEOS. To transmit a character that you want to be sure is sent to the console without any translation, you must tell THEOS that it is to pass the characters to the console device without interpretation.

First, you must tell the Session Manager to pass the characters without translation. Use the [wBypass](#) command:

```
>wbypass on
```

This can also be done with a statement function in a MultiUser BASIC (`SYS.ENV$` function) or C language program function (`ioctl`). Next, send the desired characters to the console, preceding each character with the escape code (0x1b, decimal 27). When you are finished sending the special characters, return the console to normal mode with the [wBypass](#) command.

```
>wbypass off
```

A similar mechanism exists for transmitting characters to a printer. However, it does not need to bypass the Session Manager.

An alternate method of sending data to the console device but bypassing the Session Manager is supported but not recommended for usage. It is intended for legacy applications only. You may attach the logical name `COMn` to the console.

```
>attach com1 con
```

Data sent to the COM n will be sent to the console device. How THEOS interprets this data depends upon the [COM=CON Default Bypass](#) setting in the [Sysgen](#) configuration.

If [COM=CON Default Bypass](#) is NO (default setting), the data is examined by the Session Manager to ensure that the proper session is active but it is not translated by the class code or transformed in any other way.

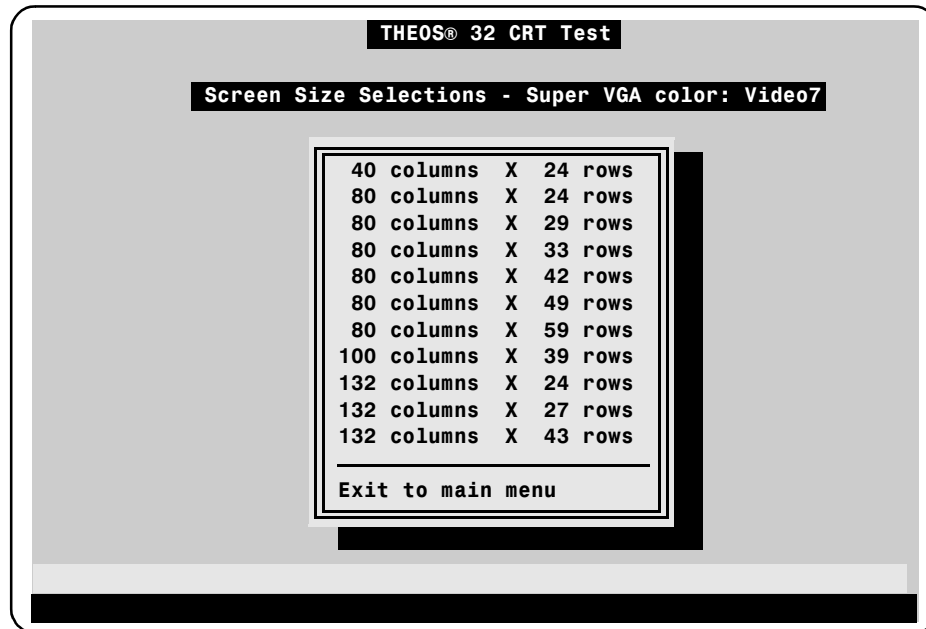
If [COM=CON Default Bypass](#) is YES, the data is completely ignored by the Session Manager. It is assumed that the data is being sent to the console device for forwarding by it to some other device connected to the console's auxiliary port.

This mechanism should not be used by new programs and particularly it should not be used if the console has multiple sessions active. Instead, use the session manager bypass functions. In MultiUser BASIC use the SYS.ENV\$ function, in C use the `ioctl()` function.

■ Console Screen Sizes

The main console on a system can display text on the screen in more than one size. You can request a different size of display by attaching the console with the number of columns and rows desired. For instance, the normal size screen uses 24 lines of 80 columns. You can use a display with smaller characters by requesting more columns or lines, such as 29 lines of 80 columns.

The specific sizes supported by the main display depend upon the video graphics adapter used. To determine the sizes that are supported on your system use the [CRT](#) command, “Screen Sizes” test.



If a size is not available on your display, it is not shown in the menu list.

Some of the sizes that are displayed may not work properly with the display adapter or may not be appropriate. Before attaching your console with a different size, you should always test it by selecting the size from the menu and viewing the result.

Some terminals also support different size displays. If you suspect that your terminal can use different sizes, use the [CRT](#) command test described above and see if multiple sizes are offered and if they work properly.

■ The Console Keyboard

Most keyboards have the same keys on them and perform the same functions. However, similar to the console display, the specific keys available and the codes transmitted when the keys are pressed vary from manufacturer to manufacturer. The console class code used for display translations also defines the various standard keys available on a keyboard and the codes that are transmitted by those keys.

■ Special Keys

There are several keyboard key sequences that are available to control various system features and actions. All of these key sequences use a lead-in key as the first key in the sequence. The **Break** key is the default key used for these sequences. The specific key used can be set with the **BREAK** environment switch described on page 98.

Key Sequence	Meaning
Break , B	This key sequence is used only when you are using THEO+COM or NetTerm and you are connected to another THEOS system. It transmits a break signal to the remote THEOS system. For instance, to send a Break , P to the remote system, you would press Break , B , P .
Break , C or Quit	Program cancel. This key sequence may terminate execution of the program that is executing. Whether or not it does is dependent upon the program itself. The Quit key is defined by the console's class code definition. See " ClassGen " on page 225.
Break , D	Debug break-point. When the Debug command is loaded, entry of this key sequence transfers control to the debugger with a "break-point" at the instruction it is executing when this key sequence is received.
Break , O	Output suppress toggle. When toggled off, all output to the console display is suppressed. When toggled on (the normal state), output to the console is displayed.

Table 6: System Control Keys

Key Sequence	Meaning
Break , P	Echo file toggle. When the Echo command has defined a file or device to be used for console display echoing, this key sequence turns on and off the actual echoing of characters to that file or device.
Break , Q	System cancel. Terminates any program that is executing. Note that this action can be disabled by setting QUIT OFF with the Set command or by disabling it in the account environment.
Break , T	<p>Clears the “type-ahead” buffer.</p> <p>When a program is executing but not waiting for keyboard input, you can type characters in anticipation of the program’s next input request. These characters are saved in the type-ahead buffer.</p> <p>If you make a mistake or change your mind, you can clear this type-ahead buffer by pressing Break, T.</p>
Break , W	<p>Toggles the screen page-wait feature.</p> <p>Commands and applications that display more information than will fit on one screen use the screen wait feature of the operating system. In this mode, an up-caret character is displayed in the bottom left corner of the screen and the program pauses until you press Enter ↵.</p> <p>Pressing Break, W disables or enables this screen page-wait feature. When disabled, the program does not pause when the screen display fills, but merely clears the screen and displays the next page of information.</p> <p>Whether or not a program uses the screen page-wait feature is dependent upon the specific program.</p>
Break , PrtScreen	Copies the current screen image to the first attached printer.
Ctrl + Alt + Del	Reboot the system. This key sequence is available on the main console only. Use the Reboot command on other consoles.

Table 6: System Control Keys

■ Display Attributes

Text displayed on the console can have various video attributes. These attributes control the intensity, color, blink, underline, *etc.* of the character font.

To control these attributes, programs output codes to the console display to turn the various attributes on or off. Most programs use the names of the attributes to identify the codes used.

Video Attribute	Meaning
BON BOFF	Enables character blink mode. All characters output after a BON is output will blink. All characters output after a BOFF is output do not blink.
FON FOFF	Enables “format mode” for some terminals. When format mode is enabled, all characters output are protected and are not cleared from the screen when an EU (erase unprotected) edit command is sent to the terminal.
KON KOFF	Enables the cursor display. Note that most programs that request input from the keyboard will enable the cursor prior to accepting the input.
MON MOFF	Enables monitor mode. Most terminals and displays support this capability. When monitor mode is enabled, the ASCII control codes are displayed with visible characters, such as F for the carriage-return character and G for the line-feed character. Refer to “ THEOS Character Sets ” on page 761 for a description of the characters displayed in monitor mode for a PC Term display.
PON POFF	Synonym to the FON and FOFF attributes.

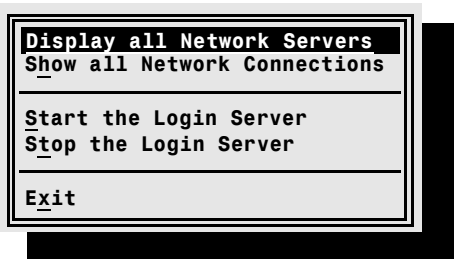
Table 7: Console Display Attributes

Video Attribute	Meaning
RVON RVOFF	<p>Enables reverse video display. The effect of this attribute depends upon the display. For monochrome displays, reverse video means just that: The characters are displayed in the opposite colors of their non-reverse video characters. For instance, instead of white characters on a black background, reverse video characters would be black characters on a white background.</p> <p>For color displays, the color of reverse video characters are defined by the Color command or the color statement in the program.</p>
SON SOFF	<p>Displays text on the status line of the terminal. All characters between the SON and the SOFF codes are displayed on the last line of the screen. If the display does not have a status line capability, the characters are not displayed at all.</p>
ULON ULOFF	<p>Enables underline display. The effect of this attribute depends upon the display. For monochrome displays (not grey-scale), underline means just that: The characters are displayed with a line under each character.</p> <p>For color and grey-scale displays, the color of underlined characters is one number less than the color code of normal characters. For instance, if normal characters are displayed in white (code 7), underlined characters are displayed in yellow (code 6).</p>

Table 7: Console Display Attributes

■ **Using Menus**

Many THEOS commands display and use menus for option and function selection. For instance, the Net command uses a menu to allow you to select the function desired:



Some menus may contain more items than will fit on the screen. When this occurs, the screen displays as many as will fit. A **scroll bar** on the right side of the menu indicates that there are more items available and shows the approximate position of the display relative to the entire set of menu items.

One of the menu items will be displayed in a reverse video **highlight bar**. This highlight bar indicates the menu item that you are currently pointing to. In the example above, the item “Display all Network Servers” is the current menu item. The highlight bar can be moved to another item quite easily with one or more of the following menu selection keys:

Key	Action
	Exit menu without selecting any item.
 (space)	Move the selection highlight bar to the next menu item. If currently positioned to the last item in the menu, it wraps to the first item.
	Move the selection highlight bar to the previous menu item. Does not wrap to the bottom of the menu.
	Move the selection highlight bar to the next menu item. Does not wrap to the top of the menu.
 or 	Move the selection highlight bar to the first menu item.

Table 8: Menu Selection Keys

Key	Action
End or Ctrl + Y	Move the selection highlight bar to the last menu item.
PageUp or Ctrl + B	When the menu has more items than will fit in one window, this key displays the previous windowful of selections. When the menu does fit, this key is synonymous with the Home key.
PageDn or Ctrl + P	When the menu has more items than will fit in one window, this key displays the next windowful of selections. When the menu does fit, this key is synonymous with the End key.
Enter ↵	Selects the menu item currently highlighted.

Table 8: Menu Selection Keys

■ Menu Hot-Keys

Besides the menu selection keys described above, you may also position to or select a menu item with a “hot-key.” A **hot-key** is a single key that, when pressed, jumps to a menu item. Menus use one of two types of hot-keys.

Unique Hot-Keys: A menu may indicate that a specific character in each menu item’s text is the hot-key character. The **Net** menu shown on page 75 uses this technique. Notice that one character in each menu item is underlined. On a color display, this character is shown in a different color.

Pressing a character that matches one of these underlined letters causes that menu item to be selected, just as if you had positioned to it with one of the arrow keys and then pressed **Enter ↵**.

First-Letter Hot-Key: Menus that do not identify a unique character for each menu item use the first letter of each menu item as the hot-key. For instance, the **Disk** command’s menu shown on page 307 uses this method. Notice that there are no underlined characters in the menu item text.

In this type of menu, pressing a letter key causes the highlight bar to move to the next menu item that starts with that letter. The item is not selected, it is merely positioned to. If there are no menu items below the current menu item, then the first menu item matching the character is selected.


For instance, when the **Disk** menu is displayed, pressing **S** moves the highlight bar to “Seek to Random Sectors.” Pressing **S** again moves to

“Surface Analysis.” Pressing **S** a third time moves the highlight bar to the top of the menu to the “Select Drive” item.

If there are no other items that start with the letter pressed, an audible alarm sounds.

■ **Multiple-page Display Browsing**

Most commands that can produce a multiple-page listing or display provide an ability to jump back and forth through the pages of the display. This is called **browsing** and uses the following keys:

Key	Action
Home or Ctrl + T	Displays the first page of the listing.
End or Ctrl + Y	Displays the last page of the listing.
PageUp or Ctrl + B	Displays the previous page.
PageDown or Ctrl + P or Space or Enter 	Displays the next page.
F9 or Ctrl + Q or Esc	Terminates the display.

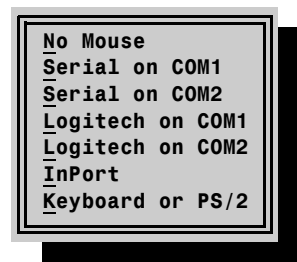
The commands that support browsing include: [Account](#) (Environment section), [FileList](#), [Help](#), [List](#), [Message](#), [Sysgen](#), and [Tree](#).

■ Mice

A mouse is an optional input pointing device attached to your console. Although associated with the console, input from the mouse is separate and distinct from input from the console's keyboard.

■ Mouse Setup

Main Console. A mouse physically connected to the computer can be used by the main console and all of its sessions (see “[Session Management](#)” on page 60). It is set up with the [Setup CRT](#) command described on page 485. THEOS supports serial mice connected to COM1 or the COM2 port, a Logitech mouse on COM1 or COM2, a Microsoft InPort mouse or a mouse connected with a PS/2 style connector.



User Terminals. Consoles other than the main console may use a mouse if they are IBM 3151 terminals, Relisys terminals, THEO+GRAFX terminals, or a computer system using ScanTerm or THEOS WorkStation software to connect to this system. Use the appropriate setup program for these consoles and work stations.

■ Using a Mouse

A mouse is only useful as a pointing device. That is, an application or program that primarily accepts text from the operator, such as a transaction entry program, or primarily produces output, such as a report, cannot use the capabilities of a mouse except in a very limited way.

Many THEOS commands do recognize and support mouse input and many existing applications support mouse input. User applications can be written or modified to support a mouse, if appropriate. Refer to the appropriate language reference manual for instructions on programming for a mouse.

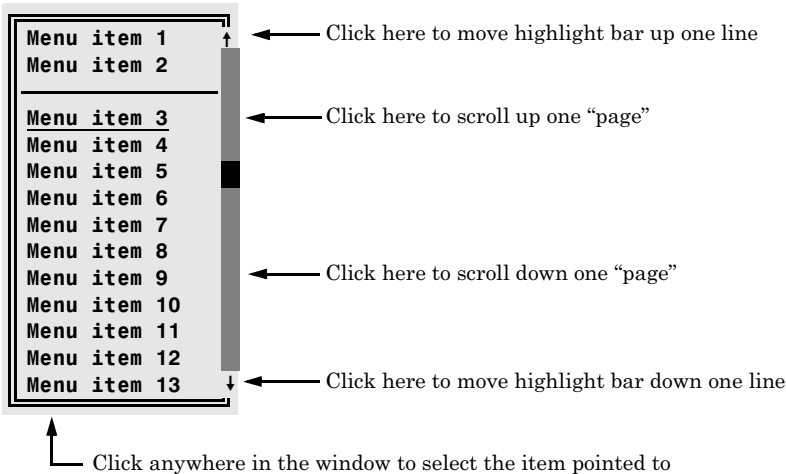
THEOS programs that do recognize a mouse include: [Account](#), [CRT](#), [Help](#), [MakeBoot](#), [Net](#), [NetTerm](#), [Setup](#), [THEO+COM](#), [WindoWriter](#) and [wMenu](#).

The THEOS mouse interface supports the following mouse actions:

Mouse Action	Description	Meaning
Click	Press and release a mouse button.	Used to move the input cursor to the location pointed to by the mouse.
Double Click	Press, release, press and release a mouse button.	Select the item pointed to by the mouse.
Drag	Press a button, move the mouse, release button.	Used only in WindoWriter, programs using menus with a “scroll bar” and custom applications.

■ Using a Mouse with Menus

Any program that uses a windowed menu for selection has the normal keyboard methods used for selection, as described on page 75. If a mouse is present, it may also be used:



5 Printers

A printer is the primary device used for producing “people-readable” reports from a computer. THEOS allows many printers to be connected to the system at the same time and each user may have as many as 64 printers attached at any one time. Printers may be privately attached or publicly attached or spooled through the print spooler.

■ Printer Class Codes

All printers have similar capabilities. However, the specific capabilities and the commands used to invoke those capabilities vary from manufacturer to manufacturer and even from model to model from the same manufacturer. For instance, one printer might use Esc, E to enable boldface printing while another uses Esc, (, s, 1, B to enable boldface printing.

THEOS provides a uniform interface to printers so a program can perform various printing functions without using different codes for each type of printer on the system. The ***printer class code*** is a table defining how to perform various functions on a specific printer.

The printer class code definitions provided with THEOS are listed in the device names file SYSTEM.TEOS32.DEVNAMES described on page 731. A new printer class code can be created with the [ClassGen](#) described on page 225.

A printer class code is specified when the printer is attached, either with the [Sysgen](#) or with the [Attach](#). The class code is identified by its number or its name, as defined in the device names file (SYSTEM.TEOS32.DEVNAMES). For instance:

```
>attach prt3 centlp1 (c135
```

```
>attach prt4 centlp2 (hplaser
```

Both of the above commands attach printers using the HP LaserJet class code.

■ Private Printers

Printers may be attached for public access (see “[Public Printers](#)” on page 84) or for private, single-user access only. A printer that is attached with the [Attach](#) is attached as a private printer by default. A printer attached privately can only be used by the user or session that attached it. Other users or sessions cannot use the printer.

One exception to this exclusive usage of a private printer is the local slave printer. See “Local Slave Printers” below.

A printer is attached as a private printer with the [Attach](#) command.

```
>attach prt3 centlp1 (c135
```

The above command attaches logical printer name PRT3 to the first parallel port and tells THEOS that the printer is an HP LaserJet compatible printer.

■ Local Slave Printers

A **slave printer** is a printer that is physically connected to a console or workstation. Text and data sent to the printer must be sent to the console or workstation, which passes it on to the printer connected to it. Since the console is a private device, the slave printer connected to it is a private device.

However, because a console can be shared between multiple sessions (see “[Session Management](#)” on page 60), the printer connected to the console can be shared between the sessions on that console. When this is done and the console is a terminal (not a TWS console), the user should be careful when sending reports to the slave printer. Only one session at a time should use the slave printer or the text printed will be intermingled, possibly on the same page.

When a printer is connected and used as a slave printer, all output to the printer is routed through the console. Because a console can only do one thing at a time, while data is being sent to the printer, data cannot be sent to the display.

THEOS tries to alternate between the two devices if it can. However, sending a lot of text to the display while printing on the slave printer will be slow or “jerky.” The specific performance will depend upon the printer and terminal and how efficiently they process data.

When a slave printer is busy or not ready, it may also make the console not ready. In other words, if you have sent text to a slave printer that is not ready you may not be able to send text to the console until the printer can print.

Each of the sessions started on a console may use the printer attached to the console as a slave printer. If the console is actually a computer that is accessing the THEOS system as a network workstation, a slave printer can be attached that is not physically connected to the console but is one of the printers attached to the client system. For instance, if you are using a THEOS system that has the print spooler started (see “[Print Spooler](#)” on page 87), then you can connect to a THEOS Network Login Server system as a client user and your spooled printers are available as a slave printer attachment.

>attach

THEOS® 32 Device Attachment

Logical Name	Physical Name	Device Number	Options
F	FLOPPY1	1:1:0	"????????", PUBLIC
G	FLOPPY2	2:1:1	"????????", PUBLIC
S	DISK1	3:2:0	"THEOS4.1", 2.6GB, PUBLIC
CON	CRT:2	5:3:0	L80,P24,PCTERM
PRT1	SPOOLER		L80,P58,QUEUE=A,HPLASER
COM1	SI01	6:5:0	B38400,CTS,PUBLIC

>netterm admin mail

>attach

THEOS® 32 Device Attachment

Logical Name	Physical Name	Device Number	Options
F	FLOPPY1	1:1:0	"????????", PUBLIC
S	DISK1	3:2:0	"THEOS4.1", 3.5GB, PUBLIC
CON	NET2	5:3:0	L80,P24,COLORPC,REMOTE=ME
PRT1	CON		L80,P58,HPLASER,REMOTE=ME
PRT2	SPOOLER		L80,P58,QUEUE=D,HPLASER

After connecting to the THEOS Network Login Server, you have access to the public printers on that system and, by default, the first printer attached on your system is a slave printer and available as PRT1 to programs that you execute on the server. In this example, your first printer is the local spooled printer. PRT2 is the spooler on the Login Server system.

■ Public Printers

On a multiuser system such as THEOS, it is best if printers are usable by all users logged onto the system. Printers should be attached as private printers only if they are physically incapable of being shared (physically connected to your console) or perform a task that is private in nature (executive printer for confidential reports, *etc.*).

A printer can be attached as a public printer in one of two ways. Any printer defined in the system configuration file ([Sysgen](#)) is attached as a public printer during system startup and is available to all users logged onto the system. A printer can be attached as a public printer after startup by using the PUBLIC option of the [Attach](#) command.

```
>attach prt7 centlp3 (public epson
```

A printer attached as public will not be accessible to any user that is already logged on. When they log off and then log on again, they will have access to all public devices attached, including any recently attached public printers.

Public printers may be spooled printers. Refer to the section “[Print Spooler](#)” on page [87](#).

■ Printer Bypass

The printer class code file causes many “generic” command characters to be translated into printer-specific command characters. For instance, the character whose value is 14 is translated into the printer command for boldface type, and carriage-returns are printed as carriage-return, line-feeds. This capability is very desirable and useful in most situations.

However, there are certain capabilities of a printer that are not supported by THEOS. To use some of these capabilities you must transmit a character code that THEOS would normally translate into something else. For example, instructing some printers to change to a different size of character requires the transmission of the character code value 31. Since the value 31 is a code that THEOS uses itself, this code would not be transmitted to the printer but instead would be translated into a carriage-return without a line-feed.

To circumvent this problem, THEOS supports a printer bypass capability. This means that there are methods to tell THEOS to not translate a character, but to merely pass it on to the printer with its original value.

THEOS supports three methods of printer bypass:

- ▶ Single-character bypass
- ▶ Multiple-character bypass
- ▶ Entire file bypass

Single- and multiple-character bypass allow a report to have some characters that are passed directly to the printer and others that are interpreted and possibly translated by THEOS. These two methods are usable by programs written in any language (C, BASIC or EXEC). The third method allows an entire file to be sent to a printer without any interpretation or translation. This third method is usable only by programs written in the THEOS C language.

■ Single-Character Bypass

Single-character printer bypass is similar to single-character console bypass: The character code value to be sent to the printer without interpretation is preceded by a special, reserved character code value. For printers, this bypass character value is 255 (0xff). For consoles the character value is 27 (0x1b).

For instance, to send the following command to a PCL printer:

```
ESC [ 4 ; 0 ; 4 {
```

You would send the string of characters:

```
255 27 91 52 59 48 59 52 123
```

If you need to send a character code value 255 to the printer, you must precede it with another character 255. The first one tells THEOS to pass the second one to the printer.

■ Multiple-Character Bypass

Sometimes it is necessary to send a stream of codes to a printer and not have any of them translated by THEOS. You could precede each character value in the stream with the single-character bypass, effectively doubling the number of characters output by the program. A more efficient method is to use multiple-character bypass.

With this method, the character stream is surrounded with a pair of character codes with a value of 254 (0xfe). This operates similarly to the quoted text capabilities in the C and BASIC languages: The character stream is

started and terminated with this special character. If the stream of characters contains a code value of 254, it is doubled.

For example, to send the stream of codes:

```
255 254 253 252 251 250
```

You would output the following stream to the printer:

```
254 255 254 254 253 252 251 250 254
```

■ File Bypass

There are some situations where you want to send an entire report or file to a printer and not have THEOS interpret any of the character codes. One example is a scanned image being reproduced on a laser printer. The scanned image is a large amount of binary data whose individual character values might correspond to codes that THEOS would normally interpret and translate.

Rather than bypassing individual codes or streams of codes, the best method is to tell THEOS that the entire file is to be sent to the printer without any analysis. To invoke this bypass method, you must be using a program written in the C language. When the printer file is first opened (using the `fopen` function), before any characters or codes have been sent to the printer, the `fctl` function is used to change the `_faccess` value in the file's `fcbl` to indicate that it is a "bypass" file. For example:

```
outfile = fopen("prt1", "wb");
```

Once this has been done, all of the subsequent characters sent to the printer file are passed onto the printer without any interpretation or translation by THEOS.

■ **Print Spooler**

The THEOS operating system provides a print spooler. The spooler solves two basic problems with printers: access by more than one user at a time and program slowdown caused by slow printers. The spooler permits any number of users to access the same printer simultaneously, while also allowing the physical printing operations to be separated from the data processing and formatting operations. As a result, applications can produce reports at a speed governed by the speed of the processor and hard disk, not the slow speed of a printer.

The presence and operation of the print spooler can be transparent to the user and to the application programs. However, a program can take advantage of the fact that a print spooler is enabled on a system. Since applications do not send reports to a physical printer, the number of “printers” available to the program is not limited by how many actual printers are attached to the computer. For instance, a computer with only one physical printer can execute a check printing program that creates the check report at the same time as the check register report. The print spooler accepts both reports and queues them for printing on the single printer when the proper forms are loaded on the printer.

The print spooler provides other advantages in creating and printing reports. Reports that require the use of special forms, such as checks or invoices, can be created and saved by the spooler. When the proper form is on the printer, the spooler can then print all reports that need the special forms. Since the spooler saves the reports prior to printing them, it is easy to ask the spooler to print several copies of a report, to reprint a report, to test the alignment of paper in a printer by printing a sample of a report, *etc.*

■ **Spooler Terms**

There are several terms and concepts unique to the print spooler that must be understood to use the spooler effectively.

Forms. A form is the paper loaded in a printer. Examples of forms are: plain paper, checks, invoices, statements, envelopes, *etc.* The spooler uses single character codes for each form that a printer might use. For example, A, B and C. There can be 64 different forms controlled by the spooler. The form letters for these 64 forms are:

A – Z	26 forms/queues
a – z	26 forms/queues
# \$ % & * + - < = > ^ ~	12 forms/queues

Queues. A queue (pronounced as “cue”) is a line or list of objects waiting for something. Reports are placed in queues waiting for a printer to become available that has the proper form mounted. Queue names are single letters that match the form letters. For instance, a report in queue c is waiting for a printer to become available that has form c loaded in it. There is one queue for each possible form.

■ Starting the Print Spooler

The print spooler is enabled in one of two ways: during system startup or on demand.

The print spooler is enabled and started during system startup if it is defined in the system configuration file by [Sysgen](#).

THEOS® 32 SYSGEN Version 4

Print Spooler

Enable Print Spooler.....(Y)	(Yes/No)
Drive Assignment.....(S)	
Begins Started.....(Y)	(Yes/No)
Check at Boot.....(N)	(Yes/No)
Secure Mode.....(N)	(Yes/No)
Default Size.....(116)	(32 - 32767)

Prt Form	Prt Form	Prt Form	Prt Form
1 A	17	33	49
2 B	18	34	50
3 C	19	35	51
4 D	20	36	52
5 E	21	37	53
6	22	38	54
7	23	39	55
8	24	40	56
9	25	41	57
10	26	42	58
11	27	43	59
12	28	44	60
13	29	45	61
14	30	46	62
15	31	47	63
16	32	48	64

If the spooler is enabled in this system configuration, then all printers defined in the configuration become spooled printers. For additional information about enabling and defining the print spooler in the system configuration, refer to the [Sysgen](#), “[Print Spooler Settings](#)” on page 543.

The spooler may be started manually with the [Spooler](#) command described on page 518.

```
>spooler init
```


When the spooler is initialized manually, all of this sessions's currently attached printers (except for a slave printer) are changed to spooled printers.

It is possible to have non-spoiled, public printers attached by attaching the printer after the spooler is initialized.

The print spooler is controlled with the [Spooler](#) command (see page 518).

■ Specifying Spooler Forms & Queues and Compatibility

Prior versions of the operating system supported 26 forms and queues. With this version 64 forms and queues are supported. To provide compatibility with programs and procedures expecting only 26 possible forms and queues a special syntax is used to request one of the new forms or queues.

For example, to specify one of the original forms or queues (uppercase A through uppercase Z) you use one of the following commands or statements:

```
>ATTACH PRT20 SPOOLER (A
>ATTACH PRT20 SPOOLER (QUEUE=A
>attach prt20 spooler (a
>attach prt20 spooler (queue=a
```

Because of the way that the CSI folds parameters to uppercase, the above four commands each specify that reports directed to PRT20 are placed in queue A by default.

```
>spooler change 12 a
>spooler change 12 (queue=a
```

Both of the above commands change spooled report number 12 to be in queue A.

```
>spooler 22 form a
```

This command tells the spooler that its PRT22 has form A mounted on it and that it is available to print any report in queue A.

```
>copy some.file report.a1h:prt22
```

This command copies the file SOME.FILE to PRT22. If that is a spooled printer it is given the name "REPORT" and placed in queue A with HOLD status.

```
OPEN #1: "REPORT.A1H:PRT22" OUTPUT SEQUENTIAL
```

Similar to the previous Copy command, this MultiUser BASIC statement opens a file for output on PRT22. If that is a spooled printer it is given the name "REPORT" and placed in queue A with HOLD status.

Notice that all of the above examples used queue or form A (uppercase A) even though they might have used a lowercase a in the specification. To specify that you want one of the new queues or forms you must use a new syntax. This new syntax is identical to the previous syntax except it uses a leading \$ character before the queue or form specification. When the \$ character is used at the start of the queue/form specification the characters following are interpreted as lowercase or special character specifiers. You may also enclose the queue or form specifier within quotation marks to indicate that its case mode is significant.

```
>ATTACH PRT20 SPOOLER (QUEUE=$A
>attach prt20 spooler ("a"
>attach prt20 spooler (queue=$a
>spooler change 12 "a"
>spooler change 12 (queue=$A
>spooler change 12 (queue=$a
>spooler 22 form "a"
>copy some.file report.$a1h:prt22
```

```
OPEN #1: "REPORT.$A1H:PRT22" OUTPUT SEQUENTIAL
```

All of the above commands and statement refer to queue or form a (lowercase "a"). When the "QUEUE=\$" syntax is used it is not necessary to enclose it in quotation marks because that syntax always forces the queue specifier to be interpreted as lowercase.

■ Default Queue Assignment

Spooled printer attachments have a default queue association. For instance:

```
>attach
```

THEOS® 32 Device Attachment

Logical Name	Physical Name	Device Number	Options
F	FLOPPY1	1:1:0	"???????",PUBLIC
G	FLOPPY2	2:1:1	"???????",PUBLIC
S	DISK1	3:2:0	"THEOS4.1",1.6GB,PUBLIC
CON	CRT:2	5:3:0	L80,P24,PCTERM
PRT1	SPOOLER		L80,P58,QUEUE=A,HPLASER
PRT2	SPOOLER		L80,P58,QUEUE=I,HPLASER

This association provides a default queue assignment for that printer. Using the above attachment, any report sent to PRT1 will be placed in queue A and any report sent to PRT2 will be placed in queue I, unless a dif-

ferent queue is specified when the report is created (see “[Overriding Queue Defaults](#)” on page 92).

Note that the queue assigned to a printer’s attachment is not necessarily the same as the form that the spooler uses for its printer. In fact, there is no connection between a user’s printer attachments and the spooler’s printers. For instance, you may only have one physical printer controlled by the spooler but you may have four printers attached.

>spooler

```
Printer #1 "HSLP1" L80,P58,HPLASER,W8
-- is waiting for work
-- and has form "IJK" mounted
```

>attach

THEOS® 32 Device Attachment

Logical Name	Physical Name	Device Number	Options
F	FLOPPY1	1:1:0	"????????",PUBLIC
G	FLOPPY2	2:1:1	"????????",PUBLIC
S	DISK1	3:2:0	"THEOS4.1",1.6GB,PUBLIC
CON	CRT:2	5:3:0	L80,P24,PCTERM
PRT1	SPOOLER		L80,P58,QUEUE=A,HPLASER
PRT2	SPOOLER		L80,P58,QUEUE=B,HPLASER
PRT3	SPOOLER		L80,P58,QUEUE=\$,HPLASER
PRT4	SPOOLER		L80,P58,QUEUE=U,HPLASER

In the above situation, unless the defaults are overridden, all reports sent to PRT1 are put in queue A, PRT2 in queue B, PRT3 in queue \$ and PRT4 in queue U. Also, because the printer has forms I, J and K mounted, none of these reports will be printed at this time.

Similarly, you can specify default hold status and number of copies to print with printer attachments.

>attach prt3 (hold copies=4

THEOS® 32 Device Attachment

Logical Name	Physical Name	Device Number	Options
F	FLOPPY1	1:1:0	"????????",PUBLIC
G	FLOPPY2	2:1:1	"????????",PUBLIC
S	DISK1	3:2:0	"THEOS4.1",1.6GB,PUBLIC
CON	CRT:2	5:3:0	L80,P24,PCTERM
PRT1	SPOOLER		L80,P58,QUEUE=A,HPLASER
PRT2	SPOOLER		L80,P58,QUEUE=B,HPLASER
PRT3	SPOOLER		L80,P58,QUEUE=m,COPIES=4,HOLD,HPLASER
PRT4	SPOOLER		L80,P58,QUEUE=U,HPLASER

When you logoff an account the printer attachments are reset to their initial state. This initial state is 1 copy and nohold. The queue is determined

by either the system configuration file for spooled printers or, if the spooler was started after the system was booted, by the printer number. For instance, PRT1 will have queue A, PRT2 will have queue B, *etc.*

■ Overriding Queue Defaults

When a report is sent to the spooler it uses the default queue, copies and hold status associated with the printer attachment. You can override these defaults and give the report a name by specifying it when you specify the destination name.

```
>copyfile some.text prt3
```

The above command specifies that SOME.TEXT file is copied to PRT3 using the default queue, copies and hold status.

```
>copyfile some.text special.y4h:prt3
```

This command specifies that SOME.TEXT file is copied to PRT3 but is given the name “SPECIAL” and placed in queue Y, marked for printing four copies with HOLD status.

The general syntax for this type of specification is:

report-name.qch:prtnn

report-name.\$qch:prtnn

report-name This is a one to eight character name and is displayed in a spooler list and status reports.

q A single character designating the uppercase queue letter for this report.

\$q This syntax is used to specify one of the extended queues (lowercase a–z, #, \$, %, &, *, +, -, <, =, >, ^ or ~).

c One or two digits specifying the number of copies to print for this report.

h An “h” or “n” specifying hold or nohold status. Held reports are not erased after they are printed.

This same syntax can be used in programs when it opens a printer for output:

```
OPEN #200: "myreport.$z5h:prt60" OUTPUT SEQUENTIAL
```

■ Spooled Printer Forms

The form mounted on a printer specifies what queues it will print. For instance, a printer with form D mounted on it will only print reports that are currently in queue D. A printer with form X mounted will only print reports in queue X, *etc.*

As indicated in the previous example, a spooled printer may have more than one form mounted on it. In that example, spooled printer 1 had forms I, J and K mounted. This means spooled printer 1 is to be used to print any report that is in queue I, J or K. A spooled printer may have as many as eight forms specified.

The initial form assigned to each of the printers controlled by the spooler is defined by the system configuration file (see [Sysgen](#), “[Print Spooler Settings](#)” on page 543). If a spooled printer is not defined in the system configuration file because the spooler was started after the system was booted, its initial form assignment is determined by its printer number.

```
>attach prt1 hslp1 (public c135

>attach prt5 hslp2 (public c135

>spooler init

>spooler status
Printer #1 "HSLP1" L80,P58,W8,HPLASER
    -- is stopped
    -- and has form "A" mounted
Printer #5 "HSLP2" L80,P58,W8,HPLASER
    -- is stopped
    -- and has form "E" mounted
```

More than one spooled printer may have the same form mounted. For instance:

```
>spooler status
Printer #1 "HSLP1" L80,P58,W8,HPLASER
    -- is waiting for work
    -- and has form "A" mounted
Printer #2 "HSLP2" L80,P58,W8,HPLASER
    -- is waiting for work
    -- and has form "B" mounted
Printer #3 "MULTI12" L80,P58,W8,EPSON,B9600
    -- is waiting for work
    -- and has form "A" mounted
Printer #4 "MULTI13" L80,P58,W8,EPSON,B19200
    -- is waiting for work
    -- and has form "A" mounted
```

In this situation, any of the printers PRT1, PRT3 or PRT4 can be used to print a report from queue A. If there are two reports in queue A ready to be printed then the first one will be sent to PRT1 and the second report will be sent to PRT3.

■ Spooler Secure Mode

The spooler saves reports that it hasn't printed yet and reports that it has printed but have "hold" status in the SYSTEM.SPOOLER library. These report files are normal stream files and it is possible for a user to copy and read them without using the spooler.

And, because they are normal files, a user may erase a spooled report with the Erase command, even though doing so will corrupt the spooler queue causing unpredictable results.

For security reasons, you may find it desirable to protect or secure these reports. For instance, one or more of the reports may be payroll checks. Normally, this report could be viewed by any user that can find and access the report file.

If the spooler is enabled in the system configuration file and the "Secure Mode" is enabled, spooled reports are protected and only the spooler can access them. With Secure Mode enabled, all spooled report files are created with read, erase and hidden attributes. This prevents all users from reading or viewing any spooled report. The spooler can still erase reports after it has printed them.

6 User Account Environments

THEOS is a multiuser operating system. As such, it allows multiple users to log onto the system at one time. THEOS manages this multiuser operation through user partitions and user accounts.

A **user partition** is an area of memory in the computer system that THEOS dedicates to a particular console or task. User partitions are described in Chapter 3 “[Multiuser and Multisession](#),” starting on page 55.

A **user account** is a name and number associated with a logged-on user. All directories, libraries and files on a disk are owned by accounts. THEOS can restrict access to files so that they are not accessible to users logged onto an account that does not own the file. A user account defines file ownership and an environment that describes how the system interacts with the operator and what types of commands they may use.

You must log onto an account before you can use the computer. Initially, there is one account on the computer with an account name of SYSTEM and a password of THEOS. Other user accounts are created and defined with the Account command described on page 156.

■ Account Name and Number

Accounts are externally identified by their account name. An **account name** is a 1–8 character name created with the Account command. The name may be composed of letters, digits, dollar signs and underscore characters.

Account names are used when you log onto an account, when you send mail to a user of an account and when you identify a file’s account owner in its path specification. For instance:

```
>logon karen

>mail boss "Is my bonus check ready yet?"

>list billing\pastdue.invoices
```

Internally, the system refers to accounts by their **account number**, which is also called the **account id**. Every account name has an account number in the range of 0–4,096.

There are two types of accounts: public and private. The public or **system account** is account number zero and normally has the name SYSTEM. All

accounts with an account number other than zero are called **private accounts**.

When logged onto a private account, a user has access to the files owned by that private account. Files owned by the system account can be accessed by private account users without specifying the account name in the file's path. Files owned by other private accounts may only be accessed by specifying the account owner in the path to the file. For example:

```
>logon private           ; log onto private account
>list my.file            ; list one of your files
>list system.theos32.devnames ; list system device names file
>list billing\late.payments ; list another user's file
```

Two or more account names may have the same account number and, when they do, they are called **synonym accounts**. A synonym account is a different account name assigned to the same account number. If two users are logged onto account names that are synonyms to each other, they will have access to the same files because they are logged onto the same account number. However, because they are logged onto different account names, they may have different passwords, privilege levels and account environments.

■ Passwords

An account may have a **password** defined and, if so, a user must enter the password correctly before logging onto the account. Passwords are encrypted and are never displayed on the console except with the Account command.

```
Logon please: BILLING
```

```
Password? *****
```

The plain-text form of a password is never displayed, even by the Account command, except when it is initially defined or changed.

■ Privilege Level

Every account has a privilege level associated with it. **Privilege levels** are numbers in the range of 0–9 and control which programs can be executed.

Programs have a privilege level requirement associated with them. When a user attempts to execute a program with a privilege level greater than the account's privilege level, the system reports “Insufficient privilege” and the user is prevented from executing the program.

Some programs have different privilege level requirements for some features. For instance, the **Set** command only requires a privilege level of one for most of its functions, but it requires a privilege level of five to change the system date or the system time.

In general, command privilege level requirements are divided into six categories:

Privilege Level	General Function
5	Command makes a change to the system, possibly affecting other users. For instance, Account, Cache and Force. This privilege level should be reserved for the system manager.
4	Command performs major system maintenance. For instance, Archive, Backup and Restore.
3	Command performs other system maintenance. For instance, Attach, Keyword and Message.
2	Program development. For instance, B32, CC32 and Link32.
1	Programs that create or modify files. For instance, Change, Create, Erase and Mailbox.
0	All other commands.

Table 9: Privilege Level Categories

Privilege levels may be used by third-party utilities and by application programs to restrict access to various features of the product.

Refer to Appendix F: “[Command Privilege Levels](#),” starting on page 741, for a complete listing of the commands and their general privilege level requirements as distributed. The privilege levels associated with a program can be displayed with the [FileList](#) command (see page 326) and it can be changed with the [Change](#) command (see page 216).

■ Account Environment

Each account name has an account environment associated with it. This environment determines how the operating system interacts with the user. The environment is defined with the [Account](#) command and can be modified with the [Set](#) command.

An account environment is composed of environment switches, environment variables and default libraries.

■ Environment Switches

Environment switches are predefined environment fields that have a limited set of possible values.

ABBREV. This on/off field controls whether command names may be abbreviated. In the [Account](#) command, this switch is referred to as “Abbreviation.”

Although most THEOS commands are English words and are easy to remember and type, some are either long or are used so often that it is desirable to be able to abbreviate them. For instance, the [FileList](#) command has an allowed abbreviation of F because it is used so frequently.

When the ABBREV field is ON (or “Abbreviation” is “Yes”), and the [STDSYN](#) field is ON or [SYNONYM](#) is set to a custom synonym file, the abbreviations in the standard synonym file or your custom synonym file may be used when specifying command names. Refer to the individual commands described in Part II of this manual for specifications of the minimum spelling for command abbreviations.

When ABBREV is OFF (or “Abbreviation” is “No”), command abbreviations are disabled and command name synonyms are not recognized.

Refer to page [734](#) for a description of the standard synonym file.

BREAK. This numeric-valued field defines the key used to start a break-key sequence. In the [Account](#) command, this field is referred to as “System BRK code.”

A value of zero means that the [Break](#) key is used for break-key sequences. Use another value if the console you are using does not have a [Break](#) key, or when usage of the [Break](#) key is needed for other purposes, such as when you are using [THEO+COM](#) to connect to another operating system.

CSI Case Mode. This switch controls the case mode for command-line entry. It may be either L or U, indicating that command lines are accepted as typed (L) or folded to uppercase (U). Although you may not change this switch with the SET command, you can toggle it to its other state by entry of **Ctrl+C** at the start of any command line.

DECIMAL. This field controls the acceptance and display of numbers. In the Account command, this switch is referred to as “Decimal is Comma.”

When DECIMAL=COMMA (or “Decimal is Comma” is “Yes”), numbers are entered and displayed using the European standard with the comma indicating the “decimal point” and the period indicating the thousands separator. When DECIMAL=DOT (or “Decimal is Comma” is “No”), numbers are entered and displayed using the American standard with the period indicating the decimal point and the comma indicating the thousands separator.

```
>show decimal
DECIMAL=DOT

>1.2345 * 987.67
1,219.278615

>set decimal=comma

>1,2345 * 987,67
1.219,278615
```

DECIMAL may only be set to COMMA, DOT or PERIOD. PERIOD is synonymous with DOT.

LANG. This numeric-valued field controls which set of files are used for help displays, menus, messages and keywords. In the [Account](#) command, this switch is referred to as “Language Code.”

The set of files affected by this switch include:

```
SYSTEM.HELP32n.*
SYSTEM.MENU32n.*
SYSTEM.TEOS32.KEYWORDn
SYSTEM.TEOS32.MESSAGEN
```

The *n* in the above file names refers to the language code. When the language code is zero, no number is used in the file name.

Most computer systems use only one language for text. As distributed by THEOS, this language is English and corresponds to a language code of

zero. When the operating system is distributed in countries using a language other than English, the distributor might translate the files into the native language and leave the language code at zero.

In countries with multiple native languages, it is possible to have multiple sets of files, one set for each language. A user would set the language code to the number of the set of files with the language preferred. In this situation, the recommended codes are:

Language Code	Language	Language Code	Language
0	English	5	Spanish
1	British	6	Swiss
2	French	7	Latin American
3	German	8	Canadian French
4	Italian	9	Belgian

Table 10: Recommended Language Codes

MSG. This on/off switch controls whether messages sent by other users are displayed on your console or sent to your mailbox. In the [Account](#) command, this switch is referred to as “Receive messages.”

When this switch is OFF (or “Receive messages” is “No”), messages sent by other users are not displayed on your console. When the sending user attempts to send a message to your console with the [Msg](#) command, they are informed that “User is not receiving message” and they are asked “Do you wish to deposit in mailbox?”

When MSG is ON, messages sent are displayed in a “pop-up” window on the receiving user’s screen and they must be acknowledged with entry of **Enter** or **Esc** before they are cleared from the screen.

QUIT. This is an on/off switch that controls whether or not the operator may abort program execution with the **Break**, **Q** key sequence. In the [Account](#) command, this switch is referred to as “Disable BRK-Q.”

When QUIT is off (or “Disable BRK-Q” is “Yes”), entry of the **Break**, **Q** key sequence is ignored and the user must use normal program exits to terminate program execution.

RDYMSG. This on/off switch controls whether or not the CSI displays the “ready message” after each command is executed. In the [Account](#) command, this switch is referred to as “Ready message.”

When the ready message is displayed, it contains four items of information: the program’s return code, the current system time when the program terminated, the total elapsed time of execution in minutes and seconds, and the total computer or CPU time used by the program during execution.

```
>disk s (verify
Cylinder: 9   Head: 5       8%

RC = 254, 16:48:33, ET = 0:47, CPU = 41.940
```

The above ready message tells you that the program was exited with a return code of 254, which is a special return code indicating that the program was aborted by the operator with a **Break**, **Q** entry. The current system time when the program exited was 4:48 p.m. The program executed for 47 seconds and used 41.940 seconds of CPU time.

SEARCH. This field contains a list of disk codes. When a file or program is requested without specifying the disk, this field controls which disks are searched and the sequence that they are searched. In the [Account](#) command, this switch is referred to as “Search sequence.”

This search sequence is used by the CSI for program searches (see “[Command Search Sequence](#)” on page 48) and for file searches (see “[File Search Sequence](#)” on page 140).

STDSYN. This on/off switch controls whether or not the CSI uses the standard synonym file when looking up command abbreviations and synonym names. In the [Account](#) command, this switch is referred to as “Standard synonym.”

The standard synonym file is SYSTEM.TEOS32.SYNONYM. It contains the minimum spelling (abbreviations) and the synonyms for the standard THEOS commands. If STDSYN is ON (or “Standard synonym” is “Yes”), this file is enabled and the synonym names defined in it may be used. Refer to page 734 for a complete description of this file

When STDSYN is OFF (or “Standard synonym” is “No”), the standard command synonyms and abbreviations cannot be used. Command names must be fully spelled out, even if the ABBREV switch is ON.

Note: You may have a private synonym file that defines non-standard synonyms and abbreviations. See the [SYNONYM](#) environment variable on page

110. A private synonym file uses the same format and syntax as the standard synonym file described on page [734](#).

WORK. This environment field tells the operating system where you want temporary and work files to be saved.

■ Environment Variables

Environment variables are environment fields that may have any value. These variables are of two types: system-defined and user-defined.

The system-defined variables are described below and are used by the operating system and the standard set of commands to customize the interaction of the system with the user.

User-defined environment variables are used by commands and applications supplied by third parties and programs that you write using MultiUser BASIC, C or the EXEC language.

Environment variables are part of the account environment and, as such, may be initialized with the account definition created with the [Account](#) command, set or changed with the [Set](#) command, and displayed with the [Show](#) command.

CREATE. This variable tells the operating system which attributes and protection codes to apply when creating files. When this variable is not defined, files are created with the following attributes: modified, not hidden, execute protected, and shared read and shared write protected.

The default attributes and protection codes defined with this variable are specified as a single, numeric value representing the combined codes. Like most numeric values used with THEOS, the value may be specified as a decimal number or a hexadecimal value. However, because the attributes and protection codes are interpreted as a bit-mapped field, it is generally easiest to specify the value in hexadecimal.

Bit-position										
7	6	5	4	3	2	1	0		Decimal	Hexadecimal
								Read protect	1	01
								Write protect	2	02
								Execute protect	4	04
								Erase protect	8	08
								Shared read protect	16	10
								Shared write protect	32	20
								Modified	64	40
								Not hidden	128	80

To set the desired default attributes and protection codes, define the `CREATE` variable to be the total value of the individual codes. For instance, to set the default to not hidden (128), erase protected (8), you would set `CREATE` to either 136 (128+8) or 0x88.

The modified attribute is always set when a file is created and cannot be changed with the `CREATE` variable.



The `CREATE` variable determines the attributes and protection codes for all files that are created. This includes the work files created by some commands and applications. Setting the attributes inappropriately can make the file unusable to a program. For instance, if the default attributes are set to read and write protected, work files will be created that cannot be used and the program will fail.

DATEFORM. This variable is not truly an account environment variable because it applies to all users logged onto the system. It is initialized by the system configuration when the system is booted. When a user changes the value of this variable, all other users are affected.

The `DATEFORM` variable tells the system how to accept and display dates. It can have one of three possible numeric values:

1. American: *mm/dd/yyyy*.
2. European: *dd-mm-yyyy*.
3. International: *yyyy.mm.dd*.

Note that the different date formats use different delimiters between the components of the date. This can assist you in determining the meaning of a date displayed.

The date format is used by the operating system for all dates displayed by the system and its commands. It also determines the meaning of dates entered. For instance, when the `DATEFORM` is 3, changing the date with the [Set DATE](#) command requires that you enter the date in the international format.

To change the `DATEFORM` value, you must have a privilege level of five.

DATEIN. This variable is not truly an account environment variable because it applies to all users logged onto the system. It controls how two-digit year number dates are interpreted by MultiUser BASIC applications. It may have one of three different values:

- 0 Two-digit year numbers are always interpreted as 20th-century dates (19nn). This default value reflects the behavior of THEOS prior to implementing this feature.
- 1 Two-digit year numbers are always interpreted as current-century dates.
- 2 Two-digit year numbers are interpreted temporally. If the difference between today's system date and the date interpreted as a 20th-century date is greater than the difference between today's date and the date interpreted as a 21st-century date, then the date is treated as a 21st-century date. Otherwise, it is treated as a 20th-century date.

DATEOUT. This variable is not truly an account environment variable because it applies to all users logged onto the system. It controls how two-digit year number dates are displayed by MultiUser BASIC applications. It may have one of three different values:

- 0 Twentieth-century dates are output with two-digit year numbers, other dates are output with four-digit year numbers. This default value reflects the behavior of THEOS prior to implementing this feature.

When a two-digit year date is input to a date-display function, the operation of this mode depends upon the [DATEIN](#) setting and the value of the date that results from that interpretation.

- 1 Dates are always output with two-digit year numbers.
- 2 Dates are always output with four-digit year numbers.

The DATEOUT environment variable affects the operation of applications using the MultiUser BASIC DTE\$ and DATE\$ functions. System utilities do not use these functions and are not affected by these settings.

DIALDIR. This variable tells the [THEO+COM](#) command where it can find the telephone number dialing directory and its modem script language files. The variable specifies the complete path to the dialing directory without specifying the file name. For instance:

```
>set dialdir=system\dialdir:s
```


This setting tells **THEO+COM** that the dialing directory file is found in the subdirectory **DIALDIR**, on the system disk, in the account **SYSTEM**.

FILETYPE. This variable is used to specify a default file-type for file specifications. For instance, when creating and maintaining programs in the C language, it is convenient to specify a default file-type of **c**.

Most commands that operate on files allow the file-name to be specified without a file-type. In these commands, when the file specification is “type-less” and there is no default library defined, the value of **FILETYPE** is appended to the file specification to form a complete file description with a file-name and a file-type. For instance:

```
>set filetype=c
```

```
>list program
```

The **List** command displays the **PROGRAM.C** file.

To indicate that the file does not have a file-type component, you should specify the file description with a period terminator:

```
>erase sample
"SAMPLE.C:S" erased.
One file erased, 7,936 bytes recovered.
```

```
>erase sample.
"SAMPLE.:S" erased.
One file erased, 11,520 bytes recovered.
```

If the file specification is typeless and there is a default library defined, the file specification is assumed to be a member of the default library:

```
>show library
LIBRARY = PROGRAM.SOURCE
```

```
>erase example
"PROGRAM.SOURCE.EXAMPLE:S" erased.
One file erased, 23,040 bytes recovered.
```

The commands that recognize and use the **FILETYPE** environment variable include: **Cat**, **Erase**, **File**, **FileList**, **Head**, **List**, **Look**, **More**, **Tail**, **Touch**, **Unique**, **WinWrite** and **WordCount**.

HISTORY. This variable is not truly an account environment variable because it applies to all users logged onto the system. It is initialized by the system configuration when the system is booted. When a user changes the value of this variable, all other users are affected.

The HISTORY switch controls whether or not command history logging is performed. In the [Sysgen](#) command, this switch is referred to as “Save History Information” and may have a value of “Y” or “N.”

With this field set to “Y,” a record is written to the SYSTEM.HISTORY file every time that the system is booted, a user logs on or off the system and when a command starts or finishes. For more information about this file, refer to “SYSTEM.HISTORY” on page [722](#).

To change the HISTORY value, you must have a privilege level of five.

HOME. This variable defines the default subdirectory used by the [ChDir](#) command. The logon process automatically defines the HOME variable to be the same as [SUBDIR](#). If [SUBDIR](#) is not defined in the account definition, HOME is set to the root directory.

```
>logon develop

>show subdir
SUBDIR = /PROGRAMS:S

>show home
HOME = /PROGRAMS:S

>cd ../doc

>show subdir
SUBDIR = /DOC:S

>cd

>show subdir
SUBDIR = /PROGRAMS:S
```

In the above example, in the last usage of the [ChDir](#) command (synonym name is CD), no directory is specified. The [ChDir](#) command uses the HOME directory in this situation.

LINEGRAPH. This variable tells THEOS commands whether or not to use line-drawing characters to box and separate columns in their displays. For instance, when LINEGRAPH is not defined, or is defined as Y or YES, the Who command displays:

```
>who
```

THEOS® 32 Who List				
PID	Account	Program	Terminal	Attach Options
1	SYSTEM	HELP	CRT1:1	C90,L80,P24
2	SYSTEM	WHO	CRT1:2	C90,L80,P24
3	SAMPLES	CSI	CRT1:3	C90,L80,P24
4	PRIVATE	WINWRITE	CRT1:4	C90,L80,P24
5		LOGON	MULTI1	C180,L80,P24,B38400,PCXON/XOFF
6		LOGON	MULTI2	C180,L80,P24,B38400,PCXON/XOFF
10	JUDITH	CHECKREG	NET1	C210,L80,P24,REMOTE=Accounting

However, when LINEGRAPH is defined as N or NO, the [Who](#) command displays:

```
>who
 1 SYSTEM  HELP      CRT1:1      C90,L80,P24
 2 SYSTEM  WHO       CRT1:2      C90,L80,P24
 3 SAMPLES CSI       CRT1:3      C90,L80,P24
 4 PRIVATE WINWRITE  CRT1:4      C90,L80,P24
 5          LOGON    MULTI1     C180,L80,P24,B38400,PCXON/XOFF
 6          LOGON    MULTI2     C180,L80,P24,B38400,PCXON/XOFF
16 JUDITH  CHECKREG  NET1       C210,L80,P24,REMOTE=Accounting
```

The commands that recognize and use the LINEGRAPH environment variable include: [Attach](#), [FileList](#), [Show](#), [Tree](#) and [Who](#).

PROMPT. This variable defines the content of the CSI prompt string. When this variable is not defined, the default CSI prompt is used (>).

A prompt string may contain plain text and ***prompt variables***. Plain text is displayed exactly as specified. Prompt variables are replaced with the current value of the variable.

Many of the following prompt variables use characters that the CSI might interpret for itself. To make sure that the prompt string is accurately passed to the Set command, enclose the string in quotes.

```
>set prompt="\!\g"
```

Variable	Meaning
\a	Current account name.
\b	Vertical bar character ().
\c	Current PATH environment.
\ c	Current PATH environment preceded by space character.
\d	Current date in current date format.
\g	Default CSI prompt character (>).
\i	System IDENT. See Sysgen command and Setup NET command.
\l	Current LIBRARY environment.
\ l	Current LIBRARY environment preceded by space character.
\n	New-line (CRLF).
\p	Current user partition number.
\s	Complete path to current working directory.
\ s	Complete path to current working directory preceded by space character.
\#s	Abbreviated path to current working directory preceded by space character.
\t	Current time of day in 24-hour format, including seconds. While the command line is empty, the time of day is updated every second. Do not use with the \n variable.
\u	Current account number.
\w	Left angle bracket character (<).
\z	Current session number.
\!	Current command line number. Command numbers are incremented every time a command is entered and reset when you Logoff.
\\	Single backslant character (\).
!crtattrib	Performs one of the CRT video display attributes described in the next table.
!nnn	Outputs the character whose value is <i>nnn</i> . See “ THEOS Character Sets ” on page 761.
!!	Single exclamation mark (!).

Table 11: CSI Prompt Variables

The variables described above, and the CRT attributes described below, are case-insensitive. That is, they can be specified with uppercase or lowercase characters.

crtattrib	Displays
home	Move cursor to top left corner of screen.
clear	Clear screen, home cursor.
up	Move cursor up one line.
down	Move cursor down one line.
left	Move cursor left one column.
right	Move cursor right one column.
eol	Erase to end-of-line.
eos	Erase to end-of-screen.
eu	Erase unprotected characters on screen.
bon	Set blink attribute on.
boff	Set blink attribute off.
ulon	Set underline attribute on.
uloff	Set underline attribute off.
rvon	Set reverse video attribute on.
rvoff	Set reverse video attribute off.
hon	Set half intensity attribute on.
hoff	Set half intensity attribute off.
fon	Set format attribute on.
foff	Set format attribute off.
kon	Set cursor on.
koff	Set cursor off.
son	Begin status line display text.
soff	End status line display text.
bel	Ring the console bell.

Table 12: CSI Prompt CRT Attributes

SUBDIR. Unless otherwise specified, all files are assumed to be in the current working directory. The current working directory is set with the Chdir command or with the SUBDIR environment variable. (ChDir sets the SUBDIR variable).

When setting SUBDIR, specify the complete path to the directory. Do not use the relative path specifiers . (dot) or .. (dot, dot). You should also specify the disk-letter. For instance:

```
>set subdir=/program/doc:s
```

SYNONYM. This variable defines a user-defined file for command synonyms and abbreviations. When used, specify the complete path and name of the file. When no file type is specified, a file type of SYNONYM is assumed.

```
>set synonym=custom/private.names:s
```

Refer to the [ABBREV](#) and [STDSYN](#) environment switches in this chapter for additional information about command synonyms.

■ Default Libraries

The default libraries defined in the account environment are special instances of predefined environment variables. Several programs, particularly the MultiUser BASIC and C language compilers, operate on sets of data files, such as the set of program source files comprising an application package, the set of object files for the program source, *etc.*

When defining any of these library environment variables, the file specified must exist and it must be a library. You may specify the complete path to the library.

LIBRARY. This variable defines the default library and is used by most THEOS commands when a file name is specified without a period. If you use a library for a collection of text files (program source, letters, memos, *etc.*), you should define a default library because it makes it much easier to use the files in the library. For instance:

```
>ww program1
```

This command line is easier to type than:

```
>ww program.source.program1
```

To use the simpler command line, specify a LIBRARY of PROGRAM.SOURCE.

The LIBRARY variable has precedence over the [FILETYPE](#) variable described earlier. When both the [FILETYPE](#) variable is defined and there is a LIBRARY defined, THEOS commands use the LIBRARY definition when given a file name that does not contain a period.

For instance:

```
>set library sample.library
>set filetype=sample
>list example
```

The file listed is SAMPLE.LIBRARY.EXAMPLE, not EXAMPLE.SAMPLE.

LINKLIB. This environment variable is used by the program linker (Link32 command) when the FILE option is specified. The library should contain member files that specify the Link32 directives for linking a program.

OBJLIB. This environment variable is used by the language compilers as a location to save the object files generated and by the Link32 command when looking for existing object files.

PATH. This variable tells the CSI where to look for programs to be executed. This is a special type of default library environment variable because it can specify either a single library or a list of paths for the locations of commands and EXEC language programs.

When PATH is defined as a single name, it must be the name of a library with a file type of CMD32. For instance:

```
>set path=user
```

This setting tells THEOS that commands may be found in the library named USER.CMD32.

When PATH is defined as a list of names, each name may be a directory specification or the path and name of a library containing programs. For instance:

```
>set path=/special,private.commands,/custom/private.commands
```

This setting tells THEOS that commands may be found in the directory /SPECIAL, in the library named PRIVATE.COMMANDS in the current directory, or in the library PRIVATE.COMMANDS in the /CUSTOM directory. These locations are searched in the sequence specified.

For additional information about the usage of the PATH variable, refer to [“Command Search Sequence”](#) on page 48.

7 Disks and Tapes

THEOS supports a wide variety of disks and tapes which can be used for on-line storage of data and for backup, archival or off-line storage.

■ Disk Drives

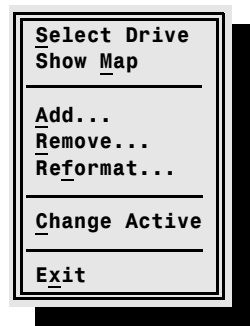
THEOS uses many types of disk drives including hard disks, removable hard disks, CD-ROM discs, floppy disks, RAM disks and image drives. The term ***drive***, or disk drive, refers to a physical disk; the term ***disk*** refers to the logical disk volume on a drive. A single physical drive may contain multiple logical disks, or a single logical disk might span two or more physical hard drives. For drives other than hard disk drives, the terms disk and drive are synonymous because a floppy disk drive, RAM disk drive and an image drive may only contain one logical disk.

■ Hard Disk Drives

THEOS supports many types of hard disk drives including IDE and SCSI. Older drive types are also supported. Because of hardware design limitations, a system may have as many as four IDE disk drives or 28 SCSI disk drives (four adapters with as many as seven disks per adapter). As long as their addressing requirements and IRQ usage do not conflict, a system may have a mixture of these drive types.

Disk drives may be of any size. Drives larger than 4GB must be partitioned into two or more logical drives with a maximum size of 4GB for any partition on a drive.

Hard disk drives can only be formatted with the [Setup DISK](#) command described on page 490. With this command you can add, change and delete partitions on a drive, add or delete logical disks in a partition, and format disks.



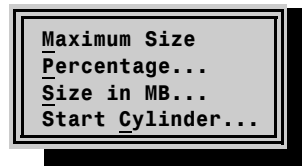
The **Setup DISK** command also changes the active partition on a drive. This is useful only for drives with more than one primary partition. For instance, a drive with a THEOS primary partition could have a DOS or Windows primary partition too. Changing the active partition on a drive with multiple primary partitions adds the THEOS MultiBoot program to the drive's boot sectors. See “**THEOS MultiBoot**” on page 23.

- **Hard Drive Partitions**

A hard disk drive may be used as a single disk or it can be divided into two or more logical disks. Dividing a hard disk drive is called partitioning and is done with the **Setup DISK** command.

THEOS Primary Partition. Except for CD-ROM and floppy drives, every drive used by the THEOS operating system must have a THEOS primary partition. This partition may be an existing partition created by a prior version of the operating system, or it may be a new partition created by this version. A THEOS primary partition has an implied logical disk of the same size as the partition size.

When a partition is created on a drive, it can be allocated to use the entire drive space, a percentage of the drive space, a size defined in megabytes or a size defined by specifying the starting cylinder number and the number of cylinders.



A drive with a primary partition that is smaller than the entire drive space can use the remaining portion of the drive for a THEOS extended partition or for another operating system's disk partitions. Partitions for other operating systems must be created by the other operating system.

Drive #1 Parameters					
SCSI Drive Target ID: 0					
MAXTOR LXT-535S					
509 cylinders, 64 heads, 32 sectors per track					
Capacity: 509MB = 533,725,184 Bytes					
Partition					
Type	Start	Size	Active		
THEOS 4.0 Primary	1	308MB	61%	✓	
DOS Primary	309	200MB	39%		

THEOS Extended Partition. A drive that has a THEOS primary partition using less than the entire drive space may have one THEOS extended partition. An extended partition is merely a secondary partition on the drive that THEOS can use for logical disks. An extended partition has one or more logical disks defined within it.

There are two basic reasons for using extended partitions on a drive. When a drive is larger than 4GB, it must be divided into two or more partitions so that each partition is 4GB or less. Another reason is to allow a single drive to be used as two or more logical disks. The secondary logical disks can be used to maintain different databases for various purposes, such as one disk for each customer supported, *etc.*

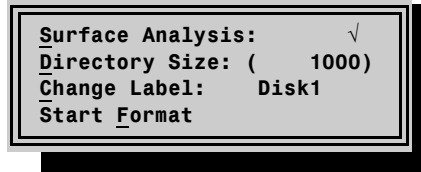
Logical Disk. A primary partition has an implied logical disk defined within it. An extended partition must have one or more logical disks explicitly created within it. The THEOS file structure is only written to logical disks and only logical disks can contain directories and files.

Spanned. A spanned disk is a logical disk that starts with a partition on one drive and continues onto partitions on other drives. This allows a logical disk to be larger than any one of the physical drives on the system. One of the advantages of spanned disks is simplified archives. On a system with multiple drives using a spanned logical disk, there need be only one logical disk attached and therefore, only one disk to archive. When there are multiple disks present on a system, each one has to be archived separately.

Another advantage of spanned disks is the ability to use very large files. A file must reside on one disk only. With spanned disks, the file can be on one disk but spanning several drives.

- **Formatting Hard Disks**

When a logical disk is created (or the implied logical disk when a primary partition is created), the disk must be formatted.



During the format process you can request surface analysis. **Surface analysis** reads and writes to every sector of the logical disk surface, attempting to detect any marginal or bad areas of the disk. Bad sectors are added to the **bad sector map** of a disk and the THEOS disk system will avoid using those sectors.

If a portion of a disk becomes unusable in the future, the only method of marking the unusable sectors as bad is to reformat the disk.

The disk label and directory size of a disk can be changed after formatting by using the [Disk](#) command described on page [296](#). However, the directory size can only be changed by clearing all files off of the disk.

■ Removable Hard Disk Drives

THEOS supports many types of removable hard disk drives. The current list of drives supported can be found on the Internet by following the link on the page:

<http://www.theos-software.com/support/>

A removable hard drive has the combined attributes of a standard, non-removable hard disk drive and a floppy disk drive. It has the capacity of hard disks but the removability and portability of a floppy.

Removable hard drives are used in the same way as non-removable hard drives. Refer to “[Hard Disk Drives](#)” on page 113 for information about attaching, partitioning, formatting and using removable hard disk drives.

One significant difference between a removable and a non-removable hard disk drive is that the disk can be physically removed from the drive. Although this seems like an obvious difference, its removability has other effects.

- ▶ A removable hard drive should not be included in the default drive search sequence (see “[Drive Search Sequence](#)” on page 126).
- ▶ Although removable hard drives can be publicly attached, users of the system must be careful when removing or changing the disk because another user might be relying upon a certain volume being present in the drive and may in fact be using it.

When a set of disks is acquired, it is best to partition and format them prior to using them, just as you would do for a box of diskettes. When a multiple-volume backup is written to a disk, you must have sufficient pre-formatted disks available.

■ CD-ROM Drives

Both SCSI and ATAPI interface CD-ROM players are supported by THEOS. They can be used like a removable hard disk drive with the exception that they are read-only devices and cannot be formatted or written to.

- ▶ They are attached like any other drive with the [Attach](#) command or in the [Sysgen](#) configuration.
- ▶ Discs can be changed with the [Eject](#) and [Mount](#) commands and the [ChDir](#) command can use the drive to specify it as the current working directory.
- ▶ The directory of files on a compact disc is viewed with the [FileList](#) command.
- ▶ Files on the disc can be copied and viewed with the standard THEOS utility command such as [CopyFile](#), [Look](#), *etc.*
- ▶ Application programs can open and read the contents of files on the disc as long as they do not attempt to open it in update or read/write mode. Files on a compact disc are always stream files.

There are a few commands that cannot access CD-ROM drives:

- ▶ [Disk](#) (because a CD-ROM does not have a THEOS file system on it).
- ▶ [LineEdit](#) (because it always opens its files in read/write mode).
- ▶ Any other command whose basic purpose is to create a file or to write data to a file.

A CD-ROM drive can be used to play audio compact discs with the [CDPlayer](#) command. The sound for the disc is played through the headphone or speaker jack located on the front of most CD players.

■ Floppy Disk Drives

THEOS supports both 3½" floppy drives and 5¼" floppy drives in various formats.

Size	Capacity and format
3½"	2.88MB, 80 cylinders, 2 surfaces, 72 sectors/track
	1.44MB, 80 cylinders, 2 surfaces, 36 sectors/track
	1.2MB, 80 cylinders, 2 surfaces, 30 sectors/track
	720K, 80 cylinders, 2 surfaces, 18 sectors/track
5¼"	1.2MB, 80 cylinders, 2 surfaces, 30 sectors/track
	720K, 80 cylinders, 2 surfaces, 18 sectors/track
	360K, 40 cylinders, 2 surfaces, 18 sectors/track

Table 13: Floppy Disk Formats Supported

• Formatting Floppy Diskettes

Floppy diskettes must be formatted before they can be used by THEOS. Preformatted diskettes can be used if they are “quick formatted” before using. A floppy is formatted with the [Disk](#) command described on page 296 or the [Setup FLOPPY](#) command described on page 494.

A quick format of a floppy can only be done with the [Setup FLOPPY](#) command. Quick formatting does not format the disk surface. It only creates a THEOS file system on the diskette.

[Setup FLOPPY](#) is used to initialize a batch of diskettes. The [Disk](#) command is used to reformat, clear or label a single diskette.

■ RAM Disk Drives

A RAM disk is a region of memory in your system that has the appearance of a disk. That is, it has a file directory and can contain files. RAM disks, being in memory, offer very fast access and are frequently used as a work drive for temporary files. You should only use RAM disks for temporary file storage because, even if they are defined in the system configuration file, they are always cleared when the system boots.

Another limitation of RAM disks is size. You may only have one RAM disk attached and it can only be as large as available memory. When a RAM disk is used as the work drive and it becomes full, the program needing additional work disk space will probably fail.

RAM disks are useful for developers as a work drive during program compilations. The process of compiling a program creates many temporary files on the work disk. Using a RAM disk as the work drive allows the compilation process to be more efficient. A RAM disk used in this manner should be about 2MB in size. When multiple compilations might occur at the same time, the RAM disk should have a size of 2MB for each session or user that will perform program compilations.

Systems that are not used extensively for program compilations should allocate memory to the disk cache because it is more efficient for this purpose. See “[Disk Caching](#)” on page 124.

• Using RAM Disk Drives

A RAM disk is defined and used by attaching a disk letter to one of the RAM disk names defined in the device names file.

RAM Disk Name	Size	RAM Disk Name	Size
RAM64K	64KB	RAM8MB	8MB
RAM128K	128KB	RAM10MB	10MB
RAM192K	192KB	RAM12MB	12MB
RAM256K	256KB	RAM16MB	16MB
RAM384K	384KB	RAM32MB	32MB
RAM512K	512KB	RAM48MB	48MB
RAM640K	640KB	RAM64MB	64MB
RAM768K	768KB	RAM96MB	96MB
RAM1MB	1MB	RAM128MB	128MB
RAM1536K	1.5MB	RAM160MB	160MB
RAM2MB	2MB	RAM192MB	192MB
RAM3MB	3MB	RAM224MB	224MB
RAM4MB	4MB		

For instance, to use a 2MB RAM disk, you would attach a disk drive letter to RAM2MB:

```
>attach m ram2mb
```

Attaching a RAM disk creates and “formats” a new disk in memory of the appropriate size. After it is attached, you may change the directory size or its label with the [Disk](#) command. The default disk label is “Ram_Disk.”

Only one RAM disk may be attached at one time.

■ Image Disk Drives

An *image disk* is a “pseudo disk” that is actually a file on a hard disk. The content of the file is an image of a disk. This image disk can be used just like any other disk.

• Creating Image Disk Drives

Image disks are created by attaching a logical drive letter to the special device names of IMAGE1 through IMAGE8.

```
>attach i image1
```

```
Enter IMAGE file name (or ESC to exit): example.image  
File not found!
```

```
Enter <CR> for new file name  
or <size>M for size in megabytes  
or <cyls>,<heads>,<sectors> --- 5M
```

```
>
```

This example creates the file EXAMPLE.IMAGE and attaches it as an image disk. This disk can be formatted, cleared, written to, archived, *etc.*, just like any disk. One advantage of the image disk is that it exists on the hard drive as a file and can be copied or archived from that disk along with the other files on the disk.

Use the <cyls>,<heads>,<sectors> option for a new image disk if you want the drive used as an image of a floppy. For instance, to make an image of a 3½", 1.44MB floppy, attach it with the following specifications:

```
>attach k image4
```

```
Enter IMAGE file name (or ESC to exit): floppy.image  
File not found!
```

```
Enter <CR> for new file name  
or <size>M for size in megabytes  
or <cyls>,<heads>,<sectors> --- 80,2,36
```

This creates an image disk that is the same format and capacity as a floppy diskette. The [Backup](#) command can write the image to a floppy

Use the size specification when all you are concerned with is the capacity of the image disk.

There are eight file names that are special to the image drive system. These names are DISK.IMAGE n where n is the number of the image disk.

Using these special file names for your image disks can simplify the subsequent attachment of the drive. (See next section for an explanation.)

Image disks can also be created with the [Backup](#) command, BUFFER TO options.

• Using Image Disk Drives

Like any other disk, an image disk can only be used after it is attached. (Image disks used by the [Backup](#) command, BUFFER option is an exception. Refer to the [Backup](#) command for a discussion of buffered backups.) Also like other disks, image drives may be attached with the [Attach](#) command or with the [Sysgen](#) command.

```
>attach i image1
```

```
Enter IMAGE file name (or ESC to exit): example.image:s
```

If the file that you specify exists, the file is attached as an image disk. If it doesn't exist you are asked if you want to create it (see previous section).

Image disk files must be accessible from the account that you are currently logged onto.

You may also specify the file name directly with the [Attach](#) command. This is useful when the disk is attached with an EXEC or with an application program.

```
>attach i example.image:s
```

This syntax requires that the image disk file exist prior to this attachment. If the file does not exist you are asked if you want to create it or if you want to specify a different file name.

If you create your image disks using the default name of DISK.IMAGE n , you may use the first syntax:

```
>attach i image1
```

Before the [Attach](#) command asks for a file name it first checks to see if the default name of DISK.IMAGE1 exists. If it does exist it uses that file as the image disk. Only when this default file name cannot be found does [Attach](#) ask you to specify a file name for the image disk.

When specifying an image disk in the system configuration ([Sysgen](#) command) the image disk file must exist prior to booting the system.

■ Disk Caching

The hard disk cache available with THEOS is a multithreaded disk cache that greatly increases the performance of disk access. With the disk cache enabled (see “[Cache](#)” on page 200), every physical read of a hard disk sector is saved in the disk cache memory. Subsequent requests for that same sector of the disk are read from the cache memory. The disk is not accessed when the requested sector is already in the cache memory.

Writes to the disk are also processed through the cache system. A request to write a sector to the disk is first written to the cache memory. With the “write delay” feature enabled, the sector is not written to disk for a period of time. Instead, sectors to be written are saved in memory and are periodically written to disk in the most efficient sequence.

To use disk caching to its fullest, allocate as much memory as possible to the disk cache, enable write delay and enable the directory write back features of the cache system. The recommended minimum size is 2MB but 16MB or more is ideal. Do not use RAM disks. Allocating the memory to cache instead of a RAM disk provides most of the speed of a RAM disk without losing any files when the system is booted. The memory that would have been allocated to a RAM disk is better used by the disk cache for all disk access rather than just the limited usage of the RAM disk.

The performance gains when using disk caching with write delay enabled are considerable. However, there is a slight risk to data integrity. If a power failure occurs, or if the system is reset between the time that a program requests a disk write operation and when the disk caching software actually performs the write to the physical disk, the data is lost and cannot be written.

If this risk is unacceptable, you can still take advantage of disk caching with write delay enabled by utilizing an on-line uninterruptable power supply (UPS) for your computer system. Otherwise, enable the disk cache with write delay and directory write back disabled.

When enabled, the disk cache is used for all hard disk and image disk accesses. Floppy disk, CD-ROM and RAM disk accesses are not processed through the cache.

Some utilities bypass disk caching because their operation is will be more efficient if the data is not cached. Specifically, [Archive](#), [Restore](#) and [TBackup](#) bypass the disk cache mechanism because they write large amounts of consecutive sectors.

■ Using Disk Drives

Like any device, a disk drive cannot be used unless it is attached. Disks may be attached during system startup by defining them in the system configuration file with the [Sysgen](#) command described on page 531. A disk may also be attached with the [Attach](#) command described on page 178.

A disk is attached with a disk drive letter assigned to it. Drive letters may be any letter from A to Z. However, disk drive letter S is reserved for the attachment of the current system disk.

```
>attach d disk2
```

When attaching a disk, it can be made a publicly accessible disk or a private disk. Disks attached with the [Sysgen](#) command are always public. Disks attached with the [Attach](#) command are private disks by default. You may use the PUBLIC option to attach a public disk drive.

```
>attach d disk2 (public
```

A **private disk** is a disk that can only be used by the current user. A **public disk** may be used by all users logged onto the system. (A disk attached as public will not be accessible to a user if they are already logged on. When they log off and then log on again, they will have access to all public devices attached, including any recently attached public disks.)

Some commands operate on entire disks. For instance, the [Backup](#) command can make a copy of the entire contents of one disk onto another. With this type of command, you specify the disk drive letter of an attached disk:

```
>backup f g
```

This command tells [Backup](#) to make a copy of the disk F by copying its contents to the disk attached as G.

However, most commands operate on files or sets of files on a disk. With these commands you specify the file name. A full file name specification includes the disk drive letter that the file resides on. For instance:

```
>list my.file:g
```

This command tells the [List](#) command to locate the file named MY.FILE on the disk attached as disk G and to display the file's contents on the screen. For more information about file name specifications, refer to “[Files](#)” on page 135.

- **Drive Search Sequence**

On large systems with several disks attached, it may be inconvenient to remember which disk a file is on. THEOS provides a mechanism to make file searching easier. Files can be located without specifying the disk drive letter. THEOS searches all of the attached disks in the defined **disk search sequence**. This sequence is defined for each account environment (see [Account](#) command on page 156 and “[SEARCH](#)” on page 101).

With the search sequence, you can specify which disks to search and in which order. For instance, if disks F, G, I, J, M and S are attached, you can specify that you want THEOS to search the disks S, I and J only, and in that order.



Drives using removable media such as floppy, removable hard disks and CD-ROM discs should not be specified in the search sequence. If they are in the search sequence but there is no disk in the drive, an error message displays every time that THEOS attempts to find a command or file and it hasn't found the file on the drives specified earlier in the search sequence.

■ Tape Drives

THEOS supports several manufacturers and models of tape drives using IDE, SCSI and ATAPI interfaces. The current list of drives supported can be found on the Internet by following the link on the page:

<http://www.theos-software.com/support/>

In a THEOS system, tapes are ideal for creating backups of the system and for off-line copies of important databases.

There are several utilities included with the operating system that facilitate this archival or backup operation:

Utility	Function
Archive, Restore	Copies files, directories, entire accounts from disk to tape or restores the copies from tape to disk. Individual files may be restored from the archived copy.
Backup	Copies the contents of an entire disk onto tape or restores the entire copy of a backed-up disk.
TBackup	Similar to Archive/Restore but with better interface and better error detection and correction.

■ Using Tape Drives

Like any device, a tape drive cannot be used unless it is attached. Tapes may be attached during system startup by defining them in the system configuration file with the [Sysgen](#) command described on page 531. A tape may also be attached with the [Attach](#) command described on page 178.

There are four logical tape names that may be used to attach a tape drive. These names are “TAP1,” “TAP2,” “TAP3” and “TAP4.” The physical name used in a tape attachment is defined the file SYSTEM.TEOS32.DEV NAMES.

```
>attach tap1 tape1
```

When attaching a tape, it can be made a publicly accessible tape or a private tape. Tapes attached with the [Sysgen](#) command are always public. Tapes attached with the [Attach](#) command are private by default. You may use the PUBLIC option to attach a public tape drive.

```
>attach tap2 tape4 (public
```

A **private tape** is a tape that can only be used by the user that attached it. A **public tape** may be used by all users logged onto the system. (A tape drive attached as public will not be accessible to a user if they are already logged on. When they log off and then log on again, they will have access to all public devices attached, including any recently attached public disks and tapes.)

Even when a tape drive is attached, a tape cartridge cannot be used in the drive for archival or backup purposes until it has been initialized. This is accomplished with the **Tape** command.

```
>tape tap2 init

Enter tape label: Tuesday1

>
```

This **Tape** command initializes the tape volume in the tape drive that is currently attached as TAP2. It does this by writing the volume label and headers at the beginning of each track on the tape. Because archives and backups to tape frequently require more than one tape volume to hold the copy of the disk, you should format all blank tapes when acquired.

The commands **Archive**, **Backup** and **TBackup** are all designed to make a backup or archive copy of a disk on a tape. For instance:

```
>archive s tape1 (volume

>backup s tape1

>tbackup tape1 (backup volume
```

The **Archive** and **TBackup** commands can also copy less than all of a disk volume to tape.

```
>archive *.command:s tape1 (account

>tbackup *.data:s *.data:a *.command:s tape1 (backup eject
```

The **Eject** command can also be used on tape drives that support this capability.

You may also use a tape for general data storage. For instance, you may write a data file to a tape that is later read back, one record at a time. When used this way you must remember that a tape is not a “file-structured device” and is sequentially accessed, not randomly accessed like disks are. To use a tape in this manner, the **Tape** command is used to rewind the tape before the file on it is opened for writing or for reading.

8 Directories and Files

One of the primary responsibilities of an operating system is to manage programs and data files on disk. THEOS supports disk directories, subdirectories and libraries of files. Data files may be either streams of data, direct-access records, indexed-access records or keyed-access records.

■ Directories

THEOS supports three types of file directories: the main or root directory of a disk, subdirectories and libraries.

■ The Main Directory of a Disk

Every THEOS disk volume has a main or **root directory**. A directory is analogous with a table of contents for a book. It contains a list of the files on the disk and information about their location and attributes.

The main directory of a THEOS disk is organized for fast and efficient access by employing the same algorithm used by keyed-access files, described on page 141. Although it is accessed with this file-access method, the main directory is not a file, cannot be accessed by programs as a file, and does not appear as an entry in a directory listing of a disk.

Unlike other files, subdirectories and libraries, the main directory of a disk is not owned by any account. All accounts and processes have access to the main directory of a disk. (They may not have access to the specific files in the main directory. See “[File Ownership](#)” on page 142.)

The size of the root directory of a disk is defined when the disk is formatted. This size cannot be changed without clearing the entire contents of the disk.

The root directory of a disk may contain subdirectories, libraries or flat files.

■ Subdirectories

Because disks frequently hold thousands of files, it can be difficult to keep track of them. Subdirectories provide a means of organizing and grouping files into more manageable sets. A **subdirectory** is a special type of file whose records are file directory entries. To provide more flexibility, the size of a subdirectory is not limited when it is created and it can grow as more files are added to the subdirectory.

Because a subdirectory is a file, it is owned by a specific account. All of the files contained in a subdirectory are owned by the same account that owns the subdirectory.

Unlike the root directory of a disk, a subdirectory is accessed as a sequential list of file names. That is, to find the 100th entry in a subdirectory, the preceding 99 entries must be read and examined. Although computers and hard disk drives are very fast, to maintain efficient access to the files in a subdirectory, its size should be limited to about 400 entries. (This is a subjective limit. You may find that, for your purposes and on your computer system, a larger or smaller size is acceptable.)

The file directory entries in a subdirectory may be other subdirectories, libraries or flat files.

Subdirectories have file names with both a file-name and a file-type component. The file-type component is frequently blank. For instance, the following are all valid subdirectory names:

```
/DIALDIR
/SAMPLES
/SAMPLES/B3220TK
/IDEAS
/IDEAS/PROGRAM
/IDEAS/DOC
/NOTES
/NOTES/SEMINAR.1994
/NOTES/SEMINAR.1995
```

Although only two levels are shown in the above examples, subdirectories may be nested to almost any level.

When specifying the path to a file in a subdirectory, the subdirectory name is terminated with a slash character (/). Thus, to specify the file BASIC.VERSION2 in the subdirectory SEMINAR.1995 in the subdirectory NOTES, you would specify:

```
/notes/seminar.1995/basic.version2
```

■ Libraries

Similar to a subdirectory, a **library** is a special type of file whose records are file directory entries. Unlike the root directory or a subdirectory, the file directory entries in a library may only be **member files**.

Libraries are used to group files that are similar in nature. For instance, all of the operating system commands are grouped together into the library SYSTEM.COMD32, all of the system help text files are grouped together into the library SYSTEM.HELP32, *etc.*

Because a library is a file, it is owned by a specific account. All of the member files contained in a library are owned by the same account that owns the library.

Similar to the root directory, the entries in a library are accessed with the same algorithm used by keyed-access files. The size of a library is determined when the library is created. Unlike the root directory, a library's size can be changed with the [Create](#) command (see page [269](#)).

Member files in a library are data files or program files. A member file cannot be another library or a subdirectory. A member file name contains only the member-name component.

■ Flat Files

A **flat file** is a file that is not a subdirectory, library or member file of a library.

Flat files have file names with both a file-name and a file-type component. The file-type component may be blank. For instance, the following are all valid flat file names:

```
SAMPLE.EXEC  
INTEREST.COMMAND  
SYSTEM.ACCOUNT  
GENERAL.LEDGER  
/SAMPLES/BJ.BASIC  
/SAMPLES/MAKEFILE.  
/PAYROLL/EMPLOYEE.MASTER
```

■ Using Subdirectories and Libraries

To access a file in a subdirectory you specify the path to the file. A **path** is the specification of the branch in the subdirectory tree to a specific file. A **tree** is the organization of a subdirectory hierarchy.

For instance, an account owns two subdirectories: IDEAS and NOTES. In the IDEAS subdirectory there are two subordinate subdirectories: DOC and PROGRAMS. In the PROGRAM subdirectory is a file named MODEM.PROGRAM. To assess this file you would specify the complete path and the file name:

```
/ideas/programs/modem.program
```

To access a member file of a library, you append the member-name to the library name. For instance, to list the device names file (member-name is DEVNAMES) in the library SYSTEM.TEOS32, you specify:

```
system.teos32.devnames
```

If the library is in a subdirectory that is not your current working directory (see below), you must specify the path along with the library and member-name:

```
/ideas/doc/user.manual.index
```

• Current Working Directory

To make it easier to use subdirectories and library members, THEOS uses the concept of the **current working directory** and the **default library**. The current working directory is the directory or subdirectory that you have specified as being the default starting point for file specifications. Unless otherwise specified, the current working directory is the root directory.

You can specify the current working directory by defining the environment variable **SUBDIR** in your account definition, by setting the **SUBDIR** environment variable with the **Set**, or by using the **ChDir**.

For instance, in the previous example showing access to the MODEM.PROGRAM file in the subdirectory /IDEAS/DOC, you could change the current working directory to /IDEAS/DOC.

```
>chdir /ideas/doc
```

```
>list modem.program
```

When a current working directory is defined, you may use path specifications that are either relative to the current working directory or to the root directory. To indicate that a path specification starts with the root directory of a disk, use a leading slash character. For instance:

```
/ideas/doc/modem.program
```

To indicate that the path is relative to the current working directory, either use no leading punctuation (go down the hierarchy) or use the double period specification (go up the hierarchy). The double period specification is normally pronounced “dot, dot.”

For instance, if the current working directory is /IDEAS/PROGRAMS, you could reference a file in a subordinate directory with:

```
new/network.programs
```

This is equivalent to a full path specification of:

```
/ideas/programs/new/network.programs
```

Alternately, to access a file in another branch of the current directory, you would reference the “parent” of the current working directory:

```
../doc/network.programs
```

This means to go up to the parent of the current working directory and then down to the DOC subdirectory.

You may use more than one double period specification, meaning to go to the parent of the parent, *etc.* For instance, ../../SAMPLE.FILE. Alternately, you may merely add one more period to the double period specification for each additional level of the tree that you want to go up.

```
../../../../sample.file  
...../sample.file
```

The above two specifications are equivalent. Note that when using the second specification it is easy to make errors when typing it.

- **Default Library**

To make it easier to use the member files of a library, THEOS allows you to use a default library. Similar to the current working directory, the default library is an environment variable ([LIBRARY](#)) and can be initialized in your account definition with the [Account](#) or changed with the [Set](#).

```
>set library sample.library
```

When a default library is defined, any typeless file name specification refers to a member of the default library. For instance:

```
example  
sample.library.example
```

The above two specifications are equivalent when the default library is defined as `SAMPLE.LIBRARY`.

Default libraries and current working directories may be used in combinations.

```
>chdir /games/programs  
  
>set library program.source  
  
>list solitary
```

The above [List](#) command lists the file `/GAMES/PROGRAMS/PROGRAM.SOURCE.SOLITARE`.

■ Files

■ File Names

THEOS flat files use a three-part name composed of a **file-name**, **file-type** and a tape or **disk-letter**. The file-name and file-type are one to eight character names consisting of letters, digits and underscore characters. The dollar sign character may be used but it is always translated into an underscore character. Although file names are recorded in uppercase, they may be specified with uppercase or lowercase characters.

file-name.file-type:disk

Colon separates file-type from disk

Period separates file-name from file-type

Some typical examples of file names are:

```
system.history:s
system.menu32:s
read.me:s
stdio.h:s
```

Files may use names without a file-type component. These are referred to as **typeless** file names. Typically, typeless file names are used for subdirectory names. However, they may be used for any file name. A typeless file name is specified with a period following the file-name component. For instance:

```
makefile.:s
```

■ Libraries and Library Members

As described earlier, a library is a special type of directory that contains member files. The name of a library may not be typeless: it must have both a file-name and a file-type component.

Files that are members of libraries use a file name that includes the library file name (file-name and file-type) and the **member-name** of the file. Member-names are composed of the same characters as file-names and file-types.

library-file-name.library-file-type.member-name:disk

Period separates file-type from member-name

```
system.b3220lib.b_close:s
system.cmd32.attach:s
system.help32._command
```

■ File Path Names

account\directory/file-name.file-type:disk

```
>logon develop
>cd /program/source
>list my.program
```

- **Wild Card Specifications**

```
>list first.file second.file third.file
```

Most of the time, when you want a command to operate on several files, the files have some similarity and that is why you want to operate on them as a set. In this situation, it is much easier to specify the file name using

wild cards. For instance, the previous `List` command could have been specified as:

```
>list *.file
```

Here, the character `*` is a wild card character that means “any.” Therefore, the command lists all files that have a file-type of “FILE.” This command is not exactly like the first example because it might find many files other than `FIRST`, `SECOND` and `THIRD` that have a file type of “FILE.”

There are four wild card characters allowed by commands that operate on multiple files:

Wild Card	Meaning
<code>*</code>	All characters match.
<code>?</code>	Any single character matches.
<code>@</code>	Any alphabetic character (A–Z and \$ or underscore) matches.
<code>#</code>	Any numeric digit character (0–9) matches.

Table 14: File Name Wild Cards

The `*` wild card always means “any zero or more characters match.” When used as the first character of a file name component, it means that all files match that component. When used at the end of a file name component, it means that any file that matches the first part of the component is included.

For instance:

```
>list *.data
```

This specification means that any file with a file-type of “data” will be included in the listing.

```
>list tsc.dat*
```

This second specification lists any file with a file-name of `TSC` and a file-type that starts with the letters `D`, `A` and `T`.

The other three wild cards specify a match on a single-character position.

For instance:

```
>list my.file#
```

This command lists only the files MY.FILE1, MY.FILE2, MY.FILE3, *etc.* The fifth character of the file-type must be a digit and there cannot be a sixth character in this component. Thus, MY.FILES and MY.FILE10 will not match this specification.

You may specify multiple wild cards in a single component. For instance:

```
>filelist sys*.*.b3#*
```

This command displays a directory listing of all files whose file-name starts with SYS, and whose member-name starts with a B followed by a 3 and another digit, and ends with any characters. This particular command specification displays all of the run-time modules for BASIC386 and MultiUser BASIC Version 2.0.

Wild Cards in Destination File Specifications

Wild cards can also be used with commands that have a “from” and “to” file specification. For instance, the [CopyFile](#) and [Rename](#) commands.

```
>copyfile *.basic =.bsource
```

This syntax instructs [CopyFile](#) to copy all files whose file-type is BASIC to new files with the same file-name but with a new file-type of BSOURCE.


For clarity purposes only, the equal sign (=) is used in the destination file specification. In actuality, any of the wild-card characters may be used in the destination file specification but they are all interpreted the same way: Each wild-card character in the destination specification is replaced with the first set of wild-card specifications in the same component of the source specification. If there is no wild cards specified in the source specification for that component, the destination uses the source file name’s component.

For instance:

```
>copyfile example.basic =.source
               ↑
```

In this command, the file is copied to EXAMPLE.SOURCE because the “=” is replaced with the source specification of “example.”

```
>rename *.data =.datafile
```




Here, all files with a file-type of DATA are renamed with the same file-name but a file-type of DATAFILE because the “=” is replaced with the matching source specification of “*.”

```
>copyfile customer.data.* cust1.=.
```




In this example, all of the members of the library CUSTOMER.DATA are copied to the library CUST1.DATA and the member-names are not changed. The first “=” in the file-type position means that DATA is to be used for the destination file-type and the second “=” in the member-name position means that the original member-name is to be used because it matches the source specification of “*” in that position.

```
>rename /logs/@@981201.log /logs/=98dec.log
```



Here, all files in the directory LOGS, with a file-type of LOG and with a file-name of eight characters with the last six characters being 981201 are renamed to have a file-name starting with the original first two characters of the current name (the “@@” in the source specification) but ending with 98DEC. The file-type is changed to LOG which could have been specified with the “=” wild card because it is the same as the current file-type.


```
>rename /logs/@##1201.log /logs/=dec.log
```



Similar to the preceding example, this command renames all of the files starting with two letters followed by two digits followed by “1201” to have file-names with these same first four characters but ending with DEC. For instance, /LOGS/BL981201.LOG is renamed to /LOGS/BL98DEC.LOG, /LOGS/BL991201.LOG is renamed to /LOGS/BL99DEC.LOG, *etc.* The example illustrates that the destination wild card matches a source wild-card specification even when it contains a mixture of wild-card characters.

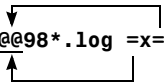
As stated earlier, any wild-card character (@, #, ?, * and =) in the destination specification is treated the same as if it were the “=” character used in these examples.

```
>rename /logs/@##1201.log /logs/@dec.log
```



Although it is unlikely that you would do so, every wild-card character in the destination specification is replaced with the matching set of wild-card characters in the source specifications. For instance:

```
>copyfile @@98*.log =x=.log
```



Here, a file named BL981201.LOG would be copied to BLXBL.LOG. Each of the “=” specifications in the destination name are replaced with the characters matching the first wild-card specification of “@@” in the source specification. The “*” in the source specification is not used in the destination names because it is not the first wild-card in that component of the source specification.

- **File Search Sequence**

When the complete path to a file name is given, and the file name does not use any wild cards, the file in that location is used. However, if the path is omitted or incomplete, or if the file name uses wild cards, the system must find a file that matches the specifications.

Account name omitted. Only directories and files owned by the account that you are currently logged onto and directories and files owned by the system (public) account are searched. For instance:

```
/my.file:s
```

This specification will locate either *account*\MY.FILE:S or SYSTEM\MY.FILE:S. The current account is always searched first.

Account name and directory path omitted. Only files owned by the current account in the current working directory and files owned by the system account’s root directory are searched.

```
my.file:s
```

This specification locates my.file:s in the current working directory of the current account. The current working directory always has a disk-letter associated with it. If the disk-letter of the file name does not match the disk-letter of the current working directory, then the root directory of the specified drive is used.

Account name specified, directory path omitted. In this situation, the directory path is assumed to be the root directory for the account.

```
private\my.file:s
```

The backslant character that terminates the account name specification also tells the system to start at the root directory.

Disk-letter omitted. When the disk-letter is omitted from a file name specification, all disks in the disk search sequence are searched, in the specified order. The disk search sequence is defined with the [SEARCH](#) environment field set in the account environment or with the [Set](#) command.

```
>show search
SEARCH      = SAB

>filelist my.file (nosort
my.file:s
my.file:a
my.file:b
```

File name specified with a single component and no periods. The meaning of this depends upon the current definition of the default library ([LIBRARY](#) environment field) and the [FILETYPE](#) environment variable. When [LIBRARY](#) is defined, the file name is assumed to be a member-name of that library. If [LIBRARY](#) is not defined but [FILETYPE](#) is, the file name is assumed to be a file-name and the value of [FILETYPE](#) is appended to it to form a file-name, file-type specification. If neither [LIBRARY](#) or [FILETYPE](#) has a value, then the file name is assumed to be the file-name of a typeless file.

■ File Organization Types

THEOS supports five types of file organization. These files may be flat files in the root directory or a subdirectory, or they may be members of a library.

Stream or Sequential. The records in these files can only be accessed in sequence. The file is updated by either adding new records to the end of the file or rewriting the entire file, record by record. This type of file is often used for text files and program source files.

Direct or Random Access. The records in a direct file are accessed by their record number and may be read or written in any sequence. The records in the file may be read sequentially with the “read next” mechanism. All records in a direct file are of the same length defined at the time the file is created.

Keyed Access. The records in a keyed file are accessed by their alphanumeric keys and may be read or written in any sequence. Although the records in a keyed file may be read sequentially, the order of a sequential

read is in physical record order, not sorted by the key. The length of the key and the length of the record are defined at the time the file is created.

Indexed Access. Similar to keyed files, the records in an indexed file are accessed by their alphanumeric keys and may be read or written in any sequence. The records in an indexed file may be read sequentially in the order of their keys. The length of the key and the length of the record are defined at the time the file is created.

Program Files. A program file is a compiled, executable program.

■ File Attributes

The directory entry for every file on a disk has many attribute fields that provide information about the file or define access restrictions to the file. The content of a directory entry is briefly described in Appendix H: “[File Systems](#),” starting on page [751](#).

■ File Ownership

Every subdirectory, library and flat file is owned by an account. The ownership of a file is set when the file is created. By default, all files created are owned by the account that you are logged onto when you create the file but you may create a file that is owned by another account. Refer to the [Create](#) command described on page [269](#). An existing file’s ownership may be changed with the [Change](#) command described on page [216](#).

When you are logged onto an account, you have direct access to the files owned by that account, and you may do anything you want to the files that their access protections allow (see page [143](#)).

As described on page [95](#), there are public accounts and private accounts. Files owned by the system or public account may be accessed by users logged onto any private account. Files owned by a private account other than the one that you are logged onto may only be accessed by using the owning account’s name in the complete path specification of the file.

For instance, if you are logged onto a private account, you may access a public account file with normal file specifications:

```
>list system.theos32.devnames
```

However, to access a file owned by another private account, you must specify the owning account’s name:

```
>list rosemary\private.file
```

The backslant character (\) is used in a complete path specification to terminate the owning account name of a file. When a path specification includes the account name, the path starts with the root directory and you do not specify any leading slash character. For instance, to access a file in another account's subdirectory, you would use a specification like:

```
>list rosemary\program\notes\private.file
```

The current working directory and the default library are used only when you are accessing a file owned by your account.

When attempting to access another account's file, either public or private, the file's shared access protections apply.

■ File Access Protection

Every file has protection codes associated with it that restrict or allow certain key types of access to the file. There are four types of access that can be protected: read, write, erase and execute. These attributes are set when the file is created (see [CREATE](#) environment variable on page 102) or when the file is altered with the [Change](#) command. A file's protection may only be changed by a user logged onto the file's owning account.

Read Protection (R). Read protection means that you may not read any of the existing information in the file. Before setting this attribute, be aware that there are no THEOS commands that can reset this attribute.

Write Protection (W). Write protection means that you cannot add to or change any information in the file. Write protection is separate from read protection. It is possible to be able to write records to a file that you cannot read.

Erase Protection (E). The erase protection attribute prevents a file from being erased, even if you are the owner of the file.

Execute Protection (X). A file with execute protection set cannot be executed by any user.

Hidden Files (H). A file may have the hidden attribute set. This attribute does not provide any protection to the file. It prevents the file from appearing in a directory listing produced with the [FileList](#) when the `HIDDEN` option is not specified.

Modified (M). The modified attribute does not provide any protection to the file. It is set every time that a file is created or written to. The modified attribute is used by the [Archive](#), [CopyFile](#) and [TBackup](#) commands to restrict

the files copied to just those that have changed. It is displayed by the [FileList](#) command to provide a visual indicator of which files have been changed recently.

The modified attribute is cleared by the [Archive](#), [Change](#), [Compress](#), [CopyFile](#) and [TBackup](#) commands.

Owner Access

Files that you own are always accessed using the erase and execute protection attributes and the owner-access read and the owner-access write protection attributes. The owner of a file may change the file's attributes.

Shared Access

Files that you do not own are always accessed using the erase and execute protection attributes and the shared-access read and the shared-access write protection attributes. A file is also erase protected from a shared-access user if it is either erase protected or has shared read or shared write protection enabled. When you do not own a file you cannot change the access attributes of the file.

■ Growth Factor

Every data file has a growth factor associated with it. When a file becomes full and a program tries to write more data to it, the system will automatically increase the size of the file (if there is sufficient disk space available). The ***growth factor*** for a file controls how much the file grows in this situation.

The default growth factor for a file is .3, meaning that the file will grow by 30% of its current size. The growth factor for a file is set when the file is created. The [Change](#) can change the growth factor of an existing file.

Growth factors are values that are either less than one or integer multiples of one. For instance, a growth factor of .5 means that the file grows by 50% of its current size. A growth factor of 2 means that the file grows by 200% of its current size.

9 Windows

THEOS provides a windowing system that can be used by any application, whether it is written using the EXEC language, MultiUser BASIC, Version 2.0, or the THEOS C language, Version 5.0. Windows can be used for menus, list boxes, message display areas, special input areas, forms input, *etc.*

Before using the windowing commands described on page 590, or the windowing statements and functions described in the BASIC or the C language reference manuals, a general understanding of the capabilities and limitations of THEOS windows is needed.

■ What is a Window?

A *window* is a rectangular area of the console display that can be used as an input area, a display area, or for both input and display. The main attribute of a window is that the contents of one window do not affect the contents of another window, even if they are displayed at the same location on the screen. When the “topmost” window is removed from the display, the underlying window is still there and its contents are the same.

■ Window Attributes

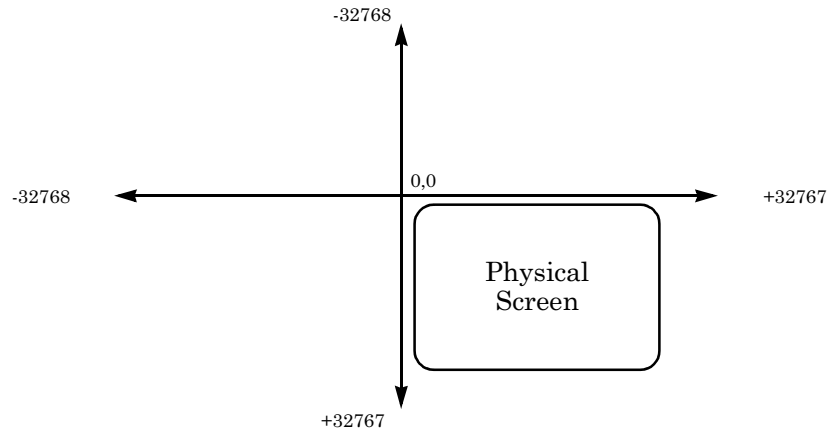
There are many attributes that define the appearance of a window.

- ▶ Position and size
- ▶ Interior color
- ▶ Frame style and color
- ▶ Shadow style
- ▶ Title text, style and color
- ▶ Display order
- ▶ Display status

Window Position and the Virtual Screen

The windows defined and used with THEOS exist in a virtual screen. The *virtual screen* is an imaginary screen that contains the physical screen seen on your console.

The dimensions of this virtual screen are -32,768 to +32,767 columns and rows.

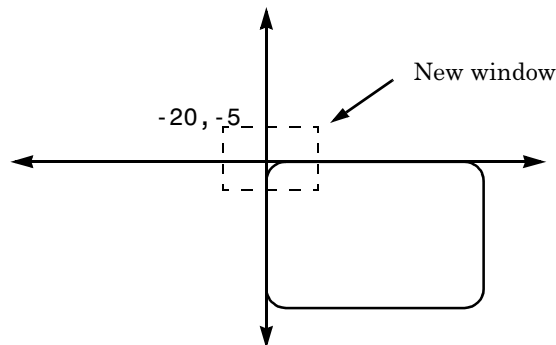


A window may be defined anywhere on this virtual screen. If a portion of the window lies within the boundaries of the physical screen, then that portion is displayed on your console. The size of the physical screen is defined by the attached page width and page depth. Frequently, this is 80 columns by 24 rows. The upper left corner of the physical screen is column zero, row zero.

For example, a window created with the command:

```
>wopen 1 -20 -5 40 10
```

has a location in the virtual screen of:



Only the portion of the window from its column 20 through 39 and its line 5 through 9 are visible. If the window has a frame or shadow, only the portions of the frame and shadow that lie within the boundaries of the physical screen are displayed.

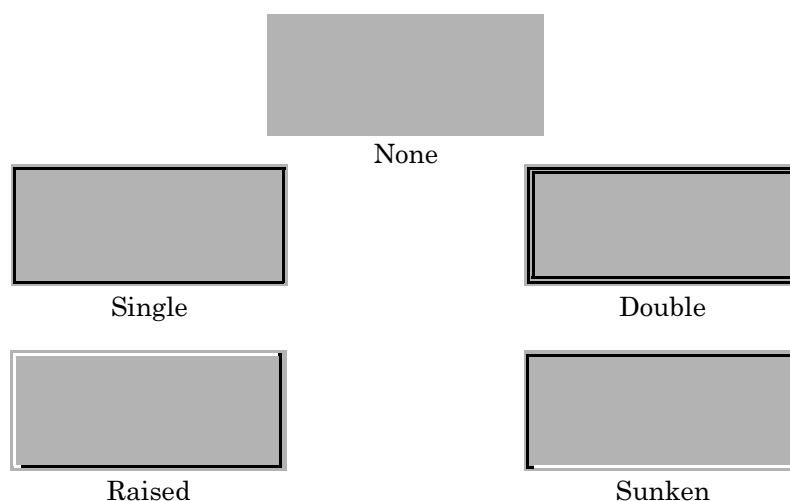
Although this window is positioned partially outside of the physical screen, it may be selected just as any window can be selected. Text may be displayed in the window and the operator may type characters into the window, even though the characters may not be visible due to the window's position on the virtual screen.

The size of a window may be as small as a single character or as large as 255 characters by 255 lines.

Frame Styles

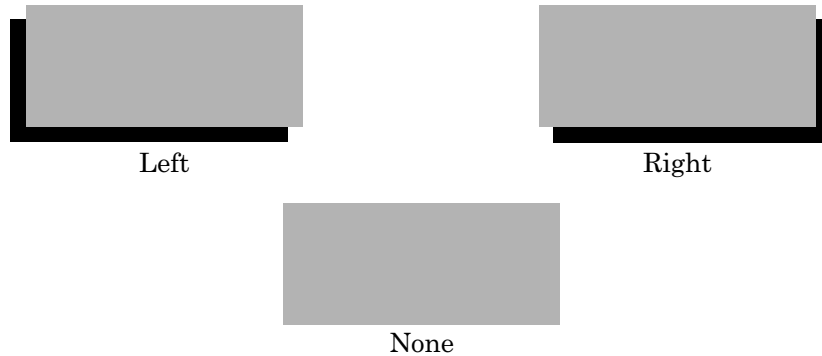
A window may have a “frame” around it. A frame requires one row above and below the window and one column to the left and right of the window.

A window's frame may be one of five styles:



Shadow Styles A window may have a “shadow.” A *shadow* requires one row below the window and two columns to the left or right of the window.

A window’s shadow may be one of three styles:



For consistency, only drop shadows on the right should be used.

Window Colors and Invert Status

On a color-capable console, the default colors for a window, its frame and its title are determined by a combination of the window number and the colors defined as the defaults used on that console. The default colors for a console are defined in the `SETUP` command using the `COLORS` setup function. Refer to “[Setup COLOR](#)” on page 484.

The default colors for a window are the colors defined for your console’s session number which are determined by taking the current session number plus the window-number, modulo 8. For instance, if you are using session number three and opening window number two, the default colors are the colors defined for session number five. If you are using session number five and opening window number 12, the default colors are the colors defined for session number one ($5+12$ modulo 8).

Because the default colors for a window will be different when a program executes on different sessions, it is always best to explicitly define the colors used in a window.

On monochrome displays, the color specifications for a window are ignored. Instead, the invert status of a window can be used to differentiate one window from another. The invert status determines whether text and lines are displayed in normal video (white on black) or reverse video (black on white).

The default invert status for a window depends upon its window number. Even-numbered windows default to invert OFF; odd-numbered windows default to invert ON.

Wyse 55 and other attribute-takes-space terminals can not use invert status.

Frame & Title Attributes

The frame and title attributes of a window can be specified with either distinct color names and attribute names or with a single attribute specification that defines the foreground color, the background color and the foreground video attributes.

```
>wframe 1 single right white red
```

```
>wframe 1 single right 0x79
```

```
>wframe 1 single right 121
```

The above three commands perform the exact same operation: They change window one's frame style to single-line with a drop shadow on the right and the color of the frame is defined as white on red. The first command defines the frame colors with separate color names. The last two commands define the frame colors with a single, frame-attribute code.

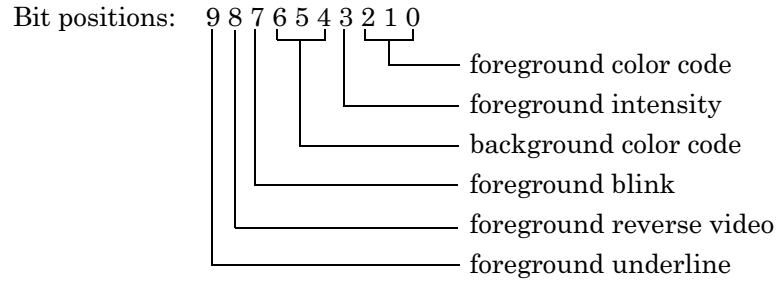
The colors that may be specified in the WFRAME, WMENU and WTITLE command may be specified with the color name. Color names must be completely spelled out.

Code	Name	Code	Name
0	Black	4	Red
1	Blue	5	Magenta
2	Green	6	Yellow
3	Cyan	7	White

The video attribute names include: NORMAL, UNDERLINE, BLINK, DIM and REVERSE. Attribute names cannot be abbreviated.

Note: On color displays, the underline and reverse attributes are displayed with different color combinations, not the actual attribute. For instance, specifying that the title text is underlined does not underline the title text. Instead, the text is displayed with the color whose code is one less than the normal text color.

The attribute code is a single bit-mapped value:



For instance, to specify that you want a title displayed with underlined, bright-white letters on a blue background, you would use one of the following commands:

```
wtitle 1 " Title Text " top center white blue blink
```

```
wtitle 1 " Title Text " top center 0xaf
```

```
wtitle 1 " Title Text " top center 175
```

Window 0

Window zero is different than other windows in several ways. It is a window that:

- ▶ Is always open and cannot be closed.
- ▶ Has no frame, title or shadow.
- ▶ Covers the entire screen, except the status line, if any.
- ▶ Has text clipping on and cannot be changed.
- ▶ When selected, covers all other windows including those marked as TOP.
- ▶ Cannot be selected with TOP or HIDDEN attributes.
- ▶ Cannot be moved.

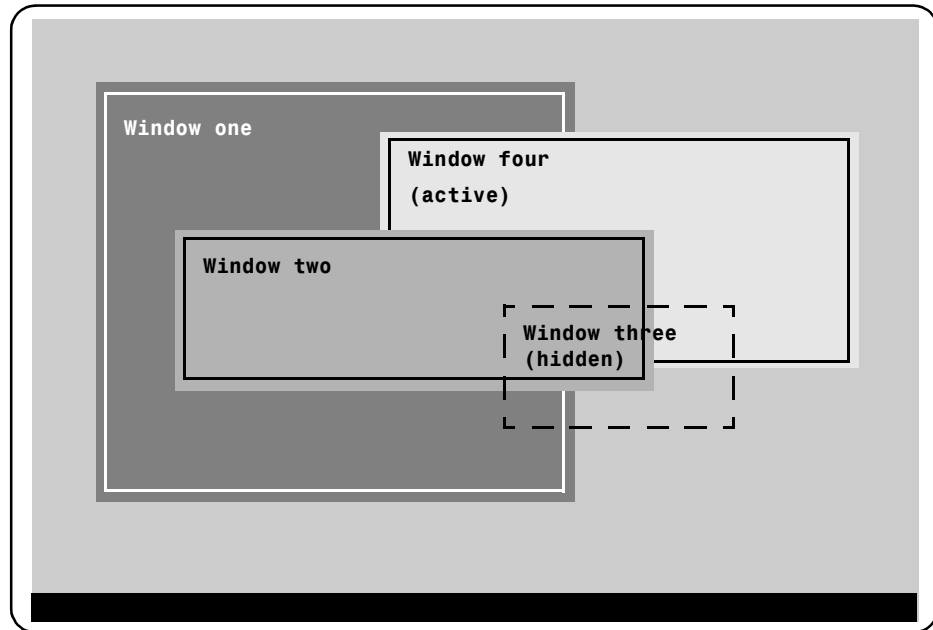
**Window
Display Order**

THEOS maintains a list of all of the open windows in their display sequence. This display order determines which portions of a window are visible when it is refreshed or selected with update ON status.

If windows 1–4 are open and selected in the following sequence:

```
>wselect 1  
  
>wselect 2 top  
  
>wselect 3 hidden  
  
>wselect 4 on
```

The screen appears as:



Window three is not truly in the window display order because it is HIDDEN.

Even though window four is the active window and it covers part of the area used by window two, all of window two appears because it was last selected with the TOP attribute. To make window four fully visible, it will either have to be selected with the TOP attribute and window two's TOP attribute will have to be cleared (select it without TOP and then reselect window four), or window two will have to be removed.

**Windowing
Commands**

There are many commands that can be used by an operator or an EXEC language program for creating and manipulating windows. Refer to “Window Management Commands” described on page [590](#). MultiUser BASIC, Version 2.0 and THEOS C, Version 5.0, have their own statements or functions that create and maintain windows. Refer to their manuals for specifics about those statements and functions.

Part II ---

Command Reference

10 Commands

This section of the manual describes each of the commands provided with the THEOS 32 operating system. They are arranged in alphabetical order by their command names.

Commands are executed by entering a command line when the CSI prompts you. Refer to Chapter 2 “[Entering Commands](#),” starting on page 33, for a description of this process and the features of the operating system available when entering commands.

Commands may also be executed by using a command line in an EXEC language program. Refer to Chapter 11 “[EXEC Job Control Language](#),” starting on page 613, for a description of the EXEC language and its capabilities.

Refer to “[Documentation Conventions](#)” on page 20 for information about typographic conventions used here and throughout this manual.

Account Command

The Account command maintains or displays user accounts.

1	<u>ACCOUNT</u>	<i>drive</i>	
2	<u>ACCOUNT</u>	<i>drive</i> (<i>option</i>	
3	<u>ACCOUNT</u>	<i>drive</i> (ADD <i>name</i> <i>add-options</i>	
<hr/>			
	<i>drive</i>	»	optional disk drive code
	<i>name</i>	»	account name
	<i>options</i>	»	DELETE <i>name</i> IDENT PRT <i>nn</i> SORT TYPE
	<i>add-options</i>	»	COPY <i>name</i> PASSWORD <i>password</i> PRIVLEV <i>nn</i> PROMPT <i>prompt-text</i> SUBDIR <i>directory</i> SYNONYM <i>name</i>

Operation For all modes the *drive* parameter is optional. It tells the Account command which drive to find the SYSTEM.TEOS32.ACCOUNT file. By default, Account uses the S drive. Specify another drive when you want to maintain the accounts on another disk, such as an emergency boot diskette.

Mode 1—Invokes the interactive mode of the Account command. In this mode you are presented with a menu that allows you to add new accounts, change existing accounts, delete accounts and list the existing accounts. This mode must be used when adding or changing an account’s environment variable definitions. For a complete description of this interactive mode, refer to “[Interactive Account Maintenance](#)” on page 160.

When invoking the interactive mode, you may specify the COPY option. Refer to the COPY option described on page 158.

Mode 2—This is a command-line mode that allows you to delete an existing account definition or to list all of the existing accounts on the console or printer. For a complete description of this mode, refer to “[Deleting Accounts](#)” on page 165 and “[Listing Accounts](#)” on page 166.

Mode 3—This is a command-line mode that allows you to add a new account definition. Only the basic components of an account can be defined with this mode by using one or more of the *add-options* described on page 158. To specify or change all of the environment switches or environment variables, you must use the interactive mode of the command.

Options

ADD *acct-name* Adds a new account definition. Specify the new account name immediately following the ADD keyword. Follow the new account name with zero or more of the *add-options* described on page 158.

```
>account ( add payroll syn accnts password aardvark
```

COPY *acct-name* Defines the account used as a “template” for adding new accounts. This option may be used in combination with the [ADD acct-name](#) option or with the interactive mode of the Account command.

When the COPY option is not used the SYSTEM account definition is used as the template for adding new accounts.

DELETE Deletes an existing account definition. Specify the account name immediately following the DELETE keyword.

```
>account ( delete accnts
```

Caution: If an account owns any files do not delete it unless it has a synonym account name defined. If you delete the account you might “orphan” the files owned by that account. See “[Deleting Accounts](#)” on page 165.

IDENT Use with the [PRTnn](#) or [TYPE](#) option to request that the account listing be in account number sequence. This is the default sequence for account listings. For a description of the account listing display, refer to “[Listing Accounts](#)” on page 166.

```
>account ( type ident
```

PRT*nn* Lists all of the existing accounts on the printer. *nn* specifies which attached printer to use and is in the range 1–16. The options [IDENT](#) or [SORT](#) may be used in combination with this option. When neither is used, the accounts are listed in account number sequence.

The alternate keywords [PRINT](#) and [PRINTER](#) may be used instead of PRT.

SORT Used with the [PRTnn](#) or [TYPE](#) option to request that the account listing is in account name sequence. For a description of the account listing display, refer to “[Listing Accounts](#)” on page 166.

```
>account ( printer4 sort
```

TYPE Lists all of the existing accounts on the console. The options [IDENT](#) or [SORT](#) may be used in combination with this option. When neither is used, the accounts are listed in account number sequence.

```
>account ( type sort
```

Add Options **COPY** *acct-name* Specifies that the new account definition will default to the switches and environment variables defined for the account *acct-name* specified. When this option is not used, the switches and environment variables defined for the SYSTEM account are used.

This option may also be specified in [Mode 1](#). This causes all new accounts added in the interactive mode to default to the switches and environment variables defined for *acct-name*.

PASSWORD Specifies the password for the account. The password is specified immediately following the PASSWORD keyword. When this option is not used the new account will have no password.

```
>acc ( add jeff password giraffe
```

For a description of the password field, its limitations and its usage, refer to “[Account Name and Number](#)” on page 95.

PRIVLEV Specifies the privilege level for the new account. Privilege levels are in the range 0–9. When this option is not used a privilege level of 1 is assigned to the account.

```
>acc ( add jeff privlev 3
```

For a description of the privlev field, its limitations and its usage, refer to “[Privilege Level](#)” on page 97.

PROMPT Specifies the CSI command prompt for the new account. When this option is not used the currently defined prompt is used for the new account.

To specify a prompt string that contains lowercase characters or special punctuation characters, surround the prompt string with a pair of double quotation marks. To use the prompt string variables described in Table 11 on page 108, precede each backslant character with an extra backslant character.

```
>account ( add rachel prompt "\\a\\g"
```

For more information about prompt strings, refer to the prompt environment variables described on page 107.

SUBDIR

Specifies the default or home subdirectory for the account. When this option is not used the account uses the root directory as its home subdirectory.

The directory specified here does not have to exist at this time. However, it should exist when the account is next used by Logon. If the directory does not exist when you log onto this account the home directory is set to the root.

```
>acc ( add susan password lion subdir private
```

For a description of the SUBDIR environment variable and its usage, refer to “SUBDIR” on page 109.

SYNONYM

Specifies the synonym account name for the new account. The synonym account name is specified immediately following the SYNONYM keyword. The synonym account must be an existing account name. The new account is assigned the same account number as the synonym account.

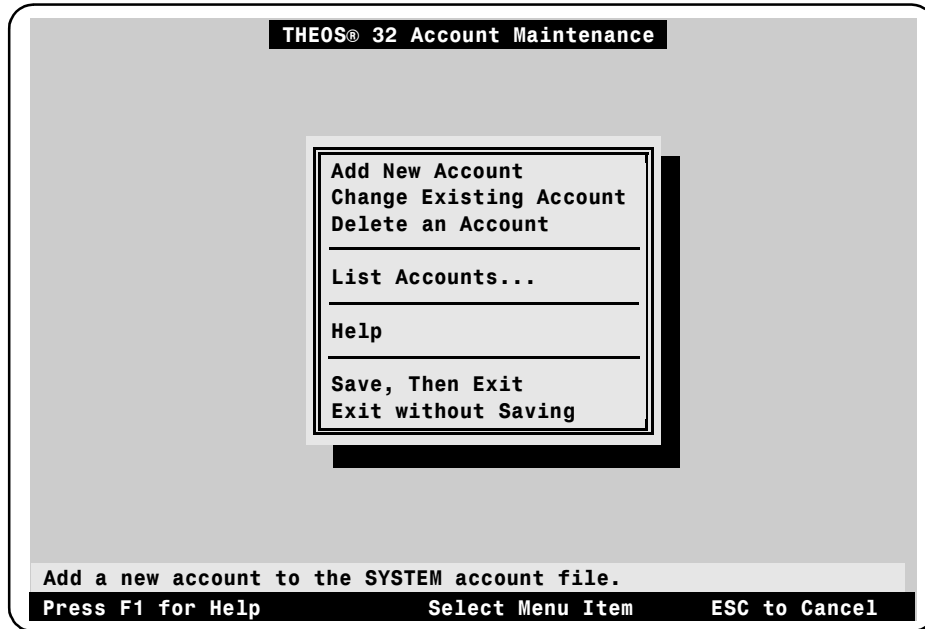
When this option is not used the new account is assigned the next account number available.

```
>acc ( add master synonym system privlev 5
```

For a description of the synonym field and its usage, refer to “Account Name and Number” on page 95.

**Interactive
Account
Maintenance**

When [Mode 1](#) is used the interactive mode of the Account command is invoked. In this mode the Account command uses a windowed menu design that offers choices for adding new accounts, changing existing accounts, deleting existing accounts, *etc.*



Use the normal menu selection keys to select the desired function. These keys are described in "[Using Menus](#)" on page [75](#).

■ Add New Account

Selecting this menu item displays the “Add New Account” screen.

The screenshot shows a terminal window titled "THEOS® 32 Account Maintenance". Inside, there's a form titled "Add New Account". The form has several sections: "Account name....." and "Password....." at the top; "Synonym to....." below; a "Switches" section with options like "User Number.....", "Privilege.....", "Decimal is comma.....", "Language code.....", "Abbreviation.....", "Standard synonym.....", "Search sequence.....", "Ready message.....", "CSI case mode.....", "Receive messages.....", "Disable BRK-Q.....", "System BRK code.....", and "Work drive....."; and an "Environment Variables" section at the bottom. At the very bottom of the terminal window, there's a prompt "Enter the account name to add." and two instructions: "Press F1 for Help" and "Esc to Cancel".

Commands

After entering the account name that you want to add, specify the synonym account name. If the new account name is not a synonym account, merely press **Enter** for the “Synonym to” field. As soon as the “Synonym to” field is filled in (or omitted), all of the “Switches” and “Environment Variables” are initialized to the current definitions of the SYSTEM account or the **COPY acct-name** (except “Privilege.”)

Note: The interactive mode may be invoked with the COPY option in effect. When this is done, then all of the “Switches” and “Environment Variables” are initialized to the current definitions of the account name specified after the COPY keyword.

Change the individual fields to the values desired. You can advance from field to field with the **↑** and **↓** keys, and you can jump from section to section with the **Tab** key.

Press **F10** when you are satisfied with the account definition and you want to return to the menu. Press **Esc** to cancel the entry of the new account.

The **F1** key can be used at any field to display information about the field.

- **Password**

After specifying the “Account name” and the “Synonym to” fields, the “Password” field is the only field that can be entered or changed in the top section.

If you enter a password and then jump back to the top section (with the **Tab** key), the password is displayed as eight asterisk characters for security purposes. This is also true in the “Change Account” screen.

- **Switches**

This section contains all of the account environment pre-defined fields. For a description of the meaning and function of these fields, refer to “[User Account Environments](#)” on page 95.

You can move from field to field with the **↑** and **↓** keys, and you can jump to the “Environment Variables” section with the **Tab** key.

- **Environment Variables**

This section is a free format area where environment variables are defined. Merely enter the environment variable name, immediately followed by an equal sign and the value of the variable. For instance:

```
PROMPT=!14\A!15\ S !4!!\!5>
DIALDIR=SYSTEM\DIALDIR:S
SUBDIR=DIRECTRY
PATH=USER
```

You can move from line to line with the **↑** and **↓** keys and you can jump to the top section with the **Tab** key. To clear or delete an environment variable definition, use **F5** or the key defined as “Erase to End of Line.”

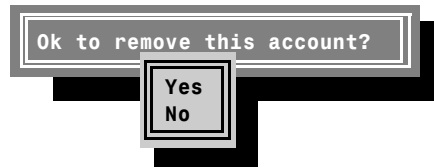
- **Change Existing Account**

This menu item operates identically to the “Add New Account” item described above, except the “Account name” field refers to the name of an existing account, not a new account name.

- **Delete an Account**

This menu item allows you to delete an existing account definition. It uses the same screen as the “Add New Account” to display the values for the existing account definition.

Enter the name of the account to delete. The existing definition is displayed and you are asked:

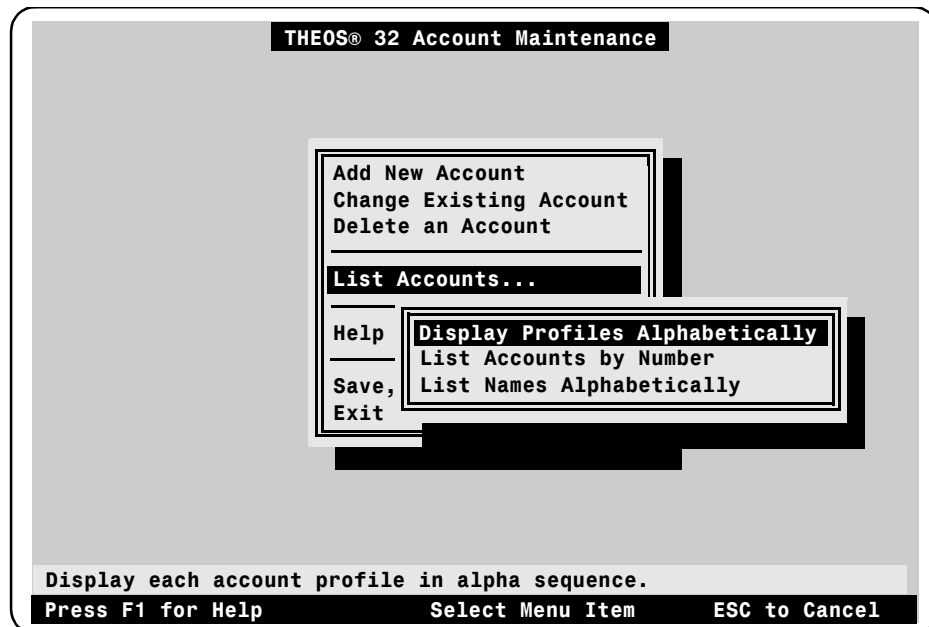


DO NOT DELETE an account name that does not have synonym accounts defined (use the “List Accounts...” menu item to find synonym accounts) if the account owns any files on any of the disks used on this system. This includes removable disks. File ownership is controlled by the account number, not the account name.

If you are at all unsure about deleting the account, respond with “No.” Once an account is deleted it is sometimes difficult to add it back with the same account number. See also “[Deleting Accounts](#)” on page 165.

■ List Accounts...

This menu item has a sub-menu associated with it that allows you to specify the type of account listing you want.



- **Display Profiles Alphabetically**

Using the same screen as the add/change/delete functions, each account definition is displayed in alphabetical sequence. The **PageUp** key is used to back up to the prior account definition. The **PageDown** or almost any other key is used to advance to the next account definition.

The **Esc** key terminates the list and returns to the main menu.

- **List Accounts by Number**

Displays all of the accounts sorted by the account or user number field. This display highlights synonym accounts because they are grouped with the main account name.

THEOS® 32 Account Maintenance				
Account name	User Number	Synonym to	Privilege	Password
SYSTEM	0		5	Yes
MANAGER	0	SYSTEM	5	Yes
COMPANY	1		3	Yes
CRAIG	1	COMPANY	3	Yes
JULIE	1	COMPANY	3	Yes
PHILLIP	1	COMPANY	3	Yes
RACHEL	1	COMPANY	3	Yes
DEVELOP	2		4	Yes
LETTERS	3		2	No
DOS	4		5	No

Press F1 for Help Esc to Cancel

The **PageUp** key is used to back up to the prior display page. The **PageDown** or almost any other key advances to the next page.

The **Esc** key terminates the list and returns to the main menu.

- **List Names Alphabetically**

Similar to the “List Accounts by Number” display, all of the existing account definitions are displayed on the screen in account name sequence.

■ Help

Displays basic information about the Account command. Specific help text is available for any menu item or entry field by pressing the **[F1]** key while the cursor is positioned on the item or field.

■ Save, Then Exit

All account additions, changes and deletions performed during this session are saved and the Account command exits.

■ Exit without Saving

Selecting this menu item causes all account additions, changes and deletions to be discarded.

Deleting Accounts

Account definitions can be deleted with the command-line option **DELETE** or by using the interactive mode of the Account command and selecting the “Delete an Account” menu item.

Before deleting an account name first check to see if there are any synonym accounts defined for it (list the accounts sorted by account number). If the account does not have a synonym account name defined, check to see if there are any files owned by the account. Remember to check any removable disks. **DO NOT DELETE** an account name that does not have any synonyms and owns files.

When deleting an account name that owns files, use the **Change** command to change the ownership of the files to another account that will not be deleted, or create a synonym account name. Access to files owned by a deleted account can be difficult.

**Listing
Accounts**

All of the currently defined accounts can be listed on the screen with either the interactive mode of the Account command (see “[List Accounts...](#)” on page 163), or to the screen or printer with the command-line options **TYPE** and **PRTnn**.

When using the command-line options to list the existing accounts, you can request that the list be sorted either by account name or by account number. The most useful, and therefore the default sequence for displaying accounts, is in account number sequence. In this sequence all of an account’s synonyms are grouped together:

Account	User	Synonym	Prv	Password
SYSTEM	0		5	Yes
MANAGER	0	SYSTEM	5	Yes
COMPANY	1		3	Yes
CRAIG	1	COMPANY	3	Yes
JULIE	1	COMPANY	3	Yes
PHILLIP	1	COMPANY	3	Yes
RACHEL	1	COMPANY	3	Yes
DEVELOP	2		4	Yes
LETTERS	3		2	No
DOS	4		5	No

In account lists for an account that has synonym account names, the primary account name is the account name that was defined first. For instance, in the above display the accounts SYSTEM and MANAGER are both account number zero and are therefore synonyms of each other. However, because the SYSTEM account was defined first and then, at a later time, the MANAGER account was defined as a synonym to SYSTEM, only the MANAGER account is identified as a synonym account.

Alternately, you can use the [SORT](#) option to display the accounts in account name sequence:

Account	User	Synonym	Prv	Password
COMPANY	1		3	Yes
CRAIG	1	COMPANY	3	Yes
DEVELOP	2		4	Yes
DOS	4		5	No
JULIE	1	COMPANY	3	Yes
LETTERS	3		2	No
MANAGER	0	SYSTEM	5	Yes
PHILLIP	1	COMPANY	3	Yes
RACHEL	1	COMPANY	3	Yes
SYSTEM	0		5	Yes

Notes

Refer to Appendix D: “[System Files](#)” for a description of the file “[ACCOUNT](#)” on page [726](#).

The Account command can also be invoked via the [Setup](#) command.

Cautions

Do not delete an account name if that account has any files and the account does not have a synonym account name already defined.

Restrictions

The Account command may be used by only one user at a time.

The Account command may only be executed when you are logged onto the SYSTEM account (user number 0).

The Account command requires a privilege level of five.

Do not edit or change the SYSTEM.TEOS32.ACCOUNT file with any program other than this Account command.

See also

[Logon](#), [Password](#), [Set](#), [Show](#), [Start](#), [Sysgen](#)

Archive Command

The Archive command makes an “archive copy” of a file, a set of files, an entire disk volume or a set of disk volumes.

- 1 ARCHIVE *from-drive to-drive* (*options*
- 2 ARCHIVE *file to-drive* (*options*
- 3 ARCHIVE *file from-drive to-drive* (FILES *options*
- 4 ARCHIVE *from-drive₁ from-drive₂ ... to-drive* (*options*

<i>file</i>	»	file name with optional path; may contain wild cards			
<i>from-drive</i>	»	drive letter of source drive to archive			
<i>to-drive</i>	»	drive letter of destination drive or logical tape name			
<i>options</i>	»	ACCOUNT	FILES	NOASK	SIZE <i>file</i>
		ASK	INCREMENTAL	NOQUERY	SUBDIR
		<i>date1</i>	LABEL <i>name</i>	NOTYPE	TYPE
		<i>date2</i>	MULTIUSER	QUERY	VOLUME
		DIFFERENTIAL	NAME <i>file</i>	SHARE	

Operation The file or files specified with *file* or *from-drive* are copied in a compressed form to an **archive volume** on the destination *to-drive*. The archive volume created can only be interpreted with the complementary [Restore](#) command described on page [458](#).

Mode 1—Unless one or more options restrict the file selection, all of the files on the *from-drive* are archived to the *to-drive*. With this mode [VOLUME](#) is a default option.

For instance, the following command archives all files in all accounts in all subdirectories on the s drive to the drive attached as TAPE1.

```
>ARCHIVE S TAPE
```


Mode 2—The file(s) specified by *file* are archived to the *to-drive*. With this mode **ACCOUNT** is a default option.

For instance, the following command archives a single file to the floppy diskette in drive F.

```
>ARCHIVE CUSTOMER.MASTER F
```

The following command archives all of the “master” files in the current account to the tape drive.

```
>ARCHIVE *.MASTER TAPE
```

Mode 3—The files listed in *file* are archived to the *to-drive*. *file* must be an ASCII stream file containing one file description per line. The **SELECTED.FILES** and **SELECTED.EXEC** files created by the **FileList** command and the **FOUND.EXEC** created by the **Look** command can be used for this specification file (see “**The EXEC and FILES Options**” on page 338). You may also create the specification file with an editor or application program.

For instance, **FileList** is used to create a list of files to be archived:

```
>FILELIST *.DATA:A (EXEC
```

```
>FILELIST A NOT *.DATA:A (10/1/97 EXEC APPEND
```

A file now exists that lists all of the “data” files and all files that have been changed since 10/01/1997. The following command will archive these files to tape:

```
>ARCHIVE SELECTED.EXEC A TAPE (FILE
```

Mode 4—Similar to **Mode 1** except that multiple volumes are archived onto *to-drive*. With this mode, **VOLUME** is a default option.

For instance, the following command archives all files in all accounts in all subdirectories on the S drive, the A drive and the B drive to the F drive.

```
>ARCHIVE S A B F
```

Options

The following options are available with the Archive command. They can be used with any of the modes described above, either singly, or in combinations.

ACCOUNT Specifies that only the files owned by the current account are candidates for archiving. This is a default option when [Mode 2](#) is used.

Use the [VOLUME](#) option to override this default.

ASK A default option that instructs Archive to ask the operator to mount the source and destination volumes, and waits for confirmation that the proper volumes are mounted.

For instance:

```
>archive s f

Source is Disk S
Destination is Disk F
Mount volumes now:

Source is labeled "SYSTEM".
Destination is labeled "ARCHIVE1".
OK to start archive (Y/N)
```

Respond with a for the “Mount volumes now:” request and a or for the “OK to start archive” question.

Use the [NOASK](#) option to override this default.

date1 The first token that looks like a date is interpreted as a selection date. Only files with a date stamp greater than or equal to this date (new files) are candidates for archiving. For instance:

```
>archive s f (10/1/97
```

With this command only those files on the s drive that have been changed on or since 10/01/1997 will be archived.

See also the [INCREMENTAL](#) and [DIFFERENTIAL](#) options.

date2 A second token that looks like a date is interpreted as a selection date. Only files with a date stamp less than or equal to this date (old files) are candidates for archiving.

```
>archive s f (1/1/86 9/30/97
```

This command archives only those files on the s drive that have not been changed since 9/30/1997. The date 1/1/86 is the earliest date maintained by the THEOS file system and is interpreted as “from the earliest date.”

DIFFERENTIAL Tells Archive to include only those files that have their modified bit set. This is the only option that prevents Archive from resetting the modified bit for files that it archives. See “[Differential Archives](#)” on page 175.

FILES Indicates that [Mode 3](#) of the Archive command is to be used. The *file* specifies an ASCII stream file with each record in the file specifying a single file name. The file name specifications in this file may include the account name and path to the file.

For instance, a line in the specification file might contain:

```
develop\custom/programs/program.source.sample:s
```

If this file is used in an archive, the display will be:

```
>archive selected.exec s f (file noask
```

```
Account: 4=DEVELOP
Subdirectory: CUSTOM
Subdirectory: PROGRAMS
Library: PROGRAM.SOURCE
Member: PROGRAM.SOURCE.SAMPLE
```

INCREMENTAL Tells Archive to include only those files that have their modified bit set. The modified bit is reset for each file archived. See “[Incremental Archives](#)” on page 176.

LABEL *label* Specifies that the *to-drive*’s volume label is set to *label*. For instance:

```
>archive s f (label "Monday1" notype
```

sets the label of the diskette in drive F to “Monday1.”

If additional disks are required, the last character of the *label* is incremented. When this might happen, try to use a starting label name that ends with a sequence identifier, such as “1” or “-A” *etc.*

MULTIUSER Allows Archive to archive a public drive even though other users may be logged on and active. Normally, when Archive is instructed to perform a full volume archive (option **VOLUME**) on a public disk, it requires single-user mode. If other users are logged onto the system, it displays the message: “Must be single user or private volume.”

Using this option tells Archive to not restrict the archive to single-user operation (the message is still displayed). **THIS CAN BE EXTREMELY DANGEROUS!** If another user changes some files while the archive is being created, the integrity of the archive is lost.

Use this option only if you are sure that all other users are inactive.

NAME Specifies that the archive volume file name is to be set to the token following this option. Additionally, the archive volume file will only use as much space as needed. None of the existing files on the *to-drive* are erased (an implied **SHARE** option).

For instance:

```
>archive s f ( account name "programs.archive"
```

This creates an archive volume file on drive F with a file name of PROGRAMS.ARCHIVE.

If the archive cannot fit in the space remaining on the destination drive, it will report that the disk is full and not create the archive volume.

Use of this option sets the **NOASK** option. To reenable this option, you must specify **ASK** after the **NAME** option and its file name.

One of the principal advantages of this option is that the resulting archive file uses only as much disk space as is needed for the actual archive volume. When this option is not used the archive volume name is ARCHIVE.VOLUME01, the archive volume uses the entire disk space of the *to-drive* and it may use multiple volumes for the output file.

NOASK Disables the source and destination volume operator confirmation at the beginning of the archive and when subsequent disks or tapes are needed.

- NOQUERY** A default option that tells Archive to not ask for confirmation on each file being archived.
- NOTYPE** Tells Archive to not display account names, subdirectory names, library names or file names on the standard output device as they are being archived.
- QUERY** Tells Archive that the operator is to be “queried” or asked if each file matching the selection criteria is to be included in the archive volume.

```
>archive s f (query
```

```
Source is Disk S
Destination is Disk F
Mount volumes now:
```

```
Source is labeled "THEOS".
Destination is labeled "ARCHIVE".
OK to start archive (Y/N)? Y
```

```
Ok to archive "SAMPLE.FILE:S" (Yes/No/All) N
Ok to archive "SELECTED.EXEC:S" (Yes/No/All) N
Ok to archive "SYSTEM.CMD32.ACCOUNT" (Yes/No/All) Y
Ok to archive "SYSTEM.CMD32.ARCHIVE" (Yes/No/All) A
94.3 MB, 8,834 files selected.
```

```
...
```

When the “Ok to archive” question is asked you may respond with a **[Y]** for yes, **[N]** for no or **[A]** for all. Responding with **[A]** means yes to this file and all remaining files are included without being queried. Respond with **[Esc]** to cancel to archive.

- SHARE** Tells Archive that the archive volume is to share the space on the first disk with any existing files. This is a default option when *to-drive* is a removable hard disk.

When this option is not used, Archive clears the directory of the first disk and creates an archive volume file that uses the entire disk space.

If multiple disks are required they will use the entire disk space for the archive file.

- SIZE file** The total size of all of the files selected for archiving is computed and appended to *file*. This is done before the archive is created.

- SUBDIR** Tells Archive that only files in the current working directory and all of its subordinate directories are included.
- TYPE** A default option that tells Archive to display each account name, subdirectory name, library name and file name on the standard output device (normally the console) as it is being archived. This display can be redirected.

For instance, the following is a typical display during an archive:

```
ACCOUNT: 2=SAMPLES
      File: CHARSET.H
      File: LIBS.EXEC
      File: MAKE.FILE
      File: READ.ME
      File: SAMPLES.EXEC
      Subdirectory: C32
            Library: C32.CMD32
                  Member: C32.CMD32.FINS
                  Member: C32.CMD32.PRTF
      ...
```

The indentation of the display indicates the hierarchy of the directory structure.

To disable this option use the **NOTYPE** option.

- VOLUME** Specifies that all accounts, all subdirectories and all files on the *from-drive* are to be archived. This option requires that the *from-drive* be a private disk volume, that all other users be logged off or that the **MULTIUSER** option be used.

Full Volume Archives

Full volume archives (option **VOLUME** in effect) are copies of the entire contents of disks. They should be created on a frequent and periodic basis to assure yourself of having adequate protection in the case of disk or computer failure. Since a full volume archive contains a copy of every file, subdirectory and account on a disk, it is the only archive volume that needs to be accessed if you must restore a system.

Because full volume archives do include all of the files on a disk, it includes files that rarely change, such as the operating system programs and data files, application programs and other static data files.

Multivolume Archives

Mode 4 of this command archives multiple disk volumes onto a single archive volume. This allows you to archive your entire system in one operation. For instance, a system with four hard drives can be archived with the command:

```
>archive s a b c tape
```

Should the need ever arise where you want to restore a volume or a file from this archive volume, use the **DRIVE** option of the **Restore** command.

```
>restore tape b (drive b
```

Differential Archives

A differential archive is an archive volume that includes only those files that have changed since the last full volume archive. For instance, if a disk contains three files:

```
PROGRAM.COMMAND
DATA1.FILE
DATA2.FILE
```

A full volume archive will create an archive volume that contains all of these files. If DATA1.FILE is changed and a differential archive is performed, it will create an archive volume that contains only that file. The other files are not included in this differential archive volume because they have not changed and there is a current copy of those files in the last full volume archive.

If DATA2.FILE is then changed and another differential archive is performed, it will create an archive volume that contains both the DATA1.FILE and the DATA2.FILE because both of those files have been changed since the last full volume archive was performed.

Full Volume	Differential 1	Differential 2
PROGRAM.COMMAND		
DATA1.FILE	DATA1.FILE	DATA1.FILE
DATA2.FILE		DATA2.FILE

Differential Archives

Should a disk failure occur, the system could be restored by first restoring the full volume archive and then the last differential archive volume.

Incremental Archives

An incremental archive is an archive volume that includes only those files that have changed since the last full volume or incremental archive was created. For instance, in the example used for the differential archive, a full volume archive is performed and then DATA1.FILE is changed. When the incremental archive is performed, it creates an archive volume that contains only the DATA1.FILE. When DATA2.FILE is changed and another incremental archive is performed, it creates an archive volume that contains only the DATA2.FILE. The DATA1.FILE is not included in this archive volume because it has not changed since the last incremental archive was created.

Full Volume	Incremental 1	Incremental 2
PROGRAM.COMMAND		
DATA1.FILE	DATA1.FILE	
DATA2.FILE		DATA2.FILE

Incremental Archives

Should a disk failure occur, the system could be restored by first restoring the full volume archive followed by the first incremental archive volume and then the second incremental archive volume.

Notes

The output of the **Archive** command is an *archive volume*. This is a special stream file that contains the compressed forms of the archived files along with their directory entries. This archive volume can be manipulated like any other file, and it can even be opened and read like any other file. However, since it is a compressed form of the original files and it contains the directory entries for the files, it is not usable except by programs that know how to interpret this information. The [Restore](#) command is a program that knows how to interpret the information and copy it back to its original, usable form.

The files in an archive volume are sorted in ascending file name sequence within account number sequence.

An archive volume is a single file named ARCHIVE.VOLUME01 (unless the [NAME](#) option is used). This single file contains all of the files included in this archive.

Frequently a file or set of files being archived from hard disk to floppy, removable hard disk or tape will be larger than the disk or tape. In that situation the **Archive** command will ask you to mount another disk or tape so that it can continue the archive of the file or files. In this situation each of the disks or tapes used will contain a single file named ARCHIVE.VOLUME nn (see [SHARE](#) option for an exception to this situation).

Unless the [DIFFERENTIAL](#) option is used, the modified bit for each file archived is reset by the archive process.

Defaults [ACCOUNT](#) ([Mode 2](#)), [ASK](#), [NOQUERY](#), [TYPE](#), [VOLUME](#) ([Mode 1](#) and [Mode 4](#)).

Cautions The [MULTIUSER](#) option tells the Archive command to not check whether or not other users are logged on or active. It does not prevent those other users from performing operations that change the database being archived. If another user does change the database during the archive operation, the integrity of the archive volume is compromised.

For instance, a disk volume being archived includes a customer master file with current balance fields and an invoice database. While the archive volume is being created, another user posts a transaction to the invoice database before it is included in the archive but after the customer master file is archived. If the archive volume is ever restored, the invoice database will not match the current balance fields in the customer master file.

Restrictions Unless the [NAME](#) option is used, the destination drive must be an image drive or some type of a removable mass-storage device such as a floppy diskette, removable hard disk or tape.

The destination media must be preformatted. If the archive requires more disks or tapes than anticipated, you can use another session or terminal to format the disk while Archive waits for you to confirm that the proper disk is mounted. To do this the [ASK](#) option must be in effect and the *to-drive* must be publicly attached.

When option [VOLUME](#) is in effect and the *from-drive* is publicly attached, all other users must be logged off or the [MULTIUSER](#) option must be specified. See “Caution” notes above regarding the [MULTIUSER](#) option.

The Archive command requires a privilege level of four.

See also [Backup](#), [CopyFile](#), [Disk](#), [Restore](#), [Tape](#), [TBackup](#)

Attach Command

This command associates logical device names such as PRINTER with physical device drivers and specifies the attachment options for a device.

Commands

1 AT TACH *logical physical* (*options*)

2 AT TACH *logical* (*options*)

3 AT TACH *logical*

4 AT TACH

<i>logical</i>	»	logical device name				
		CON		console		
		PRT <i>nn</i>		printer		
		COM <i>nn</i>		comm		
		TAP <i>E</i> <i>n</i>		tape		
		A–Z		disk		
<i>physical</i>	»	device drive name as defined in SYSTEM.THEOS32.DEV NAMES				
<i>options</i>	»	ALF	DTR	NOHOLD	PZ	XPC
		B <i>nnnnnn</i>	E <i>n</i>	NP	PUBLIC	
		BIDIR	ETX	O <i>nnn</i>	QUEUE= <i>a</i>	
		C <i>nnn</i>	FF <i>nnn</i>	P <i>nnn</i>	STP <i>nnn</i>	
		COPIES= <i>nn</i>	HOLD	PE	V <i>nnn</i>	
		COPY= <i>nn</i>	L <i>nnn</i>	PN	W <i>n</i>	
		CTS	LF <i>nnn</i>	PO	XON	

System commands and application programs perform input and output to devices by specifying a logical device name. A **logical device name** is merely a standard name that any program can use to communicate with a device such as a disk drive, tape drive, printer, console or general communications device. The operating system takes this request from the program and passes it on to the physical device driver that is currently associated with the particular logical device name.

For instance, a printer might be connected to the first serial port on the computer. When a program wants to print something on that printer, it outputs data to the logical device name PRT. The operating system passes the data to the SIO1 physical device driver which, in turn, passes it on to the printer connected to the first serial port.

The Attach command is the primary method of changing the association of logical device names with physical device drivers. Other methods include Start and Sysgen.

Operation

Mode 1—Attaches the logical device name *logical* to the physical device driver *physical*, with options. The following examples show the attachment of the second hard disk drive to logical drive letter A, the attachment of the COM1 logical device to the third serial port, with options and the attachment of logical TAPE2 to a 250MB streaming tape drive.

```
>attach a disk2

>attach com sio3 (b19200 w8 e2

>attach tape2 tape1
```

Mode 2—Changes the operating parameters or options of the logical device name *logical* without changing its assignment to its physical device driver.

The following examples show the reattachment of COM1 with a change in the specified baud rate and the reattachment of the console with a change to the line length.

```
>attach com (b38400

>attach con (l132
```

Mode 3—Detaches or disassociates the logical device driver *logical* from its current physical device driver. The following examples show the detachment of the printer4 device and disk drive G.

```
>attach prt4

>attach g
```

You cannot detach the console or disk drive s.

The physical device driver is unloaded from memory if no other logical device is associated with it by this user or any other user on the system.

Mode 4—Displays the currently attached devices for this user.

This display is always output to the standard output device and thus can be redirected to a file or another device:

```
>attach > attach.output:f
```

When the standard output device is the console, the title is displayed in reverse video and line graphics are used to box the columns as illustrated on the next page. The line graphics characters can be suppressed if the

`LINEGRAPH` environment variable is set to N (see “[Environment Variables](#)” on page 102).

When the standard output device is not the console, the title, column headings and line graphics are not output.

THEOS® 32 Device Attachment			
Logical Name	Physical Name	Device Number	Options
F	FLOPPY1	1:1:0	"????????",STP3,PUBLIC
G	floppy2	2:1:1	"????????",STP3,PUBLIC
M	RAM2MB	64:64:129	"Ram_Disk",2MB,PUBLIC
S	DISK1	3:2:0	"THEOS ',255MB,PUBLIC
CON	SVGA:1	5:3:0	L80,P24,PCTERM
PRT1	SPOOLER		L80,P58,QUEUE=A,HPLASER
COM1	MODEM	6:5:0	B19200,CTS,PUBLIC
TAP1	TAPE1	55:101:0	PUBLIC

The “Physical Name” displayed is the first name or synonym name found in the `SYSTEM.TEOS32.DEV NAMES` file that matches the “Device Number” parameters. For a description of this file, refer to Appendix D: “[System Files](#),” starting on page 721.

Options

- ALF** Applies to printer attachments only. Specifies that the printer will perform an automatic line-feed upon receipt of a carriage return character. Normally THEOS appends a line-feed to a carriage return when sent to a printer. This option tells THEOS to not append the line-feed because the printer will do that automatically.
- Bnnnnnn** Specifies the baud rate (serial devices only). Valid baud rates are dependent upon the physical device driver and are limited to the following values: 110, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200, 38400, 57600, 76800 and 115200.
- BIDIR** Indicates that bidirectional flow control is desired. May only be used on serial devices and not all device drivers support this feature.

Normally flow control refers to the transmission of characters from the computer to the peripheral device. When the **BIDIR** option is specified, the flow control also applies to the receipt of data from the peripheral. For instance, a **BIDIR** with **XON** handshaking means that the device driver will send an **XOFF** to the peripheral when it is not able to accept more characters and an **XON** when it is ready again.

This option is used in conjunction with the **CTS**, **DTR**, **ETX** or **XON** options.

- Cnnn** Applies to console and printer attachments only. Specifies the class code for input and output translation for the console or printer. For an explanation of class codes, refer to “[Console Class Codes](#)” on page 67.
- CTS** Specifies CTS hardware handshaking (serial devices only). The driver will use the DTR (Data Terminal Ready) signal to know when the peripheral device is ready to accept another character.
- DTR** Specifies DTR hardware handshaking (serial devices only). It's similar to CTS handshaking except that the driver monitors the DTR (Data Terminal Ready) signal.
- En** Specifies the handshaking protocol for flow control (serial devices only).

Enable	Meaning
E0	Disables flow control.
E1	Synonym to DTR option.
E2	Synonym to XON option.
E3	Synonym to ETX option.
E4	Synonym to CTS option.
E5	PC Term software handshaking. Similar to XON/XOFF handshaking except that scan codes 103 and 101 are used to enable/disable transmission.

- ETX** Specifies ETX/ACK software handshaking (serial devices only). The device driver sends blocks of characters. After a pre-defined number of characters are transmitted, an ETX (End of Transmission) code is sent and the driver waits for the peripheral to respond with an ACK (Acknowledge) code.

FFnnn Applies to console and nonslave printer attachments only. Specifies the time in milliseconds to delay transmission after a FF (form-feed), IL (insert line), DL (delete line), EOS (erase to end of screen) or EOL (erase to end of line) code is sent to the console or printer.

In general, this option is used only when E0 is specified and when the device is connected via a non-intelligent serial port.

Lnnn Applies to console and printer attachments only. Specifies the displayable line length for the peripheral device. The default line length value is 80.

The line-length value for printers may be any value in the range 1–254.

When attaching a console the line length specified is used in conjunction with the page size to determine the character size used. See “[Console Screen Sizes](#)” on page 70.

LFnnn Applies to console and nonslave printer attachments only. Specifies the time in milliseconds to delay transmission after a LF (line-feed) is sent to the console or printer.

In general, this option is used only when E0 is specified and when the device is connected via a non-intelligent serial port.

NP Specifies that no parity generation or checking is performed on data transmitted (serial devices only). Synonymous with the [PN](#) option. This is the default parity for serial devices.

Onnn Applies to printer attachments only. Specifies the number of “overflow lines” for the device. Overflow lines are the lines between the last printable line on a page and the first printable line on the next page. This value is used in combination with the page length specified to determine the number of lines from one page to the next.

The device driver counts each line of text transmitted. When a FF code is to be sent, the device driver sends line-feed characters instead. Line-feeds are transmitted until the page length plus overflow count is reached.

For instance, a printer attached with P5 O1 means that there are six lines on each “page.” This attachment would be appropriate for printing one-inch labels. If three lines are printed

followed by a FF, the device driver will send three line-feed characters instead of the FF code.

This option is normally only used when the peripheral device does not support form-feed controls.

P*nnn* Applies to console and printer attachments only. Specifies the displayable page length for the peripheral device. The default page length value for printers is 58; for consoles it is 24.

The page length value for printers may be any value in the range 1–254.

When attaching a console the page length specified is used in conjunction with the line length to determine the character size used. See “[Console Screen Sizes](#)” on page 70.

PE Specifies that even parity is used (serial devices only).

PN Specifies that no parity is used (serial devices only). This is the default parity for serial devices.

PO Specifies that odd parity is used (serial devices only).

PUBLIC Specifies that the attachment of this device is to be public and available to other users. Devices attached as PUBLIC are only available to other users when they log onto the system from the “Logon please” prompt. See “[Logoff](#)” on page 386. That is, a user that is already logged onto an account will not have access to a newly attached public device. They must logoff and logon to get access to the new device.

A publicly attached device cannot be reattached. To reattach a public device you must first detach it and all other users must be logged off or stopped.

PZ Specifies that zero parity is used (serial devices only).

STP*nnn* Applies to floppy disk devices only. Specifies the track-to-track step delay time in milliseconds.

V*nnn* Applies to VDI devices only and specifies the VDI device driver to use. See “[Attaching VDI Graphics](#)” on page 192.

W*n* Specifies the data word length (serial devices only). Word lengths may be either W7 or W8. W7 is the default word length for serial devices.

XON Specifies XON/XOFF software handshaking (serial devices only). The device driver monitors incoming characters from the peripheral. Receipt of an XOFF character (0x13) means that the peripheral is busy and cannot receive any characters. Receipt of an XON character (0x11) means that the peripheral is ready to receive another character.

If the device is a console with a PC Term class code specified, then an XON request is interpreted as an XPC request.

XPC Identical to the XON option except that it applies to PC Term consoles. Specifying XPC when the device is not a PC Term console is automatically translated to an XON request. The XPCON character has a value of 0x65 and the XPCOFF character is 0x67.

Spooler Options

The following options may be used when attaching a printer to the Spooler. These options specify how subsequent reports sent to this printer will be handled by the printer spooler. They do not affect reports already sent to the printer spooler.

COPIES=*nn* Specifies the number of copies to print for each report.

COPY=*nn* Synonymous with the **COPIES=*nn*** option.

HOLD Reports printed are held and not erased.

NOHOLD After a report is printed it is erased.

QUEUE=*a* Specifies the queue letter that is assigned to subsequent reports. The queue letter *a* is always interpreted as an upper-case queue identifier.

QUEUE=\$*a* Specifies the queue letter that is assigned to subsequent reports. The queue letter *a* is always interpreted as an lower-case queue identifier or as one of the 12 special character queues.

There are 64 possible single-characters queues:

A – Z	26 forms/queues
a – z	26 forms/queues
# \$ % & * + - < = > ^ ~	12 forms/queues

Previous versions of the spooler supported only the first 26 queues. To provide compatibility with programs and procedures designed for previous versions of the operating system, queue specifications default to the upper-case queue letters.

To specify one of the lowercase queue letters you must enclose the queue within quotation marks. The 12 special character queues may be specified without quotation marks. For instance:

```
>attach prt2 spooler (a      ; Refers to queue A
>attach prt2 (queue=x      ; Refers to queue X
>attach prt2 (%)           ; Refers to queue %
```

To specify one of the lowercase queue letters you must either enclose the letter in quotation marks or use the special syntax “QUEUE=\$queue.”

```
>attach prt2 ("a"          ; Refers to queue a
>attach prt2 (queue=$x     ; Refers to queue x
>attach prt2 (queue=$$     ; Refers to queue $
```

When in doubt about how a queue specification will be interpreted always use quotes and then specify the desired character with the desired case.

For additional information about controlling the printer spooler, refer to “[Spooler](#)” on page 518 and to the section “[Print Spooler](#)” on page 87.

Attaching Consoles

Your console is initially attached via the Sysgen configuration or from a Start command issued from another console or user. You cannot detach your console.

The Attach command can be used to change the attachment of your console, either to another device or merely changing one or more of the attachment options. For instance:

```
>attach con (l100 p39
```

or

```
>attach con (c94
```

Be careful when changing certain console attachment parameters such as baud rate, word length and parity because you might change to a setting that does not allow you to communicate with the computer. If that happens, you will have to go to another terminal and stop and restart your user, or you may have to reboot the computer.

The class code is very important when attaching a console. It controls both how information is displayed on the screen and how keyboard characters are interpreted.

Attaching Printers

THEOS allows each user to have as many as 64 printers attached (PRT1 through PRT64). Printers may be attached as private devices, public devices, spooled printers and slave printers.

When attaching a printer you should specify a class code. A printer class code tells THEOS how to translate the various font attribute requests such as bold and italic. It also defines the translations for graphic and international character sets.

A printer class code can define a printer setup string. This is a sequence of characters that initializes the printer. This string is sent to the printer by Attach when it is first attached. If the printer is not ready (powered off, unplugged or off-line), Attach will display a message for approximately 10 seconds as it waits for the printer to become ready. If it is not ready at the end of this time, the setup string is not sent but the printer will be attached.

■ Attaching Printers to the Spooler

To attach a public or private printer, use commands such as:

```
>attach prt3 hslp2 (hplaser p60 180
```

or

```
>attach prt48 hslp3 (epson public
```

To attach a spooled printer the printer spooler must be started (see “[Sysgen](#)” on page 531). If the spooler is started, a printer may be attached to it by using a command like:

```
>attach prt8 spooler
```

or

```
>attach prt50 spooler (1132 hold copies=2 b
```

Remember, attaching a printer to the spooler associates a logical name with the printer spooler. The physical printer attachment to the spooler is controlled by the Sysgen configuration.

■ Attaching Slave Printers

A slave printer is a printer that is physically connected to your console. It is usable only as a private printer because your console is always a private device. To attach a slave printer, use a command like:

```
>attach prt6 con (epson
```

Attaching Disks

Physical disk drives may only be attached to logical disk drive letters A–Z. Logical drive S is always attached as the “system disk” and cannot be changed with this command. Use the [System](#) command described on page [545](#) to reattach the system disk.

By convention, hard disks, RAM disks and image drives use logical disk drive letters A through E and H through Z; floppy disk drives use F and G. However, you may use any disk drive assignments.

■ Hard Disk Drives

The only option available when attaching a hard disk drive is the [PUBLIC](#) option. Hard disk drive attachments are normally performed during system bootup due to specifications in the [Sysgen](#) (see page [531](#)).

Before a hard disk can be used for the first time it must be formatted. Refer to the “[Setup DISK](#)” on page [490](#) for information about disk formatting and partitioning.

■ Floppy Disk Drives

The only options available when attaching a floppy disk drive are [PUBLIC](#) and [STPnnn](#). Floppy disk drive attachments are also normally performed during system bootup due to specifications in the [Sysgen](#) command.

To prevent access attempts to a publicly attached floppy disk drive that has no disk mounted in it, be sure to omit the floppy disk drive letter from the default search sequence. See “[SEARCH](#)” on page [101](#).

■ RAM Disk Drives

A RAM disk is a “pseudo-disk” that is actually an area of memory that is treated as a very fast access disk drive. A RAM disk is attached and used like any other disk drive.

To attach a RAM disk, use one of the physical device names in the `SYSTEM.TEOS32.DEVNAMES` file that indicate it is a RAM disk. The specific name used determines the physical size or capacity of the RAM disk.

RAM Disk Name	Size	RAM Disk Name	Size
RAM64K	64KB	RAM8MB	8MB
RAM128K	128KB	RAM10MB	10MB
RAM192K	192KB	RAM12MB	12MB
RAM256K	256KB	RAM16MB	16MB
RAM384K	384KB	RAM32MB	32MB
RAM512K	512KB	RAM48MB	48MB
RAM640K	640KB	RAM64MB	64MB
RAM768K	768KB	RAM96MB	96MB
RAM1MB	1MB	RAM128MB	128MB
RAM1536K	1.5MB	RAM160MB	160MB
RAM2MB	2MB	RAM192MB	192MB
RAM3MB	3MB	RAM224MB	224MB
RAM4MB	4MB		

Table 15: RAM Disk Names and Sizes

You cannot attach a RAM disk that is larger than available memory, and you should not attach a RAM disk that uses so much memory that it prevents users from executing their programs.

RAM disks, when attached, are automatically “formatted” and initialized as an empty disk drive. Once attached they can be “reformatted,” labeled, cleared, or have their main directory resized with the [Disk](#) command described on page [296](#).

Since a RAM disk is only a pseudo-disk, do not use it for long-term storage because its contents are lost whenever the system is rebooted.

■ Image Disk Drives

An image disk, as described on page [122](#), is a pseudo-disk that is actually a file on one of the hard disk drives. THEOS supports as many as eight image drives (IMAGE1, IMAGE2, ..., IMAGE8). To attach an image drive, use a command like:

```
>attach i image2
```

Any drive letter (other than s) can be used as the logical drive name and it may be attached with the [PUBLIC](#) option.

When an image drive is attached, the Attach command first looks to see if it can find a file named DISK.IMAGE*n*. If it can find that file, it uses it as the

image drive. When it doesn't find the file it asks you for the image file name:

```
Enter IMAGE file name (or ESC to exit):
```

Enter a file name that you want to use for the image drive. This name can include the path specification and an account name for ownership. For instance:

```
Enter IMAGE file name (or ESC to exit): PRIVATE\SUB/TEST.IMG
```

Be careful to use a name that is not an existing file name unless it is the name of an existing image drive file. If you expect to be using this image drive many times use the name `DISK.IMAGE n` where n is the number of this image drive attachment. If the file exists, Attach uses it as the image drive and does not change its contents.

You may also specify the image drive file name on the Attach command line:

```
>attach i example.img:a
```

If the file does not exist, Attach responds with:

```
File not found!
```

```
Enter <CR> for new file name
or  <size>M for size in megabytes
or  <cyls>,<heads>,<sectors> ---
```

Entry of `[Enter]` or an invalid entry causes the previous question to be asked. The other two valid entries cause Attach to create the file and initialize it to the requested size.

Entering a whole number followed by `[M]` is interpreted as a size in megabytes. For instance, 5M means five megabytes, 1M means one megabyte.

Entering three whole numbers separated by commas is interpreted as a specification of the number of cylinders, heads and sectors for the pseudo-disk. Use this type of specification when the image drive will be used as a backup of a real disk. Specify the same number of cylinders, heads and sectors as used by the real disk to be imaged. For instance: 80, 2, 36 is the format used for a 1.44MB 3½" floppy diskette.

After an image drive is attached, it can be used like any other disk drive.

Attaching CD-ROM Drives

CD-ROM drives are attached similar to hard disk drives.

```
>attach 1 cdrom1 (public
```

Attaching Tapes

THEOS allows each user to have as many as four tape drives attached at one time (TAPE1, TAPE2, TAPE3 and TAPE4). The only option available when attaching a tape drive is the [PUBLIC](#) option. For instance:

```
>attach tape tape1 (public
```

Before a tape can be used for the first time it must be initialized. Refer to “[Tape](#)” on page [549](#) for information about tape initialization.

Attaching COM Ports

A COM device is a general purpose, bidirectional communications port. It is normally a serial port on the computer but it may be a parallel port. THEOS allows each user to have as many as 16 communications devices (COM1 through COM16).

To attach a communications device use a command like:

```
>attach com multi2 (b38400 w8 cts pe
```

■ Comm and Console the Same Device

In some situations it is desirable to attach a logical COM device to the same physical port used by your console. This occurs when you are using THEO+COM or ScanTerm to be a user on a remote THEOS system, and you need to transfer files between the two systems. To perform this type of attachment use the command:

```
>attach com con
```

This attaches the logical device name COM1 to the same port used by your console and it does not change any of the existing settings used by the console. For instance, the baud rate, word length, parity, *etc.* remain unchanged.

When using this type of attachment, the operation of the comm port will depend upon the [COM=CON Default Bypass](#) setting in the system configuration file. Refer to the [Sysgen](#) command on page [533](#) for a description of this setting.

■ “Self Testing” Communications

In many cases a communications device is used to communicate with another remote computer system. To facilitate testing of programs using this type of remote access, THEOS provides a special device driver for per-

forming self-testing. That is, testing of the communications by communicating with another user on the system. This device has four names of SELF1 through SELF4 and they operate in pairs.

For instance, a pair of programs that are intended to communicate between two THEOS systems via a COM port can be tested on a single computer system by using two users on the system. The first user would use an attachment like:

```
>attach com1 self1
```

The second user on the same system would use an attachment like:

```
>attach com1 self2
```

Different logical names could be used on each attachment and different options could be specified. The important feature of these attachments is that one user is attached to SELF1 and the other is attached to SELF2. These two “physical devices” communicate with each other on the same system. The physical devices SELF3 and SELF4 form another pair of devices that communicate with each other on the system. Any data transmitted by the first user to SELF1 will be received by the other user on SELF2, and vice versa.

Attaching VDI Graphics

A VDI or Virtual Device Interface device is a peripheral capable of displaying graphics. The main console can be used as a VDI device and also some printers. Some PC Term terminals can also be used as VDI devices. Each user may have as many as four VDI devices attached at one time (VDI1, VDI2, VDI3 and VDI4).

To attach your console as a VDI device, use a command like

```
>attach vdi con
```

or

```
>attach vdi3 con (v200
```

To attach a printer as a VDI device, use a command like

```
>attach vdi2 prt (v216
```

or

```
>attach vdi4 sio3 (b38400 w8 cts EPS0N2
```


When attaching a VDI device to a serial port be sure to specify W8 for word length.

Notes	<p>Except for the console and the optional slave printer, all other private devices are detached when you log off and all publicly attached devices are reattached when you log on. See “Logoff” on page 386.</p> <p>The logical devices PRT1, COM1, TAPE1 and VDI1 can all be specified as PRT, COM, TAPE and VDI respectively.</p>
Defaults	<p>There are no built-in default options for an attachment. Any defaults used by the <code>Attach</code> command are defined in the <code>SYSTEM.TEOS32.DEV NAMES</code> file for the physical device being attached. For a description of this file see Appendix D: “System Files”, “SYSTEM.TEOS32 Library”, “DEV NAMES” on page 731.</p>
Cautions	<p>Be careful when reattaching the console. Because the console is the device used to display the result of the attachment and to make further changes, reattaching the console incorrectly might result in loss of control for your partition.</p>
Restrictions	<p>The CON and S logical devices cannot be detached. (It is possible to define a “user” that has no console. See “Start” on page 527.)</p>
See also	<p>ClassGen, Logoff, Spooler, Start, Sysgen, System</p>

Backup Command

The Backup command makes an exact copy of one disk onto another disk or tape, or it restores a backup copy from a tape to a disk.

- 1 **BACKUP** *from-drive to-drive* (*options*
- 2 **BACKUP** *from-drive to-tape* (*options*
- 3 **BACKUP** *from-tape to-drive* (*options*
- 4 **BACKUP** *drive* (*options*

<i>drive</i>	»	drive letter of source and destination disk drive
<i>from-drive</i>	»	drive letter of source drive to backup
<i>to-drive</i>	»	drive letter of destination drive to backup to
<i>from-tape</i>	»	logical tape name of source tape to restore backup from
<i>to-tape</i>	»	logical tape name of destination of backup
<i>options</i>	»	<div> <div> ASK BUFFER FROM <i>file</i> MULTIUSER </div> <div> NOASK NOVERIFY TO <i>file</i> VERIFY </div> </div>

Operation

Mode 1—This syntax causes BACKUP to copy the entire contents of *from-drive* to the *to-drive*. The two drives may be any combination of hard disks, floppy disks, RAM disks or image drives, but they must both be the same capacity and format.

Mode 2—Creates a backup copy of the *from-drive* contents onto the *to-tape*. The backup copy on the tape can only be used with a Mode 3 form of the BACKUP command.

Mode 3—Restores a backup copy on the *from-tape* onto the *to-drive*. The backup copy on the tape must have been created by a Mode 2 form of the BACKUP command, and it must be a backup copy of a disk volume that has the same capacity and format as the *to-drive*.

Mode 4—Creates a backup copy of the contents of *drive* by using one disk drive only. *drive* must be a removable disk drive.

The backup is done by copying as much of the source disk as possible to memory, asking you to change the diskette in *drive* to the destination diskette, copying the memory image to the destination diskette, and then asking you to switch back to the source diskette. This process is repeated until the entire source diskette is copied to the destination diskette.

This mode of the Backup command is most efficient when the **BUFFER**, **TO** or **FROM** options are used.

Options

ASK

A default option that instructs Backup to ask the operator to mount the source and destination volumes, and waits for confirmation that the proper volumes are mounted. For instance:

```
>backup f i

Source is Disk F
Destination is Disk I
Mount volumes now:

Source is labeled "Programs".
Destination is labeled "Image".
OK to start backup (Y/N)
```

Respond with a only for the “Mount volumes now:” request, and a or for the “OK to start backup” question.

Use the **NOASK** option to override this default.

BUFFER

Valid only with **Mode 4** of the command. When used without the **FROM** or **TO** options, this option causes Backup to use a temporary file on the S drive as an image buffer of the single drive backup.

In other words, the contents of the source diskette are written to a temporary file, the destination diskette is placed in the drive, and the contents of the temporary file are written to the destination diskette. The temporary file is then erased.

FROM

This option is used in combination with the **BUFFER** option to specify that a file created from a previous backup using the **BUFFER TO** option is to be used as the source for this backup. The file name is specified following this option:

```
>backup f (buffer from distrib.master:s
```

The file must be a file created from a previous backup using the **BUFFER TO** option. The format of this backup disk (number of cylinders, heads and sectors) must be identical to the format

of the original backup disk when the file was created. See **TO** option below.

MULTIUSER Allows Backup to copy a public drive even though other users may be logged on and active. When Backup is instructed to perform a backup of a public disk, it requires single user mode. If other users are logged onto the system, it displays the message: “Must be single user or private volume.”

Using this option tells Backup to not restrict the backup to single-user operation (the message is still displayed). **THIS CAN BE EXTREMELY DANGEROUS!** If another user changes some files while the backup is being created, the integrity of the backup is lost.

Use this option only if you are sure that all other users are inactive.

NOASK Disables the source and destination volume operator confirmation at the beginning of the backup and when subsequent disks or tapes are needed.

NOVERIFY Disables the verify after write operation. See **VERIFY** option below.

TO This option is used in combination with the **BUFFER** option to specify that the destination of the backup is not another disk but a file.

The file name is specified following this option:

```
>backup f (buffer to distrib.master:s
```

Although the file created with this option can be copied to another drive, transmitted to another computer, *etc.*, just like any other stream file, it is intended to be used as the source for another backup using the **BUFFER FROM** option or as an image drive.

VERIFY Applies only when the *to-drive* is a disk not a tape. This option causes Backup to read each track of data immediately after writing it to the *to-drive*. The information is then compared with the original information from the *from-drive*. Only when the data is the same does the backup continue to the next track.

Notes	<p>When a backup is performed, both the <i>from-drive</i> and the <i>to-drive</i> are modified to reflect that the disk was the source or destination of a backup. For instance, after a backup of the F drive:</p> <pre>>disk f Disk F label "Programs". Backup to disk "ProgCopy" on 10/26/96, at 13:20. File system = THEOS/4GB. Capacity = 1,474,560 (80 cylinders, 2 heads, 36 sectors). Main directory size = 4 files. Allocated bytes = 165,632. Available bytes = 1,308,928.</pre> <p>While the backup copy is being created the current cylinder and head number are displayed on the console. This information is provided so the operator can see how the backup is progressing.</p>
Defaults	ASK , NOVERIFY
Cautions	<p>The MULTIUSER option tells the Backup command to not check whether or not other users are logged on or active. It does not prevent those other users from performing operations that change the database that's being backed up. If another user does change the database during the backup operation, the integrity of the backup is compromised.</p> <p>Backup always destroys the prior contents of the destination drive, before any new data is written to the drive.</p>
Restrictions	<p>The <i>from-drive</i> and the <i>to-drive</i> must have the same format. That is, they must both have the same number of cylinders, heads and sectors.</p> <p>The Backup command requires a privilege level of four.</p>
See also	Archive , CopyFile , Disk , Restore , Tape , TBackup

Boot32 and Boot40 Commands

These commands change the names of the operating system files and reboot your computer to THEOS 386/486 or back to THEOS 32.

1 BOOT32

2 BOOT40

These commands are only installed and usable if THEOS 386/486 (Version 3.2) was installed on your system at the time that THEOS 32 was added to the system. That installation process renames all of the Version 3.2 libraries from SYSTEM.CMD386, SYSTEM.TEOS386, *etc.* to SYS.CMD386, SYS.TEOS386, *etc.* The BOOT32 command is installed in the SYSTEM.CMD32 library and the BOOT40 command is installed in the SYS.CMD386 library.

Operation

Mode 1—This command is used when you are using THEOS 32 and wish to use the older THEOS 386/486 operating system. It may only be used if the system disk has not been partitioned with THEOS 32 and changed to a THEOS/4G disk system.

>boot32

When executed it renames all of the THEOS 32 system file libraries from SYSTEM.CMD32, SYSTEM.TEOS32, SYSTEM.MENU32 and SYSTEM.HELP32 to SYS.CMD32, SYS.TEOS32, SYS.MENU32 and SYS.HELP32. It renames the THEOS 386/486 system file libraries from SYS.CMD386, SYS.TEOS386, SYS.MENU386 and SYS.HELP386 to SYSTEM.CMD386, SYSTEM.TEOS386, SYSTEM.MENU386 and SYSTEM.HELP386. It then changes the bootstrap loader for the system disk to the THEOS 386/486 bootstrap loader and reboots the computer.

Mode 2—This command is used when you are using THEOS 386/486 and wish to use the newer THEOS 32 operating system.

>boot40

When executed it renames all of the THEOS 386/486 system file libraries from SYSTEM.CMD386, SYSTEM.TEOS386, SYSTEM.MENU386 and SYSTEM.HELP386 to SYS.CMD386, SYS.TEOS386, SYS.MENU386 and SYS.HELP386. It renames the THEOS 32 system file libraries from SYS.CMD32, SYS.TEOS32, SYS.MENU32 and SYS.HELP32 to SYSTEM.CMD32, SYSTEM.TEOS32, SYSTEM.MENU32 and SYSTEM.HELP32. It then changes the bootstrap loader for the system disk to the THEOS 32 bootstrap loader and reboots the computer.

Note	A Cache SYNC operation is performed before rebooting to maintain data integrity.
Cautions	<p>This is an extremely dangerous command because other users are terminated without notice. If another user is in the process of updating one or more files, those files will be inaccurate because the update was not completed.</p> <p>Always perform a Show USERS command before using either of these commands and verify that all other users are logged off, at the CSI prompt or stopped.</p>
Restrictions	These commands require a privilege level of five.
See also	Reboot

Cache Command

The Cache command enables, disables, controls or reports on the disk cache. For a general description of the THEOS disk caching capabilities, refer to “Disk Caching” on page 124.

1 CACHE *function*

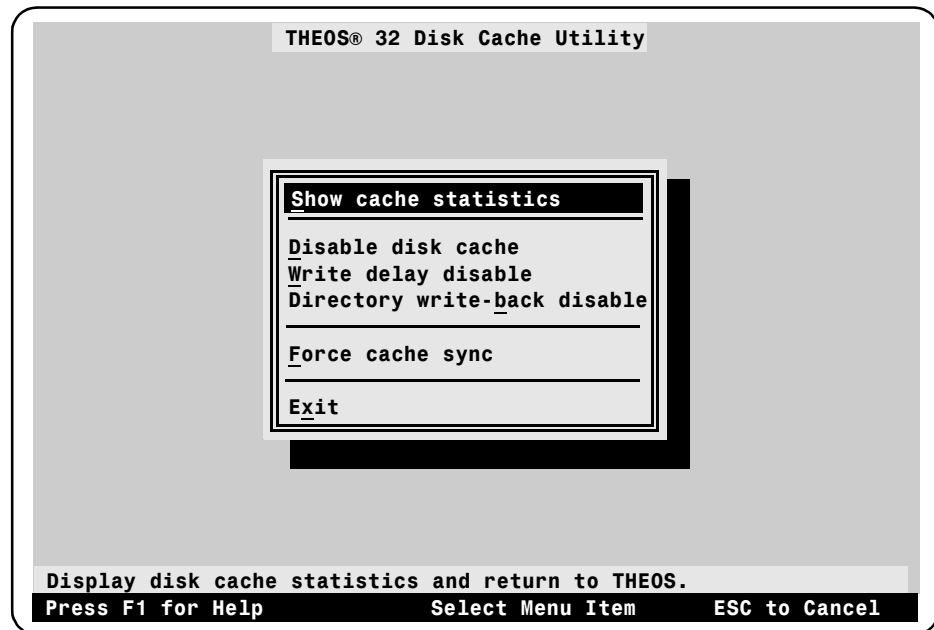
2 CACHE

<i>function</i>	»	DELAY OFF	STATUS
		DELAY ON	SYNC
		OFF	WBDIR OFF
		ON	WBDIR ON

- Operation** **Mode 1**—Performs one of the functions described below.
- Mode 2**—Invokes the Cache command menu as described on page 201.
- Functions** **DELAY OFF** Disable disk write delay. Note that this also will disable directory write caching.
- DELAY ON** Enable disk write delay.
- OFF** Disable disk caching. When disk caching is disabled, the memory used by the disk cache is freed.
- ON** Enable disk read/write caching.
- NOWBDIR** Synonymous with [WBDIR OFF](#).
- STATUS** Display disk cache statistics.
- SYNC** Force disk synchronization with memory cache.
- WBDIR OFF** Disable directory write caching.
- WBDIR ON** Enable directory write caching. The keyword “ON” may be omitted.

Cache Menu

When using Mode 2 of the Cache command, a menu is displayed allowing you to choose the function to perform or change.



Commands

Use the normal menu selection keys to select the desired function. These keys are described in [“Using Menus”](#) on page 75.

The text for the three enable/disable menu items changes according to the current status of that function. For instance, the above menu reflects a cache that has all features enabled. If the write delay feature were disabled, the third menu item would display “Write delay enable.”

Cache Status and Statistics Display

When the [STATUS](#) command line option is used, or when the “Show cache statistics” menu item is selected, the following screen appears.

```

THEOS® 32 Disk Cache Utility

Cache size (kilobytes):          6,144
Block size (bytes):             8,192
Number of blocks:               755
Write delay:                    ON
Directory write back:           ON

Hit rate: 95% (instantaneous), 86% (average)

Number of write (dirty) blocks:  0
Number of probes:                1,320,975
Number of hits:                  1,306,902
Number of syncs:                 19
Number of reads:                 1,315,355
Number of writes:                197
Number of bypasses (large r/w):  5

Press ESC to terminate.

```

This display remains on the screen and has three sections: a top section that displays the status of the current disk cache settings; a middle section showing the current cache hit rate; and a lower section displaying the disk cache usage statistics that is updated approximately once per second.

The cache status and statistics display is only available if disk caching is enabled.

■ Cache Status Display

The top portion of the display shows the current disk cache settings. These settings are controlled by the `Cache` command itself or from information recorded in the `Sysgen` configuration.

Setting	Meaning and Source
Cache size	<p>The value shown here indicates the amount of memory used by the disk cache system for caching data from physical disks. It is shown in kilobytes so the value illustrated in the previous figure reflects a six megabyte disk cache.</p> <p>The disk cache size is normally set in the <code>Sysgen</code> configuration but it may also be set by the <code>Cache</code> command itself if no size is specified in the configuration and the <code>Cache</code> command enables disk caching. A default size of 256KB is used in this situation.</p>
Block size	The block size is determined by the cache size. It indicates the smallest amount of data that is read or written to disk.
Number of blocks	This is a computed value that reflects the number of blocks of data that can be held in the disk cache memory. It is computed by dividing the cache size by the block size.
Write delay	Shows the status of the cache write delay as set by the <code>Cache DELAY</code> function or the <code>Sysgen</code> configuration.
Directory write back	Shows the status of the cache directory write back feature as set by the <code>Cache WBDIR</code> function or the <code>SYSGEN</code> configuration.

Cache Status Display

■ Cache Hit Rate

The cache hit rate shown in the middle of the display reflects the efficiency of disk caching. It shows the overall rate of success on the right side of the display reflecting the number of times a disk access request was satisfied since disk caching was enabled.

The instantaneous hit rate shown on the left side of the display reflects the rate of success for the most recent disk access requests over the last three seconds, or so.

■ Cache Statistics Display

The lower portion of the display shows the current cache statistics. This information is updated approximately once per second, thus showing the disk activity of other users and tasks.

Display Item	Meaning
Number of write (dirty) blocks:	Shows the number of blocks of data currently in the disk cache that have not been written to the physical disk yet.
Number of probes:	Shows the total number of times that a request was made for reading or writing to the disk.
Number of hits:	Shows the number of times that a request for information from the disk was able to be satisfied by information already in the disk cache memory.
Number of syncs:	Indicates the number of times that the disk cache system has been forced to synchronize itself with the physical disk. (See SYNC function.)
Number of reads:	Shows the number of times the disk cache system has provided information from a disk read request. This includes the times when the information is already in the disk cache memory and the times that it had to read from the physical disk.
Number of writes:	Shows the number of times the disk cache system was asked to write information to the disk. This includes the number of blocks still in the cache memory waiting to be written to the physical disk (dirty blocks).
Number of bypasses (large r/w):	Indicates the number of times that the disk cache system was bypassed, probably due to a request from a large read or write operation for “one time only” data.

Cache Statistics Display

Notes	A Cache SYNC operation is performed when you reboot the system using Boot32 , BOOT40 , Reboot or Ctrl + Alt + Del .
Cautions	<p>The performance gains when using disk caching with write delay enabled are considerable. However, there is a slight risk to data integrity. If a power failure occurs or if the system is reset between the time that a program requests a disk write operation and when the disk caching software actually performs the write to the physical disk, the data cannot be written.</p> <p>If this risk is unacceptable, you can still take advantage of disk caching with write delay enabled by utilizing an on-line uninterruptable power supply (UPS) for your computer system.</p>
Restrictions	<p>The Cache command requires a privilege level of five.</p> <p>If disk caching is disabled, the Cache STATUS cannot be displayed.</p>
See also	Sysgen

Calendar Command

Calendar displays a calendar for a single month or for an entire year.

- 1 CALENDAR
- 2 CALENDAR *month*
- 3 CALENDAR *month year*
- 4 CALENDAR *year*

<i>month</i>	»	calendar month number, month name or abbreviation
<i>year</i>	»	calendar year number, with or without century number

Operation **Mode 1**—Displays the calendar for the current month and year in month display format (see “[Month Display](#)” on page 207).

Mode 2—Displays the calendar for *month* in the current year in month display format (see “[Month Display](#)” on page 207). The *month* may be specified as a month number of the year (1–12), a month name (January, February, etc.) or with a month name abbreviation (Jan, Feb, Mar, *etc.*). For example:

```
>calendar 3
>calendar March
>calendar mar
```

The above three commands will display the calendar for March of the current year.

Mode 3—Displays the calendar for *month* in *year* in month display format (see “[Month Display](#)” on page 207). The *month* may be specified as it is for [Mode 2](#). The *year* may be specified as a year number in the current century (1–99) or as a complete year number (1776, 1812, 1997, 2004, *etc.*). See “[Date Limitations](#)” on page 207. For instance:

```
>cal 7 1776           ; Display July, 1776
>cal July 1776        ; Display July, 1776
>cal Jan 3            ; Display January, 1903
```

Mode 4—Displays the calendar for *year* in year display format (see “[Year Display](#)” on page 208). The *year* may be specified as in [Mode 3](#), with the exception of current century years less than 13. To avoid confusion with a [Mode 2](#) request, years between 1–12 must be specified with their century number (1901, 1902, *etc.*). For example:

```
>calendar 0           ; Display 1900
>calendar 1903        ; Display 1903
>calendar 3           ; Display March, current year
>calendar 2010        ; Display 2010
```

Date Limitations

Only Gregorian calendar dates are displayed correctly by this program. Because the Gregorian calendar was inaugurated October 15, 1582, only calendars for years greater than or equal to 1583 are displayed by this command.

Month Display When a single month is displayed by Calendar, a simulated calendar page is drawn on the screen. For instance:

April						1996
Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

When the month display is the result of a [Mode 1](#) request (display current month), the current day number will blink.

Year Display

When a full year is displayed by Calendar ([Mode 4](#)), two screen displays are used, each displaying six months of the requested year. For instance, if a CALENDAR 1999 is requested:

January 1999							February 1999							March 1999						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
					1	2		1	2	3	4	5	6		1	2	3	4	5	6
3	4	5	6	7	8	9	7	8	9	10	11	12	13	7	8	9	10	11	12	13
10	11	12	13	14	15	16	14	15	16	17	18	19	20	14	15	16	17	18	19	20
17	18	19	20	21	22	23	21	22	23	24	25	26	27	21	22	23	24	25	26	27
24	25	26	27	28	29	30	28							28	29	30	31			

April 1999							May 1999							June 1999						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
					1	2							1			1	2	3	4	5
4	5	6	7	8	9	10	2	3	4	5	6	7	8	6	7	8	9	10	11	12
11	12	13	14	15	16	17	9	10	11	12	13	14	15	13	14	15	16	17	18	19
18	19	20	21	22	23	24	16	17	18	19	20	21	22	20	21	22	23	24	25	26
25	26	27	28	29	30		23	24	25	26	27	28	29	27	28	29	30			
							30	31												

A similar screen is displayed for the months July through December.

Notes

The month names are defined in the SYSTEM.TEOS32.MESSAGE file, numbers 260–261. For a description of this file, refer to Appendix D: “[System Files](#),” starting on page [721](#).

If the calendar display is redirected to a device or file other than the console, month displays are output in a format similar to the year display, but for one month only.

Defaults

When no *month* or *year* is specified, Calendar displays a current month calendar.

Restrictions

See “[Date Limitations](#)” on page [207](#).

Cat Command Filter

The Cat command is a filter that concatenates one or more files and outputs the result to the standard output device. For a description of command filters, refer to “[Standard Input, Standard Output and I/O Redirection](#)” on page 51.

1 CAT (*options*

2 CAT *file...* (*options*

file » file name with optional path; may contain wild cards

options » NOTYPE
 TYPE

Operation

Mode 1—Copies text from the standard input device to the standard output device.

Mode 2—Each file is copied to the standard output device, one after the other.

```
>cat one.file two.file
```

This command copies ONE.FILE to the standard output device and then appends TWO.FILE to the end.

```
>cat text.file1 text.file2 text.file3 > new.text
```

This second example uses i/o redirection to change the standard output device for this command. It is equivalent to:

```
>copy text.file1 new.text
>copy text.file2 new.text (append
>copy text.file3 new.text (append
```

The standard input device file can be included in the list of *files* to be copied by using the “-” specification as a file name:

```
>cat - one.file two.file
```

This command copies the contents of the standard input device file to the standard output device and then appends ONE.FILE and TWO.FILE onto the standard output device file.

When one or more of *file* contains a wild card specification, the files matching the specification are sorted in alphabetical sequence and then processed as if they were specified individually. For instance, assume that there are files named LETTER.DOCUMENT, MEMO.DOCUMENT and SPECIAL.DOCUMENT:

```
>cat *.document
```

This command is equivalent to:

```
>cat letter.document memo.document special.document
```

Commands

Options

NOTYPE Suppresses all error message display.

-s Synonymous with the **NOTYPE** option described above. To use this option, do not use the left parenthesis. Instead, merely specify it somewhere in the command line.

```
>cat -s one.file two.file
```

```
>cat one.file two.file -s
```

TYPE A default option that allows error messages to be displayed. The most common error message is “File not found.” which is output when one or more of the input files does not exist.

Error messages are displayed on the standard error device which is normally the same as the standard output device.

Notes

A *file* specification can omit the file type if the environment variable **FILETYPE** is defined.

For more information about the **FILETYPE** variable, see “[Environment Variables](#)” on page 102.

When the standard input device is the console, the end-of-file is indicated by pressing **[Ctrl]+[D]**.

When the standard output device is a disk file, it is erased prior to outputting the first file. Subsequent files are then appended to this new file. To append a file to an existing file, use the append i/o redirection symbol:

```
>cat new.file >> old.file
```

As a command filter, Cat can be used in a pipe command:

```
>filelist *.basic | cat heading.file - > result.file
```

This command line creates a directory listing and pipes this output to the Cat command, which first outputs `HEADING.FILE` to `RESULT.FILE` and then appends the directory listing which is now in the standard input device.

See also[CopyFile](#)

CDPlayer Command

The CDPlayer command plays an audio CD in the CD-ROM drive.

Commands

- 1 `CDPLAYER drive`
- 2 `CDPLAYER drive PLAY`
- 3 `CDPLAYER drive PAUSE`
- 4 `CDPLAYER drive RESUME`
- 5 `CDPLAYER drive STOP`
- 6 `CDPLAYER drive RANDOM`
- 7 `CDPLAYER drive EDIT`
- 8 `CDPLAYER drive LIST`
- 9 `CDPLAYER drive TRACK track`

drive » drive code letter of an attached CD-ROM drive

track » track number of audio CD in drive

Operation **Mode 1**—This is the interactive play mode of the CDPlayer command. In this mode, the audio CD is played, starting at track one.

```
>cdplayer d
Lawrence Welk:  His Greatest Hits
1:52 [Track 1] Wild Irish Rose
```

If the artist, title and track titles have been entered into the CD-ROM catalog for the disc in *drive*, that information is displayed as each track plays. If the information is not available, only the time remaining and the track number are displayed.

```
>cdplayer d
1:52 [Track 1]
```

While this mode is operating, you may use the `[PageDown]` or `[→]` key to jump to the next track, and the `[PageUp]` or `[←]` key to jump to the previous track.

The **[Enter ↵]** key replays the current track and the **[Esc]** key exits the program and terminates playing the album.

Mode 2—Starts playing the audio CD in *drive*. If the artist and album title have been entered into the CD-ROM catalog for the disc in the drive, that information is displayed. The command exits but the disc continues playing until it reaches the end of the album.

```
>cdplayer c play
Don Ho: Tiny Bubbles
>
```

While the disc plays, there are very few, if any, resources used and there is minimal impact on the operation of the system.

Mode 3—Pauses but does not stop the playing of an audio disc in *drive*. If no disc is playing (see [Mode 2](#)), no error is detected or reported.

Mode 4—Resumes playing of an audio disc in *drive* that was paused with a [Mode 3](#) command. If no disc is playing (see [Mode 2](#)) or it is not paused, no error is detected or reported.

Mode 5—Stops playing the audio disc in *drive*. If no disc is playing (see [Mode 2](#)), no error is detected or reported.

Mode 6—This is an interactive play mode of the CDPlayer command, similar to [Mode 1](#). However, in this mode, the audio CD is played in random order.

```
>cdplayer d random
Lawrence Welk: His Greatest Hits
3:21 [Track 6] Beer Barrel Polka
```

If the artist, title and track titles have been entered into the CD-ROM catalog for the disc in *drive*, that information is displayed as it plays each track. If the information is not available, only the time remaining and the track number are displayed.

While this mode is operating, you may use the **[PageDown]** or **[→]** key to jump to the next track in the random order, and the **[PageUp]** or **[←]** key to jump to the previous track in the random order. The **[Enter ↵]** key replays the current track and the **[Esc]** key exits the program and terminates playing the album.

When each track has been played once, the program exits.

Mode 7—This mode allows you to define or edit the descriptive information about the audio CD in *drive*. Each audio CD is uniquely identified by a key and the database (/CDROM.CATALOG:S) uses this key to identify the matching descriptive information about the audio CD.

When this mode is used with a CD that has not been described before, the screen clears and displays:

```
Artist:
Title:
Track 1:
Track 2:
Track 3:
Track 4:
Track 5:
Track 6:
Track 7:
Track 8:
Track 9:
Track 10:
Track 11:
```

The specific number of tracks displayed reflects the actual number of tracks on the audio CD.

While entering the text, you may use some of the standard editing keys such as **Home**, **End**, **Del**, **←**, **→**, **↑**, **↓**, and **F5**. Each track description is maintained in a separate record and is written to the database as soon as you move to a different track description line.

Mode 8—Displays the descriptive information about the CD in *drive*.

```
>cdplayer d list
```

```
Don McLean: Classics
```

```
1 8:32 American Pie
2 5:18 Vincent
3 4:46 And I Love You So
4 3:49 Crying
```

The times listed for each track are the times specified in the index on the audio CD.

Mode 9—Plays the single *track* of the audio CD in *drive*. If the artist and album title have been entered into the CD-ROM catalog for the disc in the drive, that information is displayed along with the track title. The command exits but the disc continues playing until it reaches the end of the requested *track*.

```
>cdplayer c track 4
Don Ho:  Tiny Bubbles
        Hawaiian Wedding Song
>
```

Notes	A drive mount operation is performed when CDPlayer first starts. This operation closes the drive tray if it is not already closed.
Defaults	The <i>drive</i> specification may be omitted. When not specified, the first attached CD-ROM drive is used.
Restrictions	The CD-ROM disc in <i>drive</i> must be an audio CD.
See also	Attach , Eject , Mount

Change Command

The Change command changes a file's attributes.

- 1 CHANGE *file* (*options*
- 2 CHANGE *file* (FILES *options*
- 3 CHANGE (*options*

<i>file</i>	»	file name with optional path; may contain wild cards	
<i>options</i>	»	ACCOUNT <i>name</i>	PRIVATE <i>codes</i>
		FILES	PRIVLEV <i>nn</i>
		GROW <i>n.m</i>	QUERY
		NOQUERY	SHARED <i>codes</i>
		NOTYPE	TOUCH
		OWNER <i>name</i>	TYPE
		PATCH <i>nnnnnnnn</i>	

Operation **Mode 1**—Changes *file* according to *options*.

```
>change *.data (touch
Ok to change "FILE1.DATA:S" (Yes,No,All) Y
"FILE1.DATA:S" changed.
Ok to change "FILE2.DATA:S" (Yes,No,All) N
Ok to change "FILE3.DATA:S" (Yes,No,All) A
"FILE3.DATA:S" changed.
"FILE4.DATA:S" changed.
```

Mode 2—The files listed in *file* are changed according to *options*. *file* must be an ASCII stream file containing one file description per line. The SELECTED.FILES and SELECTED.EXEC files created by [FileList](#) and the FOUND.EXEC created by [Look](#) can be used for this specification file (see “[The EXEC and FILES Options](#)” on page 338). You may also create the file with an editor or application program.

For instance, [FileList](#) is used to create a list of files to be changed:

```
>FILELIST *.DATA:A (EXEC
>FILELIST A NOT *.DATA:A (10/1/95 EXEC APPEND
```


A file now exists that lists all of the “data” files and all files that have been changed since 10/01/1995. The following command will change the ownership of these files to another account:

```
>change selected.exec (file account develop
```

Mode 3—Enters the interactive mode for selecting files to be changed.

```
>change (touch
Enter file name list, terminate with empty line.
?
```

You may enter any file name desired, with optional path and possibly wild card specifications. For instance:

```
>change (touch noquery
Enter file name list, terminate with empty line.
?*.data
"FILE1.DATA:S" changed.
"FILE2.DATA:S" changed.
"FILE3.DATA:S" changed.
"FILE4.DATA:S" changed.
?*.text
"FILE1.TEXT:S" changed.
"FILE2.TEXT:S" changed.
"FILE3.TEXT:S" changed.
?
>
```

Options

ACCOUNT Changes the account ownership of the files. The new owning account is specified immediately following this keyword:

```
>change data.file (account develop
```

This option must be used by itself. That is, you may not change the ownership of a file and its protection codes, growth factor, *etc.*

You may only change the ownership of a file in the root directory. You may not change members of a library (however, the library itself may be changed).

FILES Indicates that Mode 2 of the Change command is to be used. The *file* specifies an ASCII stream file, with each record in the file specifying a single file name. The file name specifications in this file may include the path to the file.

For instance, a line in the specification file might contain:

```
custom/programs/program.source.sample:s
```

If this file is used in a Change command, the display will be:

```
>change selected.exec (file
"/CUSTOM/PROGRAMS/PROGRAM.SOURCE.SAMPLE:S" changed.
```

GROW Specifies the growth factor for the file.

```
>change data.file (grow 0.5
```

Growth factors are specified as either a whole number, such as 1, 2, 3, *etc.*, or as a fraction, such as 0.1, 0.3, 0.5, *etc.* Do not specify a combination like 1.3. If you do, only the integer portion is used. Program and stream files do not have growth factors.

For an explanation of file growth factors see “[Growth Factor](#)” on page [144](#).

NOQUERY Tells Change to not ask for confirmation before applying the requested change to a file. This is a default option for explicit *file* specifications.

NOTYPE Suppresses the display of the file changed messages.

OWNER Synonymous with the [ACCOUNT](#) option.

PATCH Changes the patch level for a program file. The new patch level is specified immediately following this keyword:

```
>change special.command (patch 10001
```

Patch levels may only be added or changed in programs, device drivers or the system nucleus.

PRIVATE Changes the owning account protection codes. The new protection codes are specified immediately following this keyword:

```
>change my.file (private nmexwr
```

See “[Changing Protection Codes](#)” on page [220](#).

PRIVLEV Changes a command’s privilege level. The new privilege level is specified immediately following the keyword:

```
>change my.command (privlev 3
```

QUERY Tells Change that you want to be “queried” or asked if each file matching the selection criteria is to be changed. This is a default option when the file specification used wild cards.

```
>change *.data (touch
Ok to change "FILE1.DATA:S" (Yes,No,All) Y
"FILE1.DATA:S" changed.
Ok to change "FILE2.DATA:S" (Yes,No,All) N
Ok to change "FILE3.DATA:S" (Yes,No,All) A
"FILE3.DATA:S" changed.
"FILE4.DATA:S" changed.
```

When asked, you may respond with a **[Y]** to change the file, an **[N]** to not change this file, or an **[A]** to change this file and all subsequent files. Pressing **[Esc]** exits the CHANGE command but, all changes made prior to the **[Esc]** entry are retained.

To disable this option use the **NOQUERY** option.

SHARED Changes the shared access protection codes. The new protection codes are specified immediately following this keyword:

```
>change my.file (shared wr
```

See “[Changing Protection Codes](#)” on page 220.

TOUCH Changes the file’s date and time to the current system date and time.

This is a default option when no other options are specified.

TYPE A default option that tells Change to display each file that is successfully changed.

To disable this option use the **NOTYPE** option.

Notes

Multiple options may be specified on a single command. For instance, you may request a change in a file’s growth factor, private and shared protection codes and its file date. However, you may not use the **ACCOUNT** or **OWNER** option with any other option.

**Changing
Protection
Codes**

A file has two types of protection codes associated with it: private and shared access. The private protection codes restrict what a user on the owning account may do with the file. The shared access protection codes restrict what a user on a non-owning account may do with the file.

Private protection codes include:

Code	Meaning
w	Write protection. The file cannot be written to.
r	Read protection. The file cannot be read.
e	Erase protection. The file cannot be deleted.
x	Execute protection. The file cannot be executed.

There are two other codes that may be specified here although they are not really protection codes:

Code	Meaning
m	Set the “modified” attribute.
h	Set the “hidden” attribute.

Shared access protection codes include:

Code	Meaning
w	Write protection. The file cannot be written to.
r	Read protection. The file cannot be read.

Shared access erase protection exists when a file has any of the following protection codes enabled: erase, shared read or shared write. Shared read and write protection is always enabled if the private protection is set to read or write protected. That is, setting private write protection enables shared write protection.

With the exception of private read protection, each of these attributes may be turned off by preceding the protection code with the letter n. Multiple attributes may be set at one time by specifying the multiple codes one after the other without any separating spaces. The protection codes may be specified in any sequence.

For example:

```
>change my.file (private nmenw shared rw
```

The above command sets the following attributes for the file: not modified, erase protected, not write protected, not read protected, shared access read and write protected.

Private read protection, once set, cannot be removed. Shared read and shared write protection cannot be disabled if the file still has private read or private write protection enabled. For instance, to disable shared write protection you must also disable private write protection.

Defaults	Touch is a default option when no other options are specified. QUERY is a default option for file specifications using wild cards. TYPE is a default option.
Cautions	<p>After changing a file to hidden, it will not appear in a directory listing (see “FileList” on page 326). However, most commands, including Change, allow access to a hidden file.</p> <p>Private read protection cannot be reset.</p>
Restrictions	<p>The ACCOUNT option must be used by itself.</p> <p>To apply any changes to a file with this command, you must be logged onto the owning account.</p>
See also	Touch

ChDir Command

The ChDir command changes the current working directory.

1 CHDIR *directory*

2 CHDIR

directory » subdirectory file name
 path to directory

Command synonym: CD

Operation **Mode 1**—Changes the current working directory to the specified directory.

>chdir programs

The *directory* may be a simple subdirectory name as shown in the above command line, or it may be a partial or complete path specification to the desired directory.

When *directory* does not start with the root directory specifier (/), it is assumed to be a subdirectory of the current working directory or a reference to a directory relative to the current working directory. For instance:

```
>pwd
/DEVELOP:S

>chdir programs

>pwd
/DEVELOP/PROGRAMS:S

>chdir ../doc/files

>pwd
/DEVELOP/DOC/FILES:S
```

The second example of the ChDir command shows a usage of the special double period directory name (..) pronounced *dot, dot*. This syntax is a reference to the parent directory of the current working directory. In the example, the current working directory was /DEVELOP/PROGRAMS:S so the parent of this directory is /DEVELOP.

Multiple dot, dot specifications can be used to indicate that the reference is to the parent of the parent of the parent, *etc.* For instance:

```
../../../../some.file
```

If the current working directory were AAA/BBB/CCC/DDD/EEE:S, this reference is to the file /AAA/SOME.FILE:S. Alternately, you could use the dot, dot specification and add additional dots to it, one for each additional parent. The above reference could be specified with:

```
...../some.file
```

A directory name may be specified with less than the complete name. For example:

```
>tree
/develop
├── doc
│   ├── files
│   └── screens
└── programs
    └── doc

>cd d
```

The ChDir command above specified a directory name that does not exist (there is no directory named “D” under DEVELOP, the current working directory). In this situation, ChDir will try to find a directory name that starts with the specified name. In this example, it will find the subdirectory “DOC” and it will change to it. When searching for a directory name that starts with the specified name, ChDir does not search the tree alphabetically (although the [Tree](#) command displays the subdirectories sorted alphabetically). Instead it performs a sequential search and uses the first one that it finds. If there is more than one directory name that matches, it might not use the one that you thought it would. For instance, if there are two directories named TEST1 and TEST2, changing to the T directory might change to TEST2.

Mode 2—When no directory name is specified in this command, the ChDir program will check to see if the [HOME](#) environment variable has a value (see “[HOME](#)” on page 106). When [HOME](#) has a value, the current working directory is set to that home directory.

If [HOME](#) is undefined or has no value, the current working directory is set to the root.

```
>show home
HOME=/DEVELOP:S

>pwd
/DEVELOP/PROGRAMS/DOC:S

>cd

>pwd
/DEVELOP:S
```

Notes

The command name CD is a synonym to the ChDir command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file. If standard synonyms are disabled (see “[Account](#)” on page 156 and “[STDSYN](#)” on page 101), this synonym name may not be allowed.

Additional information about directory names, paths and the current working directory may be found in Chapter 8 “[Directories and Files](#),” starting on page 129.

Restrictions

The *directory* specified must be owned by the current account.

See also

[MkDir](#), [PWD](#), [RmDir](#), [Set](#), [Tree](#)

ClassGen Command

The **ClassGen** command creates or changes a console or printer class code specification file.

CLASSGEN *number (option*

number » console or printer class code number (1–255)

$$option \qquad \gg \quad \text{PRT}_{nn}$$

Operation

The SYSTEM.TEOS32 library is searched for an existing class code definition file for class code *number*. If one is found, it is examined to determine if it is a console class code or a printer class code definition.

To determine the proper number to use for a new class code, refer to Appendix D: “[System Files](#)” on page 728.

If no existing class code definition can be found for *number*, you are prompted with:

Is this a CONSOLE class file?

Any response other than `Y` is interpreted as a “no” response and the class code definition will be for a printer.

If the PRT option is specified, the class code definition is output to the printer. (A new class code number will print a report with all fields blank.)

When the PRT option is not used, the appropriate set of maintenance screens are used to allow you to define or change the definitions of the class code.

Options

PRT nn Indicates that ClassGen is to print the current class code definition on the attached printer number nn . nn may be a value between 1 and 16.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

Console Class Codes ClassGen uses several screens to define a class code for consoles. These screens group the definition fields into several categories:

1. Terminal name and cursor control codes.
2. Video display attributes and cursor shape.
3. Screen editing commands.
4. Keyboard input function codes. (Two screens.)
5. Line-drawing graphics. (Two screens.)
6. International symbols. (Three screens.)
7. Keyboard international symbols. (Three screens.)

Refer to the “Notes” section for information about navigating between these screens, moving from field-to-field and inputting special character codes.

■ Console Class Code Maintenance Screens

The following shows the fields in each of the maintenance screens used for console class codes.

The first screen shows the fields for setup and cursor movement.

```

THEOS® 32 Classgen

CONSOLE Output attributes for class: 90 PcTerm U.S.A.

Terminal name..... 90 PcTerm U.S.A.
Setup..... ESC . '1'


Direct Cursor Address... ESC - 0x82 0x83
Home..... RS
Up..... VT
Down..... SYN
Left..... BS
Right..... FF

```

Figure 1: ClassGen, Consoles, Screen 1

The “Setup” field is output to the console when it is first attached.

The “Direct Cursor Address” field is coded to tell THEOS how to position the cursor on consoles using this class code. Specify the string of character that, when output to the console, positions the cursor to a specific location. Where the line or column number would be specified, use one of the following codes:

Row Number	Column Number
128 (0x80) Absolute	129 (0x81) Absolute
130 (0x82) Standard	131 (0x83) Standard
132 (0x84) ASCII	133 (0x85) ASCII

The meanings of these three types of coding are:

- Absolute** The value substituted is the row or column number. That is, positioning to row five is accomplished by outputting a character whose value is five.
- Standard** The value substituted is the row or column number plus 32. To position to row five, a character is output whose value is 37.
- ASCII** The ASCII digit characters are output for the row and column. To position to row 12, the characters ‘1’ and ‘2’ are output.

Cursor addressing, as used by THEOS class codes, is always number base 0 with the origin in the upper left corner.

The second screen shows display attribute fields.

```

THEOS® 32 Classgen

CONSOLE Output attributes for class: 90 PcTerm U.S.A.

Reverse video on..... 0xBD ESC G 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Reverse video off..... 0xBD ESC G 0x80 '0' 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Underline on..... 0xBD ESC G 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Underline off..... 0xBD ESC G 0x80 '0' 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Blink on..... 0xBD ESC G 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Blink off..... 0xBD ESC G 0x80 '0' 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Half intensity on..... 0xBD ESC G 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Half intensity off..... 0xBD ESC G 0x80 '0' 0x81 @ 0x84 '4' 0x88 '2' 0x90 '8'
Format mode on..... ESC &
Format mode off..... ESC 0x27
Cursor off..... ESC . '0'
Cursor on..... ESC . '1'

```

Figure 2: ClassGen, Consoles, Screen 2

Most terminals use one of two methods for enabling various video display attributes: bit-mapped and ANSI X3.64.

If a particular attribute uses one of these methods, the first character of the definition for the field must be a special code that the class code can recognize as a type code. It is a single character whose value reflects the binary code resulting from the following definition:

Bit	Meaning
7	Must be “on.”
6	Requires backspace for multiple attributes.
5	“On” means attributes are bit-mapped. “Off” means attributes use ANSI X3.64 coding.
4	Coding applies to UNDERLINE attribute.
3	Coding applies to BLINK attribute.
2	Coding applies to REVERSE VIDEO attribute.
1	Coding applies to FORMAT attribute.
0	Coding applies to HALF INTENSITY attribute.

For instance, a terminal using bit-maps that supports all of the attributes and doesn't require a backspace for multiple attribute encoding would use a code of 191 (0xBF).

Bit-mapped Attribute Coding

A terminal using bit-mapped coding should have the lead-in character described above (with bit position 5 on) followed by the character string that must be output to the terminal to enable the attribute. In this character string, where the specific attribute is specified, precede the attribute code with a special code derived from the following table:

Bit	Meaning
7	Must be "on."
6	Not used.
5	Not used.
4	Character following applies to UNDERLINE attribute.
3	Character following applies to BLINK attribute.
2	Character following applies to REVERSE VIDEO attribute.
1	Character following applies to FORMAT attribute.
0	Character following applies to HALF INTENSITY attribute.

For example, a terminal using bit-mapped coding for reverse video, blink and underline attributes might be coded as:

```
252 ESC G 132 '4' 136 '2' 144 '8'
```

where:

252	Specifies that this attribute requires special processing, uses a display position, is bit-mapped, and applies to underline, blink and reverse video.
ESC G	This is the lead-in used by the terminal for attribute coding.
132 '4'	Outputs the character "4" when reverse video is being selected.
136 '2'	Outputs the character "2" when blink is being selected.
144 '8'	Outputs the character "8" when underline is being selected.

Each attribute "on" field must be defined such that all of the attribute enable codes are specified, and that each attribute "off" field be defined

such that all attributes are turned off, followed by the codes that re-enable the attributes that might be still on. The example shown in Figure 2: “ClassGen, Consoles, Screen 2” on page 228 shows an example of this technique.

ANSI X3.64 Attribute Coding

A terminal using ANSI X3.64 attribute coding requires a string looking like:

```
ESC [ '7' m
```

This specific string would enable the reverse video attribute. To turn off this attribute a string looking like the following is used:

```
ESC [ '2' '7' m
```

Because ANSI encoding is common, THEOS class codes have built-in capability to output the proper character strings for the various video attributes supported by THEOS.

A terminal using ANSI X3.64 coding has the lead-in character described above followed by a 255 (0xFF). Make sure that bit position 5 is off in the lead-in character. For instance:

```
0xBD 0xFF
```

This tells the class code that the terminal uses ANSI coding for underlining, blink, reverse video and half- intensity.

The third screen shows the fields used for screen editing commands.

THEOS® 32 Classgen

CONSOLE Output attributes for class: 90 PcTerm U.S.A.

```

Insert character..... ESC Q
Delete character..... ESC W
Insert line..... ESC E
Delete line..... ESC R
Clear screen..... ESC *
Erase unprotected..... ESC & ESC ; ESC 0x27
Erase to end of line.... ESC T
Erase to end of screen.. ESC Y
Monitor mode on..... ESC U
Monitor mode off..... ESC u
Send to status on..... ESC g ESC f
Send to status off..... CR
Enable Slave Printer.... ESC `
Disable Slave Printer... ESC a
Set alpha color..... ESC ETX 0x88 0x90 0xA0 0xC0

```

Figure 3: ClassGen, Consoles, Screen 3

These fields are fairly obvious in their nature and the manual for your terminal should specify exactly what characters to use for each of these fields.

There are three fields at the bottom of this screen that are not so obvious. The “Enable Slave Printer” and “Disable Slave Printer” fields, refer to the commands that will tell the terminal to activate its auxiliary or secondary port that should be connected to a local printer. If the terminal supports this concept, find the commands in the terminal manual that instruct the terminal to pass subsequent data to the auxiliary port and to not display them on the terminal screen. This will probably be identified as “transparent” mode.

The last field, “Set alpha color,” is used on color display terminals. If the terminal does not support colors, leave this field blank.

The definition in this field tells the class code how to change colors on the terminal. Specify the command string required by the terminal for changing colors. In this command string substitute the following codes in the appropriate character positions:

Code	Meaning
136 or 0x88	Foreground color
144 or 0x90	Background color
160 or 0xA0	Reverse Video foreground color
192 or 0xC0	Reverse Video background color

When a color change is requested, THEOS changes these codes to the corresponding color code. The color codes used by THEOS are:

Code	Color	Code	Color
0	Black	4	Red
1	Blue	5	Magenta
2	Green	6	Yellow
3	Cyan	7	White

If your terminal does not use these same numbers or characters for its colors there is a mechanism for setting the “color palette.” If the first character of the “Set Alpha Color” field is a 255 (0xFF), then the next eight characters are interpreted as the color palette definition. This color palette must be specified in the same sequence that THEOS uses colors.

For instance, if your terminal uses letters for its color codes: ‘R’ for red, ‘Y’ for yellow, ‘G’ for green, ‘W’ for white, ‘C’ for cyan, ‘B’ for blue, ‘M’ for magenta and ‘N’ for black, the color palette is defined with:

```
255 N B G C R M Y W
```

The following definition is taken from class code 180. This class code is used on color PC Terminals. It has a color palette that matches the one used by THEOS but uses ASCII digits instead of the default numeric values for the colors codes.

```
0xFF '0' '1' '2' '3' '4' '5' '6' '7' ESC / 0x88 0x90 0xA0 0xC0
```


Screens four and five show the fields defining the keyboard input key translation values.

THEOS® 32 Classgen

CONSOLE Input function keys for class: 90 PcTerm U.S.A.

Up.....	0xFF H
Down.....	0xFF P
Left.....	0xFF K
Right.....	0xFF M
Escape.....	ESC
Top.....	0xFF G
Bottom.....	0xFF O
Delete character.....	0xFF S
Delete left character...	DEL
Page forward.....	0xFF Q
Page backward.....	0xFF I
Word right.....	0xFF 0xD2
Word left.....	0xFF 0xD3
Search forward.....	0xFF 0xC1
Search backward.....	0xFF 0xD1

Figure 4: ClassGen, Consoles, Screen 4

For each of the fields, specify the characters or codes transmitted by the terminal when the key is pressed that corresponds to the function.

For instance, if the terminal manual indicates that the Home key sends ESC, [, H then define the “Home” field as ESC ['H'.

If the terminal’s keyboard transmits scan codes, define the field starting with the value 255 followed by the scan code value of the key or key combination. (See above figure.)

Screens six and seven show the line graphics character definitions for the terminal.

```

THEOS® 32 Classgen

CONSOLE Line Drawing Graphics for class: 90 PcTerm U.S.A.

Upper left corner..... 0xDA
Upper right corner..... 0xBF
Lower right corner..... 0xD9
Lower left corner..... 0xC0
Four way intersection... 0xC5
Left intersection..... 0xC3
Right intersection..... 0xB4
Up intersection..... 0xC2
Down intersection..... 0xC1
Horizontal..... 0xC4
Vertical..... 0xB3
Round upper left..... 0xDA
Round upper right..... 0xBF
Round lower right..... 0xD9
Round lower left..... 0xC0

```

Figure 5: ClassGen, Consoles, Screen 6

You should define a code for each of these fields even if the terminal does not support a specific line-drawing character or even if it doesn't support any line-drawing characters.

Many programs assume that the console can display line-drawing characters and do not make any allowances for consoles that do not. For fields not supported by the terminal, use either some other line-drawing character that appears similar to the one desired or use a regular text character.

In the above example notice that the rounded corner characters (at the bottom of the display) are the same codes as the square corner characters (at the top of the display). Most terminals do not support rounded corners so the square corners should be substituted.

If the terminal does not have line-drawing graphics, leave the various fields blank. THEOS will use regular text characters for any fields that are undefined. For instance, a '+' character for all corners and the four-way intersection, a '|' for vertical and left and right intersections, and a '-' for horizontal and up and down intersections.

Screens eight, nine and ten show the fields defining the codes used to display international characters.

THEOS® 32 Classgen

CONSOLE Output International Symbols for class: 90 PcTerm U.S.A.

```

Uppercase A dieresis.... 0x8E
Lowercase a dieresis.... 0x84
Lowercase a circumflex.. 0x83
Lowercase a grave accent 0x85
Lowercase a acute accent 0xA0
Uppercase E acute accent 0x90
Lowercase e dieresis.... 0x89
Lowercase e circumflex.. 0x88
Lowercase e grave accent 0x8A
Lowercase e acute accent 0x82
Lowercase i dieresis.... 0x8B
Lowercase i circumflex.. 0x8C
Lowercase i grave accent 0x8D
Lowercase i acute accent 0xA1
Uppercase O dieresis.... 0x99
Lowercase o dieresis.... 0x94
Lowercase o circumflex.. 0x93
Lowercase o grave accent 0x95
Lowercase o acute accent 0xA2

```

Figure 6: ClassGen, Consoles, Screen 8

Symbol character terminology:

- Dieresis** Another word for umlaut (¨) which is two dots on top of the character. Example: Ä, Ë, ä, i.
- Circumflex** A diacritical mark that looks like an upside-down v (^) over the character. Example: â, î, ô.
- Grave accent** A diacritical mark that looks like a backward quote (`) over the character. Example: à, ì, ù, Ò.
- Acute accent** A diacritical mark that looks like an apostrophe (´) over the character. Example: Í, Ú, í, ó.
- Cedilla** A diacritical mark that looks like a tail (,) under the character. Example: ç, Ç.

Tilde A diacritical mark that looks like a “squiggle” (~) over the character. Example: Ã, Ñ, ñ, õ.

Diphthong A ligature or letter combination. Example: Æ, æ.

Bolle A diacritical mark that looks like a small circle (°) over the character. Example: Å, å.

Eszet The German “ss” letter combination. Example: ß.

The last three screens show the definitions of the codes received from the terminal when the keyboard generates the same international characters and symbols.

Printer Class Codes

ClassGen also uses several screens to define class codes for printers. These screens group the definition fields into several categories:

1. Printer name, initialization and display attributes.
2. Line-drawing graphics. (Two screens)
3. International symbols. (Four screens)

A printer class code uses fewer screens than a console class code mainly because there is no keyboard to define.

Refer to the “Notes” section for information about navigating between these screens, moving from field-to-field and inputting special character codes.

■ Printer Class Code Maintenance Screens

The following shows the fields used in each of the maintenance screens used for printer class codes.

THEOS® 32 Classgen

PRINTER Attributes for class: 135 HP LaserJet II & III

Printer name..... HP LaserJet II & III

Setup..... ESC E ESC ('1' '0' U

Boldface on..... ESC (s '1' B

Boldface off..... ESC (s '0' B

Underline on..... ESC & d '0' D

Underline off..... ESC & d @

Italics on..... ESC (s '1' S

Italics off..... ESC (s '0' S

Secondary color on.....

Secondary color off.....

Compress on..... ESC (s '1' '6' . '6' '6' H

Compress off..... ESC (s '1' '0' H

Double wide on.....

Double wide off.....

Double high on.....

Double high off.....

Commands

Figure 7: ClassGen, Printers, Screen 1

Of interest in this first screen is the “Setup” field. The content of this field is sent to the printer when it is first attached. For spooled printers, this string is also sent at the beginning of each report printed by the printer spooler.

The remaining fields on this screen and the other screens that follow are similar to those described in Figure 2 on page 228 and Figure 5 on page 234.

Notes

To move around a screen use the **↑** or **↓** keys, or merely enter **Enter ↵** to advance to the next field.

The **PageDown** and **PageUp** keys move from screen to screen while the **Home** and **End** keys move to the first or last screens.

To save any changes made and then exit, use the **F10** key. The **F9** or **Esc** keys exit without saving your changes.

Entering Characters

Each character in a field is separated by a space from the next character in the field.

All displayable characters except for digit characters are entered as plain text. Digit characters are enclosed in a pair of single quotation marks.

Entering Control Characters

Most fields in a class code definition will require that you indicate that a control character is part of the definition. Since entry of a control character will be treated by THEOS as a command rather than data, you will have to specify the control character value in another way.

Control characters are specified with either the numerical value or by their ASCII character name. The numerical value may be specified in decimal or hexadecimal.

Char	ASCII	Dec	Hex	Char	ASCII	Dec	Hex
^@	NUL	0	0x0	^P	DLE	16	0x10
^A	SOH	1	0x1	^Q	DC1	17	0x11
^B	STX	2	0x2	^R	DC2	18	0x12
^C	ETX	3	0x3	^S	DC3	19	0x13
^D	EOT	4	0x4	^T	DC4	20	0x14
^E	ENQ	5	0x5	^U	NAK	21	0x15
^F	ACK	6	0x6	^V	SYN	22	0x16
^G	BEL	7	0x7	^W	ETB	23	0x17
^H	BS	8	0x8	^X	CAN	24	0x18
^I	HT	9	0x9	^Y	EM	25	0x19
^J	LF	10	0xA	^Z	SUB	26	0x1A
^K	VT	11	0xB	^[ESC	27	0x1B
^L	FF	12	0xC	^\ ^_	FS	28	0x1C
^M	CR	13	0xD	^]	GS	29	0x1D
^N	SO	14	0xE	^^	RS	30	0x1E
^O	SI	15	0xF	^_	US	31	0x1F
					DEL	127	0x7F

Control characters are always displayed with the ASCII name.

Example:

```
ESC &  
27 &  
ESC g ESC f  
0xFC ESC G 132 '4' 136 '2' 144 '8'
```

Restrictions The ClassGen command requires a privilege level of five and may only be executed when you are logged onto the SYSTEM account (user number 0).


See also [Attach](#), [CRT](#), [Printer](#), [Start](#), [Sysgen](#)

Clear Command

The Clear command clears the console and positions the cursor to the upper left corner.

CLEAR

Commands

Operation	The console screen is cleared and the CSI prompt is displayed on line 2.
Notes	<p>The console status line is cleared.</p> <p>The console screen may also be cleared by pressing  at the CSI prompt. The action of this key differs from the Clear command in two ways: The CSI prompt is displayed on line 1 and the status line is not cleared.</p>

Color Command

The Color command sets the default colors used on the console.

1 COLOR

2 COLOR fg bg rvfg rvbg

fg » foreground color number or name

bg » background color number or name

rvfg » reverse video foreground color number or name

rvbg » reverse video background color number or name

Operation

Mode 1—The current colors in effect for this console are displayed.

```
>color
Normal:  WHITE on BLUE
Reverse:  WHITE on RED
```

This mode also reprograms the console to use these colors and performs an “Erase to end-of-screen” operation. This operation can be useful if the displayed colors on your console are changed without THEOS’ knowledge (for instance, by ScanTerm).

Mode 2—The default colors for this console are changed. The text color is set to *fg*, the text background color is set to *bg*, reverse video text color is set to *rvfg* and reverse video text background color is set to *rvbg*.

Before exiting, a “clear to end-of-screen” is performed.

```
>color 7 3 7 1

>color
Normal:  WHITE on CYAN
Reverse:  WHITE on BLUE

>color white red

>color
Normal:  WHITE on RED
Reverse:  RED on WHITE
```

When fewer than four colors are specified the omitted colors are set accordingly:

Omit *rvbg*: The reverse video background is set to BLACK.

```
>color white blue cyan
```

Omit *rvfg*, *rvbg*: The reverse video colors are set to the reverse of the normal video colors specified.

```
>color green black
```

```
>color
```

```
Normal: GREEN on BLACK
```

```
Reverse: BLACK on GREEN
```

Omit all but *fg*: The background color is set to BLACK (unless *fg* is BLACK, in which case the background color is set to WHITE). The reverse video colors are set to the reverse of the normal video colors.

```
>color 1
```

```
>color
```

```
Normal: BLUE on BLACK
```

```
Reverse: BLACK on BLUE
```

Colors

The colors for *fg*, *bg*, *rvfg* and *rvbg* may be specified with the color number or the color name. When a color name is used, it must be completely spelled out.

Code	Name	Code	Name
0	Black	4	Red
1	Blue	5	Magenta
2	Green	6	Yellow
3	Cyan	7	White

Table 16: Color Codes & Names

Notes

The colors changed by this command only affect the display of text output after this command. Text already on the screen is not changed.

Although a program might change the color representation of the text or graphics that the program displays, the system will restore the colors to those assigned by the last usage of this command after each program exits.

Defaults

When a console is first started, the default colors used are defined by the `SYSTEM.THEOS32.COLORCFG` file. This file is maintained by the [Setup](#) COLOR command.

Restrictions The console must be color-capable and its class code must have a properly defined “Set alpha color” field.

See also [ClassGen](#), [Setup Color](#), [Sysgen](#)

Compare Command

This command compares two files and reports any differences.

- 1 COMPARE *file1 file2* (*options*
- 2 COMPARE *file* (*FILE options*
- 3 COMPARE *file drive* (*FILE options*

<i>drive</i>	»	drive code
<i>file</i>	»	file name with optional path
<i>file1</i>	»	file name with optional path; may contain wild cards
<i>file2</i>	»	file name with optional path; may contain wild cards
<i>options</i>	»	BINARY NOTYPE PRTnn TYPE

Operation

Mode 1—Compares *file1* with *file2* and reports on the differences between the two files. The minimum report is a message: “Files are equivalent.” or “Files are not equivalent.” Specific information about the differences is displayed if the **PRTnn** or **TYPE** option is used.

Mode 2—The ASCII stream file specified by *file* is used as a source for a list of file name pairs to be compared. Each line in *file* specifies the two file names that are compared. Using this mode of the command is identical to using the **Mode 1** form for each pair of files specified in *file*.

```
>list compare.files
```

```
program.source.program1:s programs.original.program1:s
program.source.program4:s programs.original.program4:s
program.source.programx:s programs.original.programx:s
```

```
>compare compare.files (file
```

```
PROGRAM.SOURCE.PROGRAM1:S PROGRAMS.ORIGINAL.PROGRAM1:S
Files are not equivalent.
```

```
PROGRAM.SOURCE.PROGRAM4:S PROGRAMS.ORIGINAL.PROGRAM4:S
Files are equivalent.
```

```
PROGRAM.SOURCE.PROGRAMX:S PROGRAMS.ORIGINAL.PROGRAMX:S
Files are not equivalent.
```

Mode 3—Similar to [Mode 2](#) except only the first file name in each line of *file* is used. The second file name is created by using the first file name and replacing the drive-code for that file with the *drive* specified on the command line.

```
>filelist sample.*:s (exec
>compare selected.exec a (file

File: /SAMPLE.FILE1:S, A
Files are equivalent.

File: /SAMPLE.FILE2:S, A
Files are not equivalent.

File: /SAMPLE.FILE3:S, A
Files are equivalent.
```

Options

BINARY Compares the two files byte-by-byte rather than line-by-line. This is a default option when either of the files is not a stream file.

NOTTYPE A default option that suppresses the display of the differences between the two files. Only a result message is displayed: “Files are equivalent.” or “Files are not equivalent.”

Use [PRTnn](#) or [TYPE](#) to override this option.

PRTnn Displays the differences between the two files on the printer. *nn* specifies which attached printer to use and is in the range of 1–16. If *nn* is omitted, PRT1 is assumed.

The format of the display depends upon the [BINARY](#) option. See “[Binary File Display](#)” on page 247 and “[ASCII File Display](#)” on page 246.

The alternate keywords PRINT and PRINTER may be used instead of PRT. As a convenience, PRINTER1 may be specified as P.

TYPE Displays the differences between the two files on the standard output device (normally the console). The format of the display depends upon the [BINARY](#) option.

**ASCII File
Display:**

When two stream files contain ASCII data and the **BINARY** option is not specified, the files are compared on a line-by-line basis. When option **PRTnn** or **TYPE** is used, the specific differences between the two files are shown with a plus (+) or minus (-) symbol to indicate the line that must be added or deleted from *file2* to make it equivalent to *file1*.

For instance, a MultiUser BASIC language program has been edited with new comments added at the beginning of the program (line numbers 1–6). Additionally, the capitalization of one word was changed at line 110.

Commands

```
>compare julian.basic julian.backup (type
1+000001      ! Program: JULIAN Compute Julian date
2+000002
3+000003      ! Programmer: John Doe
4+000004
5+000005      OPTION VERSION 1.0,"(C) 1995 by ABC Inc."
6+000006
17-000110     PRINT "Today's julian date is: ";
17+000110     PRINT "Today's Julian date is: ";
Files are not equivalent.
```

This display indicates that the files are not the same and that the second file (JULIAN.BACKUP) must have the following changes made to it to make it the same as the first file (JULIAN.BASIC):

1. Six lines must be added at the beginning of the file.
2. The 17th line (after the six lines are added) must be deleted.
3. A new 17th line must be added.

When a difference is found, the **Compare** command uses a buffer to scan ahead in the file to see if the difference is an addition to the file (a matching line is found later in the *file1*), a deletion was made (a matching line is found later in *file2*) or if a change was made. This buffer holds approximately 40 lines of text.

Binary File Display

When two non-stream files are compared or when the **BINARY** option is specified, the files are compared on a byte-by-byte basis. When option **PRTnn** or **TYPE** is used, the specific differences between the two files are shown.

For instance, a program's version number is changed, a string literal is changed and the program is recompiled. A comparison of the new program command with the prior version of the command (renamed to have a file type of OLDCMD before the recompilation) might show:

```
>compare julian.command julian.oldcmd (type
00000105 52 53
00000107 20 21
00000109 8D 8C
00000115 31 30
00005A1D 4A 6A
00005A5D 6E 61
00005A5E 6E 61
00005A5F 6D 6E
etc.
```

Each line of the display shows a location in the files, the value of that location in *file1*, followed by the value of that location in *file2*.

Defaults

NOTTYPE is a default option. **BINARY** is a default option when either of the files is not a stream file.

Return Codes

Unless an error is detected, when the two files are equivalent the return code is zero (0); otherwise the return code is one (1).

Cautions

Comparing two stream files that contain coded or binary information without using the **BINARY** option might produce erroneous results. When in doubt, always use the **BINARY** option.

Compress Command

The Compress command compresses a file and saves the compressed form in a “compression library file.”

Commands

- 1 COMPRESS *compress-file file... (options*
- 2 COMPRESS *compress-file file (FILES options*
- 3 COMPRESS *compress-file - file*
- 4 COMPRESS *compress-file*

<i>compress-file</i>	»	file name of compression file, with optional path	
<i>file</i>	»	file name of source file, with optional path; may contain wild cards	
<i>options</i>	»	DELETE	PASSWORD
		FILES	PRESERVE
		MODIFIED	SUBDIR
		NOTYPE	TYPE

Operation

Mode 1—Each *file* specified on the command line is compressed and added to the compression library file *compress-file*.

If *compress-file* does not exist, it is created. When one or more *files* already exist in *compress-file*, they are removed before being added to the end of the *compress-file*.

```
>compress old.programs original.source.*
47% original.source.addrsort
48% original.source.dflt
46% original.source.entity
46% original.source.eod1
38% original.source.eom1
47% original.source.eom2
47% original.source.eom3
47% original.source.eom4
40% original.source.eom5
...
```

Unless the **PRESERVED** option is used, each file that is compressed has its modified attribute reset. (The modified attribute is set on the *compress-file* so it will be included on the next incremental or differential archive.)

Mode 2—The files listed in *file* are compressed and copied into *compress-file*. *file* must be an ASCII stream file containing one file description per line. The `SELECTED.FILES` and `SELECTED.EXEC` files created by [FileList](#) and the `FOUND.EXEC` created by [Look](#) can be used for this specification file (see “[The EXEC and FILES Options](#)” on page 338). You may also create the specification file with an editor or application program.

For instance, [FileList](#) is used to create a list of files to be compressed:

```
>filelist a not *.basic:a not *.command:a (10/1/96 exec
```

A file now exists that lists all of the files that have been changed since 10/01/1996. The following command will compress these files:

```
>compress old.files selected.exec (file preserve
91% dat.acc
14% dat.def
82% dat.inccod
85% dat.ass
92% dat.def
74% frn.anldat
...
```

Mode 3—Similar to Mode 1 except that each *file* preceded by a minus sign is deleted from the *compress-file* instead of being added to it.

```
>compress old.files -retail.tickets
retail.tickets removed.
```

Mode 4—No files are compressed with this mode. Instead, the list of files already contained in *compress-file* are displayed on the standard output device.

```
>compress old.files
```

<u>File name</u>	<u>Date</u>	<u>Time</u>	<u>Size</u>	<u>Rate</u>
dat.acc	12May88	11:41	1,024	91%
dat.def	31Dec69	16:00	1,024	14%
dat.inccod	31Jul90	19:31	1,804	82%
dat.ass	13Apr88	19:07	1,668	85%
dat.def	15Sep90	18:49	1,024	92%
frn.anldat	28Mar87	08:39	23,650	74%
...				

The “Date,” “Time” and “Size” columns refer to the time-stamp and size of the original, uncompressed form of the file. “Rate” refers to the compression ratio of the file.

Options **DELETE** Tells Compress to delete *file* after adding it to *compress-file*. (If *file* is preceded by a minus sign, as in Mode 3, it is removed from the *compress-file* but it is not deleted.)

FILES Invokes [Mode 2](#) of the Compress command.

MODIFIED Specifies that only files that have their modified attribute set are to be Compressed.

NOTYPE Tells Compress to not display the results of each file compressed. The general result message (the “nn files compressed.” message prior to exiting Compress) is also suppressed with this option.

```
>compress old.files myprog.basic (notype
>
```

To disable this option use the [TYPE](#) option.

PASSWORD A password is specified following the PASSWORD option keyword. All of the files added to the *compress-file* are encoded with this password and they cannot be expanded without specifying the same password.

```
>compress private.files *.text (notype password aardvark
```

PRESERVED Tells Compress to not clear the modified attribute from a file that is compressed. Without this option, every file that is compressed has its modified attribute cleared.

SUBDIR Tells Compress that the full path of the file is to be used when saving the image of *file*. When this option is not used, only the file name is saved, not its path information.

By using this option, the [Expand](#) command has sufficient information to extract and restore the file to its original location.

TYPE A default option that tells Compress to display the results of each file compression on the standard output device. This display can be redirected.

```
>compress old.files selected.exec (file
91% dat.acc
14% dat.def
82% dat.inccod
...
```

To disable this option use the [NOTYPE](#) option.

Notes

compress-file is referred to as a compression library file because it is a single file containing the compressed versions of one or more files. It is not a library in the normal usage of the term library.

When *compress-file* is specified as a typeless file specification (no period and no file-type specified), the default file type of Compress is used for the *compress-file*.

When Mode 1 or 2 is used to add files to an existing *compress-file*, any previous versions of the files with the same name are removed and then the current version is added to the end of the *compress-file*.

Typically, ASCII text files are compressed to 40% to 60% of their original size; binary files (programs and formatted data files) are compressed to 50% to 70% of their original size.

Defaults

[TYPE](#) is a default option.

See also

[Archive](#), [Backup](#), [CopyFile](#), [Expand](#), [FileType](#), [TBackup](#)

CopyFile Command

The CopyFile command copies a file from one location to another.

1	<u>COPYFILE</u>	<i>from-file to-file</i> (<i>options</i>			
2	<u>COPYFILE</u>	<i>from-file to-device</i> (<i>options</i>			
3	<u>COPYFILE</u>	<i>from-drive to-drive</i> (<i>options</i>			
4	<u>COPYFILE</u>	<i>file</i> (<u>FILES</u> <i>options</i>			
5	<u>COPYFILE</u>	<i>file from-drive to-device</i> (<u>FILES</u> <i>options</i>			
6	<u>COPYFILE</u>	(<i>options</i>			

<i>file</i>	»	file name of specifications file, with optional path			
<i>from-file</i>	»	file name of source file, with optional path; may contain wild cards			
<i>to-file</i>	»	file name of destination file, with optional path; may contain wild cards			
<i>from-drive</i>	»	disk drive letter for source files			
<i>to-device</i>	»	<i>to-drive</i> COM <i>nn</i> CON PRT <i>nn</i> TAP <i>n</i>			
<i>to-drive</i>	»	disk drive letter for destination files			
<i>options</i>	»	APPEND BYREC EOFSENT EXPORT FILES FOR <i>nnn</i> FRKEY <i>kkk</i> FRLABEL <i>lll</i> FROM <i>nnn</i> LOWCASE	MODIFIED MOUNT NEWDATE NEWFILE NOQUERY NOSORT NOTYPE NOVERIFY OLDDATE OLDER	OLDFILE PRESERVE PUBLIC QUERY REPLACE SORT SPECS SUBDIR TOKEY <i>kkk</i> TOLABEL <i>lll</i>	TRANS TRUNCATE <i>nnn</i> TYPE UPCASE VERIFY <i>date1</i> <i>date2</i>

Operation

Mode 1—Copies *from-file* to the *to-file*.

```
>copyfile one.file another.file
```

```
>copyfile *.data =.datacopy
```

Refer to “[Restrictions](#)” on page 268 for a description of general restrictions on file ownership, public files and copying files from or to subdirectories.

The *from-file* and *to-file* may contain wild-card specifications. See “[Wild Card Specifications](#)” on page 136.

Mode 2—Copies *from-file* to *to-device*. When *to-device* is a disk-drive label, the destination file will have the same file name as the source file.

```
>copyfile *.data:s f
```

```
>copyfile *.data:s =.:f
```

The above two commands produce identical results.

This mode is also used when you want to copy a file to a communications port, console, printer or to a tape volume.

```
>copyfile master.index =.:tape (norewind
```

```
>copyfile text.file com2
```

Only stream files may be copied to devices other than a disk.

Mode 3—This is a shorthand method for specifying that you want all files copied from one drive to another.

```
>copyfile s f
```

```
>copyfile *.*:s =.:f
```

The above two commands are equivalent.

Remember, unless you use the [PUBLIC](#) option, only files owned by the current account may be copied. Unless the [SUBDIR](#) option is used, only files in a single directory are copied.

This mode can also be used to copy stream data from one communications port to another, or from or to a tape drive.

```
>copyfile :com4 :com8
```

```
>copyfile :com3 sample.file:tape2
```

Mode 4—*file* is an ASCII stream file containing two file descriptions per line. The first file description in the line is treated as a *from-file* and the second file description is the *to-file*. For each line in *file*, a [Mode 1](#) or [Mode 2](#) CopyFile is performed, as appropriate.

This mode of the CopyFile command is convenient when one or more sets of files are repetitively copied. Merely edit a file containing file description pairs, such as:

```
>edit daily.files
customer.master:s /backup/customer.master:s
customer.history:s /backup/customer.history:s
customer.invoices:s /backup/customer.invoices:s
general.ledger.*:s /backup/=.*=s
check.*:s /backup/=.=
/programs/program.source.*:s a

>copyfile daily.files (file replace noquery notype
```

By using some of the file selection options in addition to this file specifications file, the copy operation can be even more powerful. For instance:

```
>copyfile daily.files (file replace noquery notype older
```

Here, the [OLDER](#) option restricts the copy operation so that only files that have been changed since the last copy are copied this time. Similarly:

```
>copyfile daily.files (file replace noquery notype modified
```

This command copies only those files that have their modified status set.

Mode 5—Similar to [Mode 4](#), *file* is an ASCII stream file but it contains one file description per line. This file description is the source file specification. If it contains a drive code, that drive code is ignored and the *from-drive* specified on the command line is used instead.

For each line in *file*, a [Mode 2](#) CopyFile is performed. For instance:

```
>edit daily.files
customer.master
customer.history
customer.invoices
general.ledger.*
check.*
/programs/program.source.*

>copyfile daily.files a b (file replace noquery notype noq
```

is equivalent to:

```
>copyfile customer.master:a b (replace notype noq
>copyfile customer.history:a b (replace notype noq
>copyfile customer.invoices:a b (replace notype noq
>copyfile general.ledger.*:a b (replace notype noq
>copyfile check.*:a b (replace notype noq
>copyfile /programs/program.source.*:a b (replace notype noq
```

Mode 6—This is the interactive mode of the CopyFile command. Because no files are specified on the command line, you are prompted to enter the file names to copy:

```
>copyfile (noquery
Enter file name list, terminate with empty line.
?sample.file* j
"SAMPLE.FILE1:S" copied to "SAMPLE.FILE1:J".
"SAMPLE.FILE2:S" copied to "SAMPLE.FILE2:J".
?sample.file* /copies/samples/=.=:j
"SAMPLE.FILE1:S" copied to "/COPIES/SAMPLES/SAMPLE.FILE1:J".
"SAMPLE.FILE2:S" copied to "/COPIES/SAMPLES/SAMPLE.FILE2:J".
?
```

Options

- APPEND** Applies only to stream files. This option tells CopyFile that, if the destination file already exists and it is a stream file, the source file is appended to the end of the existing destination file.
- BYREC** Applies only to indexed, keyed and direct files. Indicates that, if the destination file already exists, the records in the source file are added to the destination file.
- If the destination file does not exist, a new, empty file is created and the records in the source file are copied to the new destination file one record at a time. Although this copy process is slower than a file copy, the destination is a “clean” version of the source file because no deleted records are copied and the tree structure of an indexed or keyed file is built from scratch.
- EOFSENT** Indicates that the default end-of-file sentinel character is not the default EOT character (^D). You will be prompted to enter the end-of-file sentinel character before copying begins. A maximum of four characters may be entered as this sentinel.

Use this option only when the *from-file* is not a disk or tape file. That is, when the source file is coming from the console or a communications port.

EXPORT

from-file is “exported” to *to-file*. This option is ignored when the *from-file*’s organization is not direct, indexed or keyed. The purpose of the EXPORT option is to convert a file so that it can be used in an environment that does not support THEOS formatted records, or direct, indexed or keyed organization files.

A file is exported by copying the records in the *from-file* to the stream file *to-file*, changing the formatted fields in the record to quoted string fields. The key field of each record in *from-file* is output as the first field in *to-file*.

For instance, *from-file* is an indexed file with records created by the BASIC language statement:

```
WRITE #1,KEY$:STRING$,PHRASE$,INTEGER%,FLOAT
```

Each input record looks like:

```
Key: String,"Phrase needing quotes",1,2.345
```

The resulting record in *to-file* looks like:

```
"Key",String,"Phrase needing quotes",1,2.345
```

and can be read with the BASIC language statement:

```
INPUT #1:KEY$,STRING$,PHRASE$,INTEGER%,FLOAT
```

Each field is comma-separated from other fields. It is enclosed in quotes if the field is a string containing embedded spaces or quotes.

The EXPORT option implies the [BYREC](#) option and can be used with other record selection options such as [FOR](#), [FRKEY](#), [FROM](#) and [TOKEY](#). The [SPECS](#) and [TRANS](#) options are ignored when specified with EXPORT.

FILES

Indicates that [Mode 4](#) or [Mode 5](#) of the CopyFile command is used. The *file* specifies an ASCII stream file with each record in the file specifying a file to be copied. The file name specifications in this file may include the path to the file.

For a [Mode 4](#) request, the records in *file* specify both the source file name and the destination file name. In a [Mode 5](#) request, the records need only specify the source file name.

FOR

Indicates that only *nnn* records of the source file are copied to the destination file. The [FRKEY](#), [FRLABEL](#) or [FROM](#) option must be used or the FOR option will be ignored (the entire file is copied).

```
>copyfile last.letter customer.= (from 1 for 8
```

This command copies the first eight lines of one file to another file.

FRKEY

Used with indexed and keyed files to specify the first record to copy.

kkk represents the key of the first record to copy. Only records whose keys are greater than or equal to this key are copied. Can be used with the [FOR](#) and [TOKEY](#) options to limit the number of records copied after this first record is located.

```
>copy customer.master new.master (frkey "THEOS"
```

This command copies all of the records in the CUSTOMER.MASTER file whose keys start with text that is greater than or equal to "THEOS."

FRLABEL

Used with ASCII stream files to specify the first record to copy.

lll represents the starting text of the first record to copy. No records are copied until a record starting with this text is found. When found, that record is copied to the destination file along with the records that follow it in the source file. Can be used with the [FOR](#) and [TOLABEL](#) options to limit the number of records copied after this first record is located.

```
>copy memo.model new.memo (frlabel "cc:" append
```

This command copies the last lines of MEMO.MODEL, beginning with the line starting with "cc:".

FROM

Used with ASCII stream files to specify the first record to copy.

nnn represents the record number of the first record to copy. No records are copied until the *nnn*th record is found. When found, that record is copied to the destination file along with

the records that follow it in the source file. Can be used with the [FOR](#) and [TOLABEL](#) options to limit the number of records copied after this first record is located.

LOWCASE Used with ASCII stream files to translate all uppercase characters to their lowercase equivalents. All non-alphabetic characters are copied without any translation.

Most of the multinational characters (Ä, É, Ñ, *etc.*) are translated with this option. See “[Lowcase](#)” on page [397](#) for more information.

MODIFIED Copies the source file only if its modified attribute is set. After the file is copied, the modified attribute of the source file is reset (unless the [PRESERVE](#) option is also used). This option provides a method of creating a set of incremental copies of files. For instance, if the following command is executed every day:

```
>copy *.data:s f (modified
```

Then each day a copy is made of all files that have been updated or changed since the last time the copy was performed or the modified attributes were cleared. See “[Change](#)” on page [216](#).

See also “[Archive](#)” on page [168](#).

MOUNT Specifies that CopyFile is to be loaded and initialized but, before the source or destination files are located, you want to be allowed to change the disks.

```
>copyfile my.data:s your.data:f (replace mount
```

```
Mount volumes now (Y/N)?
```

Because the CopyFile command is already loaded before this question is asked, you may change the S drive volume. If the S drive is changed, be sure that you are in single-user mode.

NEWDATE The destination file’s directory entry is updated with the current date and time. This is a default option when any other option is used that causes the destination file to differ from the source file, such as: [APPEND](#), [BYREC](#), [EXPORT](#), [FOR](#), [FRKEY](#), [FRLABEL](#), [FROM](#), [LOWCASE](#), [SPECS](#), [TOKEY](#), [TOLABEL](#), [TRANS](#), [TRUNCATE](#) and [UPCASE](#).

To disable this option use the [OLDDATE](#) option.

NEWFILE Indicates that CopyFile should only attempt to copy files if the destination file name does not already exist. Note that, if [QUERY](#) is used, you are not queried about files if the destination file name already exists.

This is a default option. To disable this option use the [OLDFILE](#) or the [REPLACE](#) options.

NOQUERY Tells CopyFile to not ask for confirmation before copying each file. This is a default option when wild cards are not used.

```
>copyfile gl.* f (noq
"GL.MASTER:S" copied to "GL.MASTER:F".
"GL.JOURNAL:S" copied to "GL.JOURNAL:F".
"GL.HISTORY:S" copied to "GL.HISTORY:F".
```

To disable this option use the [QUERY](#) option.

NOSORT A default option that tells CopyFile to copy the files matching the source file specifications in the order that the file names are found on disk.

To disable this option use the [SORT](#) option.

NOTYPE Tells CopyFile to not display the results of each file copy on the standard output device. The general result message (the “nn files copied.” message prior to exiting CopyFile) is also suppressed with this option.

```
>copyfile gl.* f (not
Ok to copy "GL.MASTER:S" (Yes,No,All) Y
Ok to copy "GL.JOURNAL:S" (Yes,No,All) Y
Ok to copy "GL.HISTORY:S" (Yes,No,All) Y
```

To disable this option use the [TYPE](#) option.

NOVERIFY A default option that tells CopyFile to not verify that the copy was done successfully. Normally, the hardware verification that is always done is more than sufficient to assure that the copy is correct.

To disable this option use the [VERIFY](#) option.

OLDDATE Each destination file’s directory entry is set to the source file’s last change date. This is a default option when the destination is an exact copy of the source file.

To disable this option use the [NEWDATE](#) option.

OLDER Copies the source file to the destination file only if the destination file's last change date is older than the source file's or if the destination file does not exist. The [REPLACE](#) option is implied by this option. Note that, if [QUERY](#) is used, you are not queried about files if the destination file is newer.

OLDFILE Indicates that CopyFile should attempt to copy a file only if the destination file name already exists. This option implies a [REPLACE](#) option. Note that, if [QUERY](#) is used, you are not queried about files if the destination file name does not exist.

Note that only the destination file name is checked. It could be a totally different file. For instance, an existing command program could be replaced by a stream file or indexed file.

To disable this option use the [NEWFILE](#) option.

PRESERVE Used with the [MODIFIED](#) option to specify that source file's modified attribute is not to be reset. This option provides a method of creating a set of differential copies of files. For instance, if the following command is executed every day:

```
>copy *.data:s f (modified preserve
```

Then each day a copy is made of all files that have been updated or changed since the last time the modified attributes were cleared. See "[Change](#)" on page 216.

See also "[Archive](#)" on page 168.

PUBLIC Allows publicly owned files to be copied. Normally, when you are logged onto a private account, only files owned by the current account are searched. With this option, files owned by the SYSTEM account may be copied as long as their shared access protections allow it. See "[File Access Protection](#)" on page 143.

QUERY Tells CopyFile to "query" or ask if each file matching the source file specifications is to be copied. This is a default option when wild cards are used.

```
>copy *.data i
Ok to copy "CUSTOMER.DATA:S" (Yes,No,All)
```

When the "Ok to copy" question is asked you may respond with a **[Y]** for yes, **[N]** for no or **[A]** for all. Responding with **[A]** means

yes to this file and all remaining files are included without being queried. Respond with **[Esc]** to cancel the copy operation.

To disable this option use the **NOQUERY** option.

REPLACE Allows a file to be copied even if it already exists on the destination drive. Normally, when the destination file name already exists, CopyFile will not perform the copy. This option tells CopyFile that if it already exists it should erase the existing file and replace it with a copy of the source file.

If the destination file does not exist, this option has no effect.

```
>copy *.data:s f (replace noquery
"HISTORY.DATA:S" replaces "HISTORY.DATA:F".
"CUSTOMER.DATA:S" replaces "CUSTOMER.DATA:F".
"NEW.DATA:S" copied to "NEW.DATA:F".
3 files copied.
```

The **REPLACE** option is implied by both the **OLDER** and **OLDFILE** options. However, the **OLDFILE** option will not copy a file unless the destination file name already exists.

SORT Tells CopyFile to find all of the files that match the source file specifications and to sort the file names in ascending alphabetical order. When CopyFile has created this sorted list of possible file names to copy, it begins the normal copy operation, using the **QUERY** option if it is in effect.

```
>copy *.data:s f (replace noquery sort
"CUSTOMER.DATA:S" replaces "CUSTOMER.DATA:F".
"HISTORY.DATA:S" replaces "HISTORY.DATA:F".
"NEW.DATA:S" copied to "NEW.DATA:F".
3 files copied.
```

Compare this example display with the example used in the **REPLACE** option description.

SPECS This option allows the manipulation of each record in a stream file. With this option you can omit portions of each record, add text to each record and rearrange the contents of the records.

For a description of this option's usage, see “[Copying with Field Specifications](#)” on page 264.

SUBDIR

Indicates that all subdirectories in the source file directory are to be copied to the destination drive. If a subdirectory does not exist on the destination drive, it is automatically created.

```
>copy /art/*.*.*:s j (subdir noq
"/ART/ARTICLE.EXE:S" copied to "ARTICLE.EXEC:J".
"/ART/GENERAL/PROGRAM.NOTES.CONVERSN:S" copied
to "/GENERAL/PROGRAM.NOTES.CONVERSN:J".
...
```

In this example all of the files in /ART directory of the S drive are copied to the J drive. When a subdirectory of /ART is encountered, it is copied along with all files in that subdirectory. If a subdirectory contains a subdirectory, its files will also be copied, and so on.

If the source path contains subdirectories and the SUBDIR option is not used, then the subdirectories and the files in those subdirectories are not copied.

The [SORT](#) option is a default when SUBDIR is specified.

TOKEY

Used with indexed and keyed files to specify the last record to copy. Can be used with the [FRKEY](#) option to limit the number of records copied before this last record is located.

kkk represents the key of the last record to copy. Only records whose keys are less than or equal to this key are copied.

```
>copy customer.master new.master (tokey "THEOS"
```

This command copies all of the records in the CUSTOMER.MASTER file whose keys start with text that is less than or equal to "THEOS."

TOLABEL

Used with ASCII stream files to specify the last record to copy. Can be used with the [FRLABEL](#) option to limit the number of records copied before this last record is located.

lll represents the starting text of the last record to copy. Records are copied until a record starting with this text is found and, when found, that record is copied to the destination file.

```
>copy memo.model new.memo (tolabel "Re:"
```

TRANS This option allows the manipulation of each record in a stream file. With this option you can translate one or more characters in each input record to a different character in the output record.

For a description of this option's usage see "[Copying with Character Translation](#)" on page 266.

TRUNCATE Used with stream files to truncate each record in the destination file to *nnn* characters.

TYPE A default option that tells CopyFile to display the results of each file copy on the standard output device. This display can be redirected.

```
>copyfile *.*:s j (noquery
"/DEVELOPE.EXEC:S" copied to "DEVELOPE.EXEC:J".
"/GRAPH.SAVE1:S" copied to "GRAPH.SAVE1:J".
"/GRAPH.DATA:S" copied to "GRAPH.DATA:J".
"/GRAPH.START:S" copied to "GRAPH.START:J".
4 files copied.
```

To disable this option use the [NOTYPE](#) option.

UPCASE Used with ASCII stream files to translate all lowercase characters to their uppercase equivalents. All non-alphabetic characters are copied without any translation.

Most of the multinational characters (Ä, É, Ñ, *etc.*) are translated with this option. See "[Uppcase](#)" on page 585 for more information.

VERIFY Tells CopyFile to verify that the copy was made correctly by performing a read after each block of the file is written. The data read from the destination file is compared with the data that was written to the file at that location. A mismatch causes CopyFile to retry the write operation.

date1 Copies a file only if the source file's last change date is greater than or equal to this date. That is, if the source file was changed on or after this date.

This option may be used with the [date2](#) option.

```
>copy *.*:s f (10/15/96
```

The above command will copy only those files that have been created or changed since October 14, 1996.

date2 Copies a file only if the source file's last change date is less than or equal to this date. That is, if the source file was changed or before this date. May only be specified by first specifying the *date1* option.

```
>copy *.*:s f (10/15/96 10/30/96
```

This command copies only those files that have been created or changed since October 14, 1996, but not any files that were created or changed after October 30, 1996.

To specify a *date2* when you don't care about *date1*, use a date of 1/1/86 for the *date1* option. This is the earliest date maintained by the THEOS file system.

```
>copy *.*:s f (1/1/86 11/20/96
```

Here, since the *date1* specification is 1/1/86, only files created or changed prior to November 21, 1996, are copied.

Copying Stream Files

Many of the CopyFile options apply only to the copying of stream or sequential organization files. With stream files you can add information to the end of an existing file by using the **APPEND** option, change the case made of text in each record with the **LOWCASE** and **UPCASE** options, rearrange the contents in each record with the **SPECS** option or translate characters in each record with the **TRANS** option.

Stream files may also be copied to devices other than disks. For instance:

```
>copy system.theos32.devnames prt1
>copy private.exec con
```

If the file is not a stream file and you attempt to copy it to a device other than a disk, the message "Invalid access method" is displayed.

Copying with Field Specifications

The **SPECS** option provides a means of manipulating the contents of the records in a stream file. With this option you can delete portions of each record, add constant data to each record or rearrange the existing contents of each record. When the **SPECS** option is used, CopyFile prompts you to enter the specifications:

```
>copyfile sample.data new.data (specs
Enter specification list:
?
```


There are three types of specifications that you may enter:

- A letter, character or character code to be added to each record. Any typeable character or its numeric value in decimal or hexadecimal can be specified. Merely type the character you want added to each record. To specify the character value in hexadecimal, terminate the number with the letter h.

```
?A 1
?65 1
?41h 1
```

The above three specification are identical: They each specify that the letter “A” is to be placed in column one of each output record.

- Constant text to be added to each record. Merely type the string of characters desired surrounded by a pair of quotation marks. (Any delimiter other than a quotation mark may be used. The first character typed is always treated as the delimiter and must be matched with an identical delimiter at the end of the string of characters.)

```
? "New record: " 1
? \Some text\ 20
? +abbreviated+ 30
```

- The position of existing text in the source record. Existing text to be copied is indicated by two decimal numbers separated by a dash:

```
?1-10 10
?11-12 48
?13-13 47
?14-99 50
```

The above specifications indicate that the characters in each source record are to be rearranged: The first ten characters are moved to columns 10 through 19; the characters in columns 11 and 12 are moved to columns 48 and 49; the character in column 13 is moved to column 47; and the remainder of the source record from columns 14 through 99 is moved to columns 50 through 135.

As indicated in the example, to specify a single character of the source record, specify a column range that starts and stops on the same position (13-13).

Each specification is terminated by the column number to place the letter, constant text or source text in the destination record. The specification list is a list of one or more of these specifications terminated by an empty line. Invalid specifications are ignored with no message.

```
>copyfile sample.data new.data (specs
Enter specification list:
?* 1
?"New " 3
?1-99 10
?
```

In this example each output record will have a bullet symbol in column one, the literal text “New ” in columns three through six, and the original source record text is output starting at column 10 of each record. Columns seven through nine are output as spaces because nothing was specified for that area of the output record.

Note that the records output contain only the text and data specified in the specification list. For instance, if the previous example was changed to:

```
>copyfile sample.data new.data (specs
Enter specification list:
?* 1
?"New " 3
?10-99 10
?
```

The data in the input records, columns 1 through 9, is not copied to the output record because it is not part of the specification list.

The **SPECS** option may be used in combination with the **TRANS** option. When other text modification options are specified in combination with the **SPECS** option (i.e., **UPCASE**, **LOWCASE**), they will manipulate the source text *before* the specifications are applied and therefore do not apply to the specifications text. For instance, if **SPECS** and **UPCASE** were used with the previous specifications list, the literal text “New ” would still be output in columns three through six. It would not be converted to “NEW ”.

You can abandon the copy operation during entry of the specifications list by using the **[Esc]** or **[F9]** keys.

Copying with Character Translation

The **TRANS** option allows you to translate the characters in each source record as it is copied to the destination record. When the **TRANS** option is used, CopyFile prompts you to enter the translations:

```
>copyfile sample.data new.data (trans
Enter translation list:
?
```

Translations are entered by specifying a pair of characters separated by a space. The first character is the character in the source file that is to be

changed; the second character is the character that it will be changed to. The characters are specified with any typeable character or its numeric value in decimal or hexadecimal. Merely type the character you want translated, followed by a space and then the character to translate to. To specify the character value in hexadecimal, terminate the number with the letter h.

```
>copyfile sample.data new.data (trans
Enter translation list:
?f A
?... E
?+ O
?< U
?- N
?Y A
?
```

This translation list translates all occurrences of the accented, uppercase letters to their unaccented forms.

The **TRANS** option may be used in combination with the **SPECS** option. When other text modification options are specified in combination with the **TRANS** option (i.e., **UPCASE**, **LOWCASE**), they will manipulate the source text after the specifications are applied and therefore the “translate to” character might be changed. For instance, if **TRANS** and **LOWCASE** were used with the previous specifications list, the accented, uppercase letters would be translated to unaccented lowercase characters.

You can abandon the copy operation during entry of the translation list by using the **[Esc]** or **[F9]** keys.

Copying Indexed and Keyed Files

Of most interest when copying an indexed or keyed file is the **BYREC** option. This option creates an optimized copy of the source file with a completely rebuilt index to the records in the file. The **BYREC** option is implied by all of the options that might copy less than the entire file (**FOR**, **FRKEY**, and **TOKEY**).

When the **BYREC** option is used to optimize file access, it is best to first create an empty destination file with the **CREATE** command described on page 269. By creating the empty file first, you can specify a contiguous file and change the key length, record length and file size. The **BYREC** option will use this empty file and copy the source file to it, one record at a time.

The **BYREC** option can also be used to copy the records from one type of file organization to another. For instance, from a keyed file to an indexed file.

Copying Direct Files

Similar to indexed and keyed files, the **BYREC** option can reorganize a direct file.

Notes The [FOR](#) option requires that the [FRKEY](#) option be used. When [FOR](#) is used without the [FRKEY](#) option, the entire file will be copied.

If the source and destination files are members of a library and that library does not exist on the destination drive, CopyFile asks if you want to create it:

```
>copyfile customer.data.june j
Ok to create library "/CUSTOMER.DATA:J" (Yes,No)
```

If you respond with [Y](#), the library is created on the destination drive with a size equal to the source file's library size.

Defaults [NEWDATE](#) (when a modified copy of the source file is created), [NEWFILE](#), [NOQUERY](#) (when wild cards are not used), [NOSORT](#) (unless the [SUBDIR](#) option is used), [NOVERIFY](#), [OLDDATE](#) (when an exact copy of the source file is created), [QUERY](#) (when wild cards are used), [SORT](#) (when [SUBDIR](#) option is used), [TYPE](#).

Cautions The [MOUNT](#) option should only be used when both the source and destination drives are private disk volumes, or when no other users are active.

Restrictions By default, only files owned by the current account are copied. You can copy a public file (owned by the SYSTEM account) when logged onto a private account by specifying the [PUBLIC](#) option.

To copy a file owned by another account, you must specify the owning account name as part of the path and the file must provide shared read access:

```
>copyfile private\his.file my.file
```

This command copies the file HIS.FILE owned by the account named PRIVATE, to your account, current working directory, file name MY.FILE.

You may only copy a file to your account. By default, the destination file is in the current working directory, but you may specify a path to the directory that you want to copy it to.

```
>copyfile my.file textfile/samples/new.file
"/MY.FILE:S" copied to "/TEXTFILE/SAMPLES/NEW.FILE:S".
```

When the destination file specification includes a path, then that path must exist. CopyFile does not create subdirectories (except with the [SUBDIR](#) option).

See also [Archive](#), [Backup](#), [FileType](#), [IXDiag](#), [Receive](#), [Restore](#), [Send](#), [TBackup](#), [THEO+COM](#)

Create Command

The Create command creates direct, indexed and keyed files, libraries of files and subdirectories.

```

1  CREATE  file ( LIBRARY library-options
2  CREATE  file ( SUBDIR
3  CREATE  file ( DIRECT direct-options
4  CREATE  file ( INDEXED indexed-options
5  CREATE  file ( KEYED indexed-options
6  CREATE  file ( CLEAR

```

<i>file</i>	»	file name with optional path
<i>library-options</i>	»	REPLACE SIZE <i>nnnn</i>
<i>direct-options</i>	»	CONTIG FILESIZE <i>nnnnnn</i> GROW <i>n.m</i> RECLEN <i>nnnnn</i>
<i>indexed-options</i>	»	CONTIG FILESIZE <i>nnnnnn</i> GROW <i>n.m</i> KEYLEN <i>nnn</i> RECLEN <i>nnnnn</i>

Operation

Mode 1—Creates a new library or resizes an existing library. *file* must be a file description containing both a file name and a file type. Libraries may not be a member of another library.

```
>create source.programs (library size 44
```

```
>create source.programs (library size 100 replace
```

Mode 2—Creates a new subdirectory. *file* must be a file description containing a file name. It may have a file type but it cannot have a member name. Subdirectories may not be a member of a library.

```
>create source (subdir
```

Mode 3—Creates a new direct-access file. Direct files may be members of an existing library.

```
>create accounts.payable.control (direct file 22 rec1 128
```

Mode 4—Creates a new indexed-access file. Indexed files may be members of an existing library.

```
>create parts.inventory.sku (indexed file 20000 rec1 32 key1 10
```

Mode 5—Creates a new keyed-access file. Keyed files may be members of an existing library.

```
>create parts.inventory.list (keyed file 60000 rec1 280 key 42
```

Mode 6—Clears the contents of an existing direct, indexed or keyed file. The contents of all records in a direct file are initialized to zeros. Although indexed and keyed files are not zeroed with this command all of the records are removed and the space is returned to the file's free space list. Effectively, this is the same as erasing the file and then recreating it with its current file size. However, the clearing process is faster because the disk space does not have to be reallocated.

```
>create customer.master (clear
```

Library Options

REPLACE Indicates that the existing library is to be replaced with a library of the same name but a different size.

This option preserves all members of the existing library. It does this by renaming the existing library with a temporary name, creating the new, empty library with the size requested, and then renaming the members in the temporary library name to this new library. The temporary library name is then erased.

If the requested size of the new library is less than the number of members already in the existing library, the size is adjusted upward so that the new library can contain all of the existing member files.

See “[Cautions](#)” on page 273.

SIZE This option is required and specifies the size of the library. This size represents the minimum size to create. Because a library uses the same search algorithm as the root directory, its size is determined by the requirements of the algorithm.

For instance, requesting a size of 100 will create a library with a size of 116.

The maximum size for libraries is 262,100.

Direct File Options

CONTIG

Forces the new file to use contiguous disk space. If sufficient contiguous disk space is not available, the file is not created and a “Disk full” message is reported. See “Notes” on page 272.

FILESIZE

This option is required and specifies the number of records that the file can contain. The physical size of the file is computed by multiplying the FILESIZE by the RECLLEN.

GROW

Specifies the growth factor for the file.

Growth factors are specified as either a whole number, such as 1, 2, 3, *etc.*, or as a fraction, such as 0.1, 0.3, 0.5, *etc.* Do not specify a combination like 1.3. If you do, only the integer portion is used. The default growth factor is 0.3.

For an explanation of file growth factors see “Growth Factor” on page 144.

RECLLEN

This option is required and specifies the length of each record.

The maximum record length is 65,535. However, if the file is to be used by a BASIC16 language program, limit the record length to 2,048.

Indexed and Keyed File Options

CONTIG

Forces the new file to use contiguous disk space. If sufficient contiguous disk space is not available the file is not created and a “Disk full” message is reported. See “Notes” on page 272.

FILESIZE

This option is required and specifies the number of records that the file can contain. The physical size of the file is computed by multiplying the FILESIZE by the sum of RECLLEN plus KEYLEN plus the overhead required by the file access method (indexed or keyed).

GROW

Specifies the growth factor for the file.

Growth factors are specified as either a whole number, such as 1, 2, 3, *etc.*, or as a fraction such as 0.1, 0.3, 0.5, *etc.* Do not specify a combination like 1.3. The default growth factor is 0.3.

For an explanation of file growth factors see “[Growth Factor](#)” on page [144](#).

KEYLEN This option is required and specifies the allocated length of the keys for each record.

The maximum key length is 128.

RECLEN This option is required and specifies the length of each record. This record length is separate from the [KEYLEN](#) value.

The maximum record length is 65,535. However, if the file is to be used by a BASIC16 language program, limit the record length to 2,048.

Notes

If the [CONTIG](#) option is not used for direct, indexed and keyed files, the file is created using any available disk space, contiguous or not. If there is insufficient disk space available, the requested [FILESIZE](#) of the file is reduced so that the file can fit in the largest contiguous disk space that is available.

It is possible to create a library, subdirectory or data file in another account. Merely specify the complete path for the file description:

```
>create private\program.source (library size 44
```

In the above example PRIVATE\ is the name of another account. Access to this file is restricted unless the [CREATE](#) environment variable has been defined and it includes the attributes allowing shared file access. See “[CREATE](#)” on page [102](#) for a description of this environment variable.

The growth factor of an existing file can be changed with the [CHANGE](#) command described on page [216](#).

Defaults

When the [CREATE](#) environment variable is not defined, a data file created with this command has the following attributes: modified, shared read protected, shared write protected, execute protected, not erase protected, not read protected, not write protected and not hidden. If the [CREATE](#) variable is defined, it specifies the attributes for the new file.

The modified attribute and the file’s last change date is always set for newly created data files (direct, indexed and keyed).

Subdirectories and libraries created with this command have no protection codes set.

The default growth factor for direct, keyed and indexed files is 0.3 or 30%.

Cautions	The REPLACE option should not be used if any other user might be using one of the members of the library. The REPLACE option is actually a “macro” command in that several separate operations are performed: renaming an existing library, creating a new library, and renaming existing members and erasing the old library. It is best if all users are inactive so that all of the steps can be completed without interruption.
Restrictions	Libraries and subdirectories cannot be members of libraries. Library file descriptions must contain both a file name and a file type.
See also	Change , MkDir

CRLF Command

The CRLF command operates on stream files and converts the end-of-record mark and national characters.

Commands

- 1 CRLF *file...* (*options*
- 2 CRLF *file...* (*os options*

<i>file</i>	»	file name with optional path, may contain wild cards		
<i>options</i>	»	NOTYPE	TYPE	XLATE OEM
		NOXLATE	XLATE DOS	
<i>os</i>	»	DOS		
		THEOS		
		UNIX		

Operation CRLF only operates on stream files.

Mode 1—Performs an “auto conversion” between THEOS and DOS or UNIX or between DOS or UNIX and THEOS. The file is analyzed and, if the first record is terminated with a CR only (THEOS format), the file is converted to the DOS format using CR,LF. If the first record is terminated with an LF only (UNIX format) or a CR,LF (DOS format), the file is converted to the THEOS format using CR only.

Mode 2—The record terminators in the file are converted to the terminators used by the operating system specified.

OS	<u>DOS</u>	Indicates that the record terminator should be changed to a CR,LF.
	<u>THEOS</u>	Indicates that the record terminator should be changed to a CR only.
	<u>UNIX</u>	Indicates that the record terminator should be changed to a LF only.

Options	<u>NOTYPE</u>	Do not display the conversion messages.
	<u>NOXLATE</u>	Do not perform any translations except for record terminators.

TYPE A default option that displays the conversion message for each file converted. The messages are of the form “Changing CR into CRLF on file “SAMPLE.FILE:S”.”

XLATE DOS In addition to the translation of record terminators, all embedded national characters and other characters whose value is greater than 128 are translated. This option is effective only when *file* uses CR or CR,LF record terminators. When effective, the eight-bit characters are translated to or from the DOS character set from or to the THEOS character set.

This is a default option.

XLATE OEM In addition to the translation of record terminators, all embedded national characters and other characters whose value is greater than 128 are translated. This option is effective only when *file* uses CR or CR,LF record terminators. When effective, the eight-bit characters are translated to or from the Windows character set from or to the THEOS character set.

Notes The file is converted “in place.” That is, the output of this command is a file with the same name as the input. The actual process used is to output the file with a temporary file name, erase the input file, and then rename the temporary file to be the input file name.

Defaults [TYPE](#) and [XLATE DOS](#) are default options.

Restrictions Only ASCII stream files can be converted with this command. Other file organizations are incompatible between operating systems or do not have end-of-record marks.

See also [FileType](#)

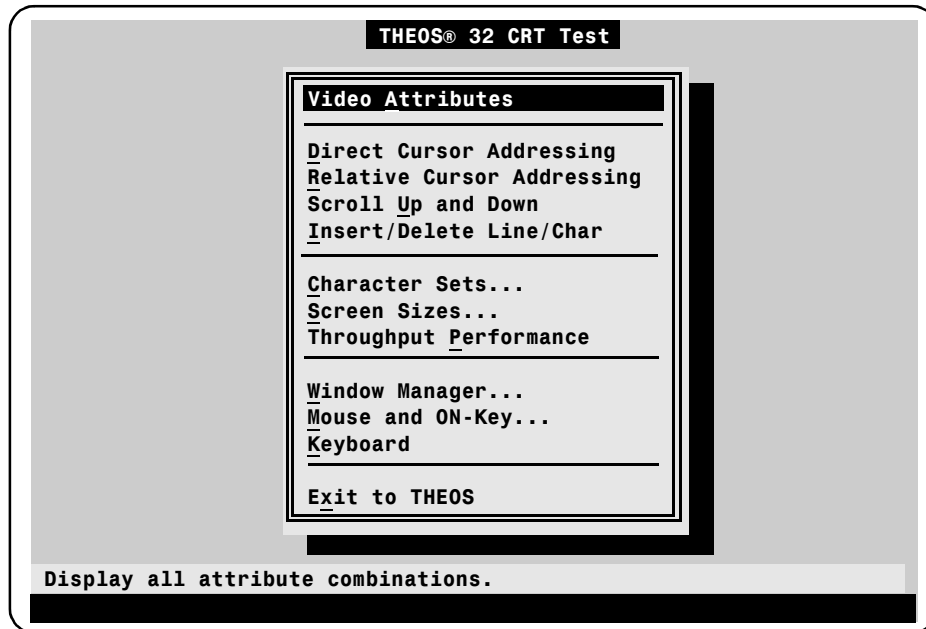
CRT Command

The CRT command demonstrates and tests the console's display capabilities and keyboard definitions.

CRT

Commands

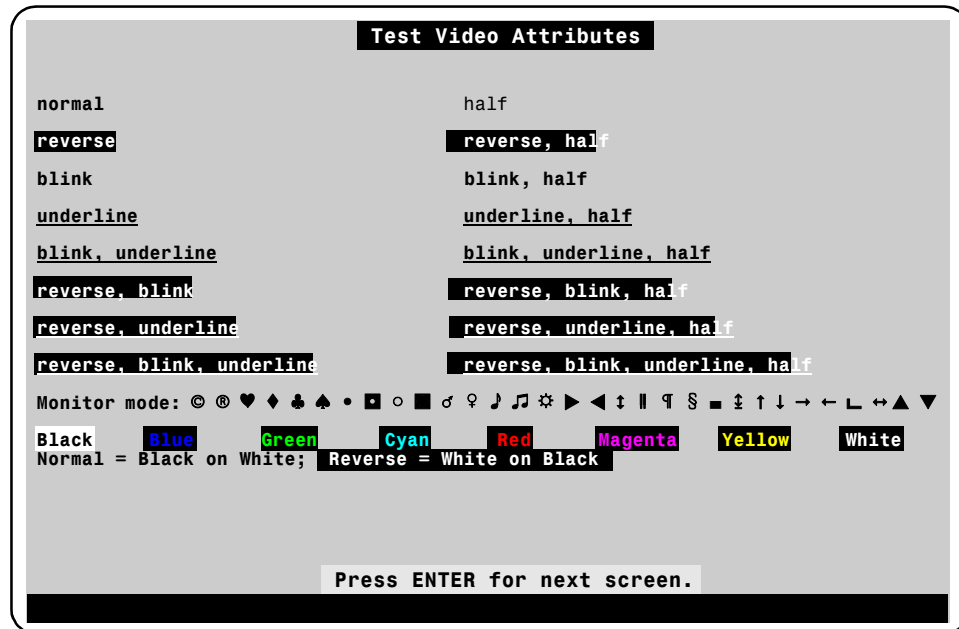
Operation When CRT is invoked the “CRT Test Menu” is displayed.



Use the normal menu selection keys to select the desired tests. These keys are described in “[Using Menus](#)” on page 75.

■ Video Attributes

This test shows the video attributes supported by THEOS and how they are displayed on this console with the current class code.



Commands

The specific information shown on your screen may differ depending upon your terminal's or monitor's capabilities and upon the definitions in the class code (see "[ClassGen](#)" on page 225).

The "Monitor mode" line shows the characters that your console will display when the monitor mode is enabled. This line is only shown if the class code defines a code for monitor mode on and off. The characters shown here are displayed when the console is a VGA monitor and class code 90 is used.

The line showing the color names is displayed only if the class code has a "set alpha color" definition.

When **Enter** is pressed an "Erase unprotected" is performed. This should clear all of the attribute displays off of the screen except for those shown in half intensity on the right.

■ Direct Cursor Addressing

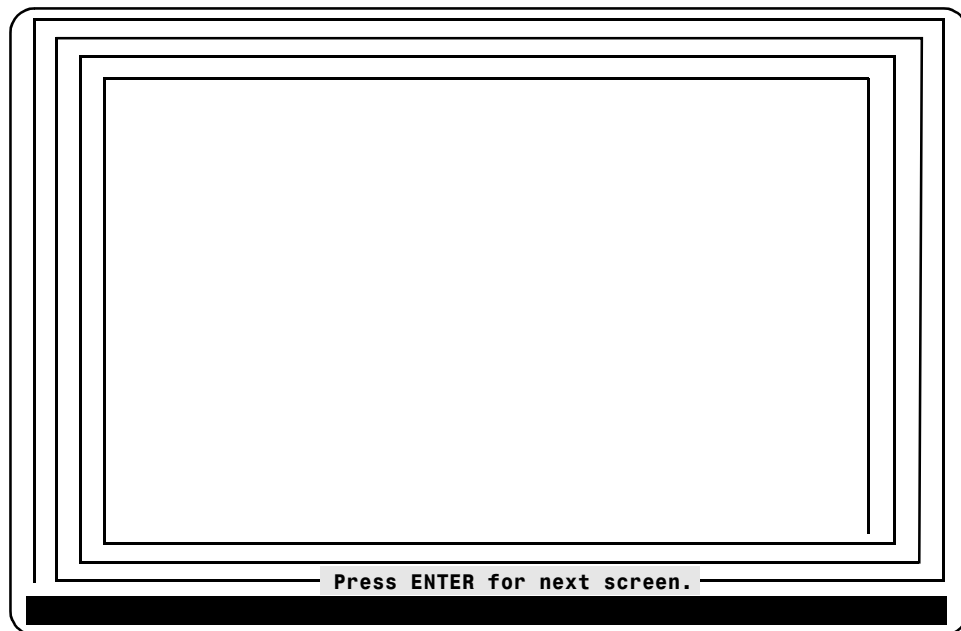
The direct cursor addressing display tests the console's and class code's ability to accurately position the cursor. The screen should fill with asterisk (*) characters in a top-left to bottom-right manner. Each asterisk is displayed by positioning the cursor to the desired location, displaying the asterisk and then positioning to the next desired location.

If the screen does not completely fill with asterisks (except for the bottom, right-hand position), there is either something wrong with the class code definition for direct cursor addressing or there is a problem with the communication's line (baud rate, parity, *etc.*).

■ Relative Cursor Addressing

This screen display tests the console's and class code's ability to move the cursor relative to its current location (left, right, up and down).

The spiraling lines shown here are drawn by moving the cursor with the left, right, up and down codes, and then outputting a single line-drawing character.



■ Scroll Up and Down

This display tests the console's ability to scroll text up and down on the screen. First, lines of text are displayed terminated by a carriage-return, line-feed. When the bottom of the screen is reached, lines continue to display which should cause the lines above this to scroll up.

```
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abc
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcd
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcde
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdef
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefg
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefgh
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghi
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghij
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijk
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijkl
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklm
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmn
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmno
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnop
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopq
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopqr
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopqrs
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopqrst
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopqrstu
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopqrstuv
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopqrstuvw
! "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`abcdefghijklmnopqrstuvwx
```

Commands

After approximately 100 lines are displayed in this manner, the screen is cleared and new lines of text are displayed. These are displayed at the top of the screen preceded by an “insert line” command. This causes the display to scroll down.

Observe these patterns as they display. Because scrolling takes a relatively long time for a terminal to perform, if there are any handshake problems with your communication's line, they will show up here with a pattern that does not appear uniform.

■ Insert/Delete Line/Char

This display tests the console's and class code's insert and delete capabilities. First, the screen is filled with rows of numbers. Then the six insert and delete commands are performed on this displayed text:

Commands

```
0123456789012345678901234567890123456789012345678901234
112356789012345678901234567890123456789012345678901234
2123456789012345678901234567890123456789012345678901234
312345 6789012345678901234567890123456789012345678901234
4123456789012345678901234567890123456789012345678901234

5123456789012345678901234567890123456789012345678901234
7123456789012345678901234567890123456789012345678901234
8123456789012345678901234567890123456789012345678901234
912345678901
0123456789012345678901234567890123456789012345678901234
11234567890123
```

```
4,1 = DC      6,3 = IC
8,5 = IL      10,7 = DL
12,9 = EOL    14,11 = EOS
```

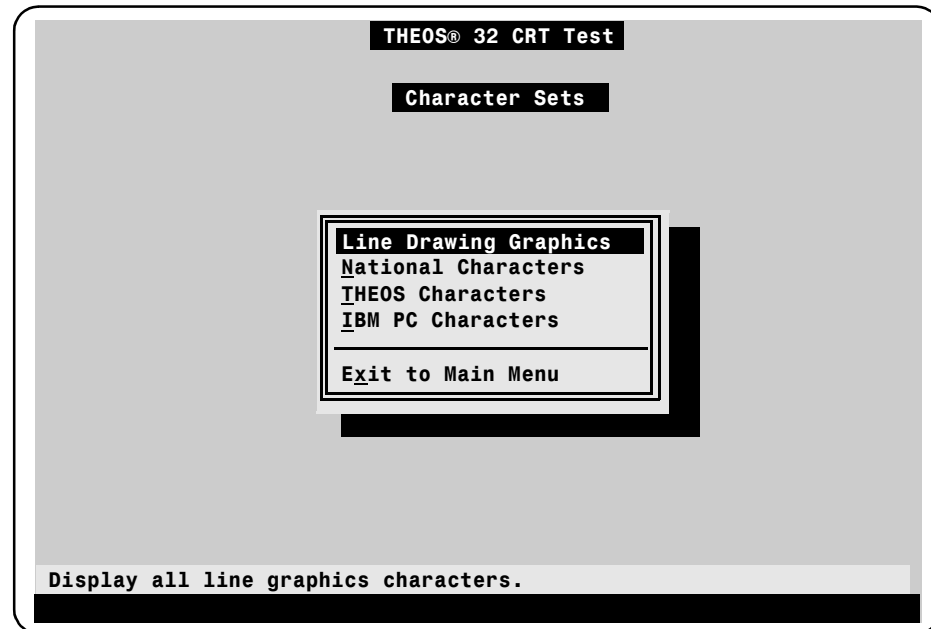
Press ENTER for menu.

If your console supports all of these insert and delete capabilities, the screen should look like the one displayed here (except yours should have your attached line length used).

The text in the middle of the screen indicates which commands were issued and at which locations. When studying these locations, note that this program addresses the cursor with a number base of zero. Therefore, the upper left corner is location 0,0.

■ Character Sets

There are multiple character sets supported by THEOS. When this option is selected from the main CRT menu, a Character Sets menu appears:



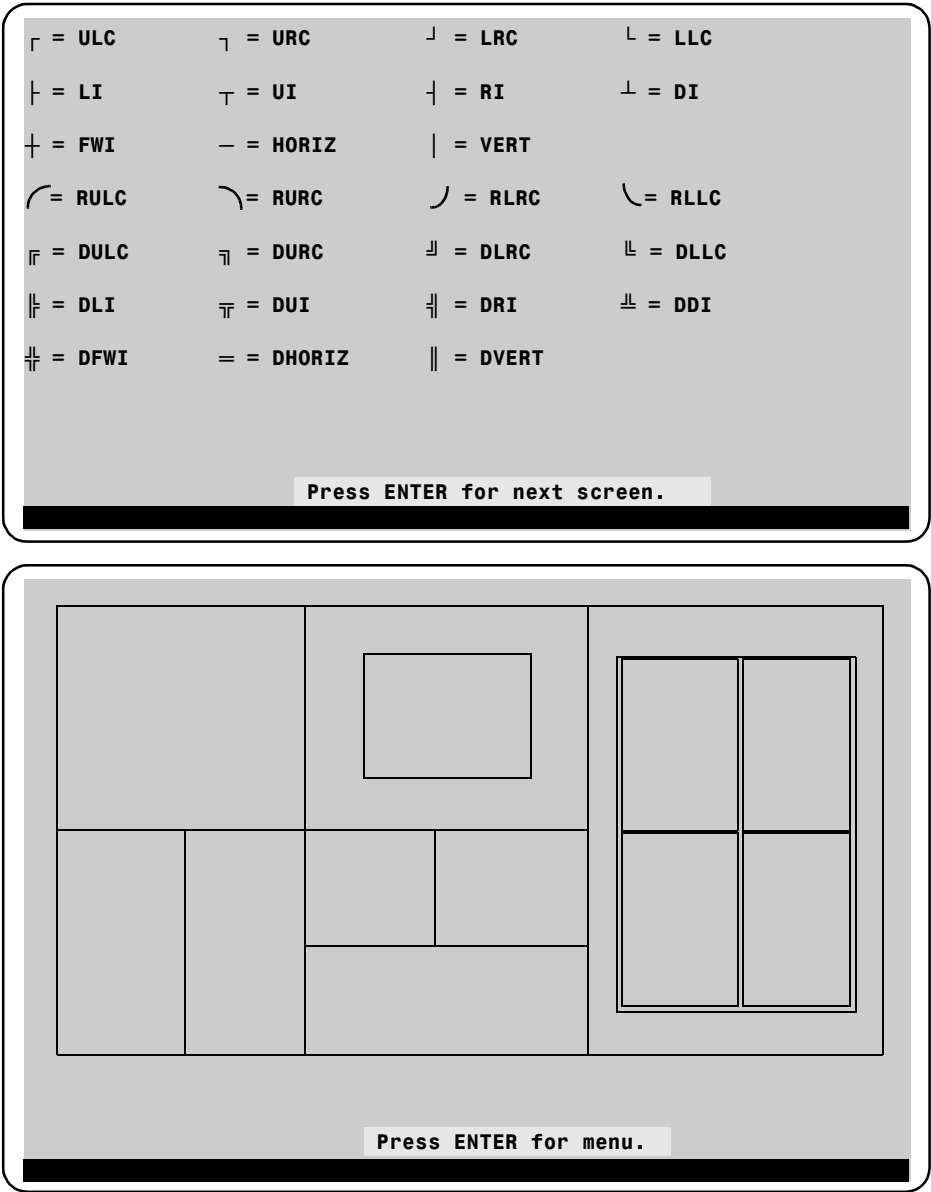
Use the normal menu selection keys to select the desired tests. These keys are described in “[Using Menus](#)” on page 75.

Note: Not all consoles support all of the character sets.

• Line Drawing Graphics

The two screens displayed by this menu item show the characters displayed when line-drawing graphics are used and a sample of the graphics display.

Commands



• **National Characters**

The National Characters display shows all of the multinational characters supported by THEOS along with their names, such as “Lowercase e grave accent.”

A special help text file is supplied that shows how to compose these characters from the keyboard.

```
>help compose
```

• **THEOS Characters**

This display shows the THEOS character set. It includes the ASCII characters, the line-drawing graphics characters and the multinational characters.

THEOS Character Set																											
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F											
0:											
1:											
2:	.	!	"	#	\$	%	&	'	()	*	+	.	.	.	/											
3:	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?											
4:	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O											
5:	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_											
6:	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o											
7:	p	q	r	s	t	u	v	w	x	y	z	{		}	~	.											
8:											
9:											
A:	┐	┌	└	┘	├	┤	┨	┩	┴	┬	┴	┬	┴	┬	┴	┬											
B:	┐	┌	└	┘	├	┤	┨	┩	┴	┬	┴	┬	┴	┬	┴	┬											
C:	Ä	ä	Â	à	Á	É	ê	é	È	é	î	ï	ì	í	Ö	ö											
D:	ô	ò	ó	û	ü	Û	ù	ú	Ç	ç	Ñ	ñ	Æ	æ	Å	å											
E:	ß	ℓ	ı	€	£	¥	₪	₹	₠	₡	₢	₣	₤	₥	₦	₧											
F:											

Press ENTER for menu.

- **IBM PC Characters**

The IBM PC character set is available on most consoles that are PC Term compatible. This characters set is “Code Page 437” and is accessed by a program by enabling monitor mode.

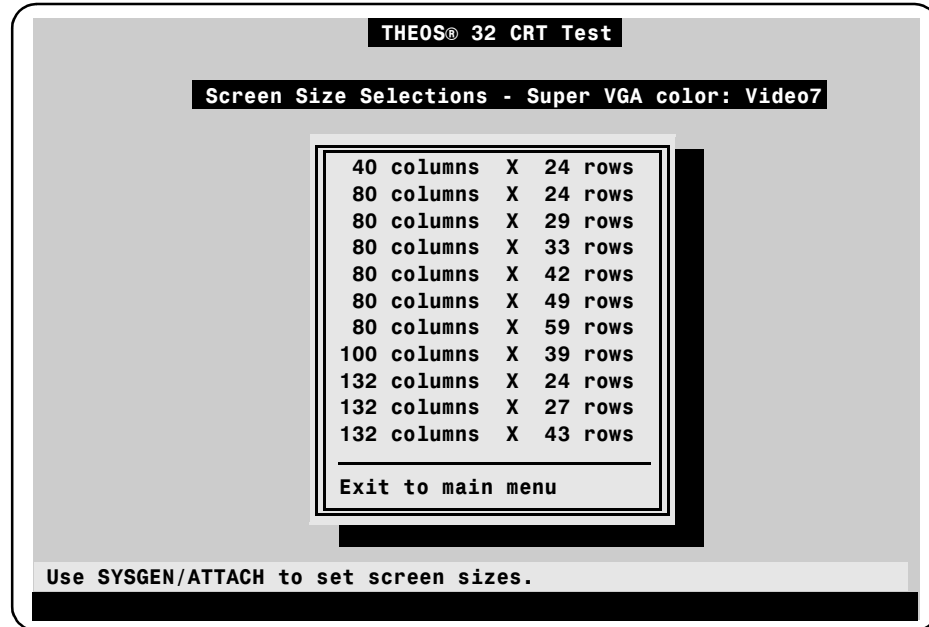
IBM-PC Character Set																
0:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1:	►	◄	†	♥	♦	♣	♠	•	◻	◊	◼	♂	♀	♪	♫	€
2:	!	"	#	\$	%	&	'	()	*	+	,	<	=	>	?
3:	0	1	2	3	4	5	6	7	8	9	:	;				
4:	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5:	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6:	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7:	p	q	r	s	t	u	v	w	x	y	z	{		}	~	␣
8:	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ï	Ä	Å
9:	É	æ	Æ	ô	ö	ò	û	°	ý	ÿ	Ü	Û	£	¥	₭	₧
A:	á	í	ó	ú	ñ	Ñ	ª	º	¿	¬	¸	½	¼	¾	»	«
B:	☒	☐	☑		└	┌	┐	┑	┒	┓	└	┌	┐	┑	┒	┓
C:	L	└	T	┌	┐	┑	┒	┓	└	┌	┐	┑	┒	┓	└	┌
D:	└	T	T	┌	┐	┑	┒	┓	└	┌	┐	┑	┒	┓	└	┌
E:	α	β	γ	π	Σ	σ	μ	τ	φ	θ	Ω	δ	∞	∅	ε	∩
F:	≡	±	≥	≤		└	┌	┐	┑	┒	┓	└	┌	┐	┑	┒

Press ENTER for menu.

This screen does not display properly if the console does not support the “Code Page 437” character set. It will not display at all if the class code number is not one of the known class codes for PC Term compatible terminals (see “[CLASSnnn \(Class Codes\)](#)” on page 728).

■ Screen Sizes

Some terminals support multiple screen sizes. For instance, most VGA display monitors support the sizes shown in the following menu. When the console supports multiple screen sizes, a menu like the following appears. This screen allows you to demonstrate the various sizes that THEOS thinks are available on your console.



Use the normal menu selection keys to select the desired tests. These keys are described in [“Using Menus”](#) on page 75.

This menu allows you to select various screen sizes and see how they appear on your console. The CRT command will not change the screen size of your console except during this demonstration. Once a desirable screen size is found, use the Attach or Sysgen command to set the attached screen size.

■ Throughput Performance

This test uses two screens to test the actual throughput performance of the console as it is currently attached. First it tests the transmission rate when one full page of text is sent to the console, and then it tests the scroll rate. The accuracy of this test may be effected by buffering if the console is attached via an intelligent multiport.

Commands

```
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll Noscroll
Full page: 81 msec; 28,000 cps (280,000) throughput bps)
```

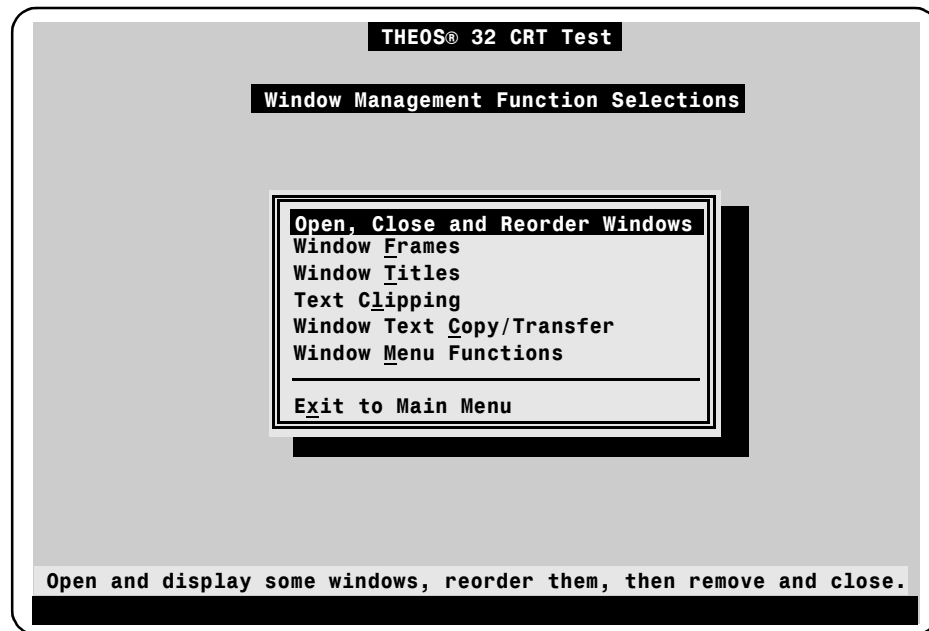
Press ENTER for next screen.

```
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll Scroll
Scroll: 1,066 msec; 54 lines/sec
```

Press ENTER for menu.

■ Window Manager

This function of the CRT command tests and demonstrates the capabilities of window management on the console.

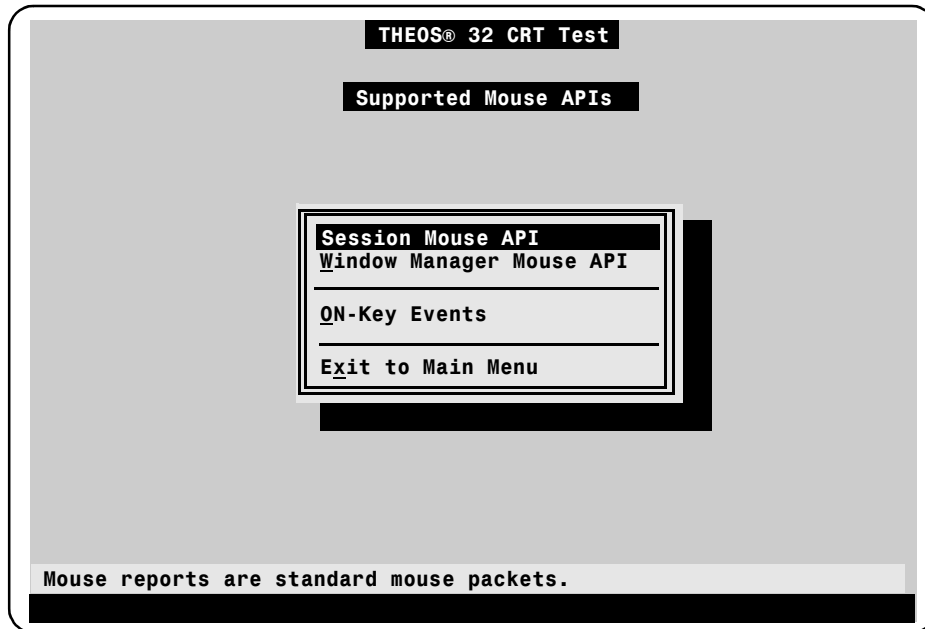


Commands

Use the normal menu selection keys to select the desired demonstrations. These keys are described in “[Using Menus](#)” on page [75](#).

■ Mouse and ON-Key

This function presents a menu that allows you to test the mouse and the “ON-KEY” capabilities of the console.



Use the normal menu selection keys to select the desired demonstrations. These keys are described in [“Using Menus”](#) on page 75.

• Session Mouse API

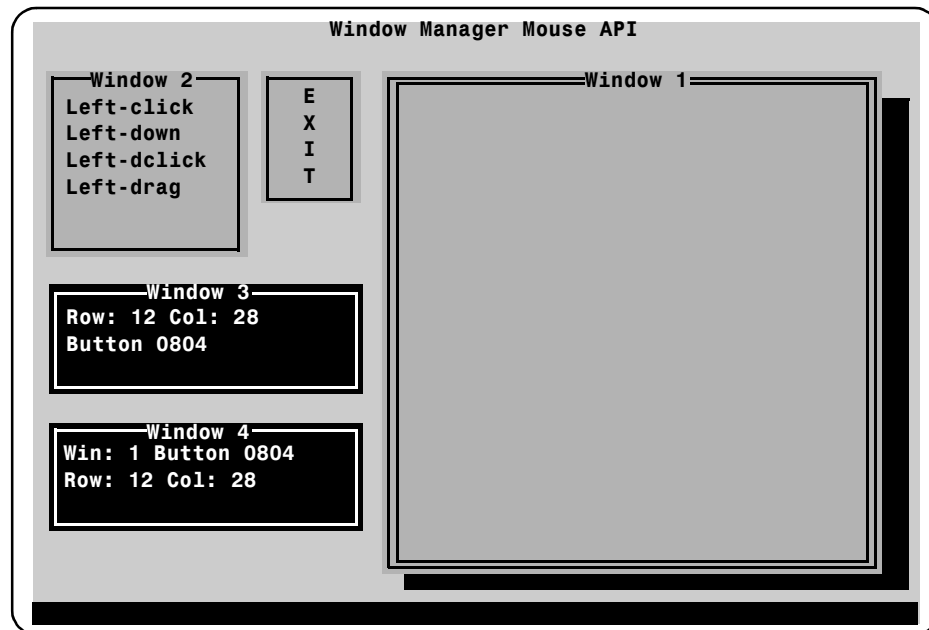
This screen provides a general test of the session mouse. A graphical representation of a mouse is displayed with buttons. When you click, double click or drag one of the mouse buttons, the button is highlighted on the screen and the action is described in text.

When the trace mode is enabled, the column and row number of the mouse cursor position is displayed. To enable trace mode, position the mouse cursor over the graphical button “TRACE:OFF” and click the left mouse button.

To exit this demonstration, click on the “EXIT” button or press the **[Esc]** key.

- **Window Manager Mouse API**

This screen tests the Window Manager Mouse capability.



The four windows in this display screen are used to report various information about mouse activity.

Window 1: This is a large target area.

Window 2: A display window that shows the last mouse event, such as: “Left-down,” “Left-click,” “Right-dclick,” *etc.*

Window 3: A display window showing the last mouse event relative to the screen origin (upper left corner of screen).

Window 4: A display window showing the last mouse event relative to the window origin.

To exit from this test click on the “Exit” display button or press **[Esc]**.

- **ON-Key Events**

This display allows you to test the keyboard for all keys that can be detected by the “ON-KEY” capability of THEOS’ Window Manager software. Merely press the keys you are interested in to determine if the ON-KEY mechanism can detect it. If nothing displays on the screen, then ON-KEY could not detect the key or key combination.

To exit from this test press **Esc** twice.

- **Keyboard**

This test allows you to test the console’s class code definitions for the input function keys. A simulated keyboard is displayed on the screen. As each key on the keyboard is pressed, the cursor is positioned to the key received and decoded by the class code.

Notes

The operation and specific displays seen when the CRT command is used will greatly depend upon your specific console and the class code associated with that console.

See also

[Attach](#), [ClassGen](#), [Printer](#), [Setup](#), [Sysgen](#)

Date Command

The Date command displays the current system date and time on the standard output device (normally the console).

1 DATE

2 DATE *+format*

format » codes specifying the format of the date and time

Commands

Operation **Mode 1**—Outputs the current date and time.

```
>date
Thursday, July 4, 1996 2:49 PM
```

The year is always displayed as a four-digit number including the century, and the time is always displayed in 12-hour format using “am” and “pm” designations. If timezone abbreviations are defined in the system configuration (see “[Standard Time Zone Abbrev](#)” on page 535), the time zone name is displayed:

```
>date
Thursday, July 4, 1996 2:49 PM PDT
```

The day of the week name and the month name are specific to the current language in use. See “[Default Language Code](#)” on page 535.

Mode 2—Outputs the current date and time formatted according to the *format* specifications. The format must start with a plus sign (+) and, if there are any lowercase characters, it must be enclosed within a pair of quotation marks.

```
>date "+DATE: %m:%d:%y%nTIME: %H:%M:%S"
DATE: 07:04:96
TIME: 10:02:15

>date "+DATE: %D%nTIME: %r"
DATE: 1998.07.04
TIME: 10:02:15AM

>date "+Event occurred on %B %d, %Y at %H:%M"
Event occurred on July 04, 1998 at 15:42
```

Format Codes The format specification is a string of characters specifying the literal characters and symbols that are output, along with codes specifying the date or time element to output. The formatting codes use the percent character (%) followed by a single letter. The letter indicates the specific date or time element to use.

%n	Starts a new display line.						
%f	Starts a new display page (form-feed output).						
%t	Tabs to the next tab stop. Tab stops are positioned every eight columns.						
%%	Outputs a single percent character.						
%A	Day of week name (Monday, Tuesday, etc.) Uses text in system messages #264 and 265.						
%B	Month name (January, February, etc.). Uses text in system messages #261, 262 and 263.						
%D	Complete date formatted using the current DATEFORM : <table data-bbox="633 829 990 913"> <tr> <td>07/04/1998</td><td>DATEFORM = 1</td></tr> <tr> <td>04-07-1998</td><td>DATEFORM = 2</td></tr> <tr> <td>1998.07.04</td><td>DATEFORM = 3</td></tr> </table> <p>Note that the date is always displayed with a four-digit year.</p>	07/04/1998	DATEFORM = 1	04-07-1998	DATEFORM = 2	1998.07.04	DATEFORM = 3
07/04/1998	DATEFORM = 1						
04-07-1998	DATEFORM = 2						
1998.07.04	DATEFORM = 3						
%H	24-hour number (00–23).						
%J	Day number of month, without leading zero (1–31).						
%M	Minute number (00–59).						
%S	Second number (00–59).						
%T	Complete time in hh:mm:ss format (i.e., 15:24:32).						
%Y	Year number including century (i.e., 1998).						
%a	Day of the week name, abbreviated to three letters (Mon, Tue, Wed, Thu, Fri, Sat and Sun). Uses text in system message #423.						
%b	Month name, abbreviated to three letters (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov and Dec). Uses text in system message #423.						

%d	Day number of month (01–31).
%h	Month name, abbreviated to three letters (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov and Dec). Uses text in system message #423. This format is identical to the %b format specification.
%j	The Julian day number (001–366).
%m	The current month number (01–12).
%r	The current time in 12-hour notation (i.e., 03:15:20PM).
%w	Weekday number with Sunday = 0.
%y	Year number in century (00–99).

Notes

Remember that the format specification must start with a plus sign. If *format* contains any spaces or lowercase letters you must enclose it within quotation marks.

This is a command that outputs to the standard output device so the output can be redirected to a disk file, printer, *etc.* See “[Standard Input, Standard Output and I/O Redirection](#)” on page 51.

See also

[Set](#), [Show](#), [Sysgen](#)

Dial Command EXEC

The Dial command is an EXEC language program that provides convenient access to the [THEO+COM](#) command's modem and telephone number dialing capabilities.

```
1  DIAL
2  DIAL  number

_____
number          »  telephone number to dial
```

Operation

Mode 1—Invokes the (see “[THEO+COM](#)” on page 570) in DIAL mode. The first attached communications port is used and the “Dialing Directory” is then invoked, allowing you to choose one of the entries to dial. Dialing directory maintenance is allowed along with manual dialing.

Refer to the *THEO+COM Installation and User's Guide* manual for a description of the dialing directory and its use.

Dial automatically exits after a number is dialed or if you abort the selection of a number. Any modem or telephone connection remains connected.

Mode 2—Invokes the [THEO+COM](#) in DIAL mode. The first attached communications port is used and *number* is dialed via the device connected to that port.

```
>dial 1 800 123-4567
Dialing 1 800 123-4567
```

The *number* may contain any of the following characters.

Character	Meaning or Effect
() - <i>space</i>	Ignored: Used for readability purposes only.
digits 0–9	Generates the tones or pulses for the digit.
,	Wait for two seconds. Multiple commas may be used for additional periods of delay.
/	Wait for 125 milliseconds. (Not all modems support this code.)
W	Wait for a secondary dial tone.
@	Wait for five seconds of silence.
!	Go <i>on-hook</i> for ½ second and then return to <i>off-hook</i> . Also called <i>flash hook</i> .

Character	Meaning or Effect
letters A–Z	Generates the tones or pulses corresponding to the telephone dial letters. For instance, the number “93THEOS” would dial “9384367.” The letters A, B and C match the digit 1, the letters D, E and F match the digit 2, <i>etc.</i>
* # A B C D	The letters A–D may not be specified with spaces surrounding each letter or they will be interpreted as the DTMF codes described next. Generates the corresponding telephone tones if tone dialing mode is enabled. The letter codes must be specified with spaces surrounding them to avoid confusing them with the telephone dialing pad letters described above.
P	Switch to pulse dialing mode. This code must be surrounded by space characters.
T	Switch to tone dialing mode. This code must be surrounded by space characters.
R	Switch to answer mode after dialing. This code can only be used at the end of the dialing string. It is used when initiating a call to an originate-only modem. After the number is dialed, your modem switches to answer mode.
;	Returns to modem command mode after the number is dialed. This code can only be used at the end of the dialing string and it must be part of a string enclosed in quotation marks (otherwise the CSI will treat it as a comment).

Dial exits after a number is dialed. Any modem or telephone connection remains connected.

Notes	To use a communications port other than the first attached COM n device, or to use any of the THEO+COM command line options, you must use the THEO+COM command (synonym name is COM).
Defaults	Dial uses the communication protocol and configuration defined by THEO+COM's configuration menu.
Restrictions	<p>A modem or computer-controlled telephone dialer must be connected to the first communications port. Dial will wait forever to get a valid response from the device. You will have to abort the command (Break, Q).</p> <p>The THEO+COM configuration and modem setup must be defined properly before using the Dial command.</p>
See also	THEO+COM

Disk Command

The Disk command formats and partitions disk volumes, reports on the status of the disk and disk allocation, and performs diagnostic tests of a disk volume.

1	<u>D</u> ISK				
2	<u>D</u> ISK	<i>drive</i>	(<i>options</i>	
3	<u>D</u> ISK	<i>drive</i>	(<u>F</u> ORMAT <i>format-options</i>	
4	<u>D</u> ISK	<i>file</i>	(<i>options</i>	
<hr/>					
	<i>drive</i>	»	disk drive attachment letter		
	<i>file</i>	»	file name with optional path		
	<i>options</i>	»	ACCOUNT	FIX	MAP
			BOOT	FREE	MULTIUSER
			CLEAN	FRAGMENT	NOASK
			CLEAR	HEX	OPTIMIZE
			FAST	LABEL	PRT <i>nn</i>
	<i>format-options</i>	»	BOOT		NOASK
			CYLINDERS <i>nnnn</i>		SECTORS <i>nnn</i>
			DENSITY <i>n</i>		SIZE <i>nn</i>
			HEADS <i>nn</i>		TRACKS <i>nnnn</i>
			INCREMENT <i>nn</i>		

Operation **Mode 1**—Invokes the Disk command menu described on page 307.

Mode 2—Performs one of the status reports or diagnostic tests indicated by the specific options specified. When no option is specified the default option of **FAST** is performed.

```
>disk i
Disk I label "Image".
File system = THEOS.
Capacity = 2,097,152 (128 cylinders, 2 heads, 32 sectors).
Main directory size = 964 files.
Allocated bytes = 64,512.
Available bytes = 2,032,640.
```

When the option is **FAST** or **SHOW** the *drive* specification may be the wild card asterisk (*). This displays the disk status for each of the currently attached drives, in drive search sequence. Drives not included in the drive

search sequence are omitted from this report. (The drive search sequence is described on page [101](#).)

```
>set search sim

>date "+Disk status as of %B %d, %Y at %H:%M" > disk.status:s

>disk * >> disk.status:s
```

The above commands define a drive search sequence that includes the system drive and two other attached drives. Using i/o redirection, a date stamp is output to a log file and then the disk status report is appended to that log file. This log file contains:

```
Disk status as of August 03, 1999 at 08:50

Disk S label "THEOS".
Archived to disk "TUESDAY" on 3 August 1999, at 18:35.
File system = THEOS/4G.
Capacity = 267,386,880 (225 cylinders, 64 heads, 64 sectors).
Main directory size = 1,004 files.
Allocated bytes = 124,427,008.
Available bytes = 142,959,872.

Disk I label "Image".
File system = THEOS.
Capacity = 2,097,152 (128 cylinders, 2 heads, 32 sectors).
Main directory size = 964 files.
Allocated bytes = 64,512.
Available bytes = 2,032,640.

Disk M label "Ram_Disk".
File system = THEOS.
Capacity = 2,097,152 (256 cylinders, 1 heads, 32 sectors).
Main directory size = 524 files.
Allocated bytes = 53,760.
Available bytes = 2,043,392.
```

Mode 3—Formats a disk volume. Formatting involves the physical formatting of every sector on the disk volume and the building of a THEOS disk volume, including the disk label information and an empty root directory. See “[Formatting Disks](#)” on page [309](#).

Mode 4—Displays the allocation information about *file*. This information includes the physical location of the directory entry for *file* and a list of all of the extents of *file* showing their locations and sizes in sectors.

```

>disk master.control:s
"MASTER.CONTROL:S" directory is at sector 92; offset 0
                        64                        137

>disk julian.*
"JULIAN.COMMAND:S" directory is at sector 97; offset 128
                        174                        376,638
"JULIAN.BASIC:S" directory is at sector 164; offset 128
                        1                        12,516
                        4                        19,898
"JULIAN.ORIGINAL:S" directory is at sector 203; offset 0
                        5                        9,157
"JULIAN.OLDCMD:S" directory is at sector 245; offset 128
                        174                        376,464

```

The first number displayed on the second line is the count of the number of sectors used by the file starting at the sector number shown on the right. Thus, in the above example, the MASTER.CONTROL:S file has a single extent that uses 64 sectors starting at sector number 137.

Options

ACCOUNT This option has meaning only when used with the [MAP](#) option. It restricts the map display to those files owned by the current account. Compare the following example with the example shown with the [MAP](#) option.

```

>logon acct9

>disk i (map account

Sectaddr    Count  Ext  File-name
      486         4    ACCT9\SAMPLE.DIRECT
      490        37    ACCT9\SAMPLE.INDEXED
      527       198    ACCT9\SAMPLE.KEYED

```

BOOT Copies the file SYSTEM.TEOS32.BOOTER1 to sectors 0–3.

CLEAN Cleans all of the unallocated sectors on the disk by writing zeros to them. This option would be appropriate if you want to insure that all erased files are truly erased.

Normally when a file is erased, its directory entry is marked as deleted and the disk space is returned to the free space map. The data in the file still resides on the disk and could be accessed by someone using the patch command. The data is not destroyed until the space is overwritten by another file.

This option destroys the contents of all erased files.

CLEAR

Clears the disk of all files and directory entries. The root directory is rebuilt as an empty directory using its current allocated size. Use the [SIZE](#) option to clear the directory to a different size.

Since this is a very destructive operation, you are asked to confirm this request before the disk is cleared.

```
>disk f (clear
```

```
Ok to erase all files on disk F (Y/N)
```

FAST

Displays a quick disk status. Unlike the [SHOW](#) option, files are not counted and misallocations are not checked. This is the default option when no options are indicated

```
>disk j
Disk I label "Image".
File system = THEOS.
Capacity = 1,048,576 (64 cylinders, 2 heads, 32 sectors).
Main directory size = 964 files.
Allocated bytes = 941,568.
Available bytes = 107,008.
```

Compare this display with the display from the [SHOW](#) option.

The *drive* may be specified with the wild card *.

FIX

Attempts to correct disk misallocations.

FRAGMENT Displays a list of all of the files that are fragmented or use more than one extent.

```
>disk s (fragment multiuse
```

```
Areas  File name
  2    SYSTEM\ATTACH.OUTPUT
  3    SYSTEM\B2OUPD.IMG
  2    SYSTEM\DIALDIR/COMPUSRV.BACKUP
  4    SYSTEM\INCLUDE
  2    SYSTEM\INCLUDE/TABULATE.BASIC
  ...
```

```
Total fragmented files: 165 out of 9,939.
2 percent of files on disk are fragmented.
```

A file may be fragmented for one of three reasons: When it was originally created there was insufficient contiguous disk space for the entire file; the file grew in size and, at that time, the

next free space available was not contiguous with the existing file; the file is larger than the maximum size for one extent.

There is a slight performance degradation with fragmented files but it is insignificant in most situations.

FREE

Displays the list of available sectors for the disk.

```
>disk j (free

      Sector    Count
      867        79
    2,189       308
    2,891        27
    4,092         4
```

HEX

This option can be used with the [MAP](#) and [FREE](#) options to cause the numbers to display in hexadecimal.

```
>disk k (map hex

Sectaddr      Count Ext File-name
      0          4   ** boot
      4          1   ** label
      5       0x35   ** main directory
    0x3A          1   ** free space map
    0x3B     0x1FC5   ** available sectors
```

LABEL

Changes the disk volume label. The new label can either be specified on the command line by following the [LABEL](#) keyword with an equal sign and the new label or, when this is not done, you are asked for the new label.

```
>disk f (label

Enter disk label: Sample

>disk f (label=Example
```

Disk labels may use upper and lowercase letters, digits, the underscore, space or period characters. Labels are limited to a maximum of eight characters.

After the label is changed a disk mount operation is performed. See “[Mount](#)” on page [414](#).

MAP

Displays a usage map of all sectors on the drive volume. This map display is in disk sector number sequence.

```
>disk i (map)
```

Sectaddr	Count	Ext	File-name
0	4		** boot
4	1		** label
5	241		** main directory
246	1		** free space map
247	4		SYSTEM\TEST.DIRECT
251	37		SYSTEM\TEST.INDEXED
288	198		SYSTEM\TEST.KEYED
486	4		ACCT9\SAMPLE.DIRECT
490	37		ACCT9\SAMPLE.INDEXED
527	198		ACCT9\SAMPLE.KEYED
725	7,467		** available sectors

The “Ext” column has numbers in it only for those files that are fragmented. See the [FRAGMENT](#) option description on page 299.

There are several identifiers used in the “File-name” column to identify disk space used by non-files. These identifiers are described in the following table.

Message	Meaning
** boot	The first four sectors of every disk are reserved for a “bootstrap loader” program.
** label	The fifth sector of every disk contains disk label information.
** main directory	The “root” directory of the drive.
** free space map	One or more consecutive sectors identifying locations on the drive that are not in use by any file. Multiple free space map sections will be used on a drive that has had files deleted or extended, causing additional areas of the disk to become available.
** available sectors	One or more contiguous sectors that are not assigned to any file or other function.

Message	Meaning
** missing sectors	THIS IS AN ERROR MESSAGE! It identifies an area of the disk that is not accounted for by any file or free space map.
** overlaps	THIS IS AN ERROR MESSAGE! It identifies an area of the disk that is “owned” by two or more files.

If disk errors are reported, they should be corrected as soon as possible!

MULTIUSER Tells Disk not to check for multiuser operation before performing the requested function. When Disk is instructed to **FORMAT** or **FIX** a public disk, it requires single user mode. If other users are logged onto the system, it displays the message: “Must be single user or private volume.”

Using this option tells Disk to not restrict the function to single-user operation (the message is still displayed). See “[Cautions](#)” on page 310.

NOASK When used with the **CLEAR** option, suppresses the request “Ok to erase all files...”. Use this option only if you are certain that you have specified the proper disk.

OPTIMIZE Cleans up and optimizes the disk’s directories and libraries by removing deleted entries (erased files), sorting subdirectories and reordering the files in subdirectories. A slight performance increase can result from this optimization.

PRT*nn* Indicates that Disk is to print the information on the attached printer number *nn* instead of the standard output device or console.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

SEEK Tests the disk’s access and reliability by performing continuous random seeks and reading the data found there.

Note that disk caching is not bypassed. For small disk volumes with disk caching enabled, this means that most if not all of the disk testing will be satisfied by the disk cache memory and

is not a true test of the disk access. In this situation, disk caching should be disabled. See “[Cache](#)” on page 200.

SHOW

Analyzes the disk and displays the disk status. All files are counted and the allocations for each file are checked. Under or over-allocated amounts are reported.

```
>disk s (show
Disk S label "THEOS".
Archived to disk "Thursday" on 8 July 1999, at 16:25.
File system = THEOS/4G.
Capacity = 1267,386,880 (255 cylinders, 64 heads, 64 sec).
Main directory uses 309 out of 1,004 files.
Total files = 9,939.
    73 directories.
    295 libraries.
    8,080 stream files.
    1,164 program files.
    227 indexed files.
    10 keyed files
    90 relative files.
Allocated bytes = 124,444,928.
Available bytes = 142,941,952.
```

Note that, if a misallocation is reported and other users are active on the system, the misallocation may be okay if it is due to another user allocating or deallocating disk space at the time this status report is generated. Check the status again after all users are logged off or at command prompts.

The “File system” displays as either “THEOS” or “THEOS/4G.” The THEOS file system is used for all floppy disks and ram disks and for hard disk partitions created with the THEOS operating system, version 3.2. THEOS/4G file systems apply only to hard disk partitions created with THEOS 32 version 4.

The *drive* may be specified with the wild card *.

SIZE

This option can be used with the [CLEAR](#) or [SHOW](#) option.

When used with the [CLEAR](#) option, you must specify the desired directory size immediately following this keyword.

```
>disk a (clear size 200

Ok to erase all files on disk A (Y/N) Y

>disk a
Disk A label "Image".
File system = THEOS.
Capacity = 2,097,152 (128 cylinders, 2 heads, 32 sectors).
```

```

Main directory size = 212 files.
Allocated bytes = 15,104.
Available bytes = 2,082,048.

```

When used by itself or with the **SHOW** or **FAST** options, Disk sets its return code to the number of megabytes of space available.

```
>set rdymsg on
```

```
RC = 0, 13:31:05, ET = 0.00, CPU = 0.040
```

```
>disk a (show size
```

```
Disk I label "Image".
```

```
File system = THEOS.
```

```
Capacity = 2,097,152 (128 cylinders, 2 heads, 32 sectors).
```

```
Main directory uses 0 out of 212 files.
```

```
Total files = 0.
```

```
    0 directories.
```

```
    0 libraries.
```

```
    0 stream files.
```

```
    0 program files.
```

```
    0 indexed files.
```

```
    0 keyed files.
```

```
    0 relative files.
```

```
Allocated bytes = 15,104.
```

```
Available bytes = 2,082,048.
```

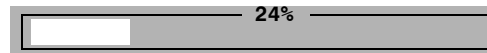
```
RC = 2, 13:31:21, ET = 0:00, CPU = 0.400
```

VERIFY

Tests a disk's readability by reading every sector on the disk. As each sector is read, its location is displayed and a progress bar is updated to reflect the amount of the disk that has been verified so far.

```
>disk s (verify
```

```
Cylinder: 60 Head: 40 24%
```



Disk caching is bypassed during this test. Any errors detected are displayed on both standard output and standard error devices.

Note: The **VERIFY** option causes the Disk command to first verify that the disk contains a THEOS File System. If it contains another file system (DOS, FAT32, CDROM, *etc.*) it reports “Must be THEOS file system” and exits without verifying the sectors on the disk.

**Format
Options**

BOOT Copies the file SYSTEM.TEOS32.BOOTER1 to sectors 0–3. Refer to page 727 in Appendix D: “[System Files](#)” for a description.

CYLINDERS Specifies the number of cylinders or tracks per surface on the disk. The number is specified following the CYLINDERS keyword. Refer to “[Formatting Disks](#)” on page 309 for recommendations on the number cylinders.

DENSITY Specifies the density of each sector on the disk and whether or not the disk is removable or fixed. The code is specified immediately following the DENSITY keyword.

Removable	Fixed	Meaning
1		128 bytes per sector, all surfaces.
2	10	256 bytes per sector, all surfaces.
3		Cylinder 0: 128 bytes per sector; remainder is 256 bytes per sector.
4		Cylinder 0, head 0: 128 bytes per sector; remainder is 256, 512 or 1024 bytes per sector.
	13	2048 bytes per sector, all surfaces.
	14	1024 bytes per sector, all surfaces.
7	15	512 bytes per sector, all surfaces. (IBM PC format)

Table 17: Disk Density Format Codes

Note: Only code 7 and 15 are used by standard THEOS disks.

HEADS Number of heads or recordable surfaces on the disk. Refer to “[Formatting Disks](#)” on page 309 for recommendations on the number of heads to use.

INCREMENT Specifies the relationship between physical and logical sectors in each cylinder. An increment value of 1 indicates that sectors are accessed consecutively (1,2,3,4, *etc.*); a value of 2 means every other sector is read (1,3,5,7, ... 2,4,6,8, *etc.*); and so on.

The increment value need only be specified on some older technology drives. Most newer drives use “on-board logic” that automatically sets the increment or interleave values.

LABEL Specifies the disk volume label for the newly formatted or built disk. The new label can either be specified on the command line by following the LABEL keyword with an equal sign and the new label or, when this is not done, you are asked for the new label after the disk is formatted or built.

Disk labels may use upper and lowercase letters, digits and the underscore character.

This is a default option whenever a disk is formatted or built.

SECTORS Specifies the number of sectors on each track. Refer to “[Formatting Disks](#)” on page 309 for recommendations on the number of sectors to use.

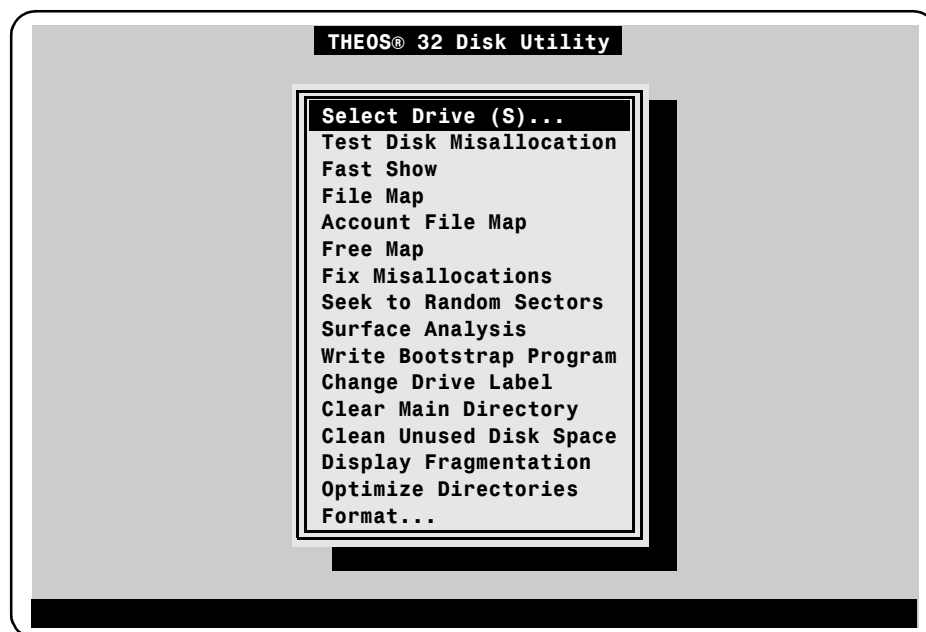
SIZE Specifies the size of the main directory. This size cannot be changed again without reformatting, building or clearing the disk.

For best performance, the size of the main directory should be at least twice as large as the expected needs of the disk. It is not necessary to specify a size when formatting a disk for usage by the [Archive](#) or the [Backup](#) because they can recreate the directory.

TRACKS Synonym to the [CYLINDERS](#) option.

Disk Menu

When no drive code or options are specified, the Disk command menu is displayed:



Commands

Use the normal menu selection keys to select the desired functions or tests. These keys are described in “[Using Menus](#)” on page 75.

Select Drive... Selects the drive that will be used in this next menu selection. Unless this item is used to select a different drive, the default drive of S is used. You cannot clear or format the system disk.

Test Disk Misallocation. Performs a Disk ([SHOW](#) on the selected drive and then exits. Refer to the [SHOW](#) option description on page 303.

Fast Show. Performs a Disk ([FAST](#) on the selected drive and then exits. Refer to the [FAST](#) option description on page 299.

File Map. Performs a Disk ([MAP](#) on the selected drive. Refer to the [MAP](#) option description on page 301.

Account File Map. Performs a Disk ([MAP ACCOUNT](#) on the selected drive and then exits. Refer to the [ACCOUNT](#) option description on page 298.

Free Map. Performs a Disk ([FREE](#) on the selected drive and then exits. Refer to the [FREE](#) option description on page 300.

Fix Misallocations. Performs a Disk ([FIX](#) on the selected drive and then exits. Refer to the [FIX](#) option description on page 299.

Seek to Random Sectors. Performs a Disk ([SEEK](#) on the selected drive and then exits. Refer to the [SEEK](#) option description on page 302.

Surface Analysis. Performs a Disk ([VERIFY](#) on the selected drive and then exits. Refer to the [VERIFY](#) option description on page 304.

Write Bootstrap Program. Performs a Disk ([BOOT](#) on the selected drive and then exits. Refer to the [BOOT](#) option description on page 305.

Change Drive Label. Performs a Disk ([LABEL](#) on the selected drive and then exits. Refer to the [LABEL](#) option description on page 300.

Clear Main Directory. Performs a Disk ([CLEAR](#) on the selected drive and then exits. Refer to the [CLEAR](#) option description on page 299. You cannot clear the system disk.

Clean Unused Disk Space. Performs a Disk ([CLEAN](#) on the selected drive and then exits. Refer to the [CLEAN](#) option description on page 298.

Display Fragmentation. Performs a Disk ([FRAGMENT](#) on the selected drive and then exits. Refer to the [FRAGMENT](#) option description on page 299.

Optimize Directories. Performs a Disk ([OPTIMIZE](#) on the selected drive and then exits. Refer to the [OPTIMIZE](#) option description on page 302.

Format... Performs a Disk ([FORMAT](#) on the selected drive and then exits. Refer to “Formatting Disks,” below. Before selecting this function, you must use the “Select Drive” function to select a drive other than the S drive. You cannot format the current system disk.

Formatting Disks

Floppy diskettes and removable hard disks can be formatted by using the Disk menu described on page 307 or Mode 3 of the Disk command. Use the [Setup](#) to perform initial partitioning and setup of hard disks and to format floppy diskettes or removable hard disks in bulk.

■ Floppy Disk Drives

Before a floppy diskette can be used for the first time by THEOS, it must be formatted or built. A “quick format” or BUILD can be done on an unused, preformatted diskette or a previously formatted diskette. This writes the THEOS file system information on the disk with an empty directory.

Floppy disk drives generally support multiple densities and capacities of diskettes. To format a floppy diskette using command-line options only, you must use the [CYLINDERS](#), [INCREMENT](#) and [SECTORS](#) options,. Use one of the following sets of values:

Cylinders	Heads	Sectors	Capacity
80	2	72	2.88MB 3½"
80	2	36	1.44MB 3½"
80	2	30	1.2MB 3½"
80	2	18	720KB 3½"
80	2	30	1.2MB 5¼"
80	2	18	720KB 5¼"
40	2	18	360KB 5¼"

Table 18: Floppy Disk Formats

For instance:

```
>disk f (format cylinders 80 head 2 sector 36
```

If all of these parameters are not specified on the command line, you are asked to select one of the known formats supported by the drive:

```
>disk f (format
```

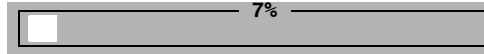
Code	Cyls	Hds	Sects	Capacity
1	80	2	36	1.44 MB 3¾"
2	80	2	30	1.2 MB 3¾"
3	80	2	18	720 MB 3¾"

```
Enter format code: 1
```

In either situation, you are next asked to put the diskette in the drive. Pressing any key other than **[Esc]** starts the formatting process:

Insert diskette to be formatted: **[Enter ↵]**

Cylinder: 5 Head: 1



After the diskette is formatted, Disk needs a label. This label can be specified on the command line with a **LABEL=** option. When it is not supplied on the command line, you are asked for the label. After the label is known, the diskette is built by writing the label sector, empty directory and free space map. The size of the directory can be specified on the command line with the **SIZE** option or, if that is not done, a default size is used.

Although space is reserved on the disk for a bootstrap loader (sectors 0–3), it is not written to the disk unless the **BOOT** option is used.

After a diskette is formatted and the THEOS information is written to it, you are asked if you want to format another diskette in the same drive. You must enter a **[Y]** to format another disk. To terminate formatting, respond with **[Enter ↵]** or an **[N]**.

Quick Formatting Floppy Diskettes

To “quick format” a diskette or batch of diskettes, use the **SETUP FLOPPY** command. A quick format does not format or verify the readability of the disk but merely writes the THEOS-specific information needed to use the disk. This information includes the THEOS disk label sector, root directory and free space map. You can quick format a diskette if it is already formatted.

Defaults

The **FAST** option is the default option when only a drive code is specified.

Cautions

The **MULTIUSER** option tells the Disk command to not check whether or not other users are logged on or active. It does not prevent those other users from performing operations that change the database that this Disk command might be using.

Do not use the **MULTIUSER** option unless you are sure that any other users logged onto the system are not going to be using the drive that you are modifying.

Restrictions The Disk command requires a privilege level of four.

To [CLEAR](#), [FIX](#), [FORMAT](#) or [OPTIMIZE](#) a disk, it must be either a privately attached drive or a publicly attached drive with the system in single-user mode or with the [MULTIUSER](#) option specified.

The current system disk (attached as drive S) cannot be formatted. To format that drive you will have to [System](#) over to another drive or reboot from another drive.

The Disk command cannot show the status of a CD-ROM disc or of a file on a CD-ROM disc, nor can it be used on a DOS-formatted disk except to reformat it as a THEOS-formatted disk.

See also [Attach](#), [MakeBoot](#), [Setup](#), [System](#), [Tape](#)

Echo Command

The Echo command defines a file or device that will be used for subsequent echoing of console displays. Alternately, it echoes the remainder of the command line to the standard output device.

1 ECHO *file*

2 ECHO *device*

3 ECHO

4 ECHO *text*

file » file name with optional path

device » name of attached output device such as COM1 or PRT3

text » any text that does not look like a file name

Operation

Mode 1—Indicates that *file* is to be opened to receive console display echoes. To specify a file that has no file type, be sure to use the period terminator after the file name.

```
>echo console.output
```

```
>echo example.
```

Console echoing is not enabled with this command. That is done by entry of **[Break]**, **[P]**.

Mode 2—Indicates that the output device is to be opened as the file to receive console display echoes. Output devices that may be used include COM1–COM16, PRT1–PRT64 and TAPE1–TAPE4. The specified device must be currently attached. If the device is a public, non-spooled device and it is in use by another user, the Echo command will wait until it is closed by that other user before proceeding.

```
>echo prt12
```

Console echoing is not enabled with this command. That is done by entry of **[Break]**, **[P]**.

Mode 3—Closes any file or device previously specified with Mode 1 or 2 of the Echo command. Any echoing enabled with **[Break]**, **[P]** is disabled.

Mode 4—Displays *text* on the console. If console echoing was enabled with Mode 1 or 2 of the Echo command and `[Break]`, `[P]` has been entered, this text is also echoed to the echo file.

```
>echo The following is for demonstration purposes only
The following is for demonstration purposes only
```

```
>echo example
example
```

Make sure that *text* does not look like a *file* or *device* name. Any text that contains two or more words or tokens will not be misinterpreted. If necessary, add a “dummy” token.

```
>echo " " prt1
prt1
```

Notes

Except for Mode 4, the Echo command does not actually echo anything to a file. After a file or device is defined with the Echo command, you must enter a `[Break]`, `[P]` to start and stop the echoing process.

The Echo command is normally used with I/O redirection or in EXEC programs to add text to a log file.

```
>echo "Begin daily archive" >> activity.log
```

See also

I/O redirection, described on page 51 in the “Fundamentals” section.

Eject Command

The Eject command ejects the media in a removable hard drive, tape or CD-ROM drive, or it ejects a page in a printer.

Commands

```
1 EJECT device
```

```
device          »   name of tape or CD-ROM device or output device such as PRT3
```

Operation	<p>When <i>device</i> is an attached CD-ROM drive, the drive's disk tray is opened.</p> <p>When <i>device</i> is an attached tape drive, the tape cartridge is ejected.</p> <p>When <i>device</i> is a removable disk drive, the disk is ejected.</p> <p>When <i>device</i> is an attached printer, a page eject or form-feed is output to that device.</p>
Cautions	<p>If the device is publicly attached, you may be interfering with the operation of another user's program.</p>
Restrictions	<p>The removable disk drive, tape or CD-ROM must support software-controlled ejecting.</p>
See also	<p>Mount</p>

Erase Command

This command erases one or more files from disk.

- 1 **ERASE** *file* (*options*
- 2 **ERASE** (*options*
- 3 **ERASE** *file* (**FILES** *options*

<i>file</i>	»	file name with optional path; may contain wild cards	
<i>options</i>	»	CLEAN	QUERY
		NOQUERY	TYPE
		NOTYPE	

Operation **Mode 1**—Attempts to erase *file* and displays the result of the attempt.

```
>erase sample.data
"SAMPLE.DATA:S" erased.
One file erased, 3,840 bytes recovered.
```

Unless the **NOTYPE** option is used, a result message is displayed after all files are erased. This shows the number of files erased and the number of bytes recovered due to erasing those files.

When *file* is a subdirectory name, the Erase command will only erase the directory if it contains no files. Use the **Rmdir** command to erase directories that contain files or erase the files first and then erase the directory.

Mode 2—Invokes the interactive mode of the Erase command. Because no files are specified on the command line, you are prompted to enter the file descriptions to erase.

```
>erase
Enter file name list, terminate with empty line.
?OUTPUT.LOG
"OUTPUT.LOG:S" erased.
?SAMPLE.OUTPUT
"SAMPLE.OUTPUT:S" erased.
?* FILE
Ok to erase "ACCOUNTS.FILE:S" (Yes,No,All) Y
"ACCOUNTS.FILE:S" erased.
Ok to erase "CUSTOMER.FILE:S" (Yes,No,All) A
"CUSTOMER.FILE:S" erased.
"EXAMPLE.FILE:S" erased.
"LEDGER.FILE:S" erased.
```

As illustrated, this interactive mode allows any file specification to be entered, including wild cards.

Mode 3—*file* is an ASCII stream file containing one file description per line. Each file description in *file* is erased. As each file is erased its file description is displayed (unless the **NOTYPE** option is specified). When the file description in *file* contains wild cards, you are queried for permission to erase each file that matches the specifications (unless the **NOQUERY** option is specified).

This mode of the Erase command is convenient when one or more sets of files are repetitively being erased. Merely edit a file containing the file description, such as:

```
>edit daily.erase
work.master:s
work.history:s
work.invoices:s
work.ledger.*:s
temp*.*:s
sort*.*:s
/programs/program.backlib.*:s

>erase daily.erase (file noquery notype
```

Options

CLEAN Specifies that the contents of *file* are cleaned by writing zeros to every byte of the file. The file is erased after it is cleaned.

NOQUERY Tells Erase to not ask for confirmation before erasing each file. This is a default option when wild cards are not used.

```
>erase gl.* (noq
"GL.MASTER:S" erased.
"GL.JOURNAL:S" erased.
"GL.HISTORY:S" erased.
```

To disable this option use the **QUERY** option.

NOTYPE Tells Erase to not display the results of each file erased on the standard output device. The general result message (the “nn files erased, nnn bytes recovered.” message displayed before exiting Erase) is also suppressed with this option.

```
>erase gl.* f (not
Ok to erase "GL.MASTER:S" (Yes,No,All) Y
Ok to erase "GL.JOURNAL:S" (Yes,No,All) Y
Ok to erase "GL.HISTORY:S" (Yes,No,All) Y
```

To disable this option use the **TYPE** option.

QUERY Tells Erase to “query” or ask if each file matching the file specifications is to be erased. This is a default option when wild cards are used. To disable this option use the [NOQUERY](#) option.

```
>erase *.data i
Ok to erase "CUSTOMER.DATA:S" (Yes,No,All)
```

When the “Ok to erase” question is asked, you may respond with a **Y** for yes, **N** for no or **A** for all. Responding with **A** means yes to this file and all remaining files are erased without being queried. Respond with **Esc** to cancel the erase operation.

TYPE A default option that tells Erase to display the results of each file erased on the standard output device. This display can be redirected.

```
>erase *.*:s j (noquery
"/DEVELOPE.EXEC:S" erased.
"/GRAPH.SAVE1:S" protected.
"/GRAPH.DATA:S" erased.
2 files erased, 15,872 bytes recovered.
```

To disable this option use the [NOTYPE](#) option.

Notes If *file* is a “typeless” file description, there is no default library defined and the environment variable [FILETYPE](#) is defined, the value of [FILETYPE](#) is appended to *file* to form a complete file description with file name and file type. To erase a typeless file you should specify the file description with a period terminator. See “[FILETYPE](#)” on page 105 for more information about this environment variable.

Defaults [QUERY](#) and [TYPE](#) are default options.

Restrictions You may erase files that are owned by the current account if the files are not erase protected. You may erase files owned by another account if they are not erase protected and they do not have shared read or shared write protection enabled.

A library cannot be erased if it has any member files. The members must be erased first and then the library may be erased.

A subdirectory cannot be erased if it has any member files. The members must be erased first and then the directory may be erased.

A file that is currently open by another user may not be erased.

See also [Rename](#), [RmDir](#)

Exit Command

This command exits from the [Shell](#) environment or, if you are not using [Shell](#) environment, this command logs off of the system.

Commands

EXIT

Operation	<p>The specific operation of this command depends upon your current environment:</p> <ul style="list-style-type: none">▶ In a Shell environment: Exits the shell, returning to the prior environment.▶ Not in Shell environment but connected via TWS, NetTerm or Telnet: Logoff the current account and disconnect.▶ Otherwise: Logoff the current account.
Notes	<p>The Shell environment is normally invoked from another environment such as WindoWriter, THEO+COM, <i>etc.</i> When Exit is used in this situation, control returns to the environment in effect before the Shell was invoked.</p>
See also	<p>Logoff, Shell</p>

Expand Command

The Expand command extracts and expands a file from a “compression library” created with the [Compress](#) command.

Commands

1

`EXPAND`

`compress-file destination (options`

2

`EXPAND`

`compress-file destination file... (options`

3

`EXPAND`

`compress-file destination file (FILES options`

`compress-file`

»

file name with optional path

`file`

»

file name of compressed file to be expanded, with optional path; may contain wild cards

`destination`

»

destination file name with optional path; may contain wild cards; may be a simple drive specification

`options`

»

`NEWFILE`

`OLDER`

`QUERY`

`date1`

`NOQUERY`

`OLDFILE`

`REPLACE`

`date2`

`NOTYPE`

`PASSWORD`

`TYPE`

Operation **Mode 1**—Each of the compressed files in *compress-file* is extracted, expanded, and written to *destination*. *destination* may be a simple drive code specification, in which case, the files are expanded and written to that drive using their saved path information.

```
>expand programs s
/bmenu.basic:s
/func_key.basic:s
/menu.basic:s
/model.basic:s
/paint.basic:s
```

Mode 2—Each *file* specified on the command line is extracted from *compress-file*, expanded and stored on *destination*.

```
>expand programs s menu.basic paint.basic (replace
/menu.basic:s
/paint.basic:s
```

If the files in *compress-file* were compressed with the SUBDIR option in effect, you must specify the original path for each file.

```
>expand vertical.programs s /vertical/progs/menu.basic
```

Mode 3—The files listed in *file* are extracted from *compress-file*, expanded and stored on *destination*. *file* must be an ASCII stream file containing one file description per line. The `SELECTED.FILES` and `SELECTED.EXEC` files created by `FileList` and the `FOUND.EXEC` created by `Look` can be used for this specification file (see “[The EXEC and FILES Options](#)” on page 338). You may also create the specification file with an editor or application program.

For instance, the `FileList` command is used to create a list of files:

```
>filelist a (10/1/96 10/8/96 exec
```

A file now exists that lists all of the files on the A disk that have been changed between 10/01/1996 and 10/08/1996. The following command will expand these files from their compressed form:

```
>expand old.files a selected.exec (files
```

Options

NEWFILE A default option that tells Expand to expand files if *file* does not already exist. Note: If `QUERY` is used, you are not queried about files if the destination file name already exists.

To disable this option use the `OLDFILE` or the `REPLACE` options.

NOQUERY Tells Expand to not ask for confirmation before expanding each file. This is a default option when `Mode 1` is used or when wild cards are not used.

```
>expand programs s
/bmenu.basic:s
/func_key.basic:s
/menu.basic:s
/model.basic:s
/paint.basic:s
/phonenbr.basic:s
```

To disable this option use the `QUERY` option.

NOTYPE Tells Expand to not display the results of each file expanded on the standard output device. The general result message (the “nn files expanded.” message prior to exiting Expand) is also suppressed with this option.

```
>expand gl.compress s gl.* (not
Ok to replace "/GL.MASTER:S" (Y|N|A|G)? Y
Ok to replace "/GL.JOURNAL:S" (Y|N|A|G)? Y
Ok to replace "/GL.HISTORY:S" (Y|N|A|G)? Y
```

To disable this option use the `TYPE` option.

OLDER

Expands the file to the *destination* only if the destination file's last change date is older than the compressed file's or if the destination file does not exist. The **REPLACE** option is implied by this option. Note: If **QUERY** is used, you are not queried about files if the destination file is newer.

This option would normally be used when a compressed library file is created and ported to another system. The **OLDER** option is used to expand only those files that need to be updated on the second system.

OLDFILE

Indicates that Expand should expand a file only if the destination file name already exists. This is the opposite of the **NEWFILE** option and implies a **REPLACE** option. Note: If **QUERY** is used, you are not queried about files if the destination file name does not exist.

Note that only the destination file name is checked. It could be a totally different file. For instance, an existing command program could be replaced by a stream file or indexed file.

To disable this option use the **NEWFILE** option.

PASSWORD

A password is specified following the **PASSWORD** option keyword. The files are expanded only if they were originally compressed with this password.

If a file is compressed with a password and it doesn't match the password specified with this Expand command or no password is specified with the Expand command, the file is not expand.

```
>compress private.files *.text (notype password aardvark
```

```
>expand private.files a (replace noq
```

```
Fatal error: File: "mydata.text:a" was corrupted or incorrect password.
```

QUERY

Tells Expand to “query” or ask if each file matching the *file* specification is to be expanded. This is a default option when wild cards are used.

```
>expand compress.data i
```

```
Ok to replace "/CUSTOMER.DATA:I" (Y|N|A|G)?
```

When the “Ok to replace” question is asked, you may respond with a **Y** for yes, **N** for no, **A** for all or **G** for go. Responding with **A** means yes to this file and all remaining files are

included without being queried. Respond with **G** or **Esc** to cancel the expand operation.

To disable this option use the **NOQUERY** option.

REPLACE Allows a file to be expanded even if it already exists on the destination drive. Normally, when the destination file name already exists, Expand will not perform the expand. This option tells Expand that it is okay if it already exists and that it should erase the existing file and replace it with a expanded file.

If the destination file does not exist, this option has no effect: The file is created just as if the REPLACE option was not specified.

```
>expand data.compress f *.data (replace noquery
/customer.data:f
/history.data:f
/accounts.data:f
```

The REPLACE option is implied by both the **OLDER** and **OLD-FILE** options. However, the **OLDFILE** option will not copy a file unless the destination file name already exists.

_TYPE A default option that tells Expand to display the results of each file expanded on the standard output device. This display can be redirected.

```
>expand programs s (noquery
/program.source.bmenu:s
/program.source.func_key:s
/program.source.menu:s
/program.source.model:s
/program.source.paint:s
/program.source.phonenbr:s
```

To disable this option use the **NOTYPE** option.

date1 Expands a file only if the *file* in *compress-file* has a last change date that is greater than or equal to this date. That is, if the compressed file was changed on or after this date.

This option may be used with the *date2* option.

```
>compress programs ; display compressed files
```

File name	Date	Time	Size	Rate
bmenu.basic	13Jan96	11:29	774	32%
func_key.basic	01Oct95	19:06	1,339	68%
menu.basic	20Dec95	14:41	2,709	43%
model.basic	11Nov95	11:53	1,618	69%
paint.basic	17Jan96	12:56	356	25%
phonenbr.basic	25Jun96	14:17	6,726	48%

```
>expand programs s (1/1/96
```

```
bmenu.basic
paint.basic
phonenbr.basic
```

The above command will expand only those files that have been created or changed since January 1, 1996.

date2

Expands a file only if the compressed file's last change date is less than or equal to this date. That is, if the compressed file was changed on or before this date. May only be specified by first specifying the *date1* option.

```
>expand programs s (10/15/96 10/30/96
```

This command expands only those files that have been created or changed since October 14, 1996, but not any files that were created or changed after October 30, 1996.

To specify a *date2* when you don't care about *date1*, use a date of 1/1/86 for the *date1* option. This is the earliest date maintained by the THEOS file system.

```
>expand programs s (1/1/0 12/31/95
```

```
func_key.basic
menu.basic
model.basic
```

Here, since the *date1* specification is 1/1/86, all files created or changed prior to January 1, 1996, are expanded.

See also

[Archive](#), [Compress](#), [CopyFile](#), [Restore](#), [TBackup](#)

File Command

The File command analyzes a file and reports on its organization and general function.

```
1  FILE  file...
2  FILE  file ( FILES

_____
file                »   file name with optional path
```

Operation

Mode 1—Each *file* specified on the command line is examined for its organization (program command, stream, indexed, *etc.*) and its contents (program object, formatted records, program source, *etc.*). The result of this analysis is output as a message on the standard output device.

```
>file system.theos32.*
"SYSTEM.THEOS32.CLASS60:S" is a Terminal Class definition file.
"SYSTEM.THEOS32.DEV34:S" is a 32-bit program file.
"SYSTEM.THEOS32.DEV64:S" is a 32-bit program file.
"SYSTEM.THEOS32.CRTCFG:S" is a stream file.
...
```

In addition to using wild card specifications, more than one *file* may be specified on the command line.

```
>file *.data *.text system.*.* sample.basic
"CUSTOMER.DATA:S" is an indexed file with formatted data.
...
```

Mode 2—The files listed in *file* are analyzed. *file* must be an ASCII stream file containing one file description per line. The SELECTED.FILES and SELECTED.EXEC files created by [FileList](#) and the FOUND.EXEC created by [Look](#) can be used for this specification file (see “[The EXEC and FILES Options](#)” on page [338](#)). You may create the file with an editor or application program.

For instance, [FileList](#) is used to create a list of files to be examined:

```
>filelist *.data:a (exec
>filelist a not *.data:a (10/1/95 exec append
```

A file now exists that lists all of the “data” files and all files that have been changed since 10/01/1995. The following command checks these files for organization and content:

```
>file selected.exec (file
```

Notes

If *file* is a “typeless” file description, there is no default library defined and the environment variable `FILETYPE` is defined, then the value of `FILETYPE` is appended to *file* to form a complete file description with file name and file type. To report on a typeless file you should specify the file description with a period terminator. See “`FileType`” on page 342 .

file may be a subdirectory name.

The file types recognized by the File command include:

File Type	Message Displayed
Library	“ <i>filename</i> ” is a library.
Directory	“ <i>filename</i> ” is a subdirectory.
Program	“ <i>filename</i> ” is a 16-bit real mode program file. “ <i>filename</i> ” is a 16-bit program file. “ <i>filename</i> ” is a 32-bit program file.
Data	“ <i>filename</i> ” is an indexed file with formatted data. “ <i>filename</i> ” is an indexed file with text data. “ <i>filename</i> ” is a keyed file with formatted data. “ <i>filename</i> ” is a keyed file with text data. “ <i>filename</i> ” is a relative file with formatted data. “ <i>filename</i> ” is a relative file with text data.
Stream	“ <i>filename</i> ” is a stream file. “ <i>filename</i> ” is a stream file with binary data. “ <i>filename</i> ” is a stream file with textdata. “ <i>filename</i> ” is an empty file. “ <i>filename</i> ” is a Printer Class Definition file. “ <i>filename</i> ” is a Terminal Class Definition file. “ <i>filename</i> ” is a Keyboard Definition file. “ <i>filename</i> ” is a MAKE source file.
Program source	“ <i>filename</i> ” is a 16-bit BASIC source file. “ <i>filename</i> ” is a 32-bit BASIC source file. “ <i>filename</i> ” is a C source file. “ <i>filename</i> ” is an EXEC source file. “ <i>filename</i> ” is a TINSTALL source file. “ <i>filename</i> ” is a TINSTALL compiled file.
Program object	“ <i>filename</i> ” is a 16-bit object file. “ <i>filename</i> ” is a 32-bit object file.
Script	“ <i>filename</i> ” is a SCRIPT file.
Other	“ <i>filename</i> ” is a COMPRESS library file.

See also

[FileList](#)

FileList Command

The FileList command displays a listing of a disk's directory.

- 1 FILELIST (options
- 2 FILELIST file... (options
- 3 FILELIST file₁... NOT file₂... NOT file₃... (options
- 4 FILELIST drive... (options
- 5 FILELIST * (options

drive	»	optional disk drive attachment letter			
file	»	optional file name with optional path; may contain wild cards			
options	»	ACCOUNT name	FT	PROTECT	SORT5
		APPEND	GROW	PRTnn	SORT6
		ASCII	HIDDEN	PUBLIC	SORT7
		CHECKSUM	KEY	SERIAL	SORT8
		COUNT	MN	SIZE	SORT sequence
		EXEC	NOTYPE	SORT1	TYPE
		FD	OFF	SORT2	VERSION
		FILES	OWNER	SORT3	date1
		FN	PRIVLEV	SORT4	date2

This command produces a directory listing of files. By default, this directory listing includes the file name description (file name, file type, member name and file drive), the last date and time that the file was changed, the file organization or type (program, indexed, library, etc.), the physical size of the file on disk and, for data files, the record and key length.

By using various options, additional information about the file can be displayed including the checksum for the file, number of records in the file, growth factor, program privilege level, protection and attribute codes for the file and program version number.

This directory listing is normally output to the console, but it also can be displayed on one of the attached printers, redirected to a file or device, or output as an EXEC language program or a data file.

Operation

Mode 1—Displays a directory listing of all “flat files” in the current directory of the current account. A flat file is a file that is not a member of a library.

Filelist *.*

10 Oct 1996 4:23pm Page 1

Filename	Filetype	Memname	:	Date	Time	Org.	Size	Rec1	Key1
C32			S	08/14/96	13:45	Directory	4,096		
EXEC			S	07/12/96	11:47	Directory	4,096		
MUBASIC			S	07/12/96	11:47	Directory	4,096		
MUBASIC	SAMPLES		S	11/18/96	09:48	Directory	10,496		
THEO_COM			S	07/12/96	11:47	Directory	4,096		
CHARSET	H		S	08/25/96	18:00	Stream	10,872		
LIBS	BACKUP		S	02/19/96	09:01	Stream	3,541		
LIBS	EXEC		S	02/19/96	09:05	Stream	3,541		
MAKE	FILE		S	02/09/96	16:00	Stream	1,899		
READ	ME		S	02/08/96	13:28	Stream	14,608		
SAMPLES	EXEC		S	07/12/96	11:49	Stream	25		

11 files, 61,365 bytes.

If a default library is defined, then a directory listing of that library is produced.

```
>show library
LIBRARY = SAMPLES.EXECS
```

```
>filelist
```

Filelist of directory /EXEC:S

10 Oct 1996 4:23pm Page 1

Filename	Filetype	Memname	:	Date	Time	Org.	Size	Rec1	Key1
SAMPLES	EXECS		S	07/12/96	13:45	Library	13,568	212	9
		CLEANUP	S	09/05/96	11:47	Stream	177		
		LIBS	S	07/13/96	11:47	Stream	952		
		LIBS_	S	07/13/96	09:48	Stream	25		
		MAKEIT	S	09/10/96	11:47	Stream	218		
		REPEAT	S	01/05/95	18:00	Stream	270		
		SHUTDOWN	S	07/27/96	09:01	Stream	81		
		SYSCMD	S	08/23/94	09:05	Stream	326		
		SYSOBJ	S	02/09/94	16:00	Stream	372		
		USERS	S	11/26/93	13:28	Stream	35		
		WHAT	S	07/12/96	11:49	Stream	87		

11 files, 61,365 bytes.

Mode 2—Displays a directory listing of the specified file description(s). This is a very powerful mode because, by using wild cards and multiple file specifications, it allows many different operations to be performed. For instance:

```
>filelist *.*:*s
```

This displays a list of all libraries, library members and files not related to any library (flat files) that are owned by the current directory of the current account on the S drive.

```
>filelist *.*
```

This displays a listing of libraries and non-library files for all drives in the default drive search sequence. No library members are included.

```
>filelist *.*?*:*s
```

This displays a listing of library member files but not the library itself or any flat files.

```
>filelist *.command *.cmd32.* *.exec
```

This displays a listing of all programs, either flat files (file type is COMMAND), members of command libraries or EXEC language flat files (file type is EXEC).

Mode 3—This mode is similar to [Mode 2](#) with the added ability to specify that groups of files are excluded from the listing. The directory listing produced includes files that match the file specification before the keyword NOT as long as they don't match the file specification following the keyword NOT. For instance:

```
>filelist *.*:*s not system.*:*s (public
```

This displays all entries for files on the S drive excluding those files with a file name of SYSTEM.

Each specification of files that are to be excluded from the listing must be preceded by the NOT keyword.

```
>filelist *.command NOT test*.command *.exec NOT test*.exec
```

Include	Exclude	Include	Exclude

This displays all files with a file type of COMMAND or EXEC excluding those files with a file name starting with TEST.

The NOT file specifications exclude files that might match the specification that immediately precedes the NOT keyword. For instance,

```
>filelist test.* example.* NOT *.data
```

This command only excludes the “*.DATA” files from the “EXAMPLE.*” specification. All files matching “TEST.*” are still included. Also, you may not specify two or more NOT specifications in a row: There must be a normal specification preceding each NOT specification.

Mode 4—This is a shorthand method of specifying that you want all files on the specified drive. The following two commands produce the same results:

```
>filelist s
>filelist *.*:s
```

As with [Mode 2](#) and [Mode 3](#), you can specify multiple drives:

```
>filelist f g
```

Mode 5—This shorthand syntax operates in one of two ways. If the environment variable [FILETYPE](#) is defined, the value of [FILETYPE](#) is appended to the “*” to form a wild card specification for files (see “[Notes](#)” on page [340](#)).

When [FILETYPE](#) is not defined, it produces a directory listing of all files in the current directory and all files on the other drives in the default drive search sequence. This mode differs from [Mode 1](#) in that the default library and the current working directory are not used.

Options

- ACCOUNT** The directory listing includes the files owned by account *name*. This option requires that you be logged onto the system account and that you have a privilege level of five.
- Normally, the directory listing is for the current account only. The [PUBLIC](#) option includes publicly owned files in addition to the files owned by the current account. The [ACCOUNT](#) and [OWNER](#) options are the only method of producing a directory listing of files owned by other private accounts.
- APPEND** Used with option [EXEC](#) or [FILES](#) to append an additional directory listing to the existing [SELECTED.EXEC](#) or [SELECTED.FILES](#) file. If neither [EXEC](#) nor [FILES](#) is specified, then [EXEC](#) is assumed.
- ASCII** Causes the listing to be sorted according to the ASCII collating sequence. The normal sort order used by FileList is a special

alphabetic and numeric sequence that orders file descriptions using numbers in a more natural sequence. Compare the two sort orders below. The list on the left is the normal order used by FileList; the list on the right is the order produced when the ASCII option is used.

CUSTOMER.ROUTE1	CUSTOMER.ROUTE1
CUSTOMER.ROUTE2	CUSTOMER.ROUTE10
CUSTOMER.ROUTE10	CUSTOMER.ROUTE102
CUSTOMER.ROUTE20	CUSTOMER.ROUTE2
CUSTOMER.ROUTE102	CUSTOMER.ROUTE20

CHECKSUM Compute and display the checksum for each file. A checksum is a 16-bit value computed by adding the values of each byte in a file.

The checksum value is output in place of the “Recl” and “Keyl” columns.

COUNT Computes and displays the record count for data files (stream, indexed, keyed and relative). This record count information replaces the “Recl” and “Keyl” columns on the right side of a normal directory listing. Files other than data files are not analyzed and there is no count information displayed for them.

EXEC Indicates that the directory listing is to be output as an EXEC language program in the file `SELECTED.EXEC`.

See “[The EXEC and FILES Options](#)” on page 338 for a description of the `SELECTED.EXEC` file produced with this option and some of its uses.

FD Used with option [EXEC](#) or [FILES](#) to indicate that the output includes only the file name, file type, member name and file drive code information for each file. Note: Member names are only included if the *file* specification indicates that library members are included in the directory listing.

FILES Indicates that the directory listing is to be output to `SELECTED.FILES`.

See “[The EXEC and FILES Options](#)” on page 338 for a description of the `SELECTED.FILES` file produced with this option and some of its uses.

FN Used with option [EXEC](#) or [FILES](#) to indicate that the output includes only the file name information for each file.

<u>FT</u>	Used with option EXEC or FILES to indicate that the output includes only the file name and file type information for each file.
<u>GROW</u>	Output the protection codes and growth factor for each file. The GROW and PUBLIC options produce the same results. The protection codes and growth factor are output in place of the “Recl” and “Keyl” columns.
<u>HIDDEN</u>	Include “hidden” files if they match file specification and the other options. (Files are marked as hidden with the CHANGE command described on page 216.)
<u>KEY</u>	Each program file matching the file specifications and any other restrictive options is analyzed for its security plug key value and its serial number. These values and the privilege level, version number and version date are output, replacing the “Org,” “Size,” “Recl” and “Keyl” columns. All files other than programs will have these columns blank.
<u>MN</u>	Used with option EXEC or FILES to indicate that the output includes only the file name, file type and member name information for each file. Note: Member names are only included if the <i>file</i> specification indicates that library members are included in the directory listing.
<u>NOTYPE</u>	Do not display the directory listing, only the summary line. This option is ignored if the option EXEC or FILES is also specified.
<u>OFF</u>	Used with option EXEC to prepend a “&control off” command as the first line of the SELECTED.EXEC file created.
<u>OWNER</u>	The directory listing will include files owned by all accounts if they match the file specifications and the other options. This option requires a privilege level of five. When the output is to the console or a printer, a page break occurs between each account’s files with the account name and number displayed at the top. EXEC and FILES note: The OWNER option does cause all qualifying files from all accounts to be included. However, the owning account name or number is <u>not</u> output.

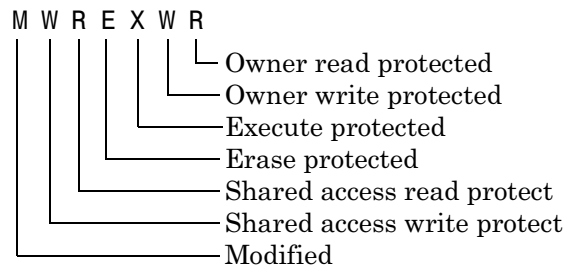
PRIVLEV Each program file matching the file specifications and any other restrictive options is analyzed for its privilege level requirements, its version and version date, and its patch level. These values are output, replacing the “Org,” “Size,” “Recl” and “Key1” columns. All files other than programs will have these columns blank.

The PRIVLEV and **VERSION** options produce the same results.

PROTECT Output the protection codes and growth factor for each file. The **GROW** and PROTECT options produce the same results.

The protection codes and growth factor are output in place of the “Recl” and “Key1” columns.

The protection codes displayed use single position-dependent letters:



Protection and attributes not enabled for a file are indicated with a period in that attribute’s position.

PRT_{nn} Indicates that FileList is to print the directory listing on the attached printer number *nn*.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

PUBLIC Includes files owned by the system account, if they match the file specifications and any other restrictive options.

SERIAL Analyzes each program file matching the file specifications and any other restrictive options. The serial number and the privilege level, version number and version date are output, replacing the “Org,” “Size,” “Recl” and “Key1” columns. All files other than programs will have these columns blank.

SIZE

Sets the return code to the total size of the files listed, in kilobytes (K).

SORT1

Sorts the output by filename, filetype, member name and drive code. This is similar to the default sequence produced by FileList.

However, there are several differences between SORT1 and the default sort sequence. The default sort sequence always lists subdirectories first and it uses line-graphics around the list and between the columns.

SORT1 sorts the subdirectory names along with file names. Additionally, by using codes for the organization, it has sufficient space to include the protection codes and the growth factor for the files. Compare the following two directory listings:

>filelist

Filelist *.* 10 Oct 1996 4:23pm Page 1

Filename	Filetype	Memname	Date	Time	Org.	Size	Recl	Keyl
C32			S 08/14/96	13:45	Directory	4,096		
EXEC			S 07/12/96	11:47	Directory	4,096		
MUBASIC			S 07/12/96	11:47	Directory	4,096		
MUBASIC	SAMPLES		S 11/18/96	09:48	Directory	10,496		
THEO_COM			S 07/12/96	11:47	Directory	4,096		
CHARSET	H		S 08/25/96	18:00	Stream	10,872		
LIBS	BACKUP		S 02/19/96	09:01	Stream	3,541		
LIBS	EXEC		S 02/19/96	09:05	Stream	3,541		
MAKE	FILE		S 02/09/96	16:00	Stream	1,899		
READ	ME		S 02/08/96	13:28	Stream	14,608		
SAMPLES	EXEC		S 07/12/96	11:49	Stream	25		

11 files, 61,365 bytes.

>filelist (sort1

Filelist *.* 10 Oct 1996 4:23pm Page 1

Filename	Filetype	Member	Dr	Date	Time	Org	Protect	Size	Recl	Keyl	Gro
C32				S 08/14/96	13:45	SD	4096			
CHARSET	H			S 08/25/96	18:00	S	.WR....	10872			
EXEC				S 07/12/96	11:47	SD	4096			
LIBS	BACKUP			S 02/19/96	09:01	S	.WR....	3541			
LIBS	EXEC			S 02/19/96	09:05	S	.WR....	3541			
MAKE	FILE			S 02/09/96	16:00	S	.WR....	1899			
MUBASIC				S 07/12/96	11:47	SD	4096			
MUBASIC	SAMPLES			S 11/18/96	09:48	SD	10496			
READ	ME			S 02/08/96	13:28	S	.WR....	14608			
SAMPLES	EXEC			S 07/12/96	11:49	S	.WR....	25			
THEO_COM				S 07/12/96	11:47	SD	4096			

11 files, 61,365 bytes.

SORT2

Sorts the output by file drive, file name, file type and member name.

SORT3

Sorts the output by file name, file type, file drive and member name.

- SORT4** Sorts the output by file type, file name, member name and file drive.
- SORT5** Sorts the output by date, time, file name, file type, member name and drive. Thus, the oldest files are listed first.
- SORT6** Sorts the output by descending date and time and ascending file name, file type, member name and drive. Thus, the newest files are listed first.
- SORT7** Sorts the output by member name, file name, file type and drive.
- SORT8** Sorts the output by organization, file name, file type, member name and drive. The sequence used for sorting organization is: indexed, keyed, relative, stream, 16-bit real mode programs, 16-bit programs, 32-bit programs, subdirectories and libraries last.

SORT *sequence* Sorts the output by a user-specified sequence. The *sequence* may be any one of: **FD**, **FN**, **FT**, **MN**, **DATE**, **SIZE** and **TYPE**. The **TYPE** sequence refers to the file's organization code (stream, indexed, program, *etc.*) Multiple sort sequences are specified by using multiple **SORT** options.

```
>filelist s (sort size
```

The above command lists the files by increasing file sizes.

```
>filelist s (sort ft sort size
```

This second examples sort the listing by file type and file size.

TYPE A default option that outputs the directory listing to the console.

VERSION Each program file matching the file specifications and any other restrictive options is analyzed for its privilege level requirements, its version and version date, and its patch level. These values are output, replacing the "Org," "Size," "Recl" and "Keyl" columns. All files other than programs will have these columns blank.

The **PRIVLEV** and **VERSION** options produce the same results.

date1 Includes a file only if the file's last change date is greater than or equal to this date. That is, if the file was changed on or after this date.

This option may be used with the *date2* option.

```
>filelist *.*:s f (10/15/96
```

The above command will include only those files that have been created or changed since October 14, 1996.

Note: Dates are specified according to the currently set date format. Refer to “Sysgen” on page 531 and “Set” on page 474.

date2 Includes a file only if the file's last change date is less than or equal to this date. That is, if the file was changed on or before this date. May only be specified by first specifying the *date1* option.

```
>filelist *.*:s f (10/15/96 10/30/96
```

This command includes only those files that have been created or changed since October 14, 1996, but not any files that were created or changed after October 30, 1996.

To specify a *date2* when you don't care about *date1*, use a date of 1/1/86 for the *date1* option. This is the earliest date maintained by the THEOS file system.

```
>filelist *.*:s f (1/1/86 11/20/96
```

Here, since the *date1* specification is 1/1/86, only files created or changed prior to November 21, 1996, are included.

FileList
Columns

Heading	Meaning	Col
Filename	These four fields comprise the file description or file name. When the output is to a file (other than SELECTED.EXEC or SELECTED.FILES), these fields are combined into one field of 28 characters.	1-8
Filetype		10-17
Memname		19-26
: or Dr		28
Date	Date file was created or last changed.	30-37
Time	Time file was created or last changed.	39-43
The following columns are <u>not</u> displayed when option PRIVLEV or VERSION is used.		
Org.	Type of file: program, library, <i>etc.</i>	45-46
Size	Size of file in bytes. This field is right-justified with spaces on the left.	55-64
Recl	The value is dependent upon the organization: 16-bit Programs... Code segment length Libraries..... Allocated size of library Indexed..... Record length Keyed..... Record length Relative:..... Record length	66-70
Keyl	The value is dependent upon the organization: 16-bit Programs... Data segment length Libraries..... Number of members Indexed..... Key length Keyed..... Key length	72-75
The following columns are displayed only when option GROW or PROTECT is used.		
Protect	Protection codes and attributes for file.	48-54
Gr or Gro	File's growth factor.	77-79

Heading	Meaning	Col
The following columns are displayed only when option VERSION or PRIVLEV is used.		
P	Privilege level of program.	46
Vers	Version number of program.	48-53
V-date	Version date of program.	55-61
The following column is displayed only when option VERSION or PRIVLEV is used.		
Patch Level	Patch level of program.	63-79
The following column is displayed only when the option KEY or SERIAL is used.		
Serial	Program serial number.	63-71
The following column is displayed only when the option KEY is used.		
Key	Program security plug key.	73-79
The following column is displayed only when the option CHECK-SUM is used.		
Checksum	File's checksum.	45-50

The EXEC and FILES Options

When the option **APPEND**, **EXEC** or **FILES** is used, the directory listing is written to a disk file, either **SELECTED.EXEC** or **SELECTED.FILES**. When these files are created with the **FileList** command, several options may have no meaning and will be ignored while several others can be used.

The options that can be used are: **FD**, **FN**, **FT**, **MN** and **OFF**. These options limit the amount of file description information included in the file.

SELECTED.EXEC

Option **APPEND** and **EXEC** create a special file named **SELECTED.EXEC**. This file contains only the file description information with command line variables added. For instance:

```
>filelist *.* (exec
>list selected.exec

&1 BACCESS.C:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 BACCESS.O:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 BJ.BASIC:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 BJ.COMMAND:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 BJ.HELP:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 BJ.MENU:S &2 &3 &4 &5 &6 &7 &8 &9 &10
...
```

Any information that may be produced by the options **CHECKSUM**, **COUNT**, **GROW**, **PRIVLEV**, **PROTECT** and **VERSION** is suppressed.

This file can be a real timesaver because the **SELECTED.EXEC** file can be used with many other commands that are used to modify or list your files. The **FileList**'s command selection capability is generally superior to most other commands. For instance, the above file could be used by executing the program:

```
>selected.exec copyfile f

>copyfile BACCESS.C:S f
>copyfile BACCESS.O:S f
>copyfile BJ.BASIC:S f
...
```

When the **SELECTED.EXEC** file is executed, each occurrence of the variable **&1** is replaced with **copyfile**, each **&2** is replaced with **f**.

The same program could be used in another command:

```
>selected.exec change (ne
>change BACCESS.C:S (ne
>change BACCESS.O:S (ne
>change BJ.BASIC:S (ne
...

>selected.exec erase

>erase BACCESS.C:S
>erase BACCESS.O:S
>erase BJ.BASIC:S
...
```

Many commands have their own FILES option that allows them to use this SELECTED.EXEC or SELECTED.FILES file as a list of files. The previous usage of the SELECTED.EXEC program that changes the protection codes and then erases the files could have been done with:

```
>change selected.exec (files ne
>erase selected.exec (files
```

The commands that have a FILES option can use a SELECTED.EXEC file because they have programming that ignores the &1, &2, *etc.* variables in each line and uses only the file name information.

SELECTED.FILES

The FILES option creates a special file named SELECTED.FILES that can be used as a data file for an application that needs directory-type information, or as a list of files used by other commands with their own FILES option.

When the FILES option is used with no options or options that do not include FD, FN, FT or MN, the SELECTED.FILES is created containing all of the information requested in the same format as a displayed or printed listing (without line graphics):

```
>filelist *.* (file size

>list selected.files

BACCESS  C                S 02/08/97 17:28 S .WR....    1875
BACCESS  O                S 03/01/97 10:49 S .WR....     541
BJ       BACKUP           S 06/29/97 10:57 P .WR....   151260
BJ       BASIC            S 06/29/97 10:54 S .WR....   50886
BJ       HELP             S 02/24/97 12:25 S .W.....   14576
...
```

The files in the above display do not have any “Recl,” “Key1” or growth factor associated with them, but this information is output for each file that has those properties. An application can use this file by opening it and reading each record. Use the column information shown in “[FileList Columns](#)” on page 336.

When any of the options [FD](#), [FN](#), [FT](#) or [MN](#) is used, the file name information is output as a packed file description:

```
>filelist s (files fd
>list selected.files

BACCESS.C:S
BACCESS.O:S
BJ.BASIC:S
...
```

This file can be used in the same way that the `SELECTED.EXEC` file was used in its example. For instance:

```
>change selected.files (files ne
>erase selected.files (files
```

Notes

Although the `FileList` command may be abbreviated to the single letter “f,” it must not be abbreviated to four characters because [File](#) is the full name of a different command.

If *file* is a “typeless” file description, there is no default library defined and the environment variable [FILETYPE](#) is defined, then the value of [FILETYPE](#) is appended to *file* to form a complete file description with file name and file type. To `FileList` a typeless file, you should specify the file description with a period terminator. See “[FileType](#)” on page 342 for more information.

```
>show filetype
FILETYPE=DATA

>filelist *

Filename Filetype Membname :   Date   Time      Org      Size
ACCOUNT  DATA                S 07/19/96 14:23 Stream      978
CAL      DATA                S 07/19/96 10:42 Stream       53
JOKES    DATA                S 11/30/90 14:58 Stream    2,390
PRINTER  DATA                S 01/18/96 11:43 Stream     843
SAMPLE   DATA                S 07/18/91 12:38 Stream    8,372
SYSTEM   DATA                S 01/27/96 16:27 Stream       9
THIRD    DATA                S 02/27/96 10:34 Stream       1
7 files, 12,646 bytes.

>filelist test.
Filename Filetype Membname :   Date   Time      Org      Size
TEST                S 08/19/96 15:46 Stream      99
One file, 99 bytes.
```

If *file* is a typeless file description and there is a default library defined, then *file* is assumed to be a member of the default library:

```
>show lib
LIBRARY = PROGRAM.SOURCE

>filelist example
Filename Filetype Membname :   Date   Time      Org      Size
PROGRAM  SOURCE  EXAMPLE S 07/19/96 14:23 Stream    2,748
One file, 2,748 bytes.
```

file may be a subdirectory name.

If the environment variable [LINEGRAPH](#) is defined and it equals N, then the directory listing is not outlined with line-graphics characters.

For multiple-page displays, the standard page browsing keys are recognized. Refer to “[Multiple-page Display Browsing](#)” on page 77. When the display is terminated early with the Quit key ([Break](#), [Q](#)), the summary line is displayed.

- Defaults
- [TYPE](#) is the only default option.
- Return Codes
- When option [SIZE](#) is used, the return code is set to the total size of the files listed, in number of kilobytes.
- Restrictions
- Options [ACCOUNT](#) and [OWNER](#) require a privilege level of five. The [ACCOUNT](#) option also requires that you be logged onto the SYSTEM account.
- See also
- [ChDir](#), [Disk](#), [File](#), [Logon](#)

FileType Command

The FileType command converts a THEOS file into an operating system independent form of the file, or it converts it back to a THEOS file.

Commands

- 1 FILETYPE file... (SAVE options
- 2 FILETYPE file... (RESTORE options

file	»	file name with optional path; may contain wild cards	
options	»	NOQUERY	QUERY
		NOTYPE	TYPE

This command converts THEOS-specific files into “THEOS File Save” format files (TFS) that are portable between operating systems. Although the contents of a TFS file may not be usable on another operating system, they can be copied and transmitted without any problems. Thus, a TFS file is useful when a THEOS file must be sent to another computer via an intermediate system such as a bulletin board system (BBS), network file server or non-THEOS FTP server.

The THEOS File Save format is a file that contains the original THEOS directory entry for the file along with the binary contents of the file. This is a stream file format that is recognized and supported by all operating systems.

Operation **Mode 1**—Converts a THEOS file into a THEOS File Save file. The SAVE keyword does not have to be specified because it is the default. The converted file uses the same file-name as the original file and a file-type of TFS. If the original file was a member of a library, the output file-name is the original file’s member-name.

```
> filetype sample.file
"SAMPLE.FILE:S" copied to "SAMPLE.TFS:S".
One file copied.

> filetype data.lib.customer
"DATA.LIB.CUSTOMER:S" copied to "CUSTOMER.TFS:S".
One file copied.
```

The file-type of file may not be TFS nor may it be a simple wildcard of *.

Mode 2—Converts a THEOS File Save file back to the original THEOS file.

<div>Options</div>	<div> <div> <div><u>NOQUERY</u></div> <div> Tells FileType to not ask for confirmation before converting each file. This is a default option when wild cards are not used. <div> <pre> >filetype *.data (noq "INDEXED.DATA:S" copied to "INDEXED.TFS:S". "DIRECT.DATA:S" copied to "DIRECT.TFS:S". "KEYED.DATA:S" copied to "KEYED.TFS:S". "DATA.DATA:S" copied to "DATA.TFS:S". 4 files copied. </pre> </div> <div> To disable this option use the QUERY option. </div> </div> </div> <div> <div><u>NOTYPE</u></div> <div> Tells FileType to not display the results of each file converted on the standard output device. The general result message (the “nn files copied.” message before exiting FileType) is also suppressed with this option. <div> <pre> >filetype *.data f (not Ok to copy "INDEXED.DATA:S" (Yes,No,All) Y Ok to copy "DIRECT.DATA:S" (Yes,No,All) Y Ok to copy "KEYED.DATA:S" (Yes,No,All) Y Ok to copy "DATA.DATA:S" (Yes,No,All) Y </pre> </div> <div> To disable this option use the TYPE option. </div> </div> </div> <div> <div><u>QUERY</u></div> <div> Tells FileType to “query” or ask if each file matching the file specifications is to be converted. This is a default option when wild cards are used. <div> <pre> >filetype *.data Ok to copy "INDEXED.DATA:S" (Yes,No,All) </pre> </div> <div> When the “Ok to copy” question is asked you may respond with a Y for yes, N for no or A for all. Responding with A means yes to this file and all remaining files are then copied without being queried. Respond with Esc to cancel the conversion operation. This option is disabled with the NOQUERY option. </div> </div> </div> <div> <div><u>TYPE</u></div> <div> A default option that tells FileType to display the results of each file copied on the standard output device. This display can be redirected. This option is disabled with the NOTYPE option. </div> </div> </div> <div> <div>Defaults</div> <div>Mode 1 is the default.</div> </div> <div> <div>Restrictions</div> <div><i>file</i> may not specify a file-type of TFS nor may the file-type use a wildcard.</div> </div> <div> <div>See also</div> <div>Compress, CopyFile (EXPORT option), Expand, Receive, Send, THEO+COM</div> </div>
--------------------	---

Find Command

The Find command searches a directory path and locates files.

FIND *file...* (*options*)

file » file name with optional path; may contain wild cards

options » PRTnn
 SORT
 SUBDIR
 date1
 date2

Operation

The current working directory, or the path specified in *file*, and all directories subordinate to it, are searched for all files matching the *file* specification. All files found are output to the standard output device.

```
>tree
/
├── data
│   ├── misc
│   │   ├── doc
│   │   └── programs
│   └── package
│       ├── doc
│       └── programs
├── vertical
│   ├── doc
│   │   ├── files
│   │   └── programs
│   └── programs
```

Any Find command executed in this example will search all of the subdirectories shown above for any and all files specified on the command line. In the following command these directories are searched for any work files:

```
>find *.work*
/vertical/checkreg.work0:s
/vertical/checkreg.work1:s
/data/routeb.workfile:s
```


In the next command, the directories are searched for all backup files, with the list of files is piped to the Erase command.

```
>find *.backlib*. * *.backup | erase
"/VERTICAL/CUSTOMER.BACKLIB.LETR0004:S" erased.
"/VERTICAL/CUSTOMER.BACKLIB.LETR0007:S" erased.
"/VERTICAL/LISTS.BACKLIB.LETR0001:S" erased.
"/VERTICAL/LISTS.BACKLIB.LETR0002:S" erased.
"/VERTICAL/LISTS.BACKLIB.LETR0003:S" erased.
"/VERTICAL/LISTS.BACKLIB.LETR0004:S" erased.
"/VERTICAL/PROGRAMS/ABC.BACKLIB.MAKEFILE:S" erased.
"/VERTICAL/PROGRAMS/ABC.BACKLIB.XFER:S" erased.
"/VERTICAL/PROGRAMS/ABC.BACKLIB.STATUS:S" erased.
"/PACKAGE/PROGRAMS/PACKAGE.BACKLIB.MENU:S" erased.
"/PACKAGE/PROGRAMS/PACKAGE.BACKLIB.CHECKREG:S" erased.
"/VERTICAL/LIBS.BACKUP:S" erased.
"/VERTICAL/DOC/FILES/CUSTOMER.BACKUP:S" erased.
13 files erased, 53,504 bytes recovered.
```

Options

PRT*nn*

Indicates that Find is to print the list of files on the attached printer number *nn*.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

SORT

Sorts the list before outputting it. This option cannot be used when the [SUBDIR](#) option is used. Compare the two lists produced with and without the SORT option.

```
>find *.work*
/vertical/checkreg.work0:s
/vertical/checkreg.work1:s
/data/routeb.sortfile:s

>find *.work* (sort
/data/routeb.sortfile:s
/vertical/checkreg.work0:s
/vertical/checkreg.work1:s
```

When the SORT option is used the lines are sorted in alphabetical order, including any path specification.

SUBDIR

Includes the names of all subdirectories searched. The subdirectory names are output after a subdirectory and all subordinate directories to this directory are searched. This option is particularly useful when you want to remove a branch of a directory tree:

```
>tree
/
├─test
└─work
```

```
>find *.*:*:s (subdir | erase (notype
```

The above command erases all of the members of all libraries, the libraries themselves and flat files in the two subdirectories. It then erases each of the subdirectories.

SUBDIR cannot be used when [SORT](#) is specified.

date1

Includes a file only if the file's last change date is greater than or equal to this date. That is, if the file was changed on or after this date.

This option may be used with the [date2](#) option.

```
>find *.*:*:s f (10/15/96
```

The above command will include only those files that have been created or changed since October 14, 1996.

date2

Includes a file only if the file's last change date is less than or equal to this date. That is, if the file was changed on or before this date. May only be specified by first specifying the [date1](#) option.

```
>find *.*:*:s f (10/15/96 10/30/96
```

This command includes only those files that have been created or changed since October 14, 1996, but not any files that were created or changed after October 30, 1996.

To specify a *date2* when you don't care about [date1](#), use a date of 1/1/86 for the [date1](#) option. This is the earliest date maintained by the THEOS file system.

```
>find *.*:*:s f (1/1/86 11/20/96
```

Here, since the [date1](#) specification is 1/1/86, only files created or changed prior to November 21, 1996, are included.

Restrictions The [SUBDIR](#) and [SORT](#) options cannot both be specified.

See also [ChDir](#), [FileList](#), [Tree](#), [WhereIs](#)

Force Command

This command attempts to force another logged-on user to cancel the command that they are executing and to optionally execute another command or EXEC program.

- 1 **FORCE** *username command*
- 2 **FORCE** *partition command*
- 3 **FORCE** *user command* (**NOQUERY**
- 4 **FORCE** **-nq** *user command*

command » an optional command line

partition » partition id number

user » *partition* or *username*

username » account name that other user is logged onto

Operation

Mode 1—Starting with partition number one, all logged-on users are examined. The first user found to be logged onto the account *username* is forced to perform a **[Break]**,**[Q]** operation and, if that is successful and there is a *command* specified, that user is forced to execute *command*.

Mode 2—If partition number *partition* is logged on it is forced to perform a **[Break]**,**[Q]** operation and, if that is successful and there is a *command* specified, the user is forced to execute *command*.

Mode 3—This mode is similar to [Mode 1](#) or [Mode 2](#) except, if the user cannot be forced on the first attempt. The **NOQUERY** option specifies that Force should not ask if it should try harder but exits after the first attempt.

With this mode of the command, if *command* contains options then the **NOQUERY** option of the Force command must be specified with a second open parenthesis.

```
>force 5 filelist a (print (noquery
```

Mode 4—Identical to [Mode 3](#) but the no query feature is specified with the “UNIX style” option. This mode is useful when *command* contains its own options.

The following two commands perform identical functions:

```
>force 5 basic test (print (noquery
>force -nq 5 basic test (print
```

Notes

It is possible that a user may not be forced with this command. A user's account environment can be set to ignore **Break**, **Q** requests. See “[Account](#)” on page 156. Additionally, some programs disable **Break**, **Q** during some or all phases of operation (for instance, WindoWriter).

If the user's **Break**, **Q** is not being honored at the time that Force attempts to make it abort its operation, Force will not know if the force was successful unless you use specify a *command* with the Force command. In that situation, the following message is displayed:



```
User won't be pushed, try harder?
```

Respond with **Y** to have Force try harder by reenabling the other user's **Break**, **Q**. If this also fails, Force exits with an error message.

Cautions

It is dangerous to force another user if you do not know what that user is doing. Always use the [Who](#) or [Show USERS](#) command to find out which program the other user is executing before using the Force command. It is always best if the user is at the CSI.

Restrictions

The Force command requires a privilege level of five.

You can only Force users or partitions that are logged onto an account.

See also

[Show USERS](#), [Who](#)

GetFile Command

This command accesses a DOS-formatted disk or partition and copies files from it. It can also receive files from a “THEOS 8-bit” operating system’s SEND command via a communications link.

- 1 **GETFILE** *drive* (**DIR**
- 2 **GETFILE** *DOS-file:drive file* (*dos-options options*
- 3 **GETFILE** **COMnn** *file* (**THEOS** *option*

<i>drive</i>	»	THEOS drive letter of DOS disk
<i>file</i>	»	file name with optional path; may contain wild cards
<i>DOS-file</i>	»	file name with optional path; may contain wild cards
<i>options</i>	»	EXEC EXPAND OWNER= <i>nn</i> QUERY NOQUERY
<i>dos-options</i>	»	BINARY HIDDEN

Operation

Mode 1—Accesses the DOS disk in *drive* and displays the directory of files.

```
>getfile f ( dir
MSDOS Path: *.*:F
Filename    Size   Date    Time
INSTALL.BAT    1 10/25/94 14:36
OEMSETUP.INF   1 10/25/94 14:51
README        DIR 10/25/94 14:28
README.EXE    55 10/21/94  9:38
```

Unless a specific path is indicated, the listing is of the root directory. For instance, to list the files in the README directory you would use:

```
>getfile readme/:f ( dir
MSDOS Path: /README/*.*:F
Filename    Size   Date    Time
DECNET.TXT    4 10/25/94 14:15
```

...

Note that the path and file description syntax of the DOS disk may use THEOS conventions or DOS file name syntax. For instance, the following two commands perform identical requests:

```
>getfile a: (dir
```

```
>getfile f (dir
```

Similarly, the following two commands are interpreted identically:

```
>getfile /dos:f (dir
```

```
>getfile a:\dos\ (dir
```

Mode 2—Copies one or more files from the DOS disk to the THEOS *file*.

```
>getfile netware/client.dos/net.cfg:f s (binary
"/NETWARE/CLIENT.DOS/NET.CFG:F" copied to "NET.CFG:S".
One file copied.
```

As indicated in the above example, specifying a drive code only for *file* tells GetFile to use the DOS file name as the destination name. Because the **BINARY** option was not used, GetFile assumes that the DOS file is a text file and converts CR,LF pairs into CR only. Also, the file transfer is terminated when the end-of-file mark is detected.

The DOS-file and drive may be specified with THEOS file name syntax or with DOS file name syntax. For instance, the following two commands copy the same file:

```
>getfile /dos/ansi.sys:f =.:s (bin
"/DOS/ANSI.SYS:F" copied to "ANSI.SYS:S".
```

```
>getfile a:\dos\ansi.sys =.:s (bin
"/DOS/ANSI.SYS:F" copied to "ANSI.SYS:S".
```

See notes about “[Wild-Card Specifications](#)” on page 352.

Mode 3—Copies one or more files from COM nn to file. The files received on COM nn must be generated by another computer using the THEOS 8 operating system’s SEND command.

Options**EXEC**

This option tells GetFile to not transfer the files from the DOS disk. Instead, the list of file names found on the DOS disk are output as an EXEC language program in the file `SELECTED.EXEC`. This file will contain only the file description information with command line variables added. For instance:

```
>getfile readme/*.txt:f (exec
>list selected.exec

&1 README/DECNET.TXT:F &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 README/IBMLAN.TXT:F &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 README/LANMAN.TXT:F &2 &3 &4 &5 &6 &7 &8 &9 &10
...
```

This file can then be edited or used as is. For instance:

```
>selected.exec getfile s (dos binary
>getfile README/DECNET.TXT:F s (dos binary
```

EXPAND

Expands compressed files to their original form. This option is only applicable when used with [Mode 3](#) and the file is a `.LISTING` or `.LINKMAP` file.

OWNER=nn Specifies the account number that currently owns the file on the transmitting THEOS system.

NOQUERY Tells GetFile to not ask for confirmation before copying each file. This is a default option when wild cards are not used.

```
>getfile readme/*.txt:f s (noq
"/README/LANMAN.TXT:F" copied to "LANMAN.TXT:S".
"/README/VINES.TXT:F" copied to "VINES.TXT:S".
"/README/PCTCP.TXT:F" copied to "PCTCP.TXT:S".
"/README/IBMLAN.TXT:F" copied to "IBMLAN.TXT:S".
4 files copied.
```

To disable this option use the [QUERY](#) option.

QUERY

Tells GetFile to “query” or ask if each file matching the file specifications is to be copied. This is a default option when wild cards are used.

```
>getfile readme/*.txt:f s
Ok to copy "/README/LANMAN.TXT:F" (Yes,No,All)
```

When the “Ok to copy” question is asked, you may respond with a **[Y]** for yes, **[N]** for no or **[A]** for all. Responding with **[A]** means yes to this file and all remaining files are then copied

without being queried. Respond with **Esc** to cancel the copy operation.

To disable this option use the **NOQUERY** option.

DOS Options **BINARY** Tells GetFile that the DOS file contains binary information and that it should transfer the entire file without any translation.

When the **BINARY** option is not used, GetFile assumes that the DOS file is a text file and converts CR,LF pairs into CR only. Also, the file transfer is terminated when the end-of-file mark is detected. When in doubt as to the type of DOS file it is, always use the **BINARY** option. The **CRLF** command can always be used on the file after it has been transferred successfully.

HIDDEN Includes DOS hidden files from the DOS disk.

Notes

Wild-Card Specifications

Similar to the **CopyFile** command, the destination file specification may use the equal sign character (=) to indicate that the destination file description uses the corresponding element of the source file description. An equal sign used as the complete term of the file description (i.e., the file-name, file-type or member-name) means that the corresponding complete term from the source file description is used. For instance:

```
>getfile *.txt:f =.text:s (noquery
"FILE1.TXT:F" copied to "FILE1.TEXT:S".
"FILE2.TXT:F" copied to "FILE2.TEXT:S".
...
```

The equal sign wild card, combined with regular characters, can have two different effects depending upon whether or not the corresponding term in the source file description used wild cards. When the source file description uses a wild card then the equal sign is replaced with the portion represented by the source file's wild card.

```
>getfile file*.dat:f abc=x.data:s (noquery
"FILE1.DAT:S" copied to "ABC1X.DATA:S".
"FILE3.DAT:S" copied to "ABC3X.DATA:S".
...
```

When the source file description does not use a wild card for the corresponding term, the equal sign is replaced with the complete source file term.


```
>getfile *.dat:f =.new=:s (noquery
"FILE1.DAT:S" copied to "FILE1.NEWDAT:S".
"FILE3.DAT:S" copied to "FILE3.NEWDAT:S".
"FILE2.DAT:S" copied to "FILE2.NEWDAT:S".
...
```

DOS Partitions and Disks

The disk drive specified by *drive* may be an attached removable disk such as a floppy or removable hard disk, or it may be a partition on an attached hard disk drive. This disk or partition may be a DOS-formatted partition (16-bit FAT) or a Windows 95 disk or partition (32-bit FAT).

When it is a partition of an attached THEOS drive, the DOS partition is referenced by specifying the attached THEOS drive code, for instance S or C: for *drive* if the DOS partition is a partition of the same physical drive that is attached as the s drive in THEOS.

```
>getfile s (dir

>getfile /windows/*.*:s

>getfile /dos/ansi.sys:s =.:s (bin
"/DOS/ANSI.SYS:S" copied to "ANSI.SYS:S".
```

Windows NT disks and partitions cannot be accessed with this command.

Return Codes

When no files are transferred, the return code is set to one, indicating failure.

See also

[CopyFile](#), [CRLF](#), [PutFile](#), [Receive](#), [THEO+COM](#)

Head Command Filter

The Head command displays the beginning of a text file on the standard output device.

- 1 HEAD (*option*
- 2 HEAD *file...* (*option*

file » file name with optional path

option » *nnn*

Operation

Mode 1—The first 10 lines from the standard input device are output to the standard output device. This form of the Head command would normally be used in a pipe command line:

```
>calendar 1997 | head
      January 1997          February 1997          March 1997
Su Mo Tu We Th Fr Sa    Su Mo Tu We Th Fr Sa    Su Mo Tu We Th Fr Sa
              1  2  3  4              1              1
 5  6  7  8  9 10 11      2  3  4  5  6  7  8      2  3  4  5  6  7  8
12 13 14 15 16 17 18      9 10 11 12 13 14 15      9 10 11 12 13 14 15
19 20 21 22 23 24 25     16 17 18 19 20 21 22     16 17 18 19 20 21 22
26 27 28 29 30 31       23 24 25 26 27 28       23 24 25 26 27 28 29
                                   30 31
      April 1997          May 1997          June 1997
>
```

Mode 2—The first lines of *file* are output to the standard output device. When more than one *file* is specified then the first lines of each of the files are output.

```
>head system.help32._commands (4
ACCOUNT      Maintain the user account names file.
ARCHIVE      Back up hard disks onto floppies or tapes.
ATTACH       Logically connect a device for future access.
BACKUP       Copy full disk to disk or tape.
```

Options

nnn Specifies the number of lines of each file to output. This specification must be in the range 1–999. The default value is 10.

Notes

When the standard output device is the console and multiple *files* are specified, the screen is cleared between each file display. When multiple files are specified, the output for each file has a heading added that identifies the file.

The *nnn* option may use the DOS/UNIX style which is a leading minus sign and the option is specified before the *file* specifications. The following two commands produce the same results:

```
>head system.help32._commands (4
```

```
>head -4 system.help32._commands
```

If *file* is a “typeless” file description, there is no default library defined and the environment variable **FILETYPE** is defined, the value of **FILETYPE** is appended to *file* to form a complete file description with file name and file type. To Head a typeless file you should specify the file description with a period terminator. See “**FILETYPE**” on page 105 for more information.

Defaults

The default number of lines displayed is 10.

See also

[List](#), [More](#), [Tail](#)

Help Command

The Help command displays a summary of all THEOS commands or a detailed description about a specific command.

Commands

1 HELP

2 F1

3 HELP *program*

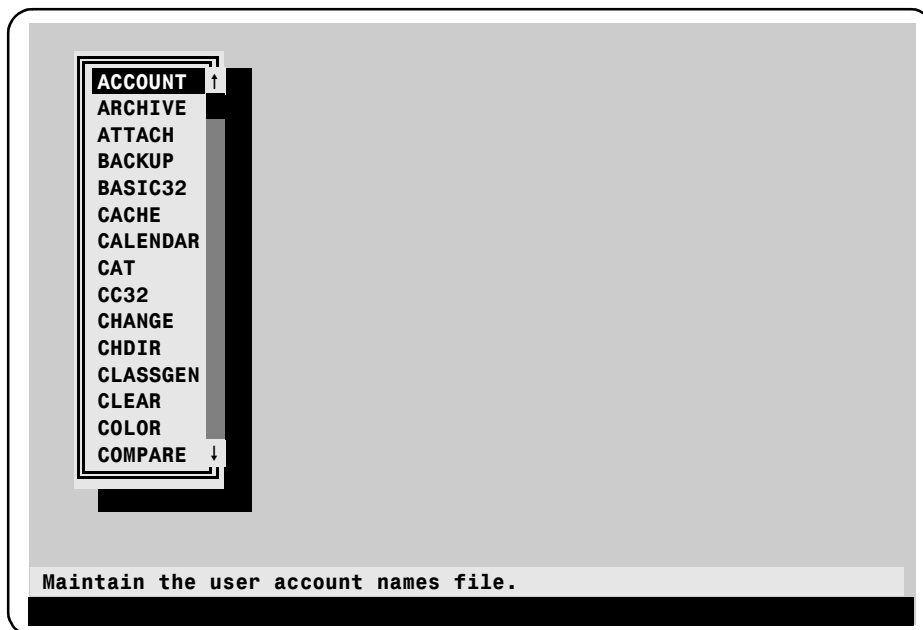
4 *program*F1

5 HELP *

program » command name or valid abbreviation

The Help command uses the library SYSTEM.HELP32 as the source of the help text displayed. If this library has been removed for any reason, the Help command cannot operate.

Operation **Mode 1**—Displays the file SYSTEM.HELP32._COMMAND, which is a list of all commands on the operating system with a brief, one-line description.



To get more information about a particular command, select the command name with the reverse-video highlight bar and then press **Enter** . The highlight bar is moved with the **↑** and **↓** arrow keys, or by entering a letter key to jump to the next command starting with that letter. As a shortcut, the **Space** advances to the next line also.

You may also use the **Home** and **End** keys to move to the beginning or end of the list. The **PageDown** and **PageUp** keys advance to the next or previous display page.

Mode 2—This is a shortcut method of invoking the [Mode 1](#) Help display.

Mode 3—Displays detailed information about the function and operation of the command *program*. *program* may be any valid name, synonym name or abbreviation of a command.

>help acc

Function: The ACCOUNT command maintains the SYSTEM.TEOS32.ACCOUNT file and the individual user account environments.

User accounts provide a method of system security by controlling access to certain files and programs, establishing privilege levels for operating system commands, and providing optional passwords that may be required to log onto certain accounts.

Syntax: ACCOUNT drive (options

Where: drive is the drive with the SYSTEM.TEOS32.ACCOUNT file to be maintained. The default drive is "S." This version of the ACCOUNT command will read a THEOS version 3.x SYSTEM.ACCOUNT file and create a new SYSTEM.TEOS32.ACCOUNT file.

Options

ADD name Add new account "name" to the SYSTEM.TEOS32.ACCOUNT file, May be used with options: PASSWORD, SYNONYM, PRIVLEV, PROMPT, and SUBDIR.

Mode 4—This is a shortcut method of invoking the [Mode 3](#) Help display for *program*.

Mode 5—Displays detailed information for all commands or files listed in the SYSTEM.HELP32 library.

Notes

Besides the standard command names, the **Help** command can display information about any subject for which there is a file in the `SYSTEM.HELP32` library. Provided with the operating system are files providing information about:

Subject	Description
_B3220	Internal help for MultiUser BASIC language.*
_LEDIT	Internal help for LineEdit command.*
_LINK	Internal help for Link command.*
_MORE	Internal help for More command.*
_PATCH	Internal help for Patch command.*
BADEDIT	Help for hard disk sector sparing.
COMPOSE	Help for composing characters on PC-Term keyboards.
CSI	Help for entering commands.
EXEC	Help for the EXEC language.
VDI	Help for using the Virtual Device Interface.
* This information is normally displayed from within the indicated program. It can be displayed at any time with the Help command.	

Ident Command

This command displays the current account number on the standard output device.

IDENT

Operation The account number of the current account is output to the standard output device (normally the console).

```
>ident  
5
```

See also [Show WHO](#)

Install Command EXEC

The Install command installs software from a floppy disk drive.

INSTALL *drive*

drive » optional disk drive letter

Commands

Operation	The installation procedure on the disk <i>drive</i> is executed. If <i>drive</i> is not specified, drive F is used.
Notes	The installation procedure on the diskette must be named INSTALL.EXEC.
Defaults	The default for <i>drive</i> is F.

IXDiag Command

This command checks the integrity of indexed files.

1 IXDIAG *file...* (*options*

<i>file</i>	»	file name with optional path; may include wildcards			
<i>options</i>	»	APPEND	EXEC	NOWAIT	SUBDIR
		DATA BASIC	FILES	RECLen	TYPE
		DATA BINARY	KEYLEN	REPAIR	VERIFY DATA
		DETAILS	LOGFILE	REPLACE	VOLUME
		EXCEPTIONS	NOTYPE	SAVE	WAIT

Operation The *file* is examined for indexed file structure integrity. Many, non-serious errors may be detected and reported along with serious, data-loss errors.

The *file* is repaired only if specifically directed to with the **REPAIR** or **SAVE** options. Otherwise, only warning messages are displayed when errors are detected.

Options **APPEND** Used with option **EXEC** to indicate that problem file names are appended to the end of any existing IXERR.EXEC file.

```
>list ixerr.exec
&1 /PRIVATE/ADDR.BOOK:S &2 &3 &4 &5 &6 &7 &8 &9 &10

>ixdiag *.* (exec
>list ixerr.exec
&1 /PRIVATE/ADDR.BOOK:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 /JONAS/DATA.BASE:S &2 &3 &4 &5 &6 &7 &8 &9 &10
```

Used with option **LOGFILE** to indicate that the error messages are appended to any existing *logfile*.

DATA BASIC Records in the *file* are formatted BASIC language records.

DATA BINARY Records in the *file* may be in any format and are not checked for consistency.

DETAILS Displays information about which files are being examined and, if an error is detected, a brief description of the error. When used with the **WAIT** option, after an error description is displayed the program pauses and you may request more information about the error condition.

Compare the following three examples to see the effect of this option:

```
>ixdiag *.*
/ DATA/CHECK.DETAIL:S needs minor repairs
/ DATA/EXPENSE.JOURNAL:S could be optimized

>ixdiag *.* (detail
Examining indexed file /DATA/CHECK.DETAIL:S
  Bad blocks on freelist - File should be rebuilt

Examining indexed file /DATA/CHECK.MASTER:S

Examining indexed file /DATA/EXPENSE.JOURNAL:S
  Indexed file has 62.9% wasted space which could be recovered if rebuilt
...

>ixdiag *.* (details wait
Examining indexed file /DATA/CHECK.DETAIL:S
  Bad blocks on freelist - File should be rebuilt
? [F1]

Indexed file has serious problems with its freelist, which could
cause problems if you tried to write new records. You may be able to
read from this file but corruption is likely if file is written
without recopying the file or rebuilding the freelist.

Examining indexed file /DATA/CHECK.MASTER:S

Examining indexed file /DATA/EXPENSE.JOURNAL:S
  Indexed file has 62.9% wasted space which could be recovered if rebuilt
? [F1]

THEOS may have been rebooted while this file was being updated and
some space has been lost for that or similar reasons. This is
basically harmless other than being a waste of disk space and this
space can be recovered by recopying the file by record or by using
the SAVE option.

...
```

Note: **DETAILS**, **NOTYPE** and **TYPE** are mutually exclusive options. Only the last one specified will have any effect.

EXCEPTIONS An *exception-file* file name is specified following the option keyword. Can only be used when **SAVE** option is used and tells IXDiag to save any suspect and bad records to this *exception-file*.

EXEC The names of all files that have problems or errors are written to the file IXERR.EXEC. If **APPEND** option is not specified, any existing IXERR.EXEC file is first erased.

```
>ixdiag *.* (exec
>list ixerr.exec

&1 /PRIVATE/ADDR.BOOK:S &2 &3 &4 &5 &6 &7 &8 &9 &10
```

FILES

Indicates that *file* is an ASCII stream file with each record in the file specifying a single file name. The file name specifications in this file may include the path and wild cards.

The `SELECTED.FILES` and `SELECTED.EXEC` files created by [FileList](#) and the `FOUND.EXEC` created by [Look](#) can be used for this specification file (see “[The EXEC and FILES Options](#)” on page 338). You may also create the file with an editor or application program.

For instance, [FileList](#) is used to create a list of files to be examined:

```
>filelist *.data:a (exec
>filelist a not *.data:a (10/1/95 exec append
```

A `SELECTED.EXEC` file now exists that lists all of the “data” files and all files that have been changed since 10/01/1995. The following command checks these files:

```
>ixdiag selected.exec (file
```

KEYLEN

The new *key-length* is specified following the option keyword. This option tells IXDiag that, when it rebuilds the file it should use the new *key-length* specified, not the key-length of the existing file.

This option is only used when [REPAIR](#) or [SAVE](#) is used.

LOGFILE

A *logfile* is specified following the option keyword. Indicates that error messages are to be written to *logfile*. May be used with option [APPEND](#) to cause the messages to be added to any existing messages in the file.

NOTYPE

Tells IXDiag to not display messages about what files it is checking.

```
>ixdiag *.* f (notype
/DATA/CHECK.DETAIL:S needs minor repairs
/DATA/EXPENSE.JOURNAL:S could be optimized
/DATA/MASTER.CONTROL:S needs minor repairs
/DATA/PAYABLES.MASTER:S needs minor repairs
```

To disable this option use the [TYPE](#) or [DETAILS](#) option.

Note: [DETAILS](#), [NOTYPE](#) and [TYPE](#) are mutually exclusive options. Only the last one specified will have any effect.

NOWAIT Used with **DETAILS** option to indicate that you want IXDiag to wait each time that it detects and reports an error in a file. This option has no effect when **DETAILS** is not specified.

RECLEN The new *record-length* is specified following the option keyword. This option tells IXDiag that, when it rebuilds the file it should use the new *record-length* specified, not the record-length of the existing file.

This option is only used when **REPAIR** or **SAVE** is used.

REPAIR Tells IXDiag that, if *file* contains errors, it should rebuild it with the corrected version.

Note: This is potentially dangerous operation. The file should be backed up first or copied to another location.

```
>ixdiag telephon.numbers (repair details
Examining indexed file /TELEPHONE.NUMBERS:S
Bad blocks on freelist - File should be rebuilt

Recovering /TELEPHON.NUMBERS:S on 23 Nov 1998 03:42PM :
205 records copied, 0 records skipped (205 total records).
```

REPLACE Used with the **SAVE** and **EXCEPTIONS** option to tell IXDiag to replace any existing *savefile* or *exception-file* with the new data. When **REPLACE** is not used with **SAVE** and **EXCEPTIONS**, the records are added to any existing *savefile* or *exception-file*.

SAVE A new *savefile* destination file name is specified following the option keyword. Tells IXDiag that it should rebuild the *file* and write the recovered records to a new file named *savefile*. Can be used with **EXCEPTIONS** to save the suspect or bad records into a new file.

```
>ixdiag suspect.file (save suspect.new except suspect.errors
Recovering SUSPECT.FILE on 23 Nov 1998 02:10PM :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
17297 records copied, 0 records skipped (17297 total records).
```

SUBDIR Tells IXDiag to check all subdirectories starting with the directory for *file*.

```
>ixdiag s (subdir details
Examining indexed file /CGI/DATA/CUSTOMER.MASTER:S
Examining indexed file /CGI/DATA/ORDERS.DETAIL:S
Examining indexed file /CGI/DATA/ORDERS.MASTER:S
Examining indexed file /FAX/FAX.LOG:S
Indexed file in use -- diagnostic check skipped
Examining indexed file /SAMPLES/SAMPLE.DATA.CUSTOMER:S
Examining indexed file /SYSTEM.CALANDAR:S
...
```

TYPE A default option that tells IXDiag to display summary information only for files with errors.

```
>ixdiag *.*
/ DATA/CHECK.DETAIL:S needs minor repairs
/ DATA/EXPENSE.JOURNAL:S could be optimized
/ DATA/MASTER.CONTROL:S needs minor repairs
/ DATA/PAYABLES.MASTER:S needs minor repairs
```

To disable this option use the **NOTYPE** option.

Note: **DETAILS**, **NOTYPE** and **TYPE** are mutually exclusive options. Only the last one specified will have any effect.

VERIFY DATA Tells IXDiag to include in its tests a check to confirm that the fields in each record of the file are BASIC formatted fields.

VOLUME Tells IXDiag to look for file in all accounts on that drive. file must be a drive specifier only:

```
>ixdiag s (volume
PACKAGE\SYSPECT.FILE:S has serious problems
SYSTEM\TELEPHON.NUMBERS needs minor repairs
```

When this option is used in combination with the **SUBDIR** option, all indexed files in all directories on all accounts are checked for integrity.

WAIT Used with the **DETAILS** option to cause IXDiag to wait each time that it reports an error in a file. In this mode you may request additional explanation of the error message by using the **F1** key.

Defaults The following options are in effect by default: **DATA BASIC**, **NOWAIT** and **TYPE**.

Notes To check all of the files on all of the accounts on all subdirectories on a disk, log onto the SYSTEM account and use the command:

```
>ixdiag s (volume subdir
```

Cautions A file should not be repaired with the **REPAIR** option unless it is backed up first. The data may be lost if there is any problems encountered during the repair operation.

Restrictions Only indexed files are checked with this command. Keyed, direct and stream files are excluded.

The file must not be open by any other user. The file is locked by IXDiag when it checks it.

Keyword Command

The Keyword command displays or maintains the SYSTEM.TEOS32.KEYWORD file.

1 KEYWORD *nnn* (*option*

2 KEYWORD * (*option*

3 KEYWORD

option» NOSORT
PRT*nn*
SORT
TYPE

Commands

- Operation**
- Mode 1**—Display keyword *nnn* from the SYSTEM.TEOS32. KEYWORD file.
- Mode 2**—Displays all of the keywords defined in the SYSTEM.TEOS32. KEYWORD file. When the display is to the console or printer, it is output in multiple columns.

>keyword * (nosort

Code	Keyword	Code	Keyword	Code	Keyword	Code	Keyword
0	Yes	26	TRANS	52	NOSORT	78	HIStory
1	No	27	SPECs	53	NOMain	79	ABbrev
2	Type	28	TRuncate	54	ATTach	80	RDYmsg
3	NOTtype	29	NEWFile	55	TOUch	81	MSG
...							

The keywords are shown in mixed case. The uppercase letters in each keyword indicate the minimum spelling or abbreviation allowed.

When the output is redirected to a disk file, it is output in a single column with no headings.

Mode 3—This is the keyword maintenance mode. When this mode is entered, you are prompted to enter the keyword number:

>keyword

Enter keyword id:

At this prompt enter either the number of the keyword you want to view or change, or press **Enter**, **Esc** or **F9** without a number to exit.

When a number is entered, the current definition is displayed and you are asked for the new definition:

```
>keyword
```

```
Enter keyword id: 3
```

```
Old text: NOnType
```

```
New text:
```

To leave the keyword unchanged, press **Enter** without any characters or spaces.

To change or define a keyword enter the new text. Keywords may be a maximum of eight characters in length. The minimum spelling or abbreviation for a keyword is specified with uppercase letters.

Care should be taken when specifying the minimum spelling for a keyword. Make sure that the abbreviation does not conflict with any other keyword's abbreviation that might be used in the same context.

When the new text for the keyword is entered, press **Enter** and you are prompted for another keyword to change.

Options

NOSORT Tells Keyword to output the list of keywords in keyword number order, not in alphabetically sorted order.

PRT*nn* Indicates that Keyword is to print the keywords on the attached printer number *nn*.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

SORT A default option that causes the keywords to be output in alphabetical order.

```
>keyword * > sorted.keywords
```

```
>list sorted.keywords
```

```
79  ABbrev
192 ABORT
109 ACCount
253 ADD
179 Alf
193 ALign
203 All
37  AAppend
```


TYPE A default option that causes Keyword to display the keywords on the console.

Defaults [TYPE](#) and [SORT](#) are default options when using [Mode 2](#).

Cautions This keyword file is used by all of the THEOS commands. Changing a keyword may affect many commands.

Restrictions You must be logged onto the system account and you must have a privilege level of five.

Keywords may be a maximum of eight characters in length.

See also [List](#), [Message](#)

Line Command Filter

This command copies one line of text from the standard input device to the standard output device.

LINE

Commands

Operation The next line of text from the standard input device is copied to the standard output device.

In the following commands a descriptive line is written to a file, followed by a [HeadingStatement](#) that appends its output to the same file.

```
>line > filelist.output
The following is a listing of all backup files.
```

```
>filelist *.backlib*. * *.backup >> filelist.output
```

In this example the first line of the KEYWORD.HEADER is copied to a file, followed by other information appended to that file.

```
>line < keyword.header > keywords.list
```

```
>keywords * >> keywords.list
```

Notes This command is normally used to write a heading or descriptive line of text to a file before writing other text to the file.

A line of text is always terminated with a CR (file) or a CR,LF (console). When the standard input device is the console, the input is terminated by pressing **Enter ↵**.

Defaults The standard input and output devices are normally the console keyboard and display.

See also [CopyFile](#), [Echo](#), [Head](#), [Tail](#), [Tee](#)

LineEdit Command

The LineEdit command is a utility program that, unlike WindoWriter, edits ASCII text files using line-edit mode rather than a full-screen editor.

LINEEDIT *file* (*option*

file » file name with optional path; wild cards not allowed

option » BACKUP
 NOBACKUP

Command synonym: **LEDIT**

Commands

Operation

This command edits an existing text file or creates a new text file. When the command is first entered, a message displays indicating whether the text file is new or an existing file:

```
>lineedit sample.text
New file "SAMPLE.TEXT".
Edit:
*
```

or

```
>ledit system.theos32.devnames
Old file: "SYSTEM.TEOS32.DEVNAMES:S".
Edit:
*
```

The edit prompt character is the asterisk (*) and, when displayed at the beginning of a line, it indicates that LineEdit is waiting for an editing command.

There is an internal help display available in LineEdit that can be invoked by pressing **F1**. This help display summarizes all of the commands available while using the LineEdit command.

Cmd	Operands	Meaning
?		Query last command
;		Comment
Again		Reexecute last "string" command
Bottom		Go to bottom of file
CAsE	[U L M]	Set case mode Upper, Lower (inverted), Mixed
Change	[/frstr/tostr[/ [#lines [#occur [#start]]]]]	Global change
COlumn		Display column ruler
COMbine		Append next line to current line
CSi	command	Execute THEOS command, then return to LINEEDIT
DElete	[#lines /str/]	Delete lines
Down	[#lines]	Go down
DUp	[#lines]	Duplicate current line
FIle	[filename]	Save the file, then exit
Find	[text]	Anchored locate
Get	[filename][frct / frstr/ [toct tostr/]]	Merge from another file
Help		Ask for help
Input	[text]	Insert a line or Insert mode
LEngth		Ask for size of file
LIst	#lines	Display lines
Locate	[/str[/]]	Locate string or Locate again
ModiFy	[#lines]	Modify lines
NAme	[filename]	Change file name
Next	[#lines]	Go down or locate
Page		Display full page and go down
Ut	[filename] [/tostr ct]	Write lines to another file
PUTD	[filename] [/tostr ct]	Write line and delete
Quit	[retcode command]	Exit and execute THEOS command
ReplacE	[text]	Replace lines
Save	[filename]	Save file
SPlit	[? /str/]	Split current line into two lines
TABset	[list of col numbers]	Set tab stops
TOp		Go to top of file
Type	[#lines]	Display lines
Up	[#lines /str/]	Go up
VERify	[on off]	Verify (display) status
X,Y,Z	[#lines statement "filename" [line#]]	Macro

Table 19: LINEEDIT Command Summary

Options

BACKUP

A default option that tells LineEdit to make a backup copy of the prior version of the text file.

When the file is saved with either the **SAVE** or the **FILE** edit commands, LineEdit checks to see if there is a file on disk with the same name as the file it is about to write to disk. If it does exist, LineEdit renames that existing file to be a backup file. The rules used for determining the name of this backup file are:

1. Is the existing file a member of a library and does a library exist with the same file-name but with a file-type of **BACKLIB**? If so, rename the file into the **BACKLIB** library, replacing any existing file with the same member-name in that **BACKLIB** library.
2. Is there a library with a file-name equal to the current account name and a file-type of **BACKLIB**? If so, rename the file into that library, replacing any existing file with the same member-name.
3. If there is no **BACKLIB** library that might apply, then rename the existing file to have a file-type of **BACKUP**, replacing any prior version of the backup file name.

Note: Unlike WindoWriter, only one level of backup is maintained for the file.

File:	TEXT.FILE	LIBNAME.LIBTYPE.TEXT
1st choice:	<i>account</i> .BACKLIB.TEXT	LIBNAME.BACKLIB.TEXT
2nd choice:	TEXT.BACKUP	<i>account</i> .BACKLIB.TEXT
3rd choice:		TEXT.BACKUP

Table 20: LINEEDIT Backups

NOBACKUP Tells LineEdit that, when an existing file is saved to disk, the prior version of any file with the same name is to be over-written without renaming it to be a backup of the original.

Notes

The LineEdit command is provided as a utility that can be used by EXEC language programs and application programs that need to edit a text file without operator assistance. WindoWriter, although a far more powerful program, is a full-screen editor and, as such, requires all of its commands to be entered from the keyboard. LineEdit, being a line-oriented editor, can receive all of its commands from a text file or an EXEC program's &stack data.

```
>list updsyn.exec

; EXEC to update standard synonym with application name
;
&begstack
find DATE
up 1
input CUSTOMER 2
file
&end

lineedit system.theos32.synonym

>updsyn
Old file "SYSTEM.THEOS32.SYNONYM:S".
Edit:
*f DATE
DATE 4
*u 1
CRT 3
*i CUSTOMER 2
*file
"SYSTEM.THEOS32.SYNONYM:S" filed.
```

Defaults

[BACKUP](#) is the default option.

Restrictions

Only stream files containing ASCII text may be edited with this command. Use Patch for making changes in other file organizations or contents.

See also

[Patch](#), [WinWrite](#)

For a complete discussion of the operation of LineEdit and the commands available for editing text files, refer to Appendix B: "[LineEdit Text File Editor](#)," on page [679](#).

List Command

List displays the contents of any data or program file on the standard output device.

1 LIST *file...* (*option*

2 LIST - (*option*

file

option

» file name with optional path; may contain wild cards

»

BINARY	HEAD	OLDATE	TRUNC <i>nnn</i>
COM <i>nn</i>	HEX	PAGE <i>nnn</i>	<i>fromline</i>
FILES	INDENT <i>nn</i>	PRT <i>nn</i>	<i>toline</i>
FORMAT	NOHEAD	TITLE <i>ttt</i>	
FROM <i>kkk</i>	NUMBERED	TO <i>kkk</i>	

Commands

Operation **Mode 1**—The *file* or *files* specified are displayed on the standard output device (console).

Mode 2—This mode can be used when standard input has been redirected (as in a pipe) or when you want to supply the list of files from the keyboard (be sure to use option [FILES](#)).

```
>filelist system.help32.1* system.help32.s* | list (file
```

When the list of files comes from the console keyboard the “-” must be specified on the command line. Terminate the list with **Ctrl**+**D**.

```
>list - (file
system.help32.1*
system.help32.s*
Ctrl+D
```

The above two commands display all of the help files starting with “L” or “S.”

Options	<u>B</u>INARY	Displays the <i>file</i> in hexadecimal. See “ Binary & Hex Displays ” on page 380.
	<u>C</u>OMnn	The display is sent to the attached COM nn device instead of the standard output device.
	<u>F</u>ILES	<p>Indicates that <i>file</i> is an ASCII stream file with each record in the file specifying a single file name. The file name specifications in this file may include the path and wild cards.</p> <p>The SELECTED.FILES and SELECTED.EXEC files created by FileList and the FOUND.EXEC created by Look can be used for this specification file (see “The EXEC and FILES Options” on page 338). You may also create the file with an editor or application program.</p> <p>For instance, FileList is used to create a list of files to be examined:</p> <pre data-bbox="639 684 1214 764"> >filelist *.data:a (exec >filelist a not *.data:a (10/1/95 exec append </pre> <p>A SELECTED.EXEC file now exists that lists all of the “data” files and all files that have been changed since 10/01/1995. The following command lists these files:</p> <pre data-bbox="639 924 959 949"> >list selected.exec (file </pre>
	<u>F</u>ORMAT	<p>The file contains first-character ANSI forms-control formatting codes. List will use these codes instead of counting lines and displaying one record per line.</p> <p>The ANSI forms-control standard specifies that the first character of each record is the page advancement control. This character is <u>never printed</u>. If these codes are not present at the beginning of each line, the first character of the line that was intended to be displayed is instead interpreted as the ANSI forms-control character and is not displayed.</p>

These codes are:

Code	Meaning
1	Eject page.
+	Do not advance—overprint the previous line. This can only be done if the printer does not perform an automatic line advance with each carriage return. (Option ALF not specified on printer attachment.)
0	Advance two lines, skipping one blank line.
-	Advance three lines, skipping two blank lines.
<i>other</i>	All other characters are not printed and cause one line to advance. By convention, the space character is used for this purpose.

Table 21: ANSI Forms Control Codes

FROM

This option can only be used when *file* is direct, indexed or keyed. For direct files, *kkk* is a number that specifies the first record number displayed. For indexed and keyed files, *kkk* is the key of the first record displayed. Only records whose keys are greater than or equal to this *kkk* will be included in the display.

This option can be used with or without the **TO** option.

```
>list system.theos32.keyword (from 20 nohead
```

```
020: 1QUERY
021: 2NOQUERY
022: 5PRIVATE
...
```

```
>list customer.master (from "Conc" trunc 47 noh
```

```
Concrete Consultants, Inc.: "1300 Southwest Eve
Continental Lumber Services: "101 West 43rd", "P.
THEOS Software Corporation: "333 Oakland Blvd,
Walnut Creek Chamber of Commerce: "100 North Mai
```

fromline

The first numeric token is assumed to be the starting line or record number. This can only be used on stream and direct files. Use the **FROM** option for keyed and indexed files.

This option indicates that the first line or record displayed is *fromline*. When the output is to the console, the browse keys

can still be used to display pages containing records before *fromline*.

HEADING A default option that causes the standard page heading to display. This standard page heading includes the file name on the left side of the page and the date, time and page number on the right.

HEX Displays the file in hexadecimal. See “[Binary & Hex Displays](#)” on page 380.

INDENT All lines output are to be indented *nn* columns.

NOHEAD Suppresses the standard page header.

NUMBERED Causes a sequentially assigned line number to display at the beginning of each line.

```
>list system.help32.list (numbered nohead
```

```
1 SELF1 248:58:0 CPSIO B38400,W8,E2
2 SELF2 249:58:1 CPSIO B38400,W8,E2
3 SELF3 246:58:0 CPSIO B38400,W8,E2
...
```

```
>list system.help32.list (nohead
```

```
SELF1 248:58:0 CPSIO B38400,W8,E2
SELF2 249:58:1 CPSIO B38400,W8,E2
SELF3 246:58:0 CPSIO B38400,W8,E2
...
```

OLDDATE Causes the date and time used in the page heading to be the *file's* last change date and time. When this option is not used the current system date and time displays.

PAGE Indicates that the first page displayed is page number *nnn*.

Note: When the output is to the console, the browse keys can still be used to display pages before page number *nnn*.

PRT*nn* Indicates that List is to print *file* on the attached printer number *nn*.

The option keyword PRT may be specified as PRT, PRINT or PRINTER.

TITLE The word or quoted term following the keyword TITLE is used in the heading line instead of the *file's* file name.

```
>list system.help32.list (title "Device Names"
```

```
Device Names                23 Oct 1996 11:55am Page 1

SELF1  248:58:0    CPSIO B38400,W8,E2
SELF2  249:58:1    CPSIO B38400,W8,E2
SELF3  246:58:0    CPSIO B38400,W8,E2
...

```

TO

This option can only be used when *file* is direct, indexed or keyed. For direct files, *kkk* is a number that specifies the last record number displayed. For indexed and keyed files, *kkk* is the key of the last record displayed. Only records whose keys are less than or equal to this *kkk* will be included in the display.

This option can be used with or without the **FROM** option.

toline

The second numeric token is assumed to be the ending line or record number. This can only be used if *fromline* is specified on a stream or direct file. Use the **TO** option for keyed and indexed files.

This option indicates that the last line or record displayed is *toline*. The browse keys cannot be used to display beyond this *toline* record.

TRUNCATE

Each line output is truncated at column *nnn*. If used in combination with the **INDENT** option, the truncation is performed after the indent spaces are added.

When **TRUNCATE** is not used, lines are not truncated and those lines that are longer than the attached length of the display device (console or printer) are wrapped to the next line.

List Displays Most files are displayed as text. For instance:

```
>list system.help32.cat

SYSTEM.HELP32.CAT                23 Oct 1996 8:26am Page 1

Function:      The CAT command will concatenate files and
                copy them to the standard output device. If a
                file name is specified, or if the file name
...

```

This type of display is used by default for all files except programs, which use the binary display format described next.

The heading line of all displays on the console or printer (not those redirected to a file) shows the file name on the left and the date, time and page number on the right. This heading line can be suppressed with the **NOHEAD** option, and the date can be set to the file's last change date with the **OLDDATE** option. C language source programs can set the left side of this heading line with the **#title** directive.

For multiple-page displays, the standard page browsing keys are recognized. Refer to “[Multiple-page Display Browsing](#)” on page 77. In addition to these standard keys, you may also use **PageUp** and **PageDown** following a number such as: **6,PageDown**. This means to advance six pages rather than the default of one.

Binary & Hex Displays Program files do not contain textual information and always display using the **BINARY** display format.

```
>list sample.command

SAMPLE.COMMAND                    23 Oct 1996 8:26am Page 1

000000: 8603DEC0 4E060000 6C010000 00080000 '...ÝfN...1.....'
000010: 940e0000 01000800 00006000 00000100 '.....`.....'
000020: 00000000 b00ac15f 35623e56 061b8161 '....l.%.5b>V...a'
...

```

In this display each line contains three pieces of information.

- ▶ On the left side is the relative location in hexadecimal.
- ▶ In the middle, 16 bytes of the file are shown in hexadecimal. This middle section is grouped in four, four-byte columns.
- ▶ On the right is the ASCII representation of those 16 bytes of the file. Any bytes that do not have a corresponding ASCII character are shown with a period.

This display format can be forced for any file by using the [BINARY](#) option.

Data files (direct, indexed and keyed) can use a third type of display invoked with the [HEX](#) option:

```
>list customer.master (hex
```

```
CUSTOMER.MASTER 23 Oct 1996 8:26am Page 1
```

```
KEY:
```

```
0000: 41414120 56656E64 696E6720 53706563 'AAA Vending Spec'
```

```
0010: 69616C69 73747300 00000000 00000000 'ialists.....'
```

```
0020: 00000000 00000000 '.....'
```

```
DATA:
```

```
0000: 041A3132 38323820 536F7574 68776573 '..12828 Southwes'
```

```
0010: 74205061 726B2050 6C616365 0400040D 't Park Place....'
```

```
0020: 43617374 726F2056 616C6C65 79040243 'Castro Valley..C'
```

```
...
```

OR

```
>list system.theos32.keyword (hex
```

```
SYSTEM.TEOS32.KEYWORD 23 Oct 1998 8:26am Page 1
```

```
Direct record: 1, reclen: 10
```

```
0000: 31594553 20202020 200D '1YES .'
```

```
Direct record: 2, reclen: 10
```

```
0000: 314E4F20 20202020 200D '1NO .'
```

```
...
```

This [HEX](#) display is similar to the [BINARY](#) display except each record is displayed separately, with the first column showing the relative location of the data within the record. For keyed and indexed files, the key is shown separately from the record. Direct files show the record number and record length on a line directly above the record contents.

Notes

If *file* is a “typeless” file description, there is no default library defined and the environment variable [FILETYPE](#) is defined, the value of [FILETYPE](#) is appended to *file* to form a complete file description with file name and file type. To list a typeless file, you should specify the file description with a period terminator. Refer to page [105](#) for more information about this environment variable.

MultiUser BASIC language source programs are displayed by passing the file name with the option LIST to the BASIC32 command.

For instance:

```
>list sample.basic
```

is identical to:

```
>basic32 sample.basic (list
```

- Defaults** [HEADING](#) is a default option. [BINARY](#) is a default option when programs are listed.
- Restrictions** The *file* must have read access enabled. See “[File Access Protection](#)” on page [143](#).
- See also** [CopyFile](#) and [WinWrite](#)

The BASIC language compiler has a built-in ability to list BASIC language source programs.

Load Command

The Load command loads a program or data file into memory for subsequent usage.

```
LOAD  program...
```

```
program          »  name of program or file to load into memory
```

Operation The *program* or *programs* are loaded into memory. When *program* is a simple name, the following locations are searched:

```
SYSTEM.CMD32.program
SYSTEM.TEOS32.program
program.COMMAND
```

If program is already loaded into memory a second copy is not loaded.

```
>load csi message keyword
```

```
>load user.cmd32.menu
```

Notes Any compiled program (organization code “P” or “Program”) may be loaded into memory. In addition, the following data files may also be loaded:

```
SYSTEM.TEOS32.CSI
SYSTEM.TEOS32.DEV NAMES
SYSTEM.TEOS32.KEYWORD
SYSTEM.TEOS32.MESSAGE
SYSTEM.TEOS32.SYNONYM
```

Programs are normally loaded into memory when needed and their use count is set to one. If a program is already in memory because another user or task is using it, or the program was loaded by this command or the system start-up process, that copy of the program is used and its use count is incremented. When a program is no longer needed by a user (program exit), its use count is decremented and, if zero, the program is unloaded.

Programs loaded by this command or the system start-up process never have their use count decremented to zero except by [Unload](#) which is described on page [582](#).

Disk Caching Loading programs and certain key data files into memory before they are needed can increase the performance of your system slightly. However, if sufficient memory is set aside for disk caching, the performance increase will be minimal, at best, and may even degrade.

When disk caching is enabled, it is best to not load any programs or files with this command or [Sysgen](#).

Caution Do not make changes to programs or files (with [LineEdit](#), [Patch](#), recompilation, *etc.*) that have been loaded into memory. Any changes to the files affect only the disk image of the file and not the loaded version. Before making changes you should first [Unload](#) the program and file.

Do not erase or change the location of a file from disk after it has been loaded. If this is done, the loaded copy of the file will not be used and cannot be unloaded except by rebooting the system.

Restrictions The Load command requires a privilege level of five.

See also [Sysgen](#), [Unload](#)

LogName Command

Displays the account name that you are logged onto.

LOGNAME

Operation The current account name is output to the standard output device.

```
>logon develop
```

```
>logname  
DEVELOP
```

See also [Ident](#), [Show WHO](#), [WhoAml](#)

Logoff Command

Logon Command

The Logoff command logs off of the current account; the Logon command logs onto a new account.

1 LOGOFF

2 LOGON account (option

account » account name

option » NOEXEC NOTERM TERM

Operation **Mode 1**—You are logged off of the current account. Logging off includes:

- ▶ All files opened by this user are closed and all file and record locks are removed. (This is actually done by the system prior to invoking the Logoff command.)
- ▶ All screen windows are closed. See wFinish command described on page 593.
- ▶ A message displays showing the time and date of the logoff and the elapsed time that you were logged onto the system, along with the amount of CPU time used while logged onto the system.
- ▶ If HISTORY is ON, a record is written to the system history log file recording the fact that you have logged off of an account. See “SYSTEM.HISTORY” on page 722.
- ▶ All privately attached devices (other than the console, slave printers and VDI devices) are detached.
- ▶ All publicly attached devices are reattached. This includes the devices attached via Sysgen and those attached by you or other users with the Attach command with option PUBLIC.

The queues assigned to the public spooled printers are reassigned to their initial values. Spooled printers defined by Sysgen are assigned the queues defined in Sysgen, other printers are assigned queues in a one-to-one basis such as queue A to PRT1, queue B to PRT2, etc.

- ▶ All user environment variables are cleared.

>logoff

Logoff at 9:12am PDT on Wednesday, October 16, 1996.

Duration 1:48:36, cpu 53.780.

After Logoff completes (and the connection was not terminated because of the Exit command or the **TERM** option), it invokes a special mode of the Logon command that prompts you for the new account name and password.

Mode 2—The Logon command has two modes of operation: one invoked by the Logoff command and the other invoked when Logon is invoked from a command line.

When Logon is invoked by Logoff, it finds and displays the contents of the file /SYSTEM.MENU32.LOGON:S. It then displays the “Logon please:” prompt, which allows you to specify the account name that you want to log onto. In this mode you can specify the password immediately following the account name by entering the account name, comma and then the password. If the account has a password and it is not entered with the account name, you are prompted for it with “Password?” In either case, when the password is entered it is not displayed. Instead, asterisks appear for each character of the password typed.

When Logon is invoked from a command line, it performs a *lateral logon*. The Logoff process is not invoked and the current elapsed time and CPU time are transferred to the new logon session. In this mode you must specify the account name on the command line and you cannot include the password with the account name. If the account has a password, you are prompted for it with “Password?” When the password is entered, it is not displayed. Instead, asterisks appear for each character of the password typed.

A lateral logon does not clear existing user environment variables, does not detach private devices and does not reattach public devices. It resets only the account name and number and the privilege level of the account. Other system environment variables are changed only if the new account’s environment definition specifies new settings.

In both modes of Logon, when the account name and password are properly entered, the logon process is performed:

- ▶ The environment switches and variables are set according to the account definition. See “[Account](#)” on page 156. If the new work drive is different than the prior work drive, the work files are copied to the new work drive.
- ▶ The system’s time-of-day clock is adjusted for any change in daylight savings time, if necessary. See “[Daylight Savings Time and Automatic Adjustment](#)” on page 757.
- ▶ If the [TERM](#) option is in effect, the EXEC program that invoked the Logon is terminated. When [NOTERM](#) is in effect (a default option), the EXEC program that invoked the Logon command is not terminated.
- ▶ If HISTORY is ON, a record is written to the system history log file recording the fact that you have logged onto an account. See “[SYSTEM.HISTORY](#)” on page 722. Unsuccessful attempts to log onto an account are also recorded there.
- ▶ If a SYSTEM.LOGON file exists it is displayed on the console.
- ▶ If *account* is not the system account or a synonym to it, the *account*.LOGON file is displayed on the console.
- ▶ If a SYSTEM.REMINDER file exists and there is one or more entries for today’s date, the reminder messages are displayed.
- ▶ If *account* is not the system account or a synonym to it and the *account*.REMINDER file exists and there is one or more entries for today’s date, the reminder messages are displayed. See “[Reminder](#)” on page 452.
- ▶ The logon time and date are displayed.
- ▶ If the [NOEXEC](#) option is not used, a search is made for an EXEC program with a file-name or member-name of *account*. The normal locations for programs are searched (user command library, SYSTEM.CMD32 library and *account*.EXEC flat files) and the environment variable [PATH](#) is used when searching for these logon files. All drives in the drive search sequence are examined. If an EXEC program is found, it is executed; otherwise processing continues at the CSI or with the next statement in the EXEC program that invoked Logon (if [NOTERM](#) is in effect).

- Options
- NOEXEC

Do not execute the *account* EXEC.
- NOTERM

Do not terminate execution of the EXEC calling the Logon command. This is a default option.
- TERM

Terminates execution of an EXEC program if it was executing when Logon was invoked.

Notes

When the user is connected via a network connection, use the Exit command to log off and disconnect the user.

Neither the Logoff nor the Logon commands clear the console display. However, one or more of the displayed files (/SYSTEM.MENU32.LOGON:S, /SYSTEM.LOGON:S, or /*account*.LOGON:S) may clear the screen by embedding a form-feed code (^L) in the text file.

Defaults

[NOTERM](#) is a default option.

When logging onto an account, the environment is set to the following defaults unless the account definition specifically states otherwise:

PROMPT

=

\g

LIBRARY

=

CMDLIB

=

OBJLIB

=

COPYLIB

=

LINKLIB

=

See also

[Account](#), [Exit](#), [LogName](#), [Show](#), [Start](#), [Stop](#), [Sysgen](#)

Look Command

The Look command searches files for specific text.

LOOK *file* (*option pattern* ...

<i>file</i>	»	file name with optional path; may include wild cards
<i>pattern</i>	»	text string to look for; may contain “regular expressions”
<i>option</i>	»	APPEND EXEC FILES

Operation *file* is examined, record by record, looking for any and all instances of *pattern*. An exact match must occur between pattern and the text in the file. This exact match includes the case mode of the characters.

To perform inexact match comparisons, the *pattern* must contain **regular expressions** (see “[Regular Expressions](#)” on page 392).

When Look displays its results on the console, it displays every line of the file that contains *pattern*. The *pattern* in the line is highlighted in reverse video and the line is identified with its line number:

```
>look system.help32.look ("THEOS"
```

```
File: SYSTEM.HELP32.LOOK:S
```

```
17          Unlike other THEOS commands, the options
38          >LOOK SYSTEM. THEOS32.DEV NAMES ("Wyse"
```

file may be a subdirectory name, meaning that all of the files in the subdirectory are to be searched.

```
>look /letters ("1994"
```

is equivalent to:

```
>look /letters/*. * ("1994"
```

When more than one *pattern* is specified, it means “or.” For instance:

```
>look some.file ("the" "and" "of"
```

Will look in SOME.FILE for the characters “the” or “and” or “of.”

Options**APPEND**

Indicates that, if *file* contains *pattern*, *file* is output to FOUND.EXEC, appending a line to any existing FOUND.EXEC.

EXEC

Indicates that *file* is output to FOUND.EXEC, similar to the [FileList](#) EXEC option. Any existing FOUND.EXEC file is erased first. Thus, if no *file* contains *pattern*, the FOUND.EXEC will be empty.

If [APPEND](#) is used instead of, or in addition to EXEC, any existing FOUND.EXEC is not erased and any *file* containing *pattern* causes lines to be appended to the end of the existing FOUND.EXEC.

```
>look system.help32.* (exec FILES
```

```
>list found.exec (nohead
```

```
&1 SYSTEM.HELP32.B3220:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 SYSTEM.HELP32.BASIC32:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 SYSTEM.HELP32.CHANGE:S &2 &3 &4 &5 &6 &7 &8 &9 &10
&1 SYSTEM.HELP32.COPYFILE:S &2 &3 &4 &5 &6 &7 &8 &9 &10
...
```

FILES

Indicates that *file* is an ASCII stream file with each record in the file specifying a single file name. The file name specifications in this file may include the path to the file and wild cards.

The SELECTED.FILES and SELECTED.EXEC files created by the [FileList](#) command and the FOUND.EXEC created by previous usage of the Look command, can be used for this specification file or you may create the file with an editor or application program.

For instance, the [FileList](#) is used to create a list of files to be examined:

```
>filelist system.help32.* (10/1/95 exec append
```

A SELECTED.EXEC file now exists that lists all help files that have been changed since 10/01/1995.

The following commands create a list of these files that contain “PRT” and then, using that list, examines the files that contain “PRT” to find files that also contain “SORT.”

```
>look selected.exec (files exec "PRT"

>look found.exec (files "SORT"

File: SYSTEM.HEL32.ACCOUNT:S
  40 SORT                When used with the TYPE or Ö
...
```

Regular Expressions

pattern is actually a string expression called a **regular expression**. A regular expression is a sequence of characters consisting of **literal characters** and **metacharacters**.

Literal characters are characters that match in a one-for-one relationship with the text in the file. For instance, the search pattern “abc” is composed solely of literal characters. This pattern matches only when the text file contains a sequence of three characters that exactly equals ‘a,’ ‘b,’ and ‘c.’

In addition to the normal ASCII characters, you may specify certain control characters as literal characters. These control characters must be specified with the following codes:

Specification	Meaning
\a	BELL or [Ctrl]+[G] code.
\b	Backspace or [Ctrl]+[H] code.
\f	Form-feed or [Ctrl]+[L] code.
\n	New-line or [Ctrl]+[J] code.
\r	Return or [Ctrl]+[M] code.
\t	Tab or [Ctrl]+[I] code.
\v	Vertical tab or [Ctrl]+[K] code.
\0	Null or [Ctrl]+[@] code.
\'	Single-quote character
\"	Double-quote character

Table 22: Regular Expression Control Character Specification

A **metacharacter** is a character or group of characters that represents something other than themselves. The wild cards allowed for file specifications by most of the THEOS commands are examples of metacharacters.

The following table shows all of the metacharacters allowed in regular expressions.

Specification	Meaning
\^	Anchor to start of line.
\\$	Anchor to end of line.
\. or \?	Any character matches.
\@	Any letter matches (A–Z and a–z).
\#	Any digit matches (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0).
\\	Metacharacter escape. This is how a backslant literal character is specified.
\[Start character set.
\]	End character set.
\{	Start general repeat specification.
\}	End general repeat specification.
*	Repeat zero or more times.
\+	Repeat one or more times.

Table 23: Regular Expression Metacharacters

Regular Expression Anchors

The first two metacharacters are called ***anchors***. These anchor other text to the beginning or end of the record searched. For instance, the pattern

```
"\^The"
```

means that “The” is searched for but only when it occurs at the beginning of a record. Similarly,

```
"the\$"
```

is a pattern that means search for “the” at the end of a record. Note that this last pattern will not find “the” if there is a space at the end of the line.

Regular Expression Wild Cards

There are three “wild card” metacharacters that allow you to search for matches on any character (`\.`), any letter (`\@`) and any digit (`\#`). For example, the pattern:

```
"THEOS\#86"
```

will match any sequence of characters starting with “THEOS,” followed by any single digit, followed by an “8” and a “6.” Thus, it will match “THEOS186,” “THEOS286,” “THEOS386,” “THEOS486,” *etc.*

Regular Expression Character Sets

Searching for one character of a set of characters is done by specifying a ***character set***. For instance, to search for a hexadecimal digit you would specify a pattern containing:

```
"\[0123456789ABCDEFabcdef\]"
```

This pattern specifies a match on any character that is either a digit, an uppercase ‘A’ through ‘F’ or a lowercase ‘a’ through ‘f.’

As a convenience, when specifying character sets there are two characters that have special meaning. The hyphen or dash character (`-`), when specified between two characters, indicates a ***character set range***. For instance, the above pattern could have been specified with:

```
"\[0-9A-Fa-f\]"
```

which means the set of characters ‘0’ through ‘9,’ ‘A’ through ‘F’ and ‘a’ through ‘f.’ Ranges specified with the hyphen character refer to ranges in the ASCII collating sequence.

The hyphen does not have a special meaning when it occurs at the beginning or end of a character set specification. Thus, the pattern:

```
"\[-0-9\]"
```

means the set of characters dash and the digits 0 through 9.

The circumflex (`^`) is the other character that has special meaning when used in a character set specification. When the circumflex is used at the start of the character set specification, it means that it is an **exception set**. That is, it is a specification of characters that do not match. For instance, the pattern:

```
"\[ ^A-Z\]"
```

means a match on any character except uppercase letters. When the circumflex is used in a position other than at the start, it is merely a character included in the set.

Regular Expression Repeat Counts

A search pattern may be repeated by using the repeat metacharacters “`\{`” and “`\}`” to enclose the minimum and maximum repeat counts. For instance, the pattern:

```
" \@{4,6}\[ .\]"
```

means that you want a match on a space, followed by four to six letters, followed by a space or period.

It is not necessary to specify both the minimum and maximum repeat count values. A missing minimum value uses the default value of one for the minimum; a missing maximum value uses infinity for the maximum. Thus, a pattern repeat count of “`\{5\}`” means five or more occurrences; a pattern of “`\{,5\}`” means one to five occurrences.

As a convenience, there are two special metacharacters for specifying common repeat counts. The metacharacter “`*`” is a shorthand specification for the repeat count “`\{0\}`” meaning zero or more occurrences. The metacharacter “`\+`” is a shorthand specification for the repeat count “`\{1\}`” meaning one or more occurrences.

*Look***Notes**

When one or more *options* are used, they must be specified before any *pattern*. If no *option* is used and the *pattern* looks like an option, a “null” option should be used.

```
>look some.file (" " "EXEC"
```

If *file* is a “typeless” file description, there is no default library defined and the environment variable `FILETYPE` is defined, the value of `FILETYPE` is appended to *file* to form a complete file description with file name and file type. To examine a typeless file, you should specify the file description with a period terminator. See “`FILETYPE`” on page 105 for more information about this environment variable.

Remember to enclose the *pattern* in quotation marks if there is any doubt about the CSI retaining the case mode of the text or to ensure that it is not broken up into multiple *patterns*. For instance:

```
>look some.file (one phrase
```

is equivalent to:

```
>look some.file ("ONE" "PHRASE"
```

and searches `SOME.FILE` for any occurrence of “ONE” or “PHRASE.” To request a search of “one phrase” it must be specified as:

```
>look some.file ("one phrase"
```

Return Codes

The return code is set to zero if *file* does not contain *pattern*; otherwise, the return code is set to one.

See also

[Compare](#), [List](#)

Lowcase Command Filter

Lowcase copies a file to the standard output device, converting all letters to lowercase.

1 LOWCASE *file*

2 LOWCASE

file » file name with optional path; wild cards are not allowed

Command synonym: LC

Commands

Operation

Mode 1—*file* is copied to the standard output device with all uppercase letters converted to lowercase. The original *file* is unchanged.

```
>lowcase system.theos32.synonym
account 3
archive 2
asm32 3
attach 1
...
```

```
>lowcase system.theos32.synonym > synonym.list
```

Mode 2—Copies standard input to standard output, converting all uppercase letters to lowercase. This form is normally used only in a pipe. However, if standard input is the console, records are copied until a **Ctrl+D** is encountered, signaling the end of the input file.

```
>filelist *.* | lowcase > file.list
```

Notes

The command name LC is a synonym to the Lowcase command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file and, if standard synonyms are disabled (see “[Account](#)” on page 156 and “[STDSYN](#)” on page 101), this synonym name may not be allowed.

Restrictions

file must be a stream file.

See also

[Upcase](#)

Mailbox Command

The Mailbox command sends a message to another user's mailbox or retrieves your messages.

Commands

- 1 **MAILBOX** *user*
- 2 **MAILBOX** *user text*
- 3 **MAILBOX** *user file*
- 4 **MAILBOX** (*option*

<i>file</i>	»	file name with optional path
<i>text</i>	»	message text to send
<i>user</i>	»	account name to send mail to
<i>option</i>	»	PURGE PURGE *

The first three modes of the Mailbox command send mail to other users. The fourth mode retrieves or removes your mail.

Operation

Mode 1—This is the normal, multi-line mode of the Mailbox command. When the command is entered, you are prompted to enter one or more lines of text. To end the “mail,” press **Enter** at the beginning of a line with no text or spaces in it.

```
>mail shirley
Enter message text terminated by empty line.
Shirley,
The company picnic has been scheduled in August.
Please call me by Friday so we can arrange a planning
meeting as we have a lot to do in the next two weeks.
>
```

To enter a blank line without ending the message, remember to enter at least one space before pressing **Enter**.

After a line is entered you cannot edit it. You can prepare a long message with an editing program and then send it with Mode 3.

Mode 2—For short, single-line messages, this mode allows you to specify the message text on the command line. If the text contains commas, quotation marks or other punctuation characters, you should enclose the entire message in quotation marks.

```
>mail dave Please call me when you get in. It's important.
```

```
>mail eric "Your package arrived, and it's big. Call me."
```

Single-word messages cannot be sent with this mode because Mailbox will assume that the single word is a file name and it will try to use Mode 3.

Mode 3—Sends a previously created text file to *account's* mailbox. Use this mode to send large messages to a user or to send the same message to several user accounts.

Create the *file* with [LineEdit](#), WindoWriter or a customized application.

```
>mail accntg my.mail
```

Mode 4—This mode retrieves your mail or removes old mail from your mailbox.

```
>logon accntg
```

```
You have mail.
```

```
>mailbox
```

```
From JS, 10:12:44 09/04/96:
```

```
To: Payroll Department
```

```
From: John Smith
```

```
Subj: Payroll Deductions
```

```
This message is to inform you that I have adopted triplets  
and need to adjust my payroll deductions accordingly.
```

```
Can you please send me a new W4 form to fill out?
```

```
Thank you.  
OK to delete letter (Y|N) Y
```

As each message or letter is displayed you are asked if you want to delete it. If you respond ☒ then the message is marked as deleted (it is not phys-

ically deleted until the system manager uses the [PURGE](#) option described below). Respond with **[N]** if you do not want it deleted at this time.

Each time that Mode 4 is used without one of the [PURGE](#) options, all mail for your account is displayed whether it has been read before or not. When a message is read before and marked as deleted it is not displayed again.

Commands

Options

The two [PURGE](#) options can be used only when you are logged onto the SYSTEM account (id zero) and require a privilege level of five.

PURGE Removes all mail that has been marked as deleted.

PURGE * Removes all mail that has been marked as deleted and all mail that has been read but not deleted.

Notes

Users are notified that there is mail waiting for them when they log onto the account (see “[Logoff](#)” on page [386](#)). They can then use [Mode 4](#) of the Mailbox command to read their mail.

Mail for all users is saved in the file SYSTEM.MAILBOX:S. This file is created automatically the first time that Mailbox is used. Mail is only removed from this file with the [PURGE](#) option.

Do not confuse this command with THEO+Mail and the mail sent and received over the Internet with that command. Mailbox operates on mail sent by the [Msg](#) command only.

Restrictions

The [PURGE](#) and [PURGE *](#) options require that you be logged onto the SYSTEM account (account id=0) and that you have a privilege level of five.

See also

[Logon](#), [Msg](#), [Reminder](#)

MakeBoot Command EXEC

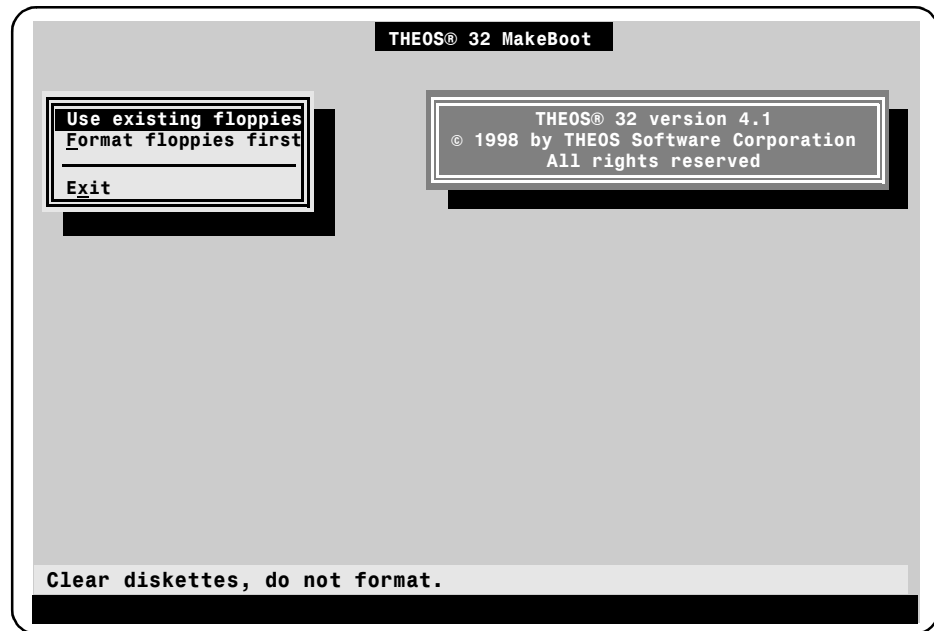
MakeBoot creates an *Emergency Boot Diskette* using the current operating system and configuration.

MAKEBOOT *drive*

drive » optional disk drive attachment letter

Operation

When MakeBoot is invoked it displays the following “windowed” menu screen.



Use the normal menu selection keys to select the desired function. These keys are described in “[Using Menus](#)” on page 75.

Use existing floppies

Choose this menu item if the diskettes are already formatted. MakeBoot clears the existing directory and resizes it to gain some additional space.

Format floppies first

Choose this menu item if the diskettes are not formatted. MakeBoot formats the diskettes using the highest capacity available on the drive. Then it copies the necessary files.

Exit

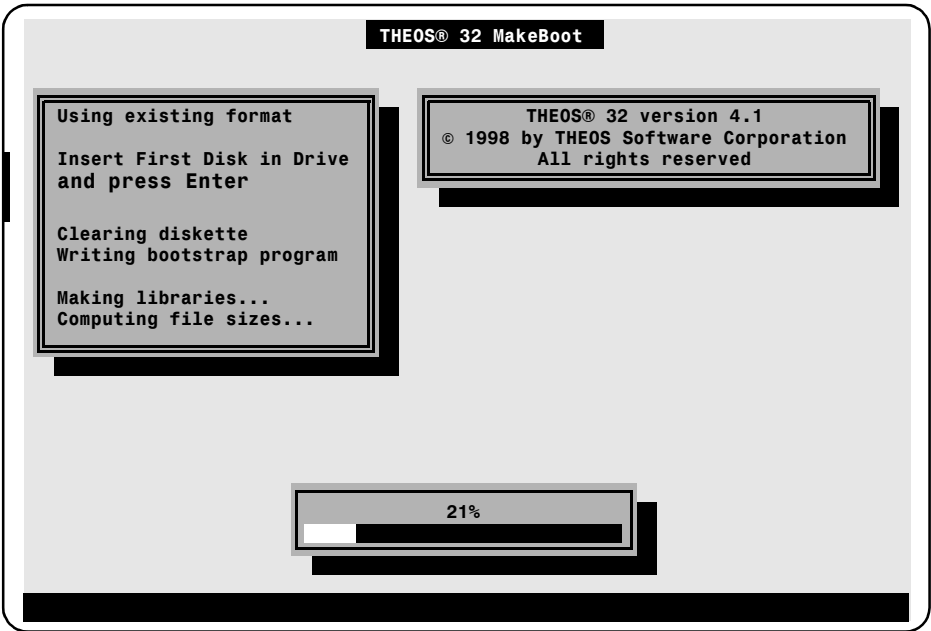
This menu item exits MakeBoot without modifying the diskette.

Notes

MakeBoot requires two diskettes to hold the necessary files for booting the computer. Be sure that you have these diskettes available before starting the process.

MakeBoot will ask you to put the first diskette in the drive and then it will format the diskette (if that option was selected), copy the necessary files to the diskette and then ask for the second diskette. It then formats that diskette (if selected) and copies the necessary files to that diskette.

As MakeBoot copies the necessary files to the floppy diskette, it reports on the files being copied in the window in the upper left of the screen. Also, there is a progress bar display near the bottom of the screen that reports on the percentage complete for the MakeBoot process.



Commands

A bootstrap loader is written to the first sectors of the first diskette. There will be some space available on the second diskette and you may copy additional files and programs if there is sufficient room for them.

MAKEBOOT Files:

The files copied to this “Emergency Boot Diskette” include:

	File
Operating system	SYSTEM.THEOS32.NUCLEUS
Command string interpreter	SYSTEM.THEOS32.CSI
Device drivers	SYSTEM.THEOS32.CLASS90 ¹ SYSTEM.THEOS32.DEV1 SYSTEM.THEOS32.DEV2 SYSTEM.THEOS32.DEV3 SYSTEM.THEOS32.DEV34 ² SYSTEM.THEOS32.DEV64 SYSTEM.THEOS32.DEV101 ²
SYSGEN configuration	SYSTEM.THEOS32.CONFIG
Accounting structure	SYSTEM.THEOS32.ACCOUNT

	File
Operating system support files	SYSTEM.TEOS32.BOOTER1 SYSTEM.TEOS32.BOOTER2 SYSTEM.TEOS32.BOOTER3 SYSTEM.TEOS32.BOOTMSG SYSTEM.TEOS32.CFGBUS SYSTEM.TEOS32.CLOCK SYSTEM.TEOS32.DEV NAMES SYSTEM.TEOS32.INSTLIST SYSTEM.TEOS32.KEYWORD SYSTEM.TEOS32.MESSAGE SYSTEM.TEOS32.SCSI ³ SYSTEM.TEOS32.SESSMAN SYSTEM.TEOS32.SYNONYM
Commands	SYSTEM.CMD32.ATTACH SYSTEM.CMD32.DISK SYSTEM.CMD32.LOGOFF SYSTEM.CMD32.LOGON SYSTEM.CMD32.MOUNT SYSTEM.CMD32.RESTORE SYSTEM.CMD32.SETUP SYSTEM.CMD32.SYSTEM SYSTEM.CMD32.TBACKUP
Command support files	SYSTEM.MENU32.ATTACH SYSTEM.MENU32.DISK SYSTEM.MENU32.FORM2 SYSTEM.MENU32.RESTORE SYSTEM.MENU32.TBACKUP SYSTEM.TEOS32.FORM2 SYSTEM.TEOS32.PART SYSTEM.TEOS32.SETDISK

1. The specific class code currently attached to the main console.
2. Only included if present in SYSTEM.TEOS32:S.
3. Only included if system contains a SCSI controller.

When this “Emergency Boot Diskette” is booted, there are sufficient commands to format the hard disk, attach different devices, restore from tape, “system” to the hard disk, *etc.* It is not intended as a general boot disk.

There may be sufficient space available on the disk to add more commands. If you do, remember to include any support files needed by the command (such as menus). You should not copy help files to this diskette.

If any significant changes are made to the system on the hard disk, you should recreate the “Emergency Boot Diskette.” Significant changes would include an upgrade to the operating system, new or changed account envi-

ronments, hard disk controller change and a change in the main console configuration or attachment.

Defaults The default drive is F, which is normally the first floppy drive.

Restrictions MakeBoot can only be run if you are logged onto the system account.

The diskette must be 1.44MB. 1.2MB diskettes are too small to hold the necessary files.

Using the Diskettes Refer to the section “[Booting with the Emergency Boot Diskettes](#)” on page 29 for a description of how these disks are used.

See also [CopyFile](#), [Disk](#)

Message Command

The Message command displays or maintains the SYSTEM.TEOS32.MESSAGE file.

Commands

- 1 MESSAGE * (PRT nn
- 2 MESSAGE
- 3 MESSAGE nn operand...

nn » message number to display

$operand$ » optional argument used by the message

Operation **Mode 1**—Displays all of the messages defined in SYSTEM.TEOS32.MESSAGE.

```
>message *
0: Logon please: \a
1: Password?
2: Command not found.\n
3: Insufficient memory.\n
...
```

Mode 2—This is the message maintenance mode. When this mode is entered, you are prompted to enter the message number:

```
>message
Enter message number:
```

At this prompt enter either the number of the message you want to view or change, or press **[Enter ↵]**, **[Esc]** or **[F9]** without a number to exit.

When a number is entered the current definition is displayed and you are asked for the new definition:

```
>message
Enter message number: 3

Old text: Insufficient memory.\n
New text:
```

To leave the message unchanged, press **[Enter ↵]**, **[Esc]** or **[F9]** without any characters or spaces.

To change or define a message, enter the new text. Messages may be a maximum of 64 characters in length. For information about message content and codes used, refer to “[Message File Syntax](#)” below.

When the new text for the message is entered, press **Enter** and you are prompted for another message to change.

Mode 3—Displays message number *nn* on the console (not the standard output device). If the message uses variables, the *operands* are used for the value of these variables. Any missing operands are displayed with an ellipsis.

```
>message 7
Invalid command syntax.

>message 19
File "... " not found.

>message 19 abc.def:g
File "ABC.DEF:G" not found.

>message 117 "Wednesday" "August" 7 1996
Date is set to Wednesday, August 7, 1996.
```

Options **PRT***nn* Indicates that Message is to print the messages on the attached printer number *nn*. The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

Message File Syntax Message text may contain plain text, display codes, video attributes, variable arguments and conditional expressions. Plain text includes all of the ASCII displayable characters (letters, digits, punctuation, *etc.*).

Display Codes

There are four display codes that can be embedded in a message:

Code	Meaning
\a	Sound the console bell.
\n	Start a new display line.
\t	Output spaces to next tab stop.
\\	Output a single backslant character.

Video Attributes

Video attributes such as reverse video, blink, *etc.* are specified by using their octal values preceded by a backslant, zero. Some of the common codes include:

Code	Meaning
\016	Reverse video on.
\017	Reverse video off.
\013	Underline on.
\026	Underline off.
\035	Blink on.
\036	Blink off.
\04	Half intensity on.
\05	Half intensity off.
\0202	Status line start.
\0203	Status line end.

Variables

Message text may specify that variable information will be inserted at the time the message is displayed. For instance, message 19 displays as “File “xxx” not found.” To indicate that the file name is inserted between the quotation marks, a variable number is used.

In message text, variables are always indicated by using a pair of “curly brace” characters to surround a variable number. Messages containing only one variable use variable number zero. Messages with more than one variable use numbers one, two, three, *etc.* For instance:

```
19: File "{0}" not found.\n
34: Device "{1}" is already attached to process {2}.\n
```

Conditional Expressions

Messages can use conditional expressions to evaluate the value of a variable and display different information depending upon that value. The general syntax for a conditional expression is:

```
{variable?value1=text1,value1=text2,...,text}
```


The value of *variable* is tested to see if it is *value1*, *value2*, etc. If it matches any of those values, then the corresponding *text* is output as the value of the expression. When a term does not have an equal sign it means that this is the “otherwise” clause and that *text* is output when the variable is not one of the previously listed values. An asterisk means that the original value of *variable* is used. For instance, message 105 is defined as:

```
105: {0?0=No,1=One,*} file{0?1=,s} changed.\n
```

This message will be displayed three different ways depending upon the value of the variable zero.

```
>message 105 0
No files changed.
```

```
>message 105 1
One file changed.
```

```
>message 105 2
2 files changed.
```

As you can see from the second variable expression {0?1=,s}, you can specify that nothing is output when the variable is a specific value. This expression states that if variable 0 is a “1,” output nothing; otherwise output an “s.”

Notes

Although the display produced by [Mode 1](#) is similar to the display produced by the [List](#) command, there are two significant differences:

1. The message numbers displayed by the **Message** command are numbered from zero, corresponding to the numbers used by programs to request one of the messages.
2. Embedded control codes are displayed graphically rather than interpreted. For instance, a new-line is displayed as \n.

Cautions

The SYSTEM.TEOS32.MESSAGE file contains all of the message text used by the operating system and its utilities. Changing an existing message can affect many programs.

You should not use any of the currently unused message numbers for your own purposes. All unused numbers are reserved for operating system usage. As updates to the software require new message text, they are added to this file. Also, this file is replaced in its entirety whenever the operating system is installed or updated.

See also

[Keyword](#)

MkDir Command

The MkDir or MD command creates a new subdirectory.

MKDIR *directory*

directory » new subdirectory name; may include path

Command synonym: **MD**

Operation Creates a new, empty subdirectory named *directory*.

```
>tree
/

>mkdir subdir

>tree
/
└─ subdir

>md subdir/sub

>tree
/
└─ subdir
   └─ sub
```

You can create a directory on another account by prepending the account name to the path for *directory*. For example:

```
>mkdir account\subdir
```

The above command creates the directory SUBDIR in the account named ACCOUNT.

Notes The command name MD is a synonym to the MkDir command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file. If standard synonyms are disabled (see “[Account](#)” on page 156 and “[STDSYN](#)” on page 101), this synonym name may not be allowed.

- Defaults** If no account name is specified for *directory*, the new subdirectory will be owned by the current account. If no path is specified for *directory*, the new subdirectory is created as a subdirectory to the current working directory.
- By convention only, directories are generally “typeless” but they may have both a file-name and a file-type.
- Restrictions** *directory* must not be the name of an existing file or subdirectory.
- directory* should not be /PIPE because this is a reserved name used by the system. You may, however, create a subordinate directory named PIPE, such as: /PRIVATE/PIPE.
- Subdirectories may be 21 levels deep. This means that /A/B/C/D/E/F/G/H/I/J/K/L/M/N/O/P/Q/R/S/T/U is valid, but adding another directory under U would be beyond the limit.
- A subdirectory may not be a member of a library.
- See also** [ChDir](#), [Create](#), [PWD](#), [RmDir](#)
- See “[Directories and Files](#)” on page 129 for additional information about directories and subdirectories.

More Command Filter

More copies a text file to the standard output device with page wait and browse capabilities.

1 MORE *file...* (*nnn*

2 MORE (*nnn*

file » file name with optional path; wild cards are not allowed

nnn » number of lines to display per page

Operation **Mode 1**—Copies *file* to the standard output device. If standard output is the console, page-waits are performed and browsing capabilities are present.

```
>more system.theos32.devnames
```

When the console screen is full, More displays its prompt on the last line. The MORE prompt consists of the word “More” and the amount of the file that has been displayed so far.

```
--More-- (52%)
```

At this time you may use any of the browse keys described on page 413.

Multiple files may be specified by listing the file names on the command line one file name after the other. When multiple files are listed this way, the **[N]** and **[P]** browse keys are enabled.

```
>more one.file two.file three.file
```

Mode 2—This mode applies when More is used in a pipe.

```
>number system.theos32.devnames | more
```

Options *nnn* Specifies the console page depth to use. When this option is not specified, the console’s attached screen size is used.

The screen size can be changed during the display with the **[Z]** browse key.

Browse Keys When the More prompt is displayed at the bottom of the screen you may use the special More browse keys.













Key	Action
 Enter	Display next line of <i>file</i> .
 Space	Display next screenful of <i>file</i> .
 D	Display next half-screenful of <i>file</i> .
<i>nnn</i>  F	Skip next <i>nnn</i> screens.
 N	Skip to next <i>file</i> .
 P	Skip back to previous <i>file</i> .
 Q	Quit.
<i>nnn</i>  S	Skip <i>nnn</i> lines.
<i>nnn</i>  Z	Set screen depth to <i>nnn</i> and display next screenful of <i>file</i> .
 / <i>expression</i>	Search for text matching <i>expression</i> . <i>Expression</i> is a regular expression as described on page 392.
 ! <i>command</i>	Execute CSI <i>command</i> and return.
 ?	Display browse help screen.

Table 24: More Command Browse Keys

Notes When used in a pipe, More should be the last command in the pipe so that its output goes to the console and you can browse through the file using the keyboard.

If *file* is a “typeless” file description, there is no default library defined and the environment variable FILETYPE is defined, the value of FILETYPE is appended to *file* to form a complete file description with file name and file type. To list a typeless file, you should specify the file description with a period terminator. See “FILETYPE” on page 105 for more information about this environment variable.

Defaults The default screen depth is the console’s attached page size.

Restrictions *file* must be an ASCII stream file.

See also CopyFile, List, Tee

Mount Command

Mount tells the operating system to reread the label information from a drive because the disk might have been changed.

MOUNT *drive*

drive » disk drive letter

Commands

Operation This program assumes that the disk in the drive might have been changed. Any current information about *drive* is disregarded and the disk drive is instructed to recalibrate its heads. This is a process that moves the read/write head to its home position. If supported by the drive mechanism and controller, the heads are moved slowly.

Once the heads are in a known position, the first sectors of the disk are read and, if it is a THEOS disk, the information is saved.

Notes If a disk is changed and the Mount command is not used, you will receive a message "Disk *drive* changed, need "XXXX" -" the next time that *drive* is accessed.

Cautions Use the **MOUNT** command every time that a disk is changed, even if the new disk has the same format as the prior disk. If the new disk has the same label as the prior disk, THEOS will not know that the disk has changed and may use information saved from the prior disk to do writes!

Return Codes This command reads the first sectors of the disk and, depending upon what it finds, sets the return code as follows:

Return Code	Meaning/Message
0	THEOS disk, mounted successfully
201	Disk not ready
203	Disk not initialized
204	Data transfer error
205	Sector not found
206	Track not found
207	Sector address error
563	Disk does not have a THEOS file system on it

Restrictions You cannot Mount the S drive. Use the [System](#) command instead.

See also [Attach](#), [Disk](#), [Reboot](#), [System](#)

Msg Command

Similar to the [Mailbox](#) command, **Msg** sends a message to another user. However, if the user is logged on, the message is displayed on their console immediately.

```
1 MSG user (option
2 MSG user message (option
3 MSG * (option
4 MSG * message (option
```

user » account name or partition number
message » optional message text to transmit
option » TITLE

Commands

Operation


Mode 1—This is the multi-line mode of the **Msg** command. When the command is invoked, you are prompted to enter one or more lines of text. To end the message press **Enter** at the beginning of a line with no text or spaces in it.

```
>msg Ted
Enter message text terminated by empty line.
Call me as soon as you get back from lunch. SomethingEnter
important has come up that changes EVERYTHING!Enter
Enter
```

On Ted's console, the message appears in a pop-up window as:

```
MSG from CHRIS (Pid=3)
Call me as soon as you get back from lunch. Something
important has come up that changes EVERYTHING!
Press ESC to clear MSG Window.
```

The receiving person must acknowledge the message by pressing **Esc** before it is erased.

To enter a blank line without ending the message, remember to enter at least one space before pressing **Enter** .

Mode 2—For short, single-line messages, this mode allows you to specify the message text on the command line. If the text contains lowercase text, commas, quotation marks or other punctuation characters that you want in the message, you should enclose the entire message in quotation marks.

```
>mail dave Please call me when you get in. It's important.
>mail eric "Your package arrived, and it's big. Call me."
```

These single-line messages are displayed on the receiving user's screen in a pop-up window just like the [Mode 1](#) messages.

Mode 3—Similar to [Mode 1](#) except that the message is transmitted to all users logged onto the system. This mode requires a privilege level of five. The message is only sent to the active session of a console with multiple sessions defined.

Mode 4—Similar to [Mode 2](#) except that the message is transmitted to all users logged onto the system. This mode requires a privilege level of five. The message is only sent to the active session of a console with multiple sessions defined.

Options

TITLE text Specifies an alternate title for the receiving user's message window. The title is displayed in the top frame, centered. When this option is not used the default title of "MSG From *your-account* (Pid=*your-pid*)" is used.

Notes

When *user* is specified with an account name, and more than one user is logged onto that account, all users logged onto that account receive the message. When there are no users logged onto the account name, you are asked if you want to put the message in the user's mailbox. When this happens, the operation is identical to the [Mailbox](#) command.

```
>msg accounts "Where is my pay check?"
User is not logged on.
Do you wish to deposit in mailbox? Y
```

The "Receive messages" field in the account environment prevents messages from being sent directly to a user's screen. Msg checks this switch on the user's environment and if it is disabled, informs you and allows you to put the message in the user's mailbox.

The "Receive messages" field status is ignored in [Mode 3](#) and [Mode 4](#): The message is sent to all users that are currently logged on.

When the message is directed to a multiple-session console, the message is displayed on that session only. If the session is not active it will not appear until the user switches to the session. Messages directed to all users (* specification) are displayed only on the active session of a multiple-session console.

When a message is sent to a user that already has an unacknowledged message on its screen, the new message is queued and is not displayed until the prior message is acknowledged and cleared from the display. As many as 50 messages for each user may be queued in this manner.

When a message is sent to an account name and there is more than one user logged onto that account name, all of those users will receive the message.

When the message is not put in the user's mailbox but is displayed in a pop-up window on the user's console, the last line of the message will prompt the receiver to "Press ESC to clear MSG Window." The text for this message is in the SYSTEM.MENU32.MSG:S file. This file may be edited to provide a custom, single-line prompt message.

Restrictions [Mode 3](#) and [Mode 4](#) require a privilege level of five.

You cannot send a message to yourself.

See also [Mailbox](#)

Number Command Filter

Number copies a file to the standard output device, numbering each line as it is copied.

1 NUMBER *file...* (*options*

2 NUMBER

file » file name with optional path

options » *start*
 increment

Operation

Mode 1—Each *file* is copied to the standard output device and numbered as it is copied. Specifying multiple *files* causes the second and remaining *files* to be appended to the *first* file copied to the standard output device and the numbering continues without being reset at the beginning of each file.

```
>number one.file
1 Line one
2 Line two
3 Line three

>number one.file one.file
1 Line one
2 Line two
3 Line three
4 Line one
5 Line two
6 Line three
```

This command is frequently used in a pipe:

```
>number some.text | tee numbered.text | more
```

Mode 2—Copies the file from the standard input device to the standard output device, numbering each line as it is copied. When the console is the standard input device you terminate the input by entering **Ctrl**, **D** on a line by itself.

- Options**
- start* The starting number to use for the first file copied to standard output. The default starting number is one.
- increment* This option may only be used in combination with *start*. Specifies the increment value for each line number used. The default *increment* is one.

```
>number one.file (100 100
100 Line one
200 Line two
300 Line three
```

Defaults *start* and *increment* have default values of one and one.

Restrictions *file* must be an ASCII stream file.

See also [Unnumber](#)

Password Command

The Password command allows you to change the password to your account.

PASSWORD

Commands

Operation	<p>The current account must have a password. If it does, you are prompted to enter the existing password to the account.</p> <pre data-bbox="495 489 738 569">>password Enter old password:</pre> <p>After entering the current password you are asked to enter the new password and then you are asked to enter it again to make sure that you did not mistype it.</p> <pre data-bbox="495 730 841 861">>password Enter old password: ***** Enter new password: ***** Enter new password: *****</pre>
Notes	<p>For information about the usage and limitations of passwords, refer to “Account Name and Number” on page 95.</p>
Notes	<p>You cannot remove a password with this command. Only the Account command can remove the password for an account.</p>
Cautions	<p>Remember to make a note of this new password. You will not be able to log onto this account without this password.</p>
Restrictions	<p>The account must have a password. If it doesn’t, ask the system administrator to add the password to the account with the Account command.</p>
See also	<p>Account, Logon</p>

Patch Command

The Patch command is a general purpose file and disk maintenance program. With it you can examine and change any file on the system or any sector of any attached disk.

1 PATCH *file* (*options*

2 PATCH *drive* (*options*

file » file name with optional path

drive » disk drive letter

options » **BINARY**
NOVIDEO

Operation

Mode 1—With this mode you can view and make changes to *file*. The *file* may be any data or program disk file. *file* may not be a subdirectory or a library name.

Refer to “[Patching Files](#)” on page 437 for a description of how Patch operates depending upon the file type (organization).

Mode 2—This mode allows you to view and make changes to the data in a specific disk sector.

Refer to “[Patching Sectors](#)” on page 438 for a description of this mode.

Options

BINARY Tells Patch that *file* is to be treated as a stream of bytes rather than records or a program.

Caution: You should only use this mode to view files. Essentially, this option tells Patch to ignore the structure of the file. If you make changes while this option is in effect, you may change a key (indexed or keyed file) to be unreachable or you may make records unreadable or programs unloadable.

NOVIDEO Tells Patch to start in the command mode rather than the full-screen display mode. This option is useful when Patch is being invoked from an EXEC program that has &STACK data that automatically updated a file.

Video Display Mode

Patch has two basic display modes: full-screen video mode and command mode. In full-screen video mode the screen is used to display as much as one sector (256 bytes) of the disk or file at a time.

```
>patch system.theos32.devnames
```

```

THEOS® 32 PATCH version 4.0

Filename: SYSTEM.LTHEOS32.DEVNAMES:S

00000000: 3B205379 6E6F6E79 6D732066 6F722041 ';' Synonyms for A'
00000010: 74746163 68206469 73706C61 790D3B0D 'ttach display.. '
00000020: 4D4F4445 4D202020 2020363A 353A3020 'MODEM      6:5:0 '
00000030: 20204350 53494F20 42333834 30302C57 ' CPSIO B38400,W'
00000040: 382C4534 2C433138 30203B20 4D6F6465 '8,E4,C180 ; Mode'
00000050: 6D20636F 6E6E6563 74696F6E 206F6E20 'm connection on '
00000060: 53494F31 0D444953 4B33E920 20202031 'SIO1.DISK3¼ 1 '
00000070: 3A313A30 20202044 20202020 20202020 ':1:0  D '
00000080: 20202020 20202020 20202020 203B2033 '          ; 3'
00000090: E920466C 6F707079 20647269 76650D44 '¼ Floppy drive.D'
000000A0: 49534B35 E8202020 20323A31 3A312020 'ISK5°  2:1:1 '
000000B0: 20442020 20202020 20202020 20202020 ' D '
000000C0: 20202020 2020203B 2035E820 466C6F70 '          ; 5° Flop'
000000D0: 70792064 72697665 0D484152 44444953 'py drive.HARDDIS'
000000E0: 4B203535 3A313030 3A302044 20202020 'K 55:100:0 D '
000000F0: 20202020 20202020 20202020 20202020 '

```

As you can see, this display is very similar to the display used by the [List](#) command when option HEX is specified. The first column shows the relative location of the data in the file. The middle portion of the screen shows the hexadecimal values for each byte of the file. And on the right is the ASCII display of those same data bytes.

In full-screen video mode you can make changes to the data on the screen or move to display other sectors of the file. The keys that can be used in this mode are:

Key	Meaning
Quit	Exits Patch without saving changes to the file.
File	Saves all changes made to the file and exits Patch. Note that saving changes when the file is direct, indexed or keyed must be done for each record with the Save key, before selecting another record in the file.
Save	Saves all changes made to the record or sector without exiting Patch.
Home	Position to the first byte of data on this screen. If you are already positioned on the first byte, then Home displays the first sector of the file or record and positions to the first byte of that sector.
End	Position to the last byte of data on this screen. If you are already positioned on the last byte, then End displays the last sector of the file and positions to the last byte of that sector.
Esc	Exits full-screen video mode and switches to command mode. Pressing Esc while in command mode returns to the full-screen video mode.
PageDown	Displays the next sector of the file. The cursor does not move. That is, if you were positioned to the third line, second byte of the current sector, you will still be positioned to the third line, second byte of the new sector.
PageUp	Displays the previous sector of the file. The cursor does not move.
Transpose or Ctrl O	Moves the cursor from the ASCII column to the hexadecimal column, or vice versa.
← → ↑ ↓	Move the cursor in the direction of the arrow. The left and right arrow keys move one byte; the up and down keys move by one line or 16 bytes. If the arrow key moves you off of this sector, the next or previous sector of the file or record is displayed. If there is no more file or record available in the direction desired, the cursor moves to the last or first byte of the file, as appropriate.

Table 25: Patch Video Mode Commands

Key	Meaning
Find	Allows you to jump to a new location in the file or disk. The operation of this key depends upon the type of Patch operation being performed. Direct, indexed and keyed files: You are asked for the record key that you want to find. Other files: You are asked for the relative location to jump to (in hexadecimal). Mode 2 of Patch: You are asked for the sector number to patch.
SchFwd SchBck	Allows you to position to the next or prior occurrence of a sequence of bytes in the file or record. The operation of these keys depends upon the location of the cursor when the key is pressed. ASCII column: Enter the ASCII text to search for. Hexadecimal columns: Enter the sequence of hexadecimal values to search for.
Again	Repeats the last SchFwd or SchBck performed.

Table 25: Patch Video Mode Commands

Changing Data on the Screen

Initially, when a sector of the file or record is displayed in full-screen video mode, the cursor is positioned to the first byte of the sector, in the ASCII display area of the screen. If the information that you want to change to is ASCII data, merely move the cursor to the position that you want changed and then type the replacement characters. (Patch only has replace mode. It is not possible to insert characters or bytes with Patch.)

If you need to change one or more locations to some binary or hexadecimal values, make sure the cursor is positioned in the middle, hexadecimal columns. Use the **Transpose** or **Ctrl**+**O** to switch back and forth. Position the cursor to the location that you want changed and type the replacement values. Note, when moving left or right with the arrow keys you move by bytes, not “nibbles” (four-bit hexadecimal characters).

**Command
Mode**

The Patch command or NOVIDEO mode provides all of the functionality of the full-screen mode plus some other important capabilities that are not easily implemented in a full-screen video mode. In command mode, Patch displays the **PATCH prompt** (/) indicating that it is ready to accept a command.

Expressions

Most of the Patch commands require an address or data values to be specified. Although these addresses and data values are normally specified with a numeric constant, they can be specified with an expression. In Patch, an expression contains one or more of the following elements:

- ▶ Numeric Constant
- ▶ ASCII String Constant
- ▶ Operator
- ▶ Variable
- ▶ Another expression enclosed within parentheses

Numeric Constants

Numeric constants are numbers specified in hexadecimal or decimal. By default, a number is a hexadecimal value unless it is terminated with the letter “t” or “T.”

1234	This is the number for 4,660 ₁₀ or 1234 ₁₆
1234t	This is the number for 1,234 ₁₀ or 04D2 ₁₆

ASCII String Constants

A value may be specified in ASCII by enclosing the ASCII characters within a pair of single quotation marks. Numeric values are always limited to 32 bits which is four bytes or four characters. When more than four characters are specified only the last four are used.

'abcd'	This is the value 1,633,837,924 ₁₀ or 61626364 ₁₆
'AbCdEfGh'	This is the value 1,164,330,856 ₁₀ or 45664768 ₁₆
'EfGh'	This is the value 1,164,330,856 ₁₀ or 45664768 ₁₆

Operators

There are many operators that may be used in an expression to modify an element or to join two or more elements together. In the following table *address*, *value₁* and *value₂* may be any value including another expression.

Operator	Meaning
<i>@address</i>	Indirection: Use value of location pointed to by <i>address</i>
<i>*</i>	Value of current address pointer
<i>~value</i>	Unary 1's complement of <i>value</i> (NOT)
<i>-value</i>	Unary 2's complement of <i>value</i> (NEG)
<i>(expression)</i>	Parenthesized subexpression
<i>value₁ value₂</i>	<i>value₁</i> is OR'd with <i>value₂</i>
<i>value₁ ^ value₂</i>	<i>value₁</i> is XOR'd with <i>value₂</i>
<i>value₁ & value₂</i>	<i>value₁</i> is AND'd with <i>value₂</i>
<i>value₁ << value₂</i>	Shift <i>value₁</i> left <i>value₂</i> bit positions
<i>value₁ >> value₂</i>	Shift <i>value₁</i> right <i>value₂</i> bit positions
<i>value₁ + value₂</i>	Add <i>value₁</i> to <i>value₂</i>
<i>value₁ - value₂</i>	Subtract <i>value₂</i> from <i>value₁</i>
<i>value₁ * value₂</i>	Multiply <i>value₁</i> by <i>value₂</i>
<i>value₁ / value₂</i>	Divide <i>value₁</i> by <i>value₂</i>
<i>value₁ % value₂</i>	Compute the remainder of <i>value₁</i> divided by <i>value₂</i>

Table 26: Patch Expression Operators

Variables

As many as 26 variables may be assigned values and used in expressions. Variable names are single letters, case insensitive (an uppercase "A" is the same as a lowercase "a").

A variable is assigned a value by using an assignment statement:

variable = expression

A variable is used in an expression by using its name followed by the dollar sign character (\$).

For instance:

```
/a=2000
0x00002000 (hex), 8192 (dec)
/b=a$+1000
0x00003000 (hex), 12288 (dec)
/
```

Patch Commands

When the Patch prompt is displayed the following commands may be used. Note that some of the commands have no meaning in certain situations. For instance, when patching a stream file the KEY command is invalid.

■ Assemble Command

Patch contains a built-in assembly language compiler that allows you to use Intel mnemonics when specifying changes to a program file.

```
A address
A
```

The A command accepts assembly language commands and stores the assembled code starting at location *address*. If *address* is omitted, the next location following the last assembled code stored is used.

```
/a 339d2
000339D2: 7470          jz    33a44
000339D4: 8D8551FFFFFF  lea   eax, (ebp-af)
000339DA: 50          push  eax
000339DB: 90          nop
000339DC: 90          nop
000339DD:             end
/
```

In the above example only the boldface text is entered. Patch supplied all of the other information and assembled the instructions as indicated. To terminate the entry of assembly language instructions, use the end pseudo-op or merely enter a blank line.

Use one or more spaces to separate the assembly language opcodes from the operand fields. Any valid Patch expression may be used in the operand.

■ Calculator

An expression calculator is available whenever Patch is waiting for a command. To use the calculator merely enter an expression. To insure that the expression is not interpreted as a command, make sure that it starts with a digit, unary operator or a question mark.

```
/?abcd
0x0000ABCD (hex), 43981 (dec)
/?2000-23
0x00001FDD (hex), 8157 (dec)
/?'abcd'
0x61626364 (hex), 1633837924 (dec)
/
```

The calculator always displays the result in both hexadecimal and decimal.

■ Checksum Command

The CHECK command computes the checksum for the entire file or for a region specified.

```
check
check checksum
check address-range
check address-range checksum
```

It either displays this checksum or compares it to the checksum specified.

```
/check
Checksum is F602
/check 1000 2000
Checksum is B75E
/check 1000 2000 abcd
Mismatch.
/
```

■ Code Command

Most programs have a code segment and a data segment. Initially, when Patch starts, it assumes that addresses requested and displayed are in the program's code segment. This assumption can be changed with the [Data Command](#). The CODE command returns to the code segment.

```
Code
```

■ Compare Command

The C command compares what is in the file at a specified location with what you specify should be in the file at that location.

C *address value-list*

For instance:

```
/d 1000 100f
001000: E8C70B07 006A0368 FB0F0000 E8BCC707 '°í...j.h....°.í.'
/c 1000 e8 c7 b
Match.
/c 1000 e8 c7 c
Mismatch.
/
```

■ Data Command

Most programs have a code segment and a data segment. Initially, when Patch starts, it assumes that addresses requested and displayed are in the program's code segment. This assumption can be changed with the DATA command. The [Code Command](#) returns to the code segment.

Data

To view or make any changes to the data portion of a program, you must use the video display mode.

■ Delete Command

This command is valid only when the file is a direct, indexed or keyed data file. It deletes the current record from the file.

De

Only the current record is deleted from the file. To get the current record use the [Key Command](#).

■ Display Command

This command displays data from the file, record or sector.

```
D address
D address-range
D
```

This command displays one or more lines of 16 bytes each, starting with *address*. If *address* or *address-range* is not specified, the next line following the last line is displayed.

When *address-range* is not specified, 16 lines of data are displayed.

```
/d 2000 200f
002000: 055B3BD8 7E0C8B45 F4488945 F8E95700 ' . [ ; « ~ . . E . H . E . ¾ W . '
/d 2000
002000: 055B3BD8 7E0C8B45 F4488945 F8E95700 ' . [ ; « ~ . . E . H . E . ¾ W . '
002010: 0000FF35 3C51FFFF 8B45F448 8D048500 ' . . . . 5 < Q . . . . E . H . . . . '
002020: 0000005B 8B04180F B6400450 8B45080F ' . . . . [ . . . . φ @ . P . E . . '
002030: B640055B 3BD87D09 8B45F440 8945FCEB ' φ @ . [ ; « } . . E . @ . E . β '
002040: 288B4508 4050FF35 3C51FFFF 8B45F448 ' ( . E . @ P . 5 < Q . . . . E . H '
002050: 8D048500 0000005B 8B04188B 40055B89 ' . . . . . . . [ . . . . @ . [ . '
002060: 038B45F4 E91F0000 00EB1683 7DF0007E ' . . E . ¾ . . . . β . . } . . ~ '
002070: 098B45F4 488945F8 EB078B45 F4408945 ' . . E . H . E . β . . E . @ . E '
002080: FCE9CCFE FFFF33C0 8BE55DC2 0400B804 ' . ¾ I . . . . 3 f . . ] , . . = . '
002090: 000000E8 34FB0600 8B450C8B 00FF308B ' . . . . ° 4 . . . . E . . . . O . '
0020A0: 45088B00 FF308B45 0C8B000F B6400450 ' E . . . . O . E . . . . φ @ . P '
0020B0: 8B45088B 000FB640 045B3BD8 7D0B8B45 ' . E . . . . φ @ . [ ; « } . . E '
0020C0: 0C8B000F B64004EB 098B4508 8B000FB6 ' . . . . φ @ . β . . E . . . . φ '
0020D0: 400450E8 538A0700 8945FC85 C074088B ' @ . P ° S . . . . E . . . f t . . '
0020E0: 45FCE93A 0000008B 450C8B00 0FB64004 ' E . ¾ : . . . . E . . . . φ @ . '
0020F0: 508B4508 8B000FB6 40045B93 2BC38945 ' P . E . . . . φ @ . [ . + † . E '
/
```

■ End Command

The E command saves the changes that have been made and exits Patch.

```
E
```

Note that when direct, indexed and keyed data files are being patched, the [Put Command](#) must be used to save the changes made to a record.

■ Fill Command

This command fills a block of the file with a single value.

F *address-range value*

Unlike the [Set Command](#) which can set a series of locations with a string of data, the F command sets the series of locations to a single value.

```
/d 2000 200f
002000: 055B3BD8 7E0C8B45 F4488945 F8E95700 '.;«~..E.H.E.¾W.'
/f 2000 2008 0
/d 2000 200f
002000: 00000000 00000000 F4488945 F8E95700 '.....H.E.¾W.'
/
```

■ Get Command

The G command reads and displays one sector of the disk. This command is only valid in Mode 2 of Patch (patching disk sectors).

G *sector*
G

If *sector* is omitted, the next sector of the disk is read and displayed.

■ Help Command

The H command displays a brief summary of all of the commands and the expression operators and elements.

H
F1

■ Key Command

The K command reads a record of a direct, indexed or keyed file. This command is valid only when patching those types of files.

K *key*
K

The *key* must match in type with the type of file. That is, for direct files the *key* must be a record number, but for indexed and keyed files the *key* must be an alphanumeric string. Indexed and keyed file *keys* may not contain the space character.

For direct files the record number specified is assumed to be a decimal number, even without the trailing “t” specifier.

■ Length Command

The LEN command displays the length of the file (stream files), the length of the file and allocated record size (direct, indexed and keyed files), or the length and type of program (program files). For program files it also allows you to change the size of the heap and stack space used by the program.

```

Len
Len HEAP size
Len STACK size

```

For instance:

```

>patch sample.stream (novideo
/len
Length = 2,031
/

>patch sample.direct (novideo
/len
Length = 280  Reclen = 28
/

>patch sample.indexed (novideo
/len
Length = 3,593,330  Keylen = 10  Reclen = 36
/

>patch sample.command
/len
Length = 92,824
      Code      = 0x00012834
      Data      = 0x00004098
      Stack     = 0x00002000
      Heap      = 0x0000C350
      Entry     = 0x00000140
      Type      = 32 bit Program
/len stack 3000
/len heap d000

```

A change to the heap or stack space is only saved when the [End Command](#) is used.

■ Move Command

The M command copies data from one location in the file, record or sector to another location.

M *from-address to-address length*

The *from-address*, *to-address*, *from-address+length* and the *to-address+length* must be within the bounds of the current file or record. This restriction does not apply when patching disk sectors.

This move operation is done as a byte-by-byte copy, not a copy and paste. Therefore, when the source and destination address ranges overlap, the result may be undesirable.

The following example wants to copy the first 32 characters of the file to location 0x10. The first attempt fails because the address ranges overlap.

```
/d 0 30
000000: 54686973 20697320 6A757374 20612074 'This is just a t'
000010: 65737420 66696C65 20746F20 62652075 'est file to be u'
000020: 73656420 62792074 68652050 41544348 'sed by the PATCH'
000030: 20636F6D 6D616E64 2E0D0D54 68697320 ' command...This '
/m 0 10 20
/d 0 30
000000: 54686973 20697320 6A757374 20612074 'This is just a t'
000010: 54686973 20697320 6A757374 20612074 'This is just a t'
000020: 54686973 20697320 6A757374 20612074 'This is just a t'
000030: 20636F6D 6D616E64 2E0D0D54 68697320 ' command...This '
/
```

To perform this type of operation properly the move must be done in two stages. First the overlapped region must be copied and then the remaining region is copied. If a larger region were copied, there might be several stages to avoid specifying an overlapped region.

```
/d 0 30
000000: 54686973 20697320 6A757374 20612074 'This is just a t'
000010: 65737420 66696C65 20746F20 62652075 'est file to be u'
000020: 73656420 62792074 68652050 41544348 'sed by the PATCH'
000030: 20636F6D 6D616E64 2E0D0D54 68697320 ' command...This '
/m 10 20 10
/m 0 10 10
/d 0 30
000000: 54686973 20697320 6A757374 20612074 'This is just a t'
000010: 54686973 20697320 6A757374 20612074 'This is just a t'
000020: 65737420 66696C65 20746F20 62652075 'est file to be u'
000030: 20636F6D 6D616E64 2E0D0D54 68697320 ' command...This '
/
```

This operation of the M command is not entirely undesirable because it can be advantageous to repetitively duplicate a region of the file or sector.

```
/d 0 30
000000: 54686973 20697320 6A757374 20612074 'This is just a t'
000010: 65737420 66696C65 20746F20 62652075 'est file to be u'
000020: 73656420 62792074 68652050 41544348 'sed by the PATCH'
000030: 20636F6D 6D616E64 2E0D0D54 68697320 ' command...This '
/m 0 5 45t
/d 0 30
000000: 54686973 20546869 73205468 69732054 'This This This T'
000010: 68697320 54686973 20546869 73205468 'his This This Th'
000020: 69732054 68697320 54686973 20546869 'is This This Thi'
000030: 73206F6D 6D616E64 2E0D0D54 68697320 's ommand...This '
```

■ Patch Level Command

The PL command displays and sets the current patch level for a program file.

```
PL patch-level
PL
```

Patch levels may only be assigned to program files. The PL command always displays the current patch level for the program.

```
/p1
Old patch level:
New patch level: 40001
/p1 40002
Old patch level: 40001
/
```

The *patch-level* must be a field using a format of @##### or #####. That is, a single letter followed by seven or fewer digits, or seven or fewer digits without the leading letter.

A program's patch level can also be set or changed with the [Change](#) command described on page 216 and it can be viewed with the [FileList](#) command described on page 326.

■ Put Command

The P command is the complement of the [Get Command](#) and [Key Command](#). P writes a record or sector back to disk.

```
P key
P sector
P
```

Use the P *key* when you are patching a direct, indexed or keyed file and you want to write the current record to the file with a different key. The *key* must match the file organization. That is, *key* must be numeric for direct files and alphanumeric for indexed and keyed files.

Use the P *sector* when you are patching disk sectors and you want to write the current sector to a different location on the disk.

To merely write the current record or sector back to the file or disk in the same place that it was read, use the P command with no argument.

■ Quit Command

The Q command exits the Patch command without updating the file or disk. Any unsaved changes are lost.

```
Q
```

Use the [End Command](#) or [Put Command](#) to save changes before quitting.

■ Replace Command

The R command changes the contents of consecutive locations to values specified.

```
R address value-list
```

The values in value-list are assigned to the locations *address*, *address*+1, *address*+2, *etc.* until the list is exhausted. The range of locations from *address* to *address* plus the number of items in *value-list* must be within the bounds of the file, record or sector.

```
/d 2000 200f
002000: 54686973 20697320 6A757374 20612074 'This is just a t'
/r 2000 15 23 0f 2d
/d 2000 200f
002000: 15230F2D 20697320 6A757374 20612074 ' .#.- is just a t'
/r 2000 'Now is the time '
/d 2000 200f
002000: 4E6F7720 69732074 68652074 696D6520 'Now is the time '
```

■ Search Command

The S command locates an occurrence of a specified series of values.

S *address value-list*

The file, record or sector is searched, starting at location *address*, for the next occurrence of the sequence of values indicated by *value-list*. The range of locations from *address* to *address* plus the number of items in *value-list* must be within the bounds of the file, record or sector.

If a match is found, its location is displayed and you are asked if you want to search for the next occurrence. Any response other than **[Y]** is treated as **[N]**.

```
/s 0 'This'
Match at 0x00000000, again? y
Match at 0x0000003B, again? y
/
```

■ Set Command

The S command allows you to set a series of locations, one location at a time.

S *address*

When S starts, it displays address and its contents in both hexadecimal and ASCII and then it allows you to change the value at that location.

```
/s 2000
0x00002000: 0F (.)
```

At this time you may enter a new value, terminate the S command or advance to the next or prior locations. Entry of **[Space]**, **[→]** or **[↓]** is interpreted as a request to advance to the next address. Entry of **[↑]** backs up to the prior address.

The location may be set to any expression value as described in “[Expressions](#)” on page 425. You may set it to a string of ASCII characters by enclosing the characters within a pair of single quotation marks. Terminate the entry of a value with **[Space]**, **[→]**, **[↓]** or **[↑]** and the locations are set to the values requested and the location pointer is advanced or backed up. Terminate entry with **[Enter ↵]** and the values are set and the S command terminates.

```

/s 2000 
0x00002000: 0F (.) 0 
0x00002001: 8B (.) 23 
0x00002002: 5D (.) 14 
0x00002003: 14 (.) 253t-48t 
0x00002004: 81 (.) 0 
0x00002005: FB (.) 'This is a test' 
0x00002013: 95 (.) 0 
/d 2000 2013
002000: 002314CD 00546869 73206973 20612074 '.#.İ.This is a t'
002010: 65737400 C083E001 741EA180 C0FFFFFF6 'est.f.™.t.L.f...'
/

```

■ Use Command

The USE command tells Patch to use either 16-bit or 32-bit instructions when using the [Assemble Command](#).

```

Use 16
Use 32

```

This command will only be necessary when you are patching disk sectors (Mode 2). When patching a program file, Patch will know whether the program is a 16-bit program or a 32-bit program.

■ Full-Screen Video Mode Command

Entry of switches Patch from the command mode to the full-screen video mode or vice versa.

Patching Files How Patch operates depends upon the type of file being patched.

Stream Files or BINARY Option

Patching a stream file or using the BINARY option, causes Patch to start in its video display mode. The entire file may be viewed or modified because all addresses are valid from zero through the length of the file.

You must use the [End Command](#) (in command mode) or the or commands (in video display mode) to save any changes made to a stream file.

Direct, Indexed and Keyed Files

Patching a direct, indexed or keyed file starts Patch in the video display mode. You may only view and change one record at a time. The key to a record cannot be changed except by using the [Put Command](#) to write the

record with a different key and the [Delete Command](#) to delete the old record (perform a DE first and then a P with the new key).

Changes made to a record must be saved with the [Put Command](#) (in command mode) or the `[Save]` command (in video display mode).

Program Files

Patching a program file causes Patch to start in command mode with the CODE segment selected.

You must use the [End Command](#) (in command mode) or the `[File]` or `[Save]` commands (in video display mode) to save any changes made to a program file.

Patching Sectors

When Mode 2 of Patch is used you can view and change one sector at a time. Patch starts out in video display mode. After specifying the first sector number you may get the next sector of the disk with either the [Get Command](#) (in command mode) or the `[PageDown]` commands (in video display mode). In video display mode you may get the prior sector with the `[PageUp]` command.

You must use the [End Command](#) (in command mode) or the `[File]` or `[Save]` commands (in video display mode) to save any changes made to a sector before getting a different sector.

Restrictions

file must not be read or write protected.

The Patch command requires a privilege level of three.

Peek Command

The Peek command allows you to peek at another user's display terminal.

```
1 PEEK name
```

```
2 PEEK partition
```

name » account name

partition » partition number

Commands

Operation **Mode 1**—The first user that is logged onto *name* (other than yourself) is peeked at.

Mode 2—The user on partition number *partition* is peeked at.

Notes When peeking at another user, all characters that are being displayed on that user's console are also displayed on your console.

Users may notice a slight degradation in performance because every character displayed on their console has to be displayed twice. There may be a significant degradation if your console is slow then the other user's, for instance, when you are connected via a slow-speed modem.

The cursor may appear on your screen.

You should always inform the user before peeking at their console. In some countries it may be illegal to peek at a user's console without their permission.

Restrictions The Peek command requires a privilege level of four.

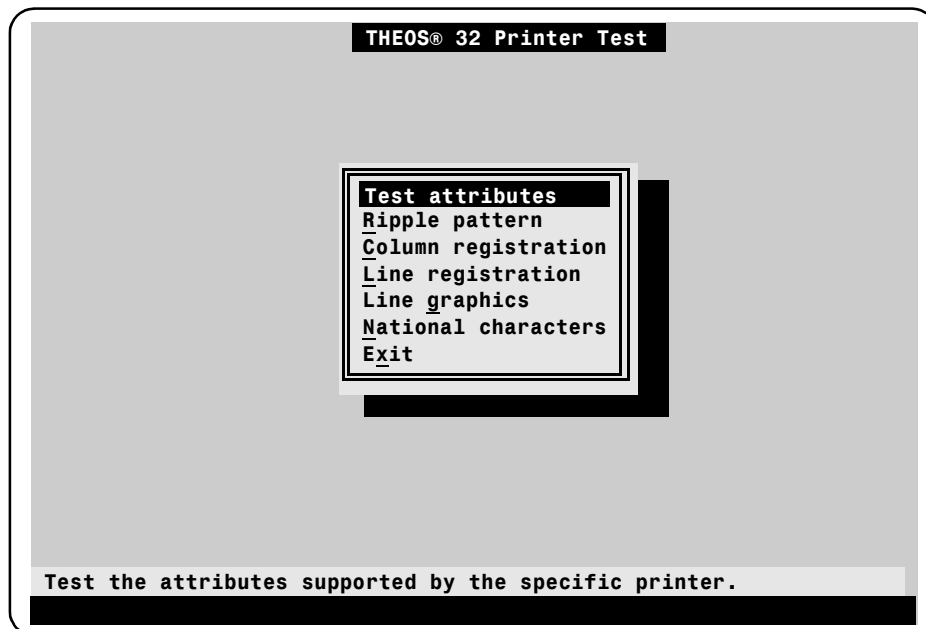
Printer Command

The Printer command is a complement to the [CRT](#) command. It tests the printer's display capabilities with the attached class code for the printer.

```
1  PRINTER
2  PRINTER  PRTnn

-----
PRTnn          »   Attached printer
```

Operation When Printer is invoked the “Printer Test Menu” is displayed.



Use the normal menu selection keys to select the desired tests. These keys are described in “[Using Menus](#)” on page 75.

Options**PRT*nn***

Indicates that Printer is to test the attached printer number *nn*. When this option is not used the first attached printer is tested.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

Tests:**Test Attributes**

Tests the printer attributes supported by THEOS including: boldface, underline, italics, second color or shading, compressed text, double-wide text and double-high text.

```

Class code: 135

Printer name: HP LaserJet II & III

Normal text (no attributes)

0x0E: Boldfaced text 0x0F

0x0B: Underline text 0x16

0x1D: Italics or alternate character set 0x1E

0x04: Second color or shading 0x05

0x02: Compressed text 0x03

0x17: Double-wide text 0x18

0x15: Double-high text 0x19

```

The display on your printer will, of course, be dependent upon your printer's capabilities.

Ripple Pattern

Displays a “ripple pattern” using the entire ASCII character set. This test can be used to check for dropped characters or improper column alignment.

```
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdef
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefg
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
```

Column Registration

Displays a columnar pattern.

```
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
! "#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
```

Line Registration

Displays lines of repetitive characters. This pattern can be used to determine if lines of text are printed straight on the printer.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#####
#####
#####
#####
```

Line Graphics

Displays all of the line-drawing graphics characters supported in the THEOS character set. See “[THEOS Character Sets](#)” on page 761. Also, a sample illustration of line graphics is printed.

┐ = ULC	┌ = URC	└ = LRC	┘ = LLC
┌ = LI	┐ = UI	└ = RI	┘ = DI
└ = FWI	─ = HORIZ	= VERT	
┐ = RULC	┌ = RURC	└ = RLRC	┘ = RLLC
┐ = DULC	┌ = DURC	└ = DLRC	┘ = DLLC
┌ = DLI	┐ = DUI	└ = DRI	┘ = DDI
└ = DFWI	─ = DHORIZ	= DVERT	

Commands

National Characters

Displays all of the international characters supported by the THEOS character set. See “[THEOS Character Sets](#)” on page 761.

Notes

The exact appearance of these tests depends upon the capabilities of your printer and the class code definition.

The “Ripple Pattern,” “Column Alignment” and “Line Registration” tests are used on character and line printers, not page printers like a laser printer.

See also [Attach](#), [ClassGen](#), [CRT](#), [Sysgen](#)

PutFile Command

The PutFile command copies a file from this system to a DOS-formatted hard disk partition or diskette.

1 **PUTFILE** *file* *DOS-file* (*options*

2 **PUTFILE** *drive* (**CLEAR**

drive » attached drive letter

DOS-file » DOS file name with optional path; may contain wild cards

file » file name with optional path; may contain wild cards

options » **BINARY** **QUERY**
 EOF **RONLY**
 HIDDEN **SUBDIR**
 NOQUERY **SYSTEM**

Operation

Mode 1—Copies the *file* from this system to a DOS formatted disk.

```
>put sample.testfile sample.txt:f
"SAMPLE.TEXTFILE:S" copied to "SAMPLE.TXT:F".
```

The *DOS-file* must be a valid DOS file description.

```
>putfile *.txt:s =.:f
Ok to copy "LETTER1.TXT:S" (Yes,No,All) Y
"LETTER1.TXT:S" copied to "LETTER1.TXT:F".
Ok to copy "LETTER3.TXT:S" (Yes,No,All) Y
"LETTER3.TXT:S" copied to "LETTER3.TXT:F".
Ok to copy "LETTER2.TXT:S" (Yes,No,All) A
"LETTER2.TXT:S" copied to "LETTER2.TXT:F".
"LETTER4.TXT:S" copied to "LETTER4.TXT:F".
"LETTER5.TXT:S" copied to "LETTER5.TXT:F".
"LETTER6.TXT:S" copied to "LETTER6.TXT:F".
```

The *DOS-file* specification may be just the drive code for the DOS disk. This is equivalent to specifying *=.:drive*. The destination name is the same as the source file name. Do not use this syntax when the THEOS file name is not a valid DOS file description. That is, do not copy library members or files whose file type is longer than three characters.

Mode 2—Clears the directory on the DOS formatted diskette.

```
>putfile f (clear
Enter disk label: DosXfer
```

Use this mode only when drive is a diskette drive. It is not designed to clear hard disks or removable hard disks.

Options

BINARY Tells PutFile that *file* may contain binary information and it is not to translate CR to CRLF. Whenever in doubt as to the content of the file, use this option.

EOF Appends a ^Z to the end of the file. This is the standard DOS end-of-file mark character.

HIDDEN Sets the “hidden” file attribute on the DOS disk for the files transferred.

NOQUERY Tells PutFile to not ask for confirmation before copying each file. This is a default option when wild cards are not used.

```
>putfile readme/*.txt:s f (noq
"/README/LANMAN.TXT:S" copied to "LANMAN.TXT:F".
"/README/VINES.TXT:S" copied to "VINES.TXT:F".
"/README/PCTCP.TXT:S" copied to "PCTCP.TXT:F".
"/README/IBMLAN.TXT:S" copied to "IBMLAN.TXT:F".
4 files copied.
```

To disable this option use the **QUERY** option.

QUERY Tells PutFile to “query” or ask if each file matching the file specifications is to be copied. This is a default option when wild cards are used.

```
>putfile readme/*.txt:s f
Ok to copy "/README/LANMAN.TXT:S" (Yes,No,All)
```

When the “Ok to copy” question is asked, you may respond with a **Y** for yes, **N** for no or **A** for all. Responding with **A** means yes to this file and all remaining files are copied without being queried. Respond with **Esc** to cancel the copy operation.

To disable this option use the **NOQUERY** option.

RDONLY Sets the “read only” file attribute on the DOS disk for the files transferred.

- SUBDIR** If the path specified in *DOS-file* does not exist on the DOS disk, this option tells PutFile to create the subdirectories that are missing.
- SYSTEM** Sets the “system” file attribute on the DOS disk for the files transferred.

Notes Unless the **BINARY** option is used, the THEOS end-of-record mark (CR) is translated to the DOS end-of-record mark (CRLF).

DOS Partitions and Disks The disk drive specified by *DOS-file* or *drive* may be an attached removable disk such as a floppy or removable hard disk, or it may be a partition on an attached hard disk drive. This disk or partition may be a DOS-formatted partition (16-bit FAT) or a Windows 95 disk or partition (32-bit FAT).

When it is a partition of an attached THEOS drive, the DOS partition is referenced by specifying the attached THEOS drive code, for instance S or C: for *drive* if the DOS partition is a partition of the same physical drive that is attached as the s drive in THEOS.

```
>putfile special.txt /windows/*.*:s
>putfile /dos/ansi.sys:s =.:s (bin
"SPECIAL.TXT:S" copied to "C:\SPECIAL.TXT".
```

Windows NTFS (NT File System) disks and partitions cannot be accessed with this command. Disks using Windows NT FAT can be used with this command.

Cautions THEOS direct, indexed and keyed files should not be transferred with this command. These file organizations are not usable by the DOS operating system. To transfer one of these types of files, first translate it into a normal stream file with the **FileType** and then copy the resulting file.

Restrictions The destination disk must be a DOS-formatted disk.

The *DOS-file* must be a valid DOS file description. DOS files have eight character file names and zero to three character file extensions.

When a path is specified for the *DOS-file*, the path must already exist on the DOS disk. The path will only be created if the **SUBDIR** option is used.

See also [GetFile](#), [FileType](#), [Send](#), [THEO+COM](#)

PWD Command

This command displays the current working directory.

PWD

Commands

Operation The current working directory is displayed on the standard output device.

```
>pwd
/LETTERS/PERSONAL:S

>show subdir
SUBDIR    = /LETTERS/PERSONAL:S
```

Notes PWD stands for print working directory.

See also [Account](#), [ChDir](#), [Logon](#), [Show](#)

Reboot Command

The Reboot command restarts and reboots the computer.

REBOOT

Operation

The computer is restarted just as if a **Ctrl+Alt+Del** were entered from the main console or the “Reset” button were pressed on the computer.

When this command is executed from the CSI command-line, you are asked to confirm this request.

```
>reboot
Ok to reboot the system (Y|N)?
```

If you respond with **N** or **Enter**, the command is exited without rebooting the system.

When a **Y** response is given, a disk cache sync is performed to make sure that the disk is current and then the computer is rebooted.

When this command is executed from an EXEC or another program, the confirmation request is not given and the computer is rebooted.

When used with a NetTerm connection to a remote system, you are warned with the message “YOU ARE ABOUT TO REBOOT A REMOTE SYSTEM.”

Note

A CACHE SYNC operation is performed before rebooting to maintain data integrity.

If history logging is enabled (see page 532), an entry is added reflecting that the system was rebooted.

Cautions

This is an extremely dangerous command because other users are terminated without notice. If another user is in the process of updating one or more files, those files will be inaccurate because the update was not completed.

Always do a [Show USERS](#) before using this command and verify that all other users are at a [Logon](#), CSI or stopped.

Restrictions

The Reboot command requires a privilege level of five.

See also

BOOT32, BOOT40

Receive Command EXEC

The Receive command is an EXEC language program giving you convenient, command-line access to the THEO+COM command's file receive capability.

Commands

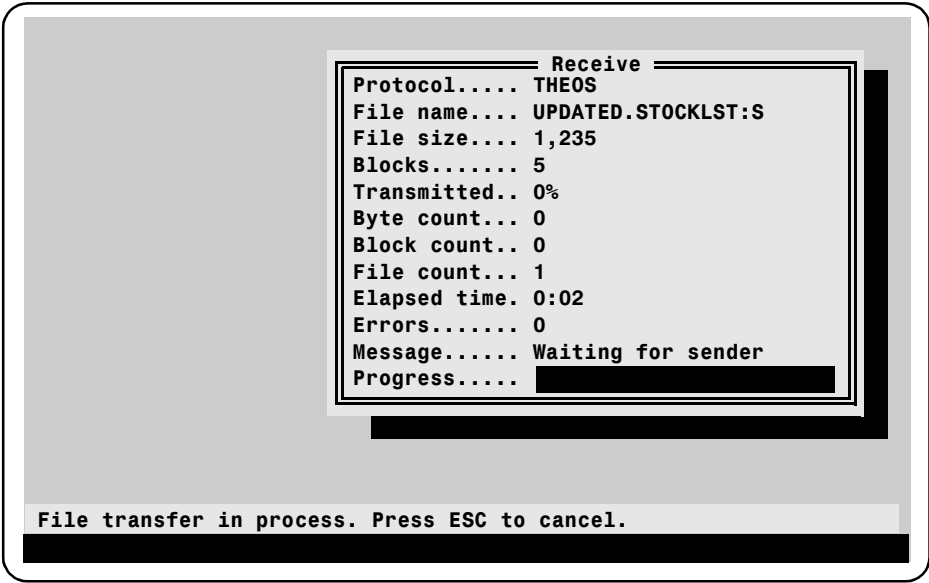
<u>RECEIVE</u> <i>file</i> (<i>options</i>					
<hr/>					
<i>file</i>	»	file name with optional path			
<i>options</i>	»	ASCII	TRACE	XMODEM	XMODEM-1K
		COM <i>nn</i>	TRACEFILE <i>fn</i>	XMODEM-CRC	YMODEM
		THEOS			

Operation Invokes the THEO+COM command in RECEIVE mode. The first attached COM device is used unless the COM*nn* option is specified. If no protocol option is specified, THEOS protocol is used.

If the connection to the other computer is via a modem, it is assumed that the telephone connection has already been established with the Dial command or by using THEO+COM directly.

```
>dial 1 800 123-4567
Dialing 1 800 123-4567

>receive updated.stocklst
```



Options	ASCII	Use the ASCII file transfer protocol. Essentially, this is no protocol and should be used only for short text files.
	<u>COMnn</u>	Use the currently attached COMnn for the communications port. When this option is not specified the first attached COM device is used.
	<u>THEOS</u>	Use the THEOS SEND/RECEIVE protocol. This is the default protocol.
	<u>TRACE</u>	Enables file transfer tracing. A window is opened in the upper-right corner of the display, showing the protocol activity during a transfer.
	<u>TRACEFILE</u> <i>fn</i>	Similar to the TRACE option except that the protocol activity is output to the file <i>fn</i> .
	<u>XMODEM</u>	Use XMODEM checksum, 256-byte protocol.
	<u>XMODEM-CRC</u>	Use XMODEM CRC-16, 256-byte protocol.
	<u>XMODEM-1K</u>	Use XMODEM-1K, CRC-16, 1024-byte protocol for single file transfers.
	<u>YMODEM</u>	Use YMODEM CRC-16, 1024-byte protocol. This protocol might be called YMODEM-BATCH in the other computer's communication program because it can receive multiple files.
Notes	Refer to the <i>THEO+COM Installation and User's Guide</i> manual for a full description of the operation of file transfers and the protocols used.	
Defaults	The first attached COM device is used by default and the THEOS protocol is the default.	
Restrictions	A COM device must be attached.	
See also	Dial , GetFile , Send , THEO+COM	

Reminder Command

This command maintains the `SYSTEM.REMINDER` file or your private `account.REMINDER` file.

REMINDER

Commands

The `SYSTEM.REMINDER` and the `account.REMINDER` files are files accessed when you log onto an account. These are keyed access files with a calendar date for the key. When you log onto an account, these files are searched for a record with today's date. If they are found, the message text for the date is displayed.

Operation A search is made for `account.REMINDER` file where *account* is the name of the account that you are currently logged onto. If the file is not found, then it is created.

The program prompts you for a reminder date:

```
>reminder
```

```
Enter date (MMDDYY):
```

You may enter a specific date such as 07/04/96, or a "generic date" such as 07/04 without the year. A specific date means that the message appears on that date only; a generic date means that the message appears every year on that month and day. See "Notes" on the next page for additional information about entering dates.

When a message already exists for the date entered, the message text is displayed and you can change it. Otherwise, you can enter the new message for the date.

```
>reminder
```

```
Enter date (MMDDYY): 07/19/96
```

```
Enter message text:
```

```
This message displays every time that I log onto this account.
```

```
Enter date (MMDDYY):
```

The message text may be as long as 256 characters, but it cannot contain any new-line characters or carriage returns. Entry of **Enter ↵** terminates the message text.

To delete an existing message, replace or add the word “DELETE” to the beginning of the message text. When the first six characters of a message are “delete” (uppercase or lowercase), the message is deleted from the file.

To exit from the Reminder command, respond with **Enter**, **Esc** or **F9** for the date.

Notes

If you are logged onto the system account, the SYSTEM.REMINDER file is always the file maintained.

The date format requested is dependent upon the current DATEFORM (see “[Set](#)” on page 474 and “[Sysgen](#)” on page 531).

	DATEFORM 1	DATEFORM 2	DATEFORM 3
Format:	MMDDYY	DDMMYY	YYMMDD
Examples:	10/15/96 101596 10/15 1015	15-10-96 151096 15-10 1510	96.10.15 961015 15.10 1510

The delimiters between the date elements can be any non-numeric character.

See also

[Logon](#)

Rename Command

Rename changes the name of an existing file, library or directory.

1 **RENAME** *from-file to-file* (*options*

2 **RENAME** *file* (**FILES** *options*

3 **RENAME** (*options*

<i>file</i>	»	file name with optional path
<i>from-file</i>	»	file name with optional path; may contain wild cards
<i>to-file</i>	»	file name with optional path; may contain wild cards
<i>options</i>	»	NOQUERY NOTYPE QUERY TYPE

Operation **Mode 1**—Changes the name of *from-file* to *to-file*.

```
>rename this.file that.file
"THIS.FILE:S" renamed "THAT.FILE:S".
1 file renamed.
```

The *from-file* and *to-file* may contain wild-card specifications. See “[Wild Card Specifications](#)” on page 136.

Mode 2—*file* is an ASCII stream file containing two file descriptions per line. The first file description in the line is treated as a *from-file* and the second file description is the *to-file*. For each line in *file*, a Mode 1 RENAME is performed.

This mode of the Rename command is convenient when one or more sets of files are repetitively renamed. Merely edit a file containing file description pairs, such as:

```
>edit daily.files
customer.master:s /prior/customer.master:s
customer.history:s /prior/customer.history:s
general.ledger.*:s /prior/=. .=:s
check.*:s /prior/=. =
>rename daily.files (file noquery notype
```

Mode 3—This is the interactive mode of the **Rename** command. Since no files are specified on the command line, you are prompted to enter the file descriptions to rename.

```
>rename (noquery
Enter file name list, terminate with empty line.
?MENU.BASIC
Destination file name missing.
?MENU.BASIC PROGRAM.BASIC.=
"MENU.BASIC:S" renamed "PROGRAM.BASIC.MENU:S".
?SAMPLE.FILE* /COPIED/SAMPLES/=
"SAMPLE.FILE1:S" renamed "/COPIES/SAMPLES/SAMPLE.FILE1:S".
"SAMPLE.FILE2:S" renamed "/COPIES/SAMPLES/SAMPLE.FILE2:S".
?
```

Options

NOQUERY Indicates that you do not want to be asked for confirmation before renaming each file. This is a default option when wild cards are not used.

```
>rename gl.* general.ledger.= (noq
"GL.MASTER:S" renamed "GENERAL.LEDGER.MASTER:S".
"GL.JOURNAL:S" renamed "GENERAL.LEDGER.JOURNAL:S".
"GL.HISTORY:S" renamed "GENERAL.LEDGER.HISTORY:S".
3 files renamed.
```

To disable this option use the **QUERY** option.

NOTYPE Tells **Rename** to not display the results of each file renamed on the standard output device. The general result message (the “nn files renamed.” message displayed before exiting **Rename**) is also suppressed with this option.

```
>rename gl.* gltest.= (not
Ok to rename "GL.MASTER:S" (Yes,No,All) Y
Ok to rename "GL.JOURNAL:S" (Yes,No,All) Y
Ok to rename "GL.HISTORY:S" (Yes,No,All) Y
```

To disable this option use the **TYPE** option.

QUERY Tells **Rename** to “query” or ask if each file matching the file specifications is to be renamed. This is a default option when wild cards are used.

```
>rename *.data =.testdata
Ok to rename "CUSTOMER.DATA:S" (Yes,No,All)
```

When the “Ok to rename” question is asked, you may respond with a **[Y]** for yes, **[N]** for no or **[A]** for all. Responding with **[A]** means yes to this file and all remaining files are renamed

without being queried. Respond with **Esc** to cancel the Rename operation.

To disable this option use the [NOQUERY](#) option.

TYPE A default option that tells Rename to display the results of each file erased on the standard output device. This display can be redirected.

```
>rename program.source.* =.basic (noquery
"PROGRAM.SOURCE.CUST:S" renamed "CUST.BASIC:S".
"PROGRAM.SOURCE.LEDGER:S" renamed "LEDGER.BASIC:S".
"PROGRAM.SOURCE.MENU:S" renamed "MENU.BASIC:S".
"PROGRAM.SOURCE.REPORTS:S" renamed "REPORTS.BASIC:S".
4 files renamed.
```

To disable this option use the [NOTYPE](#) option.

Defaults [TYPE](#) and [QUERY](#) are default options.

Restrictions You cannot rename a file that is erase, read or write protected.

You cannot rename a file to another drive. Use [CopyFile](#) for that.

You cannot rename a file to a file description of an existing file.

By default, only files owned by the current account are renamed. To rename a file owned by another account, you must specify the owning account name as part of the path:

```
>rename private\his.file my.file
```

This command renames the file HIS.FILE owned by the account named PRIVATE to your account, current working directory, with the name MY.FILE.

By default, the *to-file* is in your account in the current working directory but you may specify a path to the account and directory that you want to rename it to.

```
>rename my.file account\textfile/samples/new.file
"/MY.FILE:S" renamed to "ACCOUNT\TEXTFILE/SAMPLES/
NEW.FILE:S".
```

When the destination file specification includes a path, that path must exist. Rename does not create subdirectories.

See also [Change](#), [CopyFile](#)

Repeat Command EXEC

The Repeat EXEC executes a command several times.

REPEAT *count command-line*

command-line » any valid THEOS command

count » number of times to execute command-line

Commands

Operation

This EXEC program merely repeats the execution of some command one or more times.

```
>repeat 3 copyfile one.file to.another (append
```

```
Repeat # 1 of 3
```

```
>COPYFILE ONE.FILE TO.ANOTHER (APPEND
"ONE.FILE:S" appended to "TO.ANOTHER:S".
One file copied.
```

```
Repeat # 2 of 3
```

```
>COPYFILE ONE.FILE TO.ANOTHER (APPEND
"ONE.FILE:S" appended to "TO.ANOTHER:S".
One file copied.
```

```
Repeat # 3 of 3
```

```
>COPYFILE ONE.FILE TO.ANOTHER (APPEND
"ONE.FILE:S" appended to "TO.ANOTHER:S".
One file copied.
```

```
>
```

Restore Command

The Restore command retrieves the “archive copy” of a file, set of files or an entire disk volume.

- 1 RESTORE *from-drive to-drive* (*options*
- 2 RESTORE *file to-drive* (*options*
- 3 RESTORE *from-drive* (*SHOW options*
- 4 RESTORE *from-drive* (*VERIFY*

<i>aaa</i>	»	account name		
<i>d</i>	»	drive code or disk volume name		
<i>lll</i>	»	disk volume name of first disk in archive		
<i>file</i>	»	file name with optional path; may contain wild cards		
<i>from-drive</i>	»	drive letter of archive source or logical tape name		
<i>name</i>	»	archived file name		
<i>nnn</i>	»	new directory size		
<i>to-drive</i>	»	drive letter of destination disk		
<i>options</i>	»	ACCOUNT	LABEL <i>lll</i>	NOSYSFILE
		ALL	MULTIUSE	NOTYPE
		ASK	NAME <i>name</i>	OLDER
		CLEAR	NEWFILE	OLDFILE
		DRIVE <i>d</i>	NOASK	PRT <i>nn</i>
		FROM <i>aaa</i>	NOQUERY	QUERY
				REPLACE
				REWIND
				SIZE <i>nnn</i>
				SUBDIR
				TYPE
				VOLUME

The Restore command only restores files from an **archive volume** created with the ARCHIVE command. The archive volume contains special, compressed copies of files. See “[Archive](#)” on page 168.

Operation

Mode 1—Restores all of the files from the archive volume in *from-drive* to the disk in *to-drive*.

```
>restore tape s
```

Mode 2—Restores *file* from the archive volume to the *to-drive* in the current account. Unless one or more options are used to indicate otherwise, the *file* in the archive volume must be owned by the current account name.

```
>logon private
```

```
>restore some.file:f s (noask
```

```
Searching for account "PRIVATE".
Searching for file "SOME.FILE".
Restoring "SOME.FILE:S".
```

```
>logon develop
```

```
>restore those.files.*:f s (from private
```

```
Source is Disk F
Destination is Disk S
Mount volumes now:
```

```
Source is labeled "ArchiveD".
Archive from disk "THEOS" on 10/15/96, at 08:36.
Destination is labeled "THEOS".
OK to start restore (Y/N) Y
```

```
Searching for account "PRIVATE".
Searching for file "THOSE.FILES".
Restoring "THOSE.FILES:S".
Restoring "THOSE.FILES.FILE1:S".
Restoring "THOSE.FILES.FILE2:S".
Restoring "THOSE.FILES.FILE3:S".
Restoring "THOSE.FILES.FILE4:S".
```

In this mode files are always restored to the current account.

Mode 3—Displays a file listing of the files in the archive volume in *from-drive*.

```
>restore f (show
```

```
Account: 10\PRIVATE                                28 Aug 1996 9:01am Page 1
```

Filename	Filetype	Member	Dr	Date	Time	Org	Protect	Size	Recl	Key1
SOME	FILE			01/26/95	12:46	S	..WR....	1335		
THOSE	FILES			01/20/95	13:34	L	1280	20	
		FILE1		03/19/90	17:09	S	..W....	1284		
		FILE2		09/00/90	11:28	S	..W....	2257		
		FILE3		10/22/48	11:34	S	..W....	31		
		FILE4		12/01/87	05:00	S	..W....	9895		

The SHOW option does not have to be specified.

```
>restore f
```

```
Account: 10\PRIVATE                                28 Aug 1996 9:01am Page 1

Filename Filetype Member Dr Date Time Org Protect Size Rec1 Key1
SOME     FILE           01/26/95 12:46 S ..WR.... 1335
THOSE    FILES           01/20/95 13:34 L ..... 1280 20
          FILE1         03/19/90 17:09 S ..W.... 1284
          FILE2         09/00/90 11:28 S ..W.... 2257
          FILE3         10/22/48 11:34 S ..W.... 31
          FILE4         12/01/87 05:00 S ..W.... 9895
```

Mode 4—Verifies the integrity and readability of the archive volume in *from-drive*. It also displays a listing of the files in the archive volume similar to the display output by the [Archive](#).

This mode does not compare the archive volume to its original source disk.

Options

ACCOUNT Tells Restore to only restore those files from the archive volume that are owned by one account. This is a default option when [Mode 2](#) is used.

If the [FROM account](#) option is not specified, an implied FROM *current-account* is used. That is, only those files owned by the account that you are currently logged onto are restored.

```
>restore tape s (account noask

Searching for account "SYSTEM".
Restoring "SYSTEM.B3220LIB:S".
Restoring "SYSTEM.B3220LIB.ACCESS:S".
...
```

The ACCOUNT option is the opposite of the [VOLUME](#) option. ACCOUNT is the default option when [Mode 2](#) is used; VOLUME is the default option when [Mode 1](#) is used.

ALL Restores files from the archive volume even if the file already exists on the *to-drive* and even if the file has erase protection set. Also see the options [NEWFILE](#), [OLDFILE](#) and [REPLACE](#).

ASK

This is a default option that instructs Restore to ask the operator to mount the source and destination volumes and waits for confirmation that the proper volumes are mounted.

```
>restore tape s
```

```
Source is TAPE1
Destination is Disk S
Mount volumes now:
```

```
Source is labeled "Archived".
Archive from disk "THEOS" on 10/15/96, at 08:36.
Destination is labeled "THEOS".
OK to start restore (Y/N)
```

The first question, “Mount volumes now,” must be answered with an . All responses other than or are ignored.

The second question, “OK to start restore (Y/N),” may be answered with any response. All responses other than are treated as a response.

CLEAR

Tells Restore that, before restoring the first file, the directory of *to-drive* is to be cleared. A current directory size is used unless the SIZE option is also specified.

This option may only be used when option **VOLUME** is in effect with a **Mode 1** Restore.

When restoring to the S drive with this option you must be in single-user mode...the **MULTIUSER** option is ignored. Also, the **NOASK** option is ignored. After the restore you are prompted to reboot the system.

DRIVE *d*

Used with a multiple disk archive volume to specify that you want to restore only those files that were archived from drive *d*. *d* may be specified as a drive code or a volume name.

```
>archive s a b tape (noask notype
```

```
>restore tape c (drive a
```

This Restore command restores the files to the C drive that were archived from the A drive.

FROM *account* Tells Restore to only select those files on the archive volume that were owned by account name *account* at the time the archive was created.

See the second [Mode 2](#) example.

LABEL *label* Tells Restore that the multiple disk/tape archive volume uses disk/tape labels of *label*, with each disk/tape of the set incrementing the last character of *label*. For instance, disk one is labeled “Mon-1,” disk two is labeled “Mon-2” and so on.

This *label* is used in the prompt messages only and is not related to the disk label written when a disk is formatted.

MULTIUSER Allows Restore to restore to a public drive even though other users may be logged on and active. Normally, when Restore is instructed to perform a full volume restore (option [VOLUME](#)) on a public disk, it requires single-user mode. If other users are logged onto the system, it displays the message: “Must be single-user or private volume.”

Using this option tells Restore to not restrict the restore to single-user operation (the message is still displayed). **THIS CAN BE EXTREMELY DANGEROUS!** If another user changes some files while the restore is being done, the integrity of the files restored may be lost. Use this option only if you are sure that all other users are inactive.

NAME *name* Specifies the name of the archive volume set. When the file-type is not specified in *name* the default file-type of ARCHIVE is used.

When this option is not used the name of the archive volume must be ARCHIVE.VOLUME01.

NEWFILE Specifies that Restore will only attempt to restore a file if it does not already exist on *to-drive*.

NOASK Disables the source and destination volume operator confirmation at the beginning of the restore and when subsequent disks or tapes are needed.

NOQUERY An option that tells Restore to not ask for confirmation on each file being restored. In addition, this option suppresses the query when a file exists and the [REPLACE](#) option is not specified, as well as the query when a file exists, is protected and the [ALL](#) option is not specified.

With NOQUERY in effect the following questions are never asked:

```
Ok to restore "XXX" (Yes/No/All)
File "XXX" exists, ok to restore (Yes,No,All)
File "XXX" protected, ok to restore (Yes,No,All)
```

NOSYSFILES Do not restore operating system files. See “**NOSYSFILES**” on page 466.

NOTYPE Tells Restore to not display account names, subdirectory names, library names or file names on the standard output device as they are being restored.

OLDER Only restore files if the file does not exist on the destination or if the existing file on the destination is older than the archived file.

OLDFILE Specifies that Restore will only attempt to restore a file if it does exist on *to-drive*. This option implies the **REPLACE** option.

PRT*nn* Indicates that Restore is to print the display of account names, subdirectory names, library names and file names on the attached printer number *nn*.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. PRINTER1 may be specified as P.

QUERY Tells Restore that the operator is to be “queried” or asked if each file matching the selection criteria is to be restored.

```
>restore f s (noask query notype
```

```
Ok to restore "SAMPLE.FILE:S" (Yes/No/All) N
Ok to restore "SELECTED.EXEC:S" (Yes/No/All) N
Ok to restore "SYSTEM.CMD32.ACCOUNT" (Yes/No/All) Y
Ok to restore "SYSTEM.CMD32.ARCHIVE" (Yes/No/All) A
```

When the “Ok to restore” question is asked, you may respond with a **[Y]** for yes, **[N]** or **[Enter ↵]** for no or **[A]** for all. Responding with **[A]** means yes to this file and all remaining files are included without being queried.

Respond with **[Esc]** to cancel the restore (files already restored remain restored).

REPLACE This option tells Restore that it is okay to attempt to restore a file even if it already exists on the *to-drive*. When this option is not used (and the **NOQUERY** option is not used), an attempt to restore an existing file causes you to be queried:

```
File "XXX" exists, ok to restore (Yes,No,All)
```

Valid responses are identical to the **QUERY** option responses.

REWIND When source is tape, rewind to start of tape before beginning the restore operation. This is a default option.

SIZE *nnnn* Used in conjunction with the **CLEAR** option. This option tells Restore what size to make the newly cleared directory on the *to-drive*.

SUBDIR Tells Restore to restore the files into the current working directory. When this option is not used, files are restored to the *to-drive*'s root directory.

This option is normally only used when restoring files that were archived from a root directory. When restoring a file that was in a subdirectory, it is restored to that same directory but subordinate to the current working directory.

For instance: /SUBDIR/SOME.FILE:S is archived and then you CHDIR to the SUBDIR directory and perform a restore with the SUBDIR option.

The file is restored to /SUBDIR/SUBDIR/SOME.FILE:S.

TYPE A default option that tells Restore to display each account name, subdirectory name, library name and file name on the standard output device (normally the console) as it is being restored. This display can be redirected.

The display with this option differs between an **ACCOUNT** restore and a **VOLUME** restore. A **VOLUME** restore displays like the **Archive** display, showing the account names, subdirectory names, library names and file names with indentation to indicate the hierarchy of the account and directory structure.

For instance, the following is a typical display during a full volume restore:

```
ACCOUNT: 2=SAMPLES
  File: READ.ME
  File: SAMPLES.EXEC
  Subdirectory: C32
    Library: C32.CMD32
      Member: C32.CMD32.FINS
      Member: C32.CMD32.PRTF
  ...
```

An **ACCOUNT** restore displays a simple message for each file restored:

```
Searching for account "SAMPLES".
Restoring "READ.ME:S".
Restoring "SAMPLES.EXEC:S".
Restoring "/C32/C32.CMD32:S".
Restoring "/C32/C32.CMD32.FINS:S".
Restoring "/C32/C32.CMD32.PRTF:S".
...
```

When a qualifying file cannot be restored for some reason, the **TYPE** option displays an appropriate message:

```
File "XXX" not restored because file exists.
File "XXX" not restored because disk full.
File "XXX" not restored because directory full.
```

To disable this option use the **NOTYPE** option.

VOLUME Restores the entire archive volume to the *to-drive*. This is the default option with **Mode 1** and can only be used with **Mode 1**.

Defaults **ASK** and **TYPE** are default options. **VOLUME** is a default option with **Mode 1**, **ACCOUNT** is a default option with **Mode 2**.

Restrictions The Restore command requires a privilege level of four.

Individual files on a multivolume archive can only be restored with the **Mode 2** form of this command if the files are on the first volume of the archive set. When the files are on secondary volumes of the set, use the **Mode 1** form of the command with the **QUERY** option.

NOSYSFILES When the [NOSYSFILES](#) option is specified, the following sets of files are skipped if found on the archive volume.

Command files:	System files:
/SYS.CMD386.*	/SYS.TEOS386.*
/SYS.CMD386#.*	/SYS.TEOS386#.*
/SYSTEM.CMD286.*	/SYSTEM.ACCOUNT
/SYSTEM.CMD386.*	/SYSTEM.MAILBOX
/SYSTEM.CMD386#.*	/SYSTEM.SPOOLER.*
/SYSTEM.CMD32.*	/SYSTEM.TEOS386.*
/SYSTEM.CMD32#.*	/SYSTEM.TEOS386#.*
	/SYSTEM.TEOS32.*
Help files:	/SYSTEM.TEOS32#.*
/SYS.HELP386.*	/THEOS286.COPYLIB.*
/SYS.HELP386#.*	/THEOS386.COPYLIB.*
/SYSTEM.HELP286.*	
/SYSTEM.HELP386.*	Language files:
/SYSTEM.HELP386#.*	/SYSTEM.B286LIB.*
/SYSTEM.HELP32.*	/SYSTEM.B32CTLIB.*
/SYSTEM.HELP32#.*	/SYSTEM.B3220LIB.*
	/SYSTEM.B386LIB.*
Menu files:	/SYSTEM.C286LIB.*
/SYS.MENU386.*	/SYSTEM.C386LIB.*
/SYS.MENU386#.*	/SYSTEM.C32LIB.*
/SYSTEM.MENU286.*	
/SYSTEM.MENU386.*	
/SYSTEM.MENU386#.*	
/SYSTEM.MENU32.*	
/SYSTEM.MENU32#.*	

Table 27: NOSYSFILES Files

See also [Archive](#), [Backup](#), [CopyFile](#), [Disk](#), [Tape](#), [TBackup](#)

Rmdir Command

The Rmdir or Remove Directory command, erases a subdirectory and all of its files.

RMDIR *directory...* (*options*)

directory » subdirectory name; may contain path; may contain wild cards

options » NOQUERY
 NOTYPE
 QUERY
 TYPE

Command synonym: **RD**

Commands

Operation The *directory* or *directories* specified are erased.

```
>rmdir subdir1 subdir2
"/SUBDIR1:S" erased.
"/SUBDIR2:S" erased.
```

If a directory contains files or subordinate directories, you are asked to confirm that you want to remove the directory and all of its subordinate files and directories.

```
>rd subdir1 (notype
Ok to completely remove directory "/SUBDIR1:S" [Y|N]? Y
```

Options **NOQUERY** Indicates that you do not want to be asked for confirmation before erasing a subdirectory containing files.

```
>rd subdir1 (noquery
"/SUBDIR1/TEST1.FILE:S" erased.
"/SUBDIR1/TEST2.FILE:S" erased.
"/SUBDIR1/TEST3.FILE:S" erased.
"/SUBDIR1/SUBDIR11/TEST1.FILE:S" erased.
"/SUBDIR1/SUBDIR11/TEST2.FILE:S" erased.
"/SUBDIR1/SUBDIR11/TEST3.FILE:S" erased.
"/SUBDIR1/SUBDIR11:S" erased.
"/SUBDIR1/SUBDIR12:S" erased.
"/SUBDIR1:S" erased.
```

Subdirectories that do not contain files are not queried, even without this option.

NOTYPE Tells RmDir to not display the results of each file or directory erased on the standard output device.

```
>tree
/
├── subdir1
│   ├── subdir11
│   └── subdir12
└── subdir2

>rd subdir1 (not
Ok to completely remove directory "/SUBDIR1:S" [Y|N]?Y

>tree
/
└── subdir2
```

QUERY Tells RmDir to “query” or ask if each directory matching the file specifications is to be removed. This is a default option when wild cards are used.

```
>rd * (query notype
Ok to completely remove directory "/SUBDIR1:S" [Y|N]?Y
Ok to completely remove directory "/SUBDIR2:S" [Y|N]?Y
```

When the “Ok to remove” question is asked, you may respond with a **[Y]** for yes, **[N]** for no or **[A]** for all. Responding with **[A]** means yes to this directory and all remaining directories are removed without being queried. Respond with **[Esc]** to cancel the remove operation.

TYPE A default option that tells RmDir to display the results of each file and directory erased on the standard output device. This display can be redirected.

```
>rmdir *
"/SUBDIR1:S" erased.
"/SUBDIR2:S" erased.
```

To disable this option use the [NOTYPE](#) option.

Notes The command name RD is a synonym to the RmDir command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file. If standard synonyms are disabled (see “[Account](#)” on page [156](#) and “[STDSYN](#)” on page [101](#)), this synonym name may not be allowed.

Defaults [TYPE](#) is a default option.

See also [Erase](#)

See Command Filter

The See command copies a file to the standard output device, making all nonprintable characters visible.

1 SEE *file...*

2 SEE

file » file name with optional path; may contain wild cards

Operation

Mode 1—Each *file* in the list of files is copied to the standard output device. Each nonprintable character in *file* is displayed with two or three displayable characters:

Nonprintable character values less than the space character (32) display with a leading up-caret (^) followed by the control character. For instance, a tab character displays as ^I.

Nonprintable character values greater than 127 display with a leading tilde (~) followed by the display for the character value minus 128. For instance, the character value 137 displays as ~^I and the value 159 displays as ~^_.

The DEL character (value 127) displays as ^?. The end-of-line character (carriage return) and NULL characters (0) display as a dollar sign (\$) unless the *file* specification is preceded with a minus sign character.

```
>see system.cmd32.repeat
system.cmd32.repeat:s:
; get type of first argument$
&t = &typ &1$
$
; validate count and command$
&if (&index < 2) | (&t <> N)$
^I&control off$
^Ihelp repeat$
...
```

Multiple files can be specified on the command line:

```
>see file1 file2 file3
```

Each file is displayed on the standard output device. The minus sign specification can be used to indicate that the end-of-line character is not displayed as a dollar sign:

```
>see - system.cmd32.repeat
system.cmd32.repeat:s:
; get type of first argument
&t = &typ &1

; validate count and command
&if (&index < 2) | (&t <> N)
^I&control off
^Ihelp repeat
...
```

Multiple files can be specified, some without the minus sign specification and some with:

```
>see file1 file2 file3 - file4 file5
```

In the above command, the first three files are output with the dollar sign character displayed and the last two files are displayed without this character.

Mode 2—Copies the file from standard input to standard output, displaying the nonprintable characters as described above. This mode is normally used when the See command is part of a pipe.

```
>upcase system.cmd32.repeat | see
; GET TYPE OF FIRST ARGUMENT$
&T = &TYP &1$
$
; VALIDATE COUNT AND COMMAND$
&IF (&INDEX < 2) | (&T <> N)$
^I&CONTROL OFF$
...
```

The minus sign specification can be used with this mode:

```
>upcase system.cmd32.repeat | see -
; GET TYPE OF FIRST ARGUMENT
&T = &TYP &1

; VALIDATE COUNT AND COMMAND
&IF (&INDEX < 2) | (&T <> N)
```

Restrictions *file* must be a stream file.

See also [List](#), [More](#)

Send Command EXEC

The `Send` command is an EXEC language program that gives you convenient, command-line access to the `THEO+COM` command's file send capability.

```
1  SEND  file ( options
```

```
2 SEND file ( FILES options
```

file » file name with optional path; may contain wild cards

<i>options</i>	»	ASCII	NOEOT	TRACEFILE <i>fn</i>	XMODEM-1K
		COM nn	THEOS	XMODEM	YMODEM
		EOT	TRACE	XMODEM-CRC	

Operation

Mode 1—Invokes the **THEO+COM** command in SEND mode. The first attached COM device is used unless the **COMnn** option is specified. If no protocol option is specified, the THEOS protocol is used.

If the connection to the other computer is via a modem, it is assumed that the telephone connection has already been established with the `Dial` command or by using `THEO+COM` directly.

```
>dial 1 800 123-4567
Dialing 1 800 123-4567
```

```
>send updated.stocklst
```

Mode 2—Similar to [Mode 1](#), except file is an ASCII stream file that contains one file description per line. The `SELECTED.FILES` and `SELECTED.EXEC` files created by [FileList](#) and the `FOUND.EXEC` created by [Look](#) can be used for this specification file (see “[The EXEC and FILES Options](#)” on page 338). You may also create the file with an editor or application program.

For instance, `FileList` is used to create a list of files to be sent:

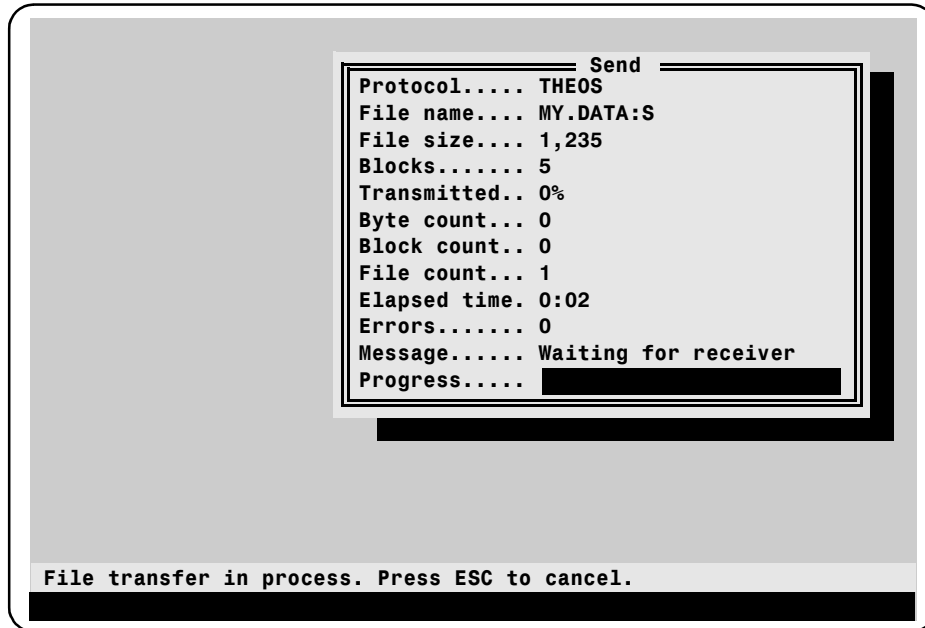
```
>filelist *.data:a (exec
```

```
>filelist a not *.data:a (10/1/95 exec append
```

A file now exists that lists all of the “data” files and all files that have been changed since 10/01/1995. The following command sends these files to another computer connected via a COM attachment:

```
>send selected.exec (file
```

The THEOS protocol must be used in this mode. The files are sent with **NOEOT** in effect for all files except the last file, which uses **EOT**.



Options

- ASCII** Use the ASCII file transfer protocol. Essentially, this is no protocol and should be used only for short text files.
- COMnn** Use the currently attached COMnn for the communications port. When this option is not specified, the first attached COM device is used.
- EOT** A default option that tells **THEO+COM** to send the end-of-transmission codes after the file is sent. To disable this option use the **NOEOT** option.
- NOEOT** Tells **THEO+COM** to not send the end-of-transmission codes after the file is sent. This option is used when several files will be sent.

```
>send this.file (theos noeot
```

```
>send that.file (theos eot
```

The above two commands send the two files to another computer. The other computer used the **Receive** command only once

because the first file used NOEOT and the transmission was not terminated.

- THEOS** Use the THEOS SEND/RECEIVE protocol. This is the default protocol.
- TRACE** Enables file transfer tracing. A window is opened in the upper-right corner of the display, showing the protocol activity during a transfer.
- TRACEFILE** *fn* Similar to the [TRACE](#) option except that the protocol activity is output to the file *fn*.
- XMODEM** Use XMODEM checksum, 256-byte protocol.
- XMODEM-CRC** Use XMODEM CRC-16, 256-byte protocol.
- XMODEM-1K** Use XMODEM-1K, CRC-16, 1024-byte protocol for single file transfers.
- YMODEM** Use YMODEM CRC-16, 1024-byte protocol. This protocol might be called YMODEM-BATCH in the other computer's communication program because it can receive multiple files.

- Notes** Refer to the *THEO+COM Installation and User's Guide* manual for a full description of the operation of file transfers and the protocols used.
- Defaults** The first attached COM device is used by default and the THEOS protocol is the default.
- Restrictions** A COM device must be attached.
- See also** [Dial](#), [PutFile](#), [Receive](#), [THEO+COM](#)

Set Command

The Set command sets and changes system and user-defined environment parameters.

1 **SET** *system-parameter value*

2 **SET** *user-variable=value*

<i>system-parameter</i>	»	ABBREV <i>on-off</i>	HISTORY <i>on-off</i>	QUIT <i>on-off</i>
		BREAK <i>c</i>	LIBRARY <i>lib</i>	RDYMSG <i>on-off</i>
		COPYLIB <i>lib</i>	LINKLIB <i>lib</i>	SEARCH <i>drive-seq</i>
		CWD <i>dir</i>	MSG <i>on-off</i>	SUBDIR <i>dir</i>
		DATE <i>date</i>	OBJLIB <i>lib</i>	SYNONYM <i>file</i>
		DATEFORM <i>n</i>	PATH <i>path</i>	TIME <i>time</i>
		DATEIN <i>n</i>	PRIORITY <i>n</i>	WORKVOL <i>drive</i>
		DATEOUT <i>n</i>	PROMPT <i>string</i>	
<i>user-variable</i>	»	user-defined environment variable		
<i>value</i>	»	value to assign to environment variable		

Operation

Mode 1—Changes a system-defined environment parameter. These parameters are normally set in the [Account](#) environment for each user.

```
>set date 10/26/2001
Date is set to Friday, October 26, 2001.
```

Although the syntax for [Mode 1](#) is similar to [Mode 2](#), a system-defined environment parameter cannot be set using the equal sign operator.

The Set command with no arguments performs a [Show](#) command.

```
>set
RDYMSG    = OFF
MSG        = ON
WORK       = M
SEARCH     = S
PROMPT     = !14\a!15\ s !4!!\!5>
DIALDIR    = SYSTEM\DIALDIR:S
HOME       = /:S
```

Mode 2—Changes a user-defined environment parameter. User-defined environment parameters may be any name that uses alphanumeric characters, does not contain a space character, and is not the same as a system-defined environment parameter.

```
>set filetype=basic
```

```
>set username="Abraham Lincoln"
```

The value of a user-defined environment parameter may be any string of characters. If the value contains characters that might be misinterpreted by the CSI, you should enclose the value in a pair of double quotation marks.

To clear a user-defined environment parameter, specify nothing after the equal sign.

```
>set username=Enter ↵
```

System Parameters

ABBREV *on-off* Sets the status of the command abbreviations switch to ON or OFF. When you log onto an account, the initial state of this switch is set by the account environment's "Abbreviation" field.

```
>set abbrev on
```

BREAK *c* Sets the key or character used to invoke Break-key sequences. The character can be specified with a character or the numeric value of the character.

```
>set break '+'
```

The first command sets the Break key to be the Plus key. To ensure that the character is accepted correctly, you may have to enclose it in a pair of single quote characters. After this command is executed, entry of **+**, **Q** will act just like entry of **Break**, **Q**.

```
>set break 27
```

The second command sets the Break key to be the Escape key. This setting is useful when using a terminal that does not have a **Break** key.

To disable this special Break key, use a setting of zero:

```
>set break 0
```

Even when the Break key is set to another key, the **Break** key is still effective.

COPYLIB *library* Defines the default copy library used by the language compilers and editors. *library* may be a complete path specification of the library including the drive code. The file-type of the library must be COPYLIB.

```
>set copylib /program.copylib:i
```

CWD *directory* This parameter is merely a synonym to the SUBDIR parameter and operates as if you had used the [ChDir](#) command to change the current working directory.

DATE *date* Changes the current system date. The *date* is specified using the currently set DATEFORM format.

Unlike the most other parameters, changing the DATE or TIME requires a privilege level of five and it affects all users on the system.

```
>set date 7/4/98
Date is set to Saturday, July 4, 1998
```

DATEFORM *n* Sets the format for date entry and display. The initial status of this parameter is set in the system environment by the [Sysgen](#) command, "Date Format" field.

DATEFORM:	1	2	3
Meaning:	American	European	International
Format:	mm/dd/yy	dd-mm-yy	yy.mm.dd
Example:	07/04/98	04-07-98	98.07.04

Table 28: DATEFORM Codes

Unlike the most other parameters, changing the DATEFORM requires a privilege level of five and it affects all users on the system.

DATEIN *n* Defines the algorithm used for interpreting two-digit year dates in MultiUser BASIC.

- 0** Two-digit year dates are interpreted as twentieth-century dates.
- 1** Two-digit year dates are interpreted as current-century dates.
- 2** Two-digit year dates are interpreted as twentieth-century or twenty-first century, depending upon how close that date is to the current system date.

Unlike the most other parameters, changing DATEIN or DATEOUT requires a privilege level of five and it affects all users on the system.

Refer to Appendix I: “[System Date and Time](#)” for additional information.

DATEOUT *n* Defines the algorithm used for outputting dates specified with two-digit years in MultiUser BASIC.

- 0** Twentieth-century dates are output with two-digit year numbers, other dates are output with four-digit year numbers.
- 1** Dates are always output with four-digit year numbers.
- 2** Dates are always output with two-digit year numbers.

Unlike the most other parameters, changing DATEIN or DATEOUT requires a privilege level of five and it affects all users on the system.

Refer to Appendix I: “[System Date and Time](#)” for additional information.

HISTORY *on-off* Sets the status of the command history save switch. With this field set to ON, a record is written to the SYSTEM.HISTORY file every time that the system is booted, a user logs on or off the system, and when a command starts or finishes. The initial value of the switch is defined by the system configuration file (see “[Sysgen](#)” on page [531](#)).

Unlike the most other parameters, changing HISTORY requires a privilege level of five and it affects all users on the system.

LIBRARY *library* Defines the default library. *library* may be a complete path specification of the library including the drive code.

```
>set library program.source
```

LINKLIB *library* Defines the default link library used by the language compilers. *library* may be a complete path specification of the library including the drive code. The file-type of the library must be LINKLIB.

```
>set linklib /program.linklib:i
```

MSG *on-off* Sets the status of the receive messages switch to ON or OFF. The initial state of this switch is set when you Logon to an account by the “Receive messages” field.

```
>set msg off
```

OBJLIB *library* Defines the default object library used by the language compilers. *library* may be a complete path specification of the library including the drive code. The file-type of the library must be OBJLIB.

```
>set objlib /program.objlib:i
```

PATH *path* Defines the default library or path used by the CSI and the operating system when trying to locate a program. *path* may be a complete path specification of the library including the drive code. The file-type of the library must be CMD32.

```
>set path user
```

Alternately, path may be a comma-separated or space-separated list of locations to find commands and EXECs.

```
>set path /cmds,private.commands,/special
```

The locations may be directory names or library names. For additional information, refer to “[Command Search Sequence](#)” on page 48.

PRIORITY *n* Sets the priority level for your partition. Priority level is a number from zero to seven.

Partitions are all started with a priority level of four.

When a partition or task has a higher priority than another task, it is allocated CPU time until it gives up its time due to a

“wait for interrupt” or a “wait for resource” condition. Only then will lower priority tasks be allocated time. Thus, a CPU-bound task with a high priority can have exclusive usage of the computer at the expense of all other users and tasks.

In general, do not change your priority level. Giving yourself a higher priority means that other users may not get any time.

This option requires a privilege level of three.

PROMPT *string* Sets the prompt string used by the CSI when it prompts you for a new command. The default CSI prompt used by THEOS is a single character: `>`.

For information about setting this parameter, refer to the [PROMPT](#) environment variable, described on page 107.

QUIT *on-off* Sets the status of the `[Break]`, `[Q]` enabled switch to ON or OFF. The initial state of this switch is set when you Logon to an account by the “Disable BRK-Q” field.

```
>set quit off
```

RDYMSG *on-off* Sets the status of the command “ready message” switch to ON or OFF. The initial state of this switch is set when you Logon to an account by the “Ready message” field.

```
>set rdymsg on
```

```
RC = 0, 13:27:13, ET = 0:00, CPU = 0.040
```

```
>
```

SEARCH *sequence* Sets the drive search sequence. *sequence* is a list of the drive letter codes to be searched when a file is requested without a drive specification. For additional information, refer to “[Command Search Sequence](#)” on page 48.

Do not specify a sequence that does not include the s drive.

SUBDIR *directory* This parameter changes the current working directory. It is equivalent to using the [ChDir](#) to set the current directory path.

SYNONYM *file* Defines a user-defined command synonym file. For a description of this file, refer to Appendix D: “[System Files](#)”, “[SYSTEM.THEOS32 Library](#)”, “[SYNONYM](#)” on page 734. Specify only the

file name, not the file type. User-defined command synonym files must have a file type of SYNONYM.

```
>set synonym private
```

An existing setting for a user-defined command synonym file can be disabled by using the CLEAR option:

```
>set synonym (clear
```

The system-defined command synonym file cannot be enabled or disabled with this command. It can only be set in the account environment with the [Account](#) command.

TIME *time* Changes the current system time.

Unlike the most other parameters, changing the DATE or TIME requires a privilege level of five because it affects all users on the system.

Since time can be set to the nearest second, you are prompted to press a key when you want the system time set to the specified *time*.

```
>set time 12:15:38
```

Type any character to set time to 12:15:38

WORKVOL *drive* Sets the drive used for temporary work files. The initial value is set when you Logon to an account by the “Work drive” field.

Notes

The Logoff command clears all user-defined and system-defined environments. The Logon command resets only those parameters that are defined in the [Account](#) environment.

The parameters DATE, DATEFORM, DATEIN, DATEOUT, HISTORY and TIME affect all users on the system. These parameters (except for DATE and TIME) are normally only set in the system configuration file ([Sysgen](#) command).

Defaults

The default or initial values for most of the environment variables are set during system boot using information from the [Sysgen](#) command or when an account is logged onto.

Restrictions

User-defined environment parameter values may not contain a double quotation mark. That character is ignored when assigning the value to the name.

Changing the DATE, DATEFORM, DATEIN, DATEOUT, HISTORY or TIME parameters requires a privilege level of five.

Changing the PRIORITY for your partition requires a privilege level of three.

See also

[Account](#), [ChDir](#), [Logon](#), [Mailbox](#), [Msg](#), [Show](#), [Sysgen](#)

Setup Command

The Setup command provides a single command to configure and initialize the major components of THEOS and various types of devices.

1	SETUP				
2	SETUP	function			
	function	»	ACCOUNT COLOR CRT	DIGIXE DISK	FLOPPY MAXSPEED SIO SYSGEN

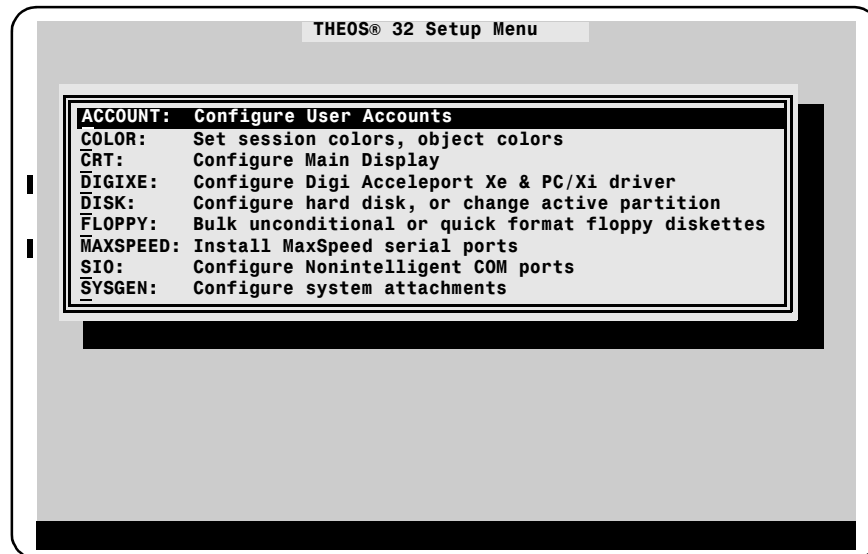
- Operation

Mode 1—Invokes Setup in its menu mode. See “[Setup Menu](#)” below.

Mode 2—Bypassing the [Setup Menu](#), the *function* screen is displayed. Refer to “[Functions](#)” on page 483.
- Setup Menu

When Setup is invoked with [Mode 1](#), the Setup Menu is displayed. This menu is dynamic because only those components installed on your system are presented in the menu. For instance, if you do not have the DigiBoard CX software installed on the system, the CX menu item is not offered.

There may be additional items displayed on the menu, depending upon any add-on products that you may have installed on your system.

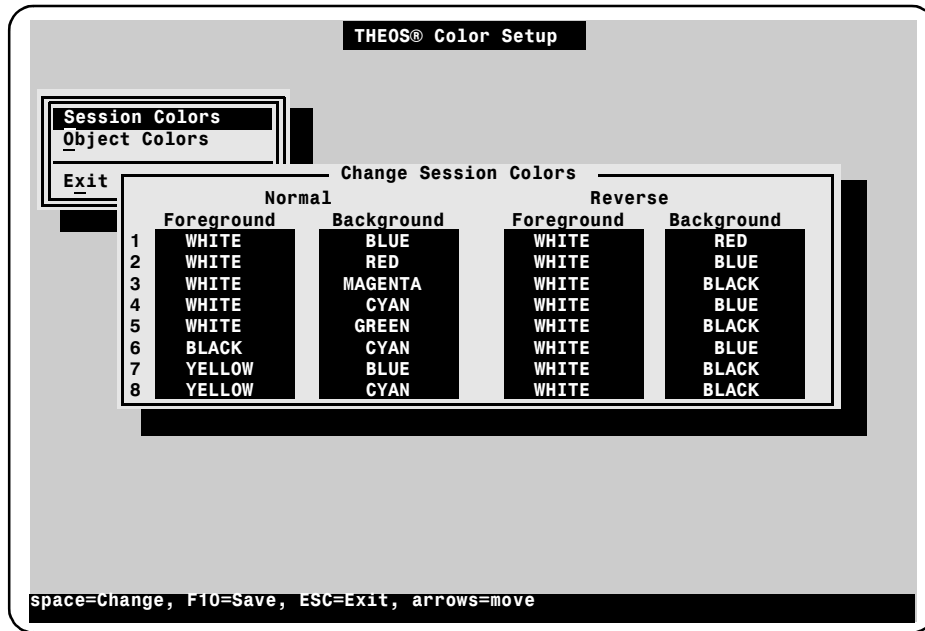


Use the normal menu selection keys to select the desired function. These keys are described in “Using Menus” on page 75.

Functions

- ACCOUNT** This function invokes the [Account](#) described on page 156.
- COLOR** Invokes the session Color Configuration screen described on page 484.
- CRT** Invokes the main console CRT Setup screen described on page 485.
- DIGIXE** Invokes the Digi Acceleport Xe and PC/Xi Configuration screen described on page 488.
- DISK** Invokes the [Disk](#) command hard disk formatter and partition setup procedures described on page 490.
- FLOPPY** Invokes the [Disk](#) command floppy diskette formatter procedures described on page 494.
- MAXSPEED** Invokes the MaxSpeed driver installation routine.
- SIO** Invokes the serial port configuration screen described on page 496.
- SYSGEN** This function invokes the [Sysgen](#) command described on page 531.

Setup COLOR This set of screens allow you to change the colors used for each of the sessions on your console and to define the colors used for various object windows.



The **Space** key cycles through each of the available eight colors: black, blue, green, cyan, red, magenta, yellow and white. The **Tab** or **Enter** keys advance to the next position.

Change Session Colors

Define the normal video text color and reverse video text color for each of the eight possible console sessions.

Object Colors

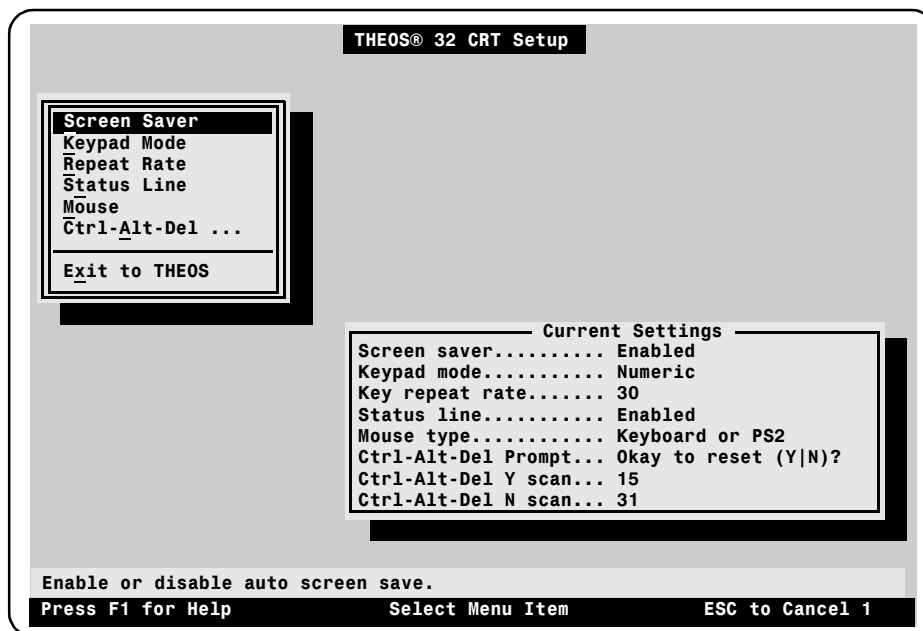
Define the text and background colors used for the various windows objects.

Change Object Colors		
	Foreground	Background
Menu	BLUE	WHITE
Help	BLACK	WHITE
Prompt	BLACK	WHITE
Backdrop	WHITE	BLUE
Dialog	BLACK	CYAN
Error	WHITE	RED
Warn	BLACK	CYAN
Progress	WHITE	BLUE

These colors defined for these objects will be used by THEOS utilities for all color consoles, not just the current console.


Setup CRT

The main console keyboard/monitor used with PC-compatible computers is different than other consoles because it is completely controlled by software. This screen allows you to configure that software.

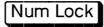



Use the normal menu selection keys to select the desired item. These keys are described in [“Using Menus”](#) on page 75.

- **Screen Saver**

The screen saver, when enabled, blanks the monitor after 10 minutes of keyboard inactivity. When blanked you can press any key to “wake up” the screen. The key pressed to wake up the screen is processed the same whether the screen was blanked or not. To wake up the screen without actually entering a character, press the  key.

- **Keypad Mode**

The keypad mode refers to the initial state of the  key. Selecting “Numeric” enables the  key; selecting “Control” disables it. In control mode the 10-key pad operates as arrow keys, home, end, *etc.*

- **Repeat Rate**

The repeat rate refers to the speed that characters are generated when a key is pressed and held down. Select one of the repeat rates offered: 5, 10, 15, 20, 25 and 30 characters per second.

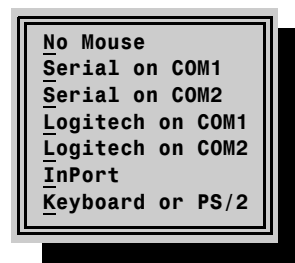
- **Status Line**

The status line is the bottom line of the display. When enabled the status line is used by programs for displaying a status message and by session management to display the current session number.

When disabled the bottom line of the display is available for normal display text and the session number is not displayed on the screen.

- **Mouse**

The mouse field allows you to specify whether or not you have a **session mouse** connected for this console, the type of mouse and where it is connected. A session mouse is a mouse device that is available to all sessions of this console. It can be used by “mouse-aware” programs such as Multiuser BASIC, Version 2.0. WindoWriter and THEO+Mail.



- **Ctrl-Alt-Del**

This field defines the text displayed and the response keys allowed when the operator presses **Ctrl**+**Alt**+**Del**.

The “Reset Literal” is the text message displayed. If the native language is other than English, enter the appropriate message here. Be sure to include an indication of the acceptable responses in the message.

The “Y scan code” and “N scan code” fields specify the keyboard scan-code values that are recognized as the positive and negative responses to the prompt message. Enter the hexadecimal values for the keys desired.

- **Exit to THEOS**

This menu item saves any changes made and exits Setup. The changes are not effective until the system is rebooted.

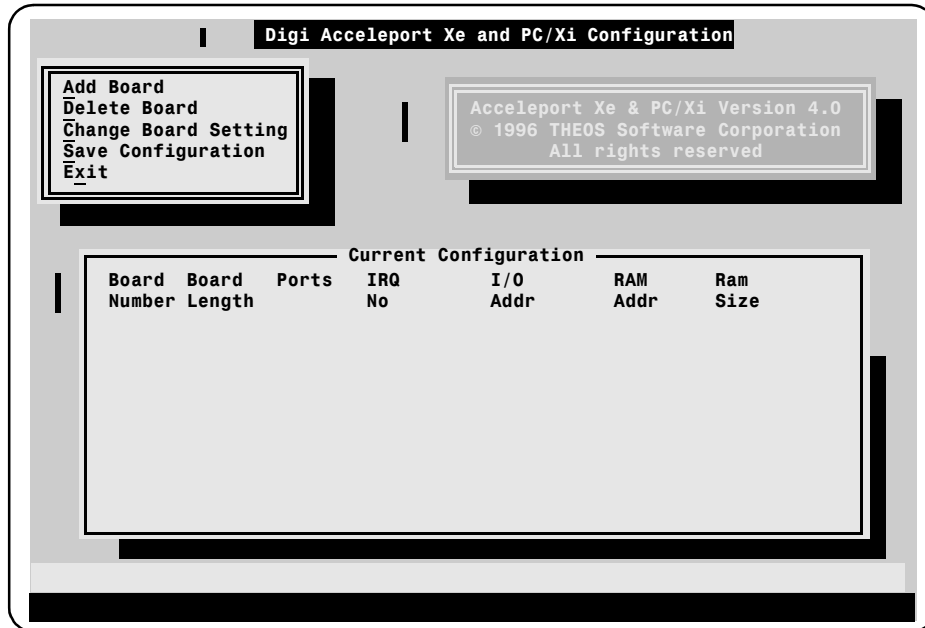
You can exit without saving your changes by pressing **Esc**. If any changes were made, you are asked if you want to save them or not.

Setup DIGIXE

This menu allows you to add, change and delete intelligent, Digi Acceleport Xe and PC/Xi multiport serial boards.



When adding or changing these board definitions, remember that you are only telling the software what boards you have in the system and what their operating parameters are. The actual setting of these parameters for the board must be done manually by changing jumpers or switches on the card.



Use the normal menu selection keys to select the desired item. These keys are described in [“Using Menus”](#) on page 75.

Add Board. When selected, a menu appears allowing you to select the type of board that you want to add:



Select the type of board installed or press **[Esc]** to cancel. For a description of each of these types of boards, press **[F1]** and the help text for the board displays.

After selecting a board you are asked to identify the interrupt request number (IRQ), the base I/O address and the memory base address for the board.

Although only valid and unused addresses and numbers are offered, it is your responsibility to use the address and IRQ number that matches the settings for the actual port.

You can cancel or backup to the prior selection by pressing **[Esc]**.

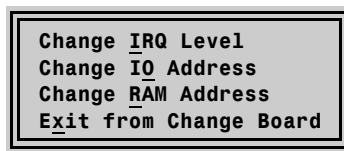
After you have answered all of the questions about the new port, it is added to the display and you are returned to the “Add, Delete, Change” menu. Note that the new port is not saved until you select the “Save Configuration” item.

You may have a maximum of seven Digi Acceleport multiport boards in a system. You can have more ports defined if you use nonintelligent multiport boards ([Setup SIO](#)) in addition to the intelligent ports defined here. Alternately, you may use a different type of intelligent multiport boards that allow more ports. You may not mix types of intelligent multiport boards. For instance, you may use Digi Acceleport PC/Xe or Digi CX, but not both.

Delete Board. When selected, a menu appears allowing you to select which board definition to delete. Use the arrow keys to position the highlight bar to the desired item or press **[Esc]** to cancel.

Change Board Setting. When selected, a menu appears allowing you to select which board definition to change. Use the arrow keys to position the highlight bar to the desired item or press **[Esc]** to cancel.

After you select the item a menu appears allowing you to select the item to change in the port definition:



Select and change each item desired. Select the “Exit from Change Board” or press **[Esc]** to return to the “Add, Delete, Change” menu.

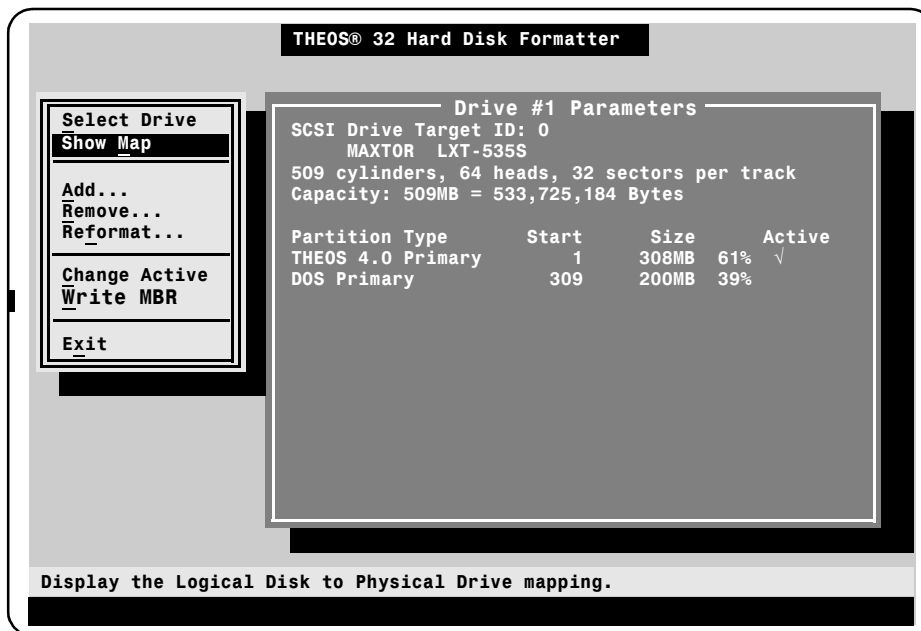
Save Configuration. You must select this menu item to save any additions, deletions or changes that you have made.

Exit. Exits the Setup DIGIXE configuration program without saving any changes.

Setup DISK

The “Hard Disk Formatter” screen provides all of the tools necessary to setup a hard disk drive and its logical disks. The term **drive**, or disk drive, refers to the physical disk; the term **disk** refers to the logical disk volume on a drive. Thus, a single physical drive may contain multiple logical disks, or a single logical disk might span two or more physical drives.

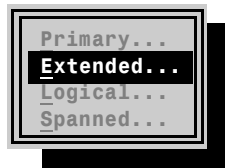
When Setup DISK is invoked, it offers a menu of functions and displays the parameters and current setup of the system disk. Items in the menu are grayed or shown in reduced intensity if the function is not available for the drive at this time. Note that the first item, “Select Drive,” is enabled if you have more than one hard disk drive attached.



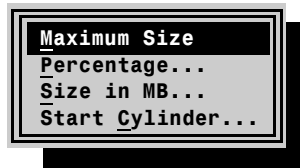
Use the normal menu selection keys to select the desired item. These keys are described in “Using Menus” on page 75.

- Select Drive** Select a drive other than physical drive one. This function is only available when two or more hard disks are installed on your system and are currently attached.
- Show Map** Displays a summary of the disk to drive mapping.
- Add** Adds a THEOS primary, extended, logical or spanned partition to the hard drive. You may add a partition to a drive only when there is unallocated space on the drive. You may only add a disk to unallocated space in an extended partition.

A disk may have only one THEOS primary partition and one THEOS extended partition. A THEOS extended partition may have multiple logical disks defined in it.



When adding a primary, extended or spanned partition to a drive, you may specify the partition size and location in one of four ways:



Note: Removable hard disks may only have one THEOS partition.

Maximum Size. SETUP uses the largest unallocated portion of the drive and defines a new partition in that space.

Percentage... With this option, you specify the percentage of the entire drive that you want allocated to the new partition. The maximum percentage available is displayed and it is the default size.

Size in MB... This option shows the maximum size available on the drive and allows you to specify the size for this new partition.

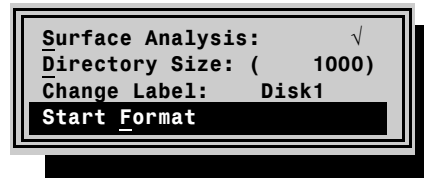
Start Cylinder... SETUP displays the starting cylinder number and the number of cylinders to specify if you want the maximum size partition cre-

ated. Enter the starting cylinder number and the number of contiguous cylinders to be allocated to this new partition.

Adding a Primary Partition to a Drive

A drive may contain one THEOS Primary partition. A primary partition is a bootable partition and is also a THEOS logical disk. As such, it needs to be formatted before it can be used.

After the primary partition is created, Setup needs to format the new disk.



Surface Analysis. When this item is checked, the disk formatter will perform an exhaustive write and readback analysis of the entire disk surface. Any sectors found to be suspect are added to the disk's bad sector map. These sectors will never be used by the operating system for directories or files.

Directory Size. Specify the size of the root or main directory of the disk. Be sure to use a size that is large enough for the expected uses of the disk. The directory size cannot be changed without clearing the current contents of the directory and thus losing all of the files on the disk.

Change Label. Specify the disk's volume label.

Start Format. This item starts the formatting of the new disk. If "Surface Analysis" was checked, the formatting process may take some time and a progress bar is displayed.

Adding a Logical Disk to an Extended Partition

When an extended partition is added to a drive, SETUP remains in the add menu and allows you to add logical disks to the extended partition. Unlike a primary partition, an extended partition is unusable until logical disks are defined within it. An extended partition may have more than one logical disk defined within its boundaries.

When adding a logical disk to an extended partition, you may specify the size as the **Maximum Size**, a **Percentage** or as a **Size in MB**. These size options are the same as described above except that they are relative to the size of the extended partition, not the drive.

After the logical disk is created, it must be formatted. SETUP uses the same formatting menu and procedures as described above.

Adding a Spanned Partition

A spanned partition is special type of logical disk that uses two or more physical drives.

Remove

Removes an existing THEOS primary, extended, logical or spanned partition, or a partition created by another operating system. You cannot remove any partition if it is currently in use as the system disk.

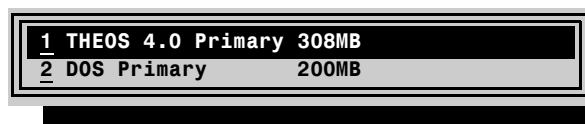
Before SETUP removes a partition, it displays a warning message about the consequences of doing so and allows you to cancel the operation. The default reply is to cancel. You must select the “Yes” response to actually remove a partition or logical disk.

Reformat

Reformats an existing THEOS primary, logical or spanned partition. You cannot format the current system disk. Refer to “[Formatting Hard Disks](#)” on page 116.

Change Active

Changes the active partition. The active partition is the disk partition that is booted by the computer’s firmware or THEOS MultiBoot when the computer is restarted. This function is enabled only when there are multiple primary partitions. When selected, SETUP offers a list of the available partitions. Non-primary partitions are grayed and cannot be selected.



After selecting the active drive, SETUP returns to the menu. The change in active drive has no effect until the system is rebooted.

Changing the active drive should not be necessary because the THEOS MultiBoot allows you to specify the active partition during bootup. The THEOS MultiBoot is described on page 23.

Write MBR

Replaces the Master Boot Record (MBR) on the drive using the THEOS MultiBoot program. When this disk is booted in the future, it will prompt you to select one of “OS (THEOS, Win, NT)?”

The THEOS MultiBoot program is automatically written to a disk when it is first partitioned and the THEOS partition is the only partition on the disk.

Setup FLOPPY The “Floppy Diskette Formatter” screen prepares one or more diskettes for usage by THEOS. For instance, when you want to [Archive](#) the files owned by an account to diskette, you must have sufficient blank diskettes available before you start the ARCHIVE. The Setup FLOPPY command provides an easy method of preparing these diskettes.

When Setup FLOPPY is invoked, it offers a short menu of the floppy drives attached and the standard sizes supported by those drives. For instance, if you have two floppy drives on the system, a 3½" drive on F and a 5¼" drive on G, the menu appears as:



Use the normal menu selection keys to select the desired item. These keys are described in “[Using Menus](#)” on page 75.


If a floppy drive supports 2.88MB floppies, that size will be offered as a menu option.

After a drive and size are selected, it then offers you a choice of formatting levels:



Unconditional Format. This level assumes that the diskette is either unformatted or that it is formatted but needs to be reformatted. First, every track and sector of the diskette is formatted and then a THEOS file system is created on the diskette. A progress bar displays during the formatting phase to show the relative progress of the format operation.

Quick Format. This level assumes that the disk is already formatted and only needs the THEOS file system written to it.

With either format level, you are prompted to insert a diskette in the drive and to respond with **Enter**  when you have done so. The diskette is then formatted with a THEOS file system. The label for the diskette is a generic “Floppy3½” or “Floppy5¼,” as appropriate. The main directory size is set to 20.

After the diskette is successfully formatted, you are prompted to insert another diskette to repeat the operation.

To change the label of the diskette or the size of the directory, use the DISK LABEL option or the DISK SIZE option described on page [303](#).

Note: The DISK command can also format diskettes. For instance:

```
>disk f (format
```

```
>disk f (clear
```

The first command performs an unconditional format. The second command performs a quick format.

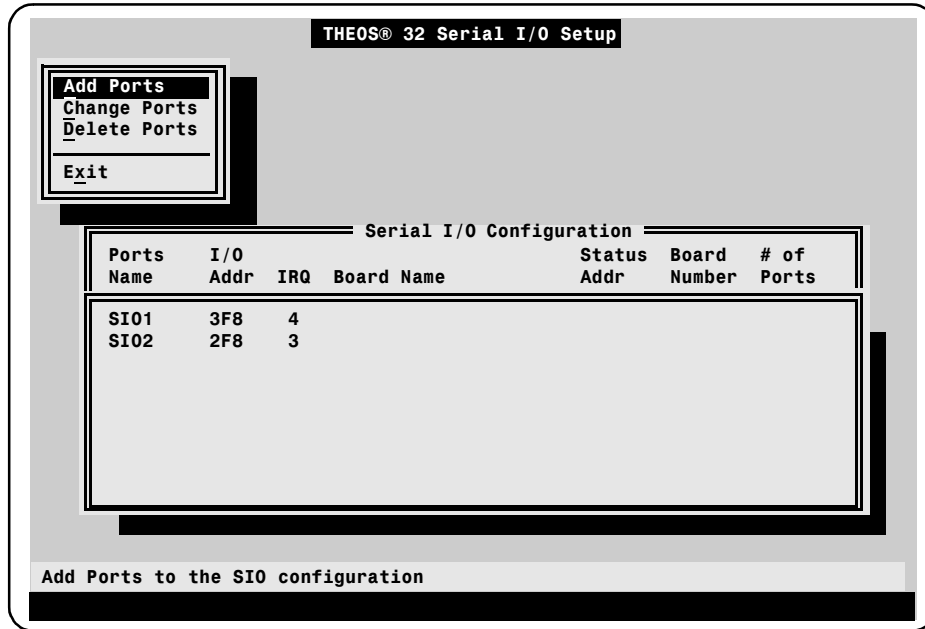
Use DISK to format a single diskette or to format a diskette with an unusual configuration (low density, single sided, *etc.*).

Setup SIO

This menu allows you to add, change and delete nonintelligent serial ports such as SIO1, SIO2, SIO3 and SIO4 and nonintelligent serial boards such as the Digiboard PC/X series of boards.



When adding or changing these port definitions, remember that you are only telling the software what ports you have in the system and what their operating parameters are. The actual setting of these parameters for the hardware port must be done manually by changing jumpers or switches on the card.



Use the normal menu selection keys to select the desired item. These keys are described in “[Using Menus](#)” on page 75.

Add Ports. When selected, a menu appears allowing you to select the type of port that you want to add:



Select the type of port desired or press **[Esc]** to cancel.

When you add COM1 through COM4 you are asked to choose the base I/O address and the IRQ number for the port.

When you add a nonintelligent DigiBoard you are asked to identify the number of ports on the board (4 Port or 8 Port), the base I/O address, the status port address and IRQ number for the board.

Although only valid and unused addresses and numbers are offered, it is your responsibility to use the address and IRQ number that matches the settings for the actual port.

You can cancel or backup to the prior selection by pressing **[Esc]**.

After you have answered all of the questions about the new port, it is added to the display and you are returned to the Add, Change, Delete menu. Note that the new port is not saved until you press **[F10]** or use the "Exit" item to save the additions.

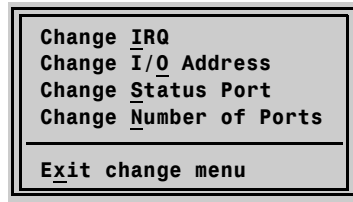
You may only have a maximum of four standard comm ports (COM1 through COM4) and four DigiBoard multiport boards. If each multiport board has the maximum of eight ports, you may have a total of 36 nonintelligent ports defined on a system. You can have more ports defined if you use intelligent multiport boards in addition to the nonintelligent ports defined here.

Change Ports. When selected, the a highlight bar appears in the lower window allowing you to select the port definition to change. Use the arrow keys to position the highlight bar to the desired item or press **[Esc]** to cancel.

After you select the item by pressing **[Enter ↵]** key, a menu appears allowing you to select the item to change in the port definition. For ports one through four, the selection menu is:



For DigiBoard definitions, the selection menu is:



Delete Ports. When selected, a highlight bar appears in the lower window allowing you to select the port to delete. Use the arrow keys to position the highlight bar to the desired item or press **[Esc]** to cancel.

When you select an item to delete by pressing the **[Enter ↵]** key, you are asked to confirm your request. You must select “Yes” to delete the port definition.

Exit. If any ports were added, changed or deleted, you are asked if you want to save the changes.

You may also save the changes by pressing **[F10]** or you can exit without saving any changes by pressing **[F9]**.

Setup Command Restrictions

The Setup command can only be used when you are logged onto the SYSTEM account (account id zero).

Although the Setup command requires only a privilege level of one, some of the functions require higher privilege levels.

See also [Account](#), [ClassGen](#), [Disk](#), [Sysgen](#), [Tape](#)

Shell Command

The Shell command is designed to be called from another program. It provides access to the “CSI shell” so users may temporarily leave a program, enter commands and, upon completion, return to that program.

SHELL

Operation

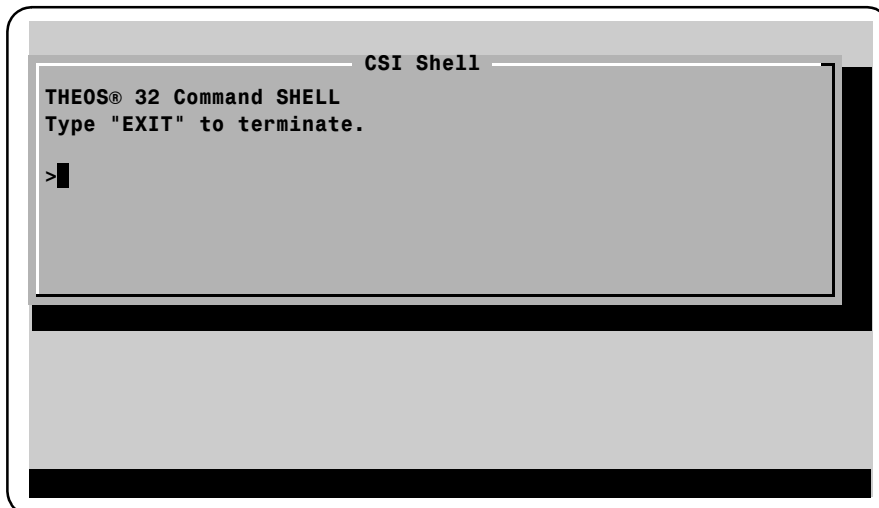
The current program’s environment is saved and the CSI Shell is entered. Upon entry, the CSI clears the currently active window and displays the reminder message:

```
THEOS® 32 Command Shell
Enter "EXIT" to terminate
```

If the current prompt string contains the default CSI prompt character (>), it is displayed with the blink attribute as a reminder to the operator that the user is in the CSI Shell and should return back to the calling program.

An example usage from a MultiUser BASIC Version 2.0 program might be:

```
001000 window open 1,2,10,76,10; color 7,4
001010 window frame 1, raised, right, color 7,4
001020 window title 1," CSI Shell ", center top, color 6,4
001030 window select 1
001040
001050 system "shell"
001060 print "I'm back...";
...
```



Notes

As illustrated in the example, the `Shell` command uses the current window for its display and input. That window is cleared before `Shell` displays its reminder message.

Files are not closed by this command. However, the statement or function that you use to invoke the command in your program may. For instance, in a MultiUser Basic language program, if you use the `SYSTEM` statement to invoke the `Shell` command, no files are closed. If you use the `CSI` statement to invoke the `Shell` command, your files will be closed before invoking the `Shell` command.

Use the [Exit](#) to exit the shell environment and return to the calling environment.

The ability to use this command can be disabled by renaming or deleting the file `SYSTEM.CMD32.SHELL:S`.

Show Command

The Show command displays the current value of *system-parameters*, *user-variables* and other information about the system.

1 SHOW

2 SHOW *function*

3 SHOW *

<i>function</i>	»	ABBREV	HISTORY	PCI	TAPE
		BREAK	IDE	PRIORITY	TIME
		CLOCK	IDENT	PRIVLEV	USER
		COPYLIB	IRQ	PROMPT	USER *
		CWD	LIBRARY	QUIT	USER <i>n mm-nn</i>
		DATE	LINKLIB	RDYMSG	VERSION
		DATEFORM	MEMORY	SCSI	WHO
		DATEIN	MEMORY *	SEARCH	WORKVOL
		DATEOUT	MSG	SERIAL	<i>name</i>
		DEVICE	OBJLIB	SUBDIR	
		DISK	PATH	SYNONYM	

Commands

Operation

Mode 1—Displays the current status of the system environment variables and all user-defined variables.

>logon develop

>show

```
RDYMSG      = OFF
MSG         = ON
SYNONYM     = PRIVATE.SYNONYM
WORK        = M
SEARCH      = S
PROMPT      = !rvon\alrvoff\g
HOME        = /:S
DIALDIR     = SYSTEM\DIALDIR:S
SUBDIR      = /PROGRAMS:S
CLIST       = YES
```

>

The minimum parameters displayed with this command include:

```
>show
RDYMSG      = OFF
MSG         = ON
WORK        = M
SEARCH      = S
HOME        = /:S
PROMPT      = \g

>
```

Any system or user-defined variable that has no value is not displayed.

Mode 2—This mode displays a specific system-parameter, user-defined variable or system function.

```
>show search
SEARCH      = S
```

Mode 3—Displays all of the information from [Mode 1](#) plus all of the standard system parameters.

```
>show *
ACCOUNT     = SYSTEM
USERNUM     = 0
PORT        = 16
PRIVLEV     = 9
LOGON       = 15:29:09 09/04/99
ABBREV      = ON
MSG         = ON
RDYMSG      = OFF
SEARCH      = S
WORKVOL     = M
SERIAL      = 102-12345
IDENT       = "TheosServer"
SYNONYM     = USER.SYNONYM
SUBDIR      = /
LIBRARY     =
PATH        =
OBJLIB      =
COPYLIB     =
LINKLIB     =
```

Unlike [Mode 1](#), a parameter that is undefined is displayed as a blank definition (see LINKLIB above).

Functions	<u>ABBREV</u>	Shows the status (ON or OFF) of the command abbreviation switch. This switch is set in the account environment when you Logon to an account and by the Set command.
	<u>BREAK</u>	Shows the value of the key defined to simulate a Break key. This key is set in the account environment when you Logon to an account and by the Set command.
	<u>CLOCK</u>	Displays the current time and date continuously. The time is updated once per second. Use the Esc or F9 to quit. <pre>>show clock 16:12:24 PDT Monday, April 20, 1998.</pre>
	<u>COPYLIB</u>	Shows the default copy library used by the language compilers and editors. This library is set in the account environment when you Logon to an account and by the Set command.
	<u>CWD</u>	This is a synonym to the SUBDIR function. It displays the current working directory.
	<u>DATE</u>	Displays the current time and date, once. <pre>>show date 16:12:24 PDT Monday, April 20, 1998.</pre>
	<u>DATEFORM</u>	Displays the current date format code. Refer to Table 28: “DATEFORM Codes” on page 476 for a description of these codes and their meanings.
	<u>DATEIN</u>	Displays the current code for interpreting two-digit year dates in MultiUser BASIC. <ul style="list-style-type: none"> 0 Two-digit year dates are interpreted as twentieth-century dates. 1 Two-digit year dates are interpreted as current-century dates. 2 Two-digit year dates are interpreted as twentieth-century or twenty-first century, depending upon how close that date is to the current system date. <p>Refer to Appendix I: “System Date and Time” for additional information.</p>

DATEOUT Displays the current code used by MultiUser BASIC when converting dates specified with two-digit years.

- 0** Twentieth-century dates are output with two-digit year numbers, other dates are output with four-digit year numbers.
- 1** Dates are always output with four-digit year numbers.
- 2** Dates are always output with two-digit year numbers.

Refer to Appendix I: “[System Date and Time](#)” for additional information.

DEVICE Displays information about all devices currently attached.

>show device

Name	Device Name
F	1.44 MB 3½ Floppy 1
G	1.2 MB 5¼ Floppy 2
P	SCSI[0:0] MAXTOR LXT-535S 8.57
S	SCSI[0:0] MAXTOR LXT-535S 8.57
CON	Main Display
PRT1	Print Spooler
COM1	COM1,IO=3F8,IRQ=4
PRT16	

In the above example, the PRT16 device does not support this capability so no information was displayed for it.

DISK Displays information about all currently attached disk devices.

>show disk

Name	Type	Bus	Id	Part.	Device Name
DISK1	IDE	0	M	Primary	WDC AC23 200L
DISK2	IDE	0	M	Extended	WDC AC23 200L
DISK3	IDE	1	M	None	SyQuest EZ230A

HISTORY Shows the status (ON or OFF) of the command history save switch. This switch is set in the system configuration by Sysgen.

IDE

Shows the IDE devices currently configured on this system.

```
>show ide
```

Bus	M/S	Proto	Rem	Type	Device Name	Serial
0	Master	IDE	N	Disk	IBM-DPRA-21215	

The “Rem” column indicates whether or not the device uses removable media.

IDENT

Shows the value of the identification field. This field is set in the system configuration by Sysgen or by the Network Login Server, if active. The setting by the Network Login Server takes precedence over the Sysgen setting.

IRQ

Shows the 16 IRQ numbers and the currently attached devices that use these IRQ numbers and the addresses associated with them. When adding a new device to the computer, this information is useful in determining which IRQs are already in use and which ones are available for use by the new device.

```
>show irq
```

IRQ	Device	Default	Address
0	CLOCK	(Timer)	0038:590
1	DEV3	(Keyboard)	04a0:2100
3	DEV3	(Com2)	04a0:2548
4	DEV5	(Com1)	0038:15a
5			0038:15a
6	DEV1	(Floppy)	3f70:1aa0
7			0000:0
8		(RT Clock)	0038:15a
9			0038:15a
10	DPTHA	(SCSI)	00c8:1f47
11	ET_DRV	(Network)	3de0:146d
12	DIGICX		3ba0:1b45
13		(Math chip)	0038:15a
14		(IDE)	0038:15a
15			0000:0

LIBRARY

Shows the default library used by the language compilers, editors and most commands. This library is set in the account environment when you Logon to an account and by the [Set](#) command.

LINKLIB

Shows the default link library used by the language compilers. This library is set in the account environment when you Logon to an account and by the [Set](#) command.

MEMORY Displays a summary of the current memory usage.

>show memory

THEOS 32 Memory Usage

Region Name	Length (bytes)	Size (Kbytes)
Boot Ram Size	0xf80000	15,872
Nucleus	0x42820	267
Device Drivers	0x354ec	214
Control Blocks	0x15e72	88
I/O Buffers	0x5a3d	23
Disk Cache	0x600000	6,144
Ram Disk	0x200000	2,048
Loaded Files	0x97c2	38
Program Code	0x57b94	351
Program Data	0x5dc94	376
Unused Memory	0x53e2b4	5,369
Largest Unused	0x51d33c	5,237

MEMORY * Displays a map of all of memory, followed by the summary display above.

>show memory *

```

Address  Length  Name
00000010  27F0 0008,93 Global Descriptor Table
00002800    800 0010,93 Interrupt Descriptor Table
00003004   2040 005B,F1 Unit Control Block
00005048  42820 0018,9B THEOS/32 NUCLEUS
0004786C   EB4 0038,9B THEOS/32 CLOCK
00048724  3DFB 00C8,9B SCSI Transport
0004C6F8    1C      ** Unused Memory **
0004C718   3B8 04A3,F1 CLASS90 Translate Table
0004CAD4    94 22B8,93 RPB for SYSTEM.CMD32.SHOW
0004CB68    F4      ** Unused Memory **
0004CC9C    94 22C8,93 RPB for SYSTEM.CMD32.KEYWORD
...
```

MSG Shows the status (ON or OFF) of the receive messages switch. This switch is set in the account environment when you Logon to an account and by the [Set](#) command.

OBJLIB Shows the default object library used by the language compilers. This library is set in the account environment when you Logon to an account and by the [Set](#) command.

PATH Shows the default command search path used by the language compilers and the CSI. This path is set in the account environment when you Logon to an account and by the [Set](#) command.

PCI Shows the PCI devices currently configured on this system.

```
>show pci
```

Bus	Dev	Func	VendID	DeviceID	Device Class	IRQ	Port
0	7	1	8086	7010	IDE controller	14	01F0
0	17	0	114F	0006	COM Controller	12	
0	18	0	1044	A400	SCSI controller	10	6000
0	19	0	1002	4354	VGA controller		
0	20	0	10EC	8029	Ethernet controller	11	6200

PRIORITY Displays the current priority level for your partition. The default priority of four is set when the partition is started. It can be raised or lowered with the [Set](#) command.

PRIVLEV Displays your current privilege level. The privilege level controls which commands you may use. It is set by your account environment when you Logon to an account. (The privilege level cannot be changed with the [Set](#) command.)

PROMPT Displays the current prompt string. This prompt string is set in the account environment when you Logon to an account and by the [Set](#) command.

QUIT Shows the status (ON or OFF) of the [Break](#), [\[Q\]](#) enabled switch. This switch is set in the account environment when you Logon to an account and by the [Set](#) command.

RDYMSG Shows the status (ON or OFF) of the command “ready message” switch. This switch is set in the account environment when you Logon to an account and by the [Set](#) command.

SCSI Shows the SCSI devices currently configured on this system.

```
>show scsi
```

Bus	Id	Width	Rem	Type	Device Name	Serial
0	0	16	N	Disk	SEAGATE ST31055W	002093420Y6BFY31313
0	2	8	Y	Tape	ARCHIVE Python 25501	
0	6	8	Y	CD-ROM	MATSHITA CD-ROM CR-504	

The “Rem” column indicates whether or not the device uses removable media.

SEARCH Displays the disk drive search sequence. This search sequence is set in the account environment when you Logon to an account and by the [Set](#) command.

SERIAL Displays the THEOS serial number for this operating system.

SUBDIR The current working directory is displayed. The current directory is set in the account environment when you Logon to an account, by the **Set** command and by the **ChDir** command.

SYNONYM Shows the user-defined command synonym file. This synonym file is set in the account environment when you Logon to an account and by the **Set** command.

TAPE Displays information about all, currently attached TAPE devices.

>show tape

Name	Type	Bus	Id	Device Name
TAPE1	SCSI	0	2	TANDBERG TDC 3600

TIME Displays the current time and date, once.

>show date







16:12:24 PDT Monday, April 20, 1998.

USER Displays the status of all defined partitions. The number of partitions or tasks is defined in the system configuration with the **Sysgen** command.

THEOS® 32 Show Users							
Pid	Username	Programe	Status & Info	Pid	Username	Programe	Status & Info
1	SYSTEM	SHOW	*	17			Stopped
2	SYSTEM	CSI	I	18			Stopped
3	TEST	CSI	I	19			Stopped
4	MANUALS	WINWRITE	I	20			Stopped
5		LOGON	I	21			Stopped
6		LOGON	I	22	SYSTEM	CSI	I
7	ACCTNG	MENU	I	23	THEO+FAX	FAXSCHED	Z Task of 24
8	ORDERS	INVOICE	I	24	THEO+FAX	FAXSERVE	I
9	ORDERS	ORDENTRY	I	25	HTTP	WEBSERVE	I
10	LETTERS	WINWRITE	I	26	FTP	FTP SERVE	I
11				27	PPP	PPPSERVE	I
12			Stopped	28	SERVICES	TCP SERVE	I
13			Stopped	29	NETLOGIN	NETLOGIN	I
14			Stopped	30	SPOOLER	DESPPOOL	I Task of 31
15			Stopped	31	SPOOLER	DESPPOOL	I
16			Stopped	32	CACHE	SYNC	ZN

This display differs from the other **USER** displays because all partitions are displayed at one time.

The “Status & Info” column uses codes for the status condition of the partition.

Code	Meaning
*	Indicates that this is the partition performing the Show USERS.
E= <i>n</i> <i>n</i>	Partition is waiting for semaphore <i>nn</i> to be set.
I	Waiting for interrupt. Usually, the program is waiting for another character from the console.
L	Waiting for a locked resource.
N	 ,  is disabled for the partition.
O	 ,  is in effect for the partition.
P	Stopped by  ,  or screen pause command.
R	The partition is running a program.
Z	Partition is “sleeping.”

This function can be used with the option PRIORITY.

>show user (priority

When this option is used, the priority for the user is displayed immediately following the status codes.

USER *

Displays a continuous status report of all defined partitions. The display is identical to the [USER](#) function except it is refreshed every second and the “Status & Info” column also displays the program name and address in memory that the program is executing at this moment in time.

This function can be used with the option PRIORITY.

>show user * (priority

When this option is used, the priority for each of the users is displayed immediately following the status codes.

USER *n*... Displays a status report for partition *n*. Multiple partitions can be specified by listing the partition numbers:

```
>show user 1 2 5 6 7 10
```

This command displays the status for the six partitions listed.

Ranges of partitions can be specified by using the syntax *n-m* where *n* is the from partition number and *m* is the to partition number. For instance:

```
>show user 1-4 10 16-20
```

This command displays the status for partitions 1, 2, 3, 4, 10, 16, 17, 18, 19 and 20.

The asterisk can be used to request a continuous display of the status of the partitions.

```
>show user 1-4 10 16-20 *
```

This function can be used with the option **PRIORITY**.

```
>show user 1-4 (priority
```

When this option is used, the priority for each of the users is displayed immediately following the status codes.

VERSION Displays the version and date for the operating system and all major components installed on your system.

THEOS® 32 Show Version			
Program name	Date	Version	Patch
THEOS® Operating System	15JUN98	4.1, 129 user	
THEO+DOS®	10JAN98	4.0	
MultiUser BASIC®	09DEC97	2.0	20252
MultiUser BASIC Runtimes	09DEC97	2.0	20252
THEOS C 32 Compiler	24JAN98	5.0 Build 27	
THEO+Com Communications	28OCT97	1.2	
THEO+Script	14NOV97	4.0 Build 28	
WindoWriter® Text Editor	21NOV97	4.0	
THEO+Net Network Connectivity	18DEC97	4.0, 8 user	
THEO+Fax Server	15DEC97	1.0	
THEO+Mail Internet Mail Client	17FEB98	1.1	
THEO+Server FTP & HTTP Server	12APR98	1.0	

The list of programs displayed by the **VERSION** function comes from the file `SYSTEM.MENU32.SHOW_VER`.

WHO Displays information identifying you and your partition.

```
>show who
ACCOUNT  = SYSTEM
USERNUM  = 0
PORT     = 16
PRIVLEV  = 5
LOGON    = 15:29:09 09/04/99
```

WORKVOL Displays the current work volume drive code. This drive code is set in the account environment when you Logon to an account and by the [Set](#) command.

name Displays the user-defined variable *name*.

Notes When the environment variable [LINEGRAPH](#) is set to “N,” the line graphics used in the display for [MEMORY](#) and [VERSION](#) are suppressed.

Restrictions The continuous display for Show [USER *](#) and the Show [MEMORY *](#) functions requires a privilege level of five.

See also [Account](#), [Set](#), [Sysgen](#), [Who](#), [WhoAml](#)

Sleep Command

The Sleep command causes your partition or process to suspend execution for an interval of time or until a specified time-of-day.

1 SLEEP *seconds*

2 SLEEP *time-of-day*

seconds » number of seconds to sleep

time-of-day » time to wake up

Operation **Mode 1**—Sleeps for *seconds* number of seconds.

>sleep 30

This command puts your partition to sleep for 30 seconds.

Mode 2—Sleeps until the system clock is equal to *time-of-day*.

>sleep 23:30

This command puts your partition to sleep until 11:30 pm.

Notes Once “sleeping” has started with this command it may only be awakened early by entry of the **Break**, **Q** or by a **Force** from another user.

The EXEC language has its own **&sleep** statement, BASIC has its own **SLEEP** statement and C has its own **sleep** function. If you need to put the process to sleep while executing a program, it is more efficient to use one of these statements or functions.

Restrictions *seconds* must be an integer number.

Sort Command Filter

This command sorts a stream file using the entire record as a sort key or designated portions of each record as the sort key.

- 1 **SORT** *-options -o output infile...*
- 2 **SORT** *-options +position1 -position2 -field-options -o output infile...*

<i>+position1</i>	»	sort key start position			
<i>-position2</i>	»	sort key end position			
<i>output</i>	»	optional output file name with optional path			
<i>infile</i>	»	file name with optional path			
<i>field-options</i>	»	b	f	n	
		d	i	r	
<i>options</i>	»	b	f	n	u
		c	i	r	
		d	m	tn	

Sort is a filter program and, as such, defaults to using the standard input device for its input file and the standard output device for its output file. It is suitable for use in pipes.

```
>list vendor.names | sort -d | more
```

The above command sorts the output of the LIST command using dictionary order and then displays the sorted output with the MORE command.

Operation

Mode 1—Sorts the *infile*s into one *output* file using the entire record as the sort key.

```
>sort -o sorted.data unsorted.data
```

```
>sort -briu -o sorted.data unsorted.data
```

The above two commands sort the file UNSORTED.DATA. The result of the sort is output as SORTED.DATA. The first command sorts the file using the sort order of the ASCII collating sequence. The second command also sorts the file but it ignores leading blanks in the records and it ignores any characters outside the range of ASCII characters. It also ignores duplicate records and outputs the file in reverse sequence.

Mode 2—Sorts the *infile*s into one file using the designated sort keys for determining the sort order of the records.

Input Files

Sort can sort multiple input files into a single output file. Merely list the input files on the command line, one after the other.

```
>sort -o sort.output sort.input1 sort.input2 sort.input3
```

The sequence of the listed input files does not matter unless the *-m* option is used.

If no input file is specified, the standard input device is used for the input file.

Sort can sort as large an input file or files as will fit in available memory.

Sort Keys

Unless otherwise specified, the entire record is used as a sort key. By using the *+position* and *-position* fields, portions of the record can be specified as the sort key(s).

The format for *+position* and *-position* is *f.c* where *f* is the number of fields to skip from the beginning of the record and *c* is the number of characters to skip from the beginning of the field. Fields are separated in records with a tab character. If a different character is used to separate fields, the *-t* option must be used to identify the separating character.

For instance, a specification of *+3.2* indicates that the sort key starts with the third character of the fourth field in the record.

Multiple sort keys can be specified. For example:

```
>sort +3.2 -3.10 +1.0 -1.20 sort.input -o sort.output
```

This command line states that the file *sort.input* is to be sorted using two keys, one starting with the third character of the fourth field through the eleventh character of the fourth field and the other using the first 21 characters of the second field.

Each sort key may have its own *field-options* specified immediately following the sort key specifications. For example:

```
>sort +3.2 -3.10 -ri +1.0 -1.20 -n sort.input -o sort.output
```

Sort keys that do not have their own *field-options* use the *options* specified for the entire sort. These *options* are specified before the sort key specifications.

Options

- b** Ignores leading blanks and other white space characters in the sort key for comparison purposes.

Input Keys	Comparison Keys
John F. Kennedy	John F. Kennedy
John F. Kennedy	John F. Kennedy
John F. Kennedy	John F. Kennedy

- c** Checks if input files are already sorted.
- d** Uses dictionary order when comparing sort keys. In dictionary order the sort order is changed so that digits come last, preceded by letters. All other characters (punctuation and other nonalphanumeric characters) are ignored when comparing two sort keys.
- f** For sort purposes, folds uppercase sort key characters to lowercase. Thus, “AARDVARK” sorts to the same location as “aardvark.”
- i** Ignores characters outside of the ASCII set of 7-bit characters for sort purposes only. This ASCII set of characters includes all of the characters from character value 32 (space) through 127 (tilde).
- m** Indicates that no sorting is to be done; the *infile*s are merged into the *outfile* in the sequence that the *infile*s are specified.
- n** The key is a number and should be sorted according to the value of the number.

Sort Keys	w/o -n option	with -n option
423	2	2
32	20	20
2	200	32
20	32	200
200	423	423

- r** The key is sorted in descending order, not the normal ascending sequence.

tc Specifies that the tab, or field delimiting character, is the character *c*. This option is used with the *+position* and *-position* options to specify that a character other than Tab separates the fields in the record.

```
>sort "-t," +2.0 +0.0 names.data
```

This command sorts a file of names and specifies that the comma character separates the fields in the records. It is placed in quotation marks to prevent the CSI from interpreting the comma as one of its characters.

Input Keys	Sorted Keys
Ken, E., Larvik	Joseph, William, Cone
Robert, G., Holbrook	Duncan, S., Holbrook
Michael, K., Malley	Robert, G., Holbrook
Robert, L., Lewison	Ken, E., Larvik
Shirley, L., McCartney	Robert, T., Lewis
Duncan, S., Holbrook	Robert, L., Lewison
Robert, T., Lewis	Michael, K., Malley
Joseph, William, Cone	Shirley, L., McCartney

u Specifies that only records with unique keys are output. When this option is not used, records with duplicate sort keys are output in the sequence that they were found in the input file(s).

Defaults The default *infile* is the standard input device and the default *outfile* is the standard output device.

The default sort sequence is ascending by character value. Refer to Appendix J: “[THEOS Character Sets](#),” starting on page [761](#) for a list of the characters and their values.

Restrictions Only stream files are sorted with this command.

The file is sorted using the currently available memory.

Maximum line length of records is 2,048 characters.

Split Command Filter

The Split command splits a stream file into multiple files, each one containing a portion of the original file.

1 SPLIT *infile outfile (option*

2 SPLIT *-nnn infile outfile*

infile » optional input file name with optional path

outfile » optional output file name with optional path

option » *nnn*

Commands

Operation Both Modes 1 and 2 operate the same. The only difference is that Mode 2 is “UNIX-like” in syntax.

The *infile* is read and output in *nnn* line chunks to the *outfile*. *outfile* name is modified with a two-letter suffix added. The first output file is *outfileAA*, the second is *outfileAB*, and so on.

```
>split system.history system.hist (1000
```

This command divides the SYSTEM.HISTORY file into 1,000 line chunks. The first 1,000 lines are written to SYSTEM.HISTAA, the second 1,000 lines are written to SYSTEM.HISTAB, *etc.*

Option *nnn* Specifies the number of lines each output file will contain.
When not specified the default value of 100 is used.

Defaults The default number of lines for each of the output files is 100.

When *outfile* is not specified the output file name is XAA.

When *infile* is not specified the input comes from the standard input device.

Restrictions The *infile* must be a stream file.

The *nnn* option must be a nonzero positive value.

See also [Cat](#)

Spooler Command

The Spooler command controls the print spooler.

Commands

1	<u>S</u> POOLER	<i>printer</i>		
2	<u>S</u> POOLER	<i>printer manager-function</i>		
3	<u>S</u> POOLER	<i>printer function</i>		
<hr/>				
<i>copies</i>	»	number of copies to print		
<i>function</i>	»	ABORT		KILL <i>reports</i>
		ALIGN <i>report nnn</i>		LIST
		BACKUP <i>nnn</i>		PRINT <i>report nnn</i>
		CHANGE <i>report queue copies hold</i>		STATUS
<i>hold</i>	»	HOLD		
		NOHOLD		
<i>manager-function</i>	»	ATTACH <i>options</i>	INIT	STOP
		BUILD <i>nnn drive</i>	QUIT	VERIFY
		FORM <i>queue</i>	START	
<i>printer</i>	»	optional spooled printer number		
<i>queue</i>	»	report or form queue letter		
<i>report</i>	»	report number		
<i>reports</i>	»	*, report number, range of report numbers and/or list of report numbers		

Operation **Mode 1**—Displays the status of a spooled printer or on all of the spooled printers.

```
>spooler 1
Printer #1 "CENTLP1" L80,P58,HPLASER,W8
-- is waiting for work
-- and has form "A" mounted

>s
Printer #1 "CENTLP1" L80,P58,HPLASER,W8
-- is waiting for work
-- and has form "A" mounted
Printer #2 "SI03" L80,P58,CANON2
-- is printing report # 9 "CHECKREG"
-- created by account ACCTNG - on page 7 of 9
-- and has form "B" mounted
```

Mode 2—Changes the status of a spooled printer or enables or disables the print spooler.

```
>spooler 3 stop
```

```
>spooler 3
Printer #3 "SIO4" L80,P58,EPSON
-- is stopped
-- and has form "C" mounted
```

Mode 3—Changes the attributes of a spooled report or performs print management of a spooled report.

```
>spooler change 11 2 d hold
```

```
>s list
```

```
File Acc-name Rpt-name   Date    Time  Q Pages C Status
11 SYSTEM    DEVNAMES 09/05/96 16:25 D    9  2 Closed, Hold
```

Spooler Manager Functions

ATTACH *options* Changes the attachment parameters of the spooler's printer. Note that this attachment refers to the spooler's printer and not the attachment of a logical printer to the spooler.

You may change any of the attachment parameters except the actual device. For instance, if the printer is currently connected and attached to CENTLP1, you cannot change it to SIO4 with this function. To add or change the physical device used for a spooled printer, you must use the [Sysgen](#) command and then reboot the system.

You may change any of the other parameters, such as baud rate, line and page length, class code, *etc.*

```
>spooler 3 attach 1132 p60 b38400 w8 e2 c135
```

```
>spooler 3
Printer #3 "SIO4" L132,P60,HPLASER,B38400,W8,XON/XOFF
-- is waiting for work
-- and has form "C" mounted
```

The *printer* must be either stopped or idle (waiting for work). You cannot change the attachment of a printer that is currently printing a report.

BUILD *nnn drv* This function creates a new SYSTEM.SPOOLER library. The only time that you might need to use this function is when you are first installing the system and want to create a spooler queue library that is larger than the 116 member default size used when the spooler is first installed by the [Sysgen](#) command.

```
>spooler build 2000 s
```

The *nnn* parameter specifies the size of the new SYSTEM.SPOOLER library and the *drv* specifies the drive used. If *nnn* is not specified, a library size of 116 is used. If *drv* is not specified, drive s is used. Make sure that the *drv* used here is the same drive specified in the INIT function or in the system configuration for the spooler. See “[Sysgen](#)” on page 531.

Note: If you want to change the size of an existing SYSTEM.SPOOLER library, you must stop the print spooler, erase the existing library, and then use the BUILD option to create a new library.

FORM *queues* Specifies which queues are printed on *printer*.

```
>spooler 4 form r
```

This command tells the spooler that spooled printer 4 has form R mounted on it and that it can print any reports in the R queue. Refer to the description of “[Forms and Queues](#)” on page 525 for more information about form specifications.

printer must be specified if there is more than one spooled printer. A printer may be printing from as many as eight queues.

```
>spooler 3 form abcdefgh
```

This command tells the spooler that spooled printer 3 can print reports in queue A, B, C, D, E, F, G or H.

```
>spooler 3 form "AX$g"
```

This command tells the spooler that spooled printer 3 can print reports in queue A, X, g or \$.

If a report is currently printing on *printer*, it will finish printing even if the report's queue no longer matches the printer's

form. The *queue* of a report is only matched to the *form* of a printer when it first starts to print.

INIT *drive* This function initializes and starts the print spooler. This initialization can also be done automatically at boot time if the “Enable Print Spooler” is set in the system configuration (see “Sysgen” on page 531).

drive is the disk drive code for the drive containing the SYS-TEM.SPOOLER library. If *drive* is not specified, s is used.

At least one printer must be attached when this INIT function is performed. All printers that are currently attached are transferred to the print spooler. (Slave printers are not transferred and cannot be spooled.)

For each physical printer that is transferred to the print spooler, a logical printer is attached to the spooler. These logical printers are available to you and to any other user when they logon to the system. (If they are already logged on they will have to Logon again to get these attachments.)

Assuming that three printers are attached:

```
>spooler init

>spooler
Printer #1 "CENTLP1" L80,P58,HPLASER,W8
    -- is stopped
    -- and has form "A" mounted
Printer #2 "SI03" L80,P58,CANON2
    -- is stopped
    -- and has form "B" mounted
Printer #3 "SI04" L80,P58,EPSON
    -- is stopped
    -- and has form "C" mounted
```

QUIT This function stops the print spooler. It is the opposite of the INIT function. You may only stop the spooler if the spooler is idle (not printing any reports) and the system is in single-user mode. In this case, single-user mode means that no other partitions are started.

```
>spooler quit
```

START This function activates a specific spooled printer or all spooled printers. This is the opposite of the STOP function.

```
>spooler 2 start
```

STOP

Stops a specific spooled printer. *printer* must be specified if there is more than one spooled printer. If a report is currently printing on *printer*, it is finished before the printer is actually stopped.

```
>spooler 3 stop
```

```
>spooler
```

```
Printer #1 "CENTLP1" L80,P58,HPLASER,W8
-- is waiting for work
-- and has form "A" mounted
Printer #2 "SI03" L80,P58,CANON2
-- is waiting for work
-- and has form "B" mounted
Printer #3 "SI04" L80,P58,EPSON
-- is stopped
-- and has form "C" mounted
```

It is best to STOP a printer before changing the paper in the printer. This insures that no report will print on the new paper until you tell the spooler that it is ready. Use the [START](#) function to restart the printer.

VERIFY

Verifies that the spooler's queue file matches the spooled reports that are in the spooler queue library and that the spooler queue library matches the spooler queue.

This function can also be performed when the system is booted by enabling the "[Check at Boot](#)" field in the system configuration (see "[Sysgen](#)" on page 531).

If errors are detected and it advises you to rebuild the spooler queue you must:

1. Use the [QUIT](#) function to stop the spooler (or boot the system in maintenance mode).
2. Erase the file SYSTEM.SPOOLER:S and all of its member files.
3. Rebuild the file by using the [BUILD nnn drv](#) function. Or, if the spooler is configured and enabled in the system configuration file, reboot the system. The boot process will create a missing spooler library and queue when it starts the spooler (see "[Sysgen](#)" on page 531).

Functions

ABORT

Stops printing the current report on the specified printer. *printer* must be specified if there is more than one spooled printer.

When a report is aborted, the spooler prints the message “****
A B O R T ****” and performs a form-feed. The report is marked as printed and the report copies is set to zero. If the report does not have HOLD status, it is deleted.

To successfully abort a report may depend upon the printer’s status and the transmission protocol. If the printer is off-line or powered off and the transmission protocol requires a response from the printer, then the report will not abort until the printer is powered on and made ready to print.

ALIGN *report page* Specifies that an alignment pattern is printed for spooled report number *report* starting with page number *page*. If the page number is not specified, page one is assumed. *printer* must be specified if there is more than one spooled printer.

The alignment pattern printed is a copy of the report’s page *page* with all letters on that page replaced with X’s and all digits on the page replaced with 9’s.

After an alignment pattern is printed you are asked “Do you wish another alignment page (Y/N).” Respond with **[N]** to cause the report to print. Before responding with **[Y]** make any alignment adjustments on the printer. When the alignment pattern is accepted the report is printed starting with the same page number.

A report is never printed nor is an alignment pattern created unless the report’s *queue* is the same as the requested printer’s *form*. If no *printer* is specified, the single spooled printer must have the proper *form* mounted.

BACKUP *pages* After the current page of the current report being printed on *printer* is printed, the report is backed up *pages* number of pages and printing resumes.

Omitting *pages* causes the report printout to back up one page. *printer* must be specified if there is more than one spooled printer.

CHANGE *report queue copies hold* Changes the attributes of spooled report number *report*. You must be logged onto the same account, or a synonym account, as the account used to create the report. The *queue*, *copies* and *hold* attributes can be specified in any order and one or more may be omitted, indicating that that attribute is not changed.

The *queue* is a single character identifying one of the 64 possible queues. Refer to the description of “[Forms and Queues](#)” on page [525](#).

copies is specified as one or two digits in the range 0–99. For descriptive purposes, you may specify the *copies* with the syntax COPIES=*copies* or COPY=*copies*.

hold is specified with either HOLD or NOHOLD.

If the report is currently printing and you change the *queue* to a letter that the printer is not set for, the report will continue to print on that printer. The *queue* of a report is only matched to the *form* of a printer when it first starts to print.

KILL *report* Removes and deletes spooled report number *report*. If the report is currently printing, it must be aborted first with the ABORT function. The ABORT function will kill the report unless it is marked as HOLD, in which case you must use the KILL function to remove it.

report may be specified with * or it may specify a single report number, a range of report numbers, a list of reports numbers, or any combination of these.

```
>spooler kill *
```

Removes all reports in the spooler queue.

```
>spooler kill 8-11
```

Removes reports numbered 8, 9, 10, and 11.

```
>spooler kill 2 6 7 33
```

Removes reports numbered 2, 6, 7 and 33.

```
>spooler kill 1, 3, 30-33, 45, 50-55
```

Removes reports numbered 1, 3, 30, 31, 32, 33, 45, 50, 51, 52, 53, 54 and 55.

LIST Displays a list of the spooled reports on the standard output device. The information displayed includes:

Column	Content
File	The report number.
Acc-name	Owning account name.
Rpt-name	Report name
Date Time	Date and time report created.
Q	Queue of report.
Pages	Number of pages in report. This is only an estimated number based upon the number of form-feeds in the report.
C	Number of copies still to print.
Status	Current status of the report. Status messages include Open, Closed, Printing, Printed, Hold.

The return code is set to 1 if one or more reports listed. Otherwise it is set to zero.

PRINT *report page* Specifies that spooled report number *report* is to be printed on *printer* starting with page number *page*. If the page number is not specified, page one is assumed. *printer* must be specified if there is more than one spooled printer. The *report* is printed even if *printer* is currently stopped.

A report is never printed unless the report's *queue* is the same as the requested printer's *form*. If no *printer* is specified, the single spooled printer must have the proper *form* mounted.

STATUS Reports on the status of the spooled *printer* or on all of the spooled printers. This is equivalent to a [Mode 1 Spooler](#) command.

Forms and Queues

Forms and queues are identified with single-characters. There are 64 possible forms and queues:

A – Z	26 forms/queues
a – z	26 forms/queues
# \$ % & * + - < = > ^ ~	12 forms/queues

Previous versions of the spooler only supported the first 26 forms and queues. To provide compatibility with programs and procedures designed

for previous versions of the operating system, form and queue specifications default to the uppercase form and queue letters.

To specify one of the lowercase form or queue letters you must enclose the form or queue within quotation marks. The 12 special character forms and queues may be specified without quotation marks. For instance:

```
>spooler 1 form a           ; Refers to form A
>spooler 1 form "a"        ; Refers to form a
>spooler 1 form x           ; Refers to form X
>spooler 1 form %           ; Refers to form %
>spooler 1 form abc$        ; Refers to forms A, B C and $
>spooler 1 form "Aix$%"    ; Refers to forms A, i, x, $ and %
```

To specify one of the lowercase queue letters you must enclose the letter in quotation marks or use the special syntax "QUEUE=\$*queue*."

```
>spooler change 2 a         ; Refers to queue A
>spooler change 2 "a"       ; Refers to queue a
>spooler change 2 $         ; Refers to queue $
>spooler change 2 queue=a   ; Refers to queue A
>spooler change 2 queue=$a  ; Refers to queue a
>spooler change 2 queue=$$  ; Refers to queue $
```

When in doubt about how a form or queue specification will be interpreted by the Spooler command, always use quotes and then specify the desired character with the desired case.

Restrictions

The functions [ABORT](#), [BACKUP](#), [PRINT](#), [ATTACH](#), [FORM](#), [START](#) and [STOP](#) all require a privilege level of one.

For all functions other than [BUILD](#) and [INIT](#), the print spooler must be initialized before the function can be used.

See also

[Attach](#), [Sysgen](#)

Start Command

Stop Command

The Start command starts a user, session or background task. The Stop command stops a user, session or background task.

1

START *partition console* (*attach-options*

2

START *partition console* (*attach-options* ACCOUNT *name* PASSWORD *password*

3

START *partition console* (*attach-options* MODEM

4

START *command*

5

STOP *partition*

attach-options

»

console attachment parameters

command

»

any THEOS or user-supplied command name with parameters

console

»

physical device name for console attachment

name

»

account name for automatic logon

partition

»

an unused partition id number

password

»

account password

Operation **Mode 1**—Starts a partition as a user with console. *partition* must be the number of an unused memory partition. The number of memory partitions is defined in the field “Maximum Number of Tasks” by the [Sysgen](#).

console must be a physical device name listed in SYSTEM.TEOS32.DEV NAMES. The *attach-options* are any valid console attachment parameters as defined by the [Attach](#) command (see “[Attach](#)” on page [178](#)).

```
>start 20 multi5 ( c190 b38400 w8 e5
```

When a user is started with this command, it has all publicly attached devices available and the privately attached console as specified in this command. Once the user environment is defined, it executes the Logon command and awaits the operator’s response to the “Logon please:” prompt.

See “[Session Management](#)” on page [60](#) for a description of starting a new session on your console.

Mode 2—Identical in operation to [Mode 1](#) except the new user is automatically logged onto *account*. The PASSWORD parameter is optional and is used when the account has a password. If the account has no password, the PASSWORD parameter is ignored.

```
>start 20 multi5 (c190 b38400 w8 e5 acc develop password supreme
```

If the PASSWORD parameter is not used and *account* has a password, the new user is started and Logon displays its “Password” prompt.

Mode 3—This mode starts a user similar to [Mode 1](#) and [Mode 2](#). However, when the MODEM keyword is used it tells Start that the user is connected via a remote modem connection.

A user started with this mode differs from both [Mode 1](#) and [Mode 2](#) in that the user’s console device is attached with this mode but no initialization strings are sent for the console class code. A special indicator is set that [Logon](#) uses.

Instead of starting the user with the “Logon please” prompt, the [Logon](#) command programs the modem to auto-answer the telephone line. It then waits for an incoming call and connection. After the connection is established between the two modems [Logon](#) then sends the classcode initialization string and requests that the user logs on.

When the remote user performs a [Logoff](#) (not lateral [Logon](#)), the connection is terminated and the modem is reinitialized to wait for the next incoming call.

Mode 4—This mode starts a background task. A ***background task*** is a user partition started without a console. *command* must be specified and may be any valid command line.

```
>start archive s tape (multiuse noask noquery volume
Task started as process #29
```

The partition number used for the new background task is the highest numbered partition currently available. It has all of the publicly attached devices available to it but does not have a console display or keyboard.

Instead of executing the Logon command, *command* is executed. *command* must be a command that does not require any console input. *command* may be an EXEC language program that executes a series of commands before exiting.

Whenever a background task command requests console input or exits to the CSI, it is stopped. Background tasks do have access to the standard

input and standard output devices and can use i/o redirection to simulate console input. Background tasks are described on page 58.

Mode 5—Stops user partition number *partition*. The partition must be in the program Logon. If necessary, use the **Force** to force the user to Logoff first before stopping the partition.

```
>stop 20
Partition is still active.
```

```
>force 20 logoff
```

```
>stop 20
```

Before using the **Force**, you should first check to make sure that the user is not in the process of updating any files.

If the user partition was started with a **Mode 3** start (modem server), the modem connection is first terminated before stopping the user.

Notes

In **Mode 1**, **Mode 2** and **Mode 3**, when a partition is started with a console on a serial port, a modem initialization string is sent to the partition's console port. The specific string of characters transmitted is defined by a file in the SYSTEM.MODEM library or by the file SYSTEM.TEOS32.MODEM. These files can be edited to customize the modem initialization string to that required by your modems. Refer to “**Starting Users**” on page 56 for a description of this user startup process. See also “**SYSTEM.MODEM Library**” on page 725.

For compatibility, the **Start** command accepts the command syntax used in prior versions:

```
>Start partition ( console attach-options start-options
```

Defaults

New users started with this command always have the devices that are defined in the system configuration file (see “**Sysgen**” on page 531) and any other devices that are publicly attached.

Spooled printers defined by the system configuration file are assigned the queues defined in that file, other printers are assigned queues in a one-to-one basis such as queue A to PRT1, queue B to PRT2, *etc.*

Return Codes

The **Start** command sets the return code to a non-zero value less than 1,000 when an error occurs and to a value greater than 1,000 when the partition is successfully started.

Return values greater than 1,000 specify that the partition was started successfully and give the specific partition number used for the new task. Subtract 1,000 from the return code to produce the partition number.

Restrictions

The Stop command requires a privilege level of five. [Mode 1](#) and [Mode 2](#) of the Start command require a privilege level of five.

When attempting to Start a new user or task, the message “All available tasks are in use” indicates that all partitions are already in use. You must either wait for a partition to become available (another task stops) or increase the number of partitions available by setting a larger “Maximum Number of Tasks” in your system configuration (see “[Sysgen](#)” on page [531](#)) and then reboot.

When starting a new user, if the device specified as console is already in use by another user or task, the message “Device is attached to user ...” or “Device ... is already attached to process ...” is displayed. Either choose a different device for this user or free up the device by stopping the other users.

See also

[Attach](#), [Sysgen](#)

Sysgen Command

Sysgen maintains the system configuration file (SYSTEM.TEOS32.CONFIG).

SYSGEN *drive* (*option*)

drive » optional disk drive letter for SYSTEM.TEOS32.CONFIG file

option » PRT*nn*

Operation

Maintains the system configuration file on *drive*. If drive is not specified, S is assumed. Refer to “[Sysgen Screens](#)” below for a description of each of the configuration screens used.

The drive parameter is used to change the system configuration file for a boot disk other than the current system disk. For instance, the Emergency Boot Floppy.

If the **PRTnn** option is specified, the configuration is printed and no maintenance is performed.

Option

PRTnn Sysgen prints the current system configuration on the attached printer number *nn*.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

Notes

The system configuration file contains information used when THEOS is booted. Changes made to the configuration are not effective until the system is reset or rebooted.

Sysgen Screens

Sysgen uses six different screens to maintain the system configuration. You may use the normal **PageUp** and **PageDown** keys to move from screen to screen. Use the **↑** and **↓** keys to move from field to field or line to line on a screen. The **F10** key saves the current configuration and exits Sysgen.

The **Esc** key cancels all changes made and exits Sysgen.

General System Environment

THEOS® 32 SYSGEN Version 4

```

Date Format.....(A)          (American/European/International)
System Identification.....(Administration )
Save History Information.....(W)      (None/Single/Daily/Weekly/Monthly)
History File Path.....(/LOGS:S
Math Processor Enabled.....(Y)        (Yes/No)
COM=CON Default Bypass.....(N)        (Yes/No)
Disk Cache Size.....(6M ) (0 - 262144 kilobytes)
Cache Write Delay.....(Y)            (Yes/No)
Directory Write Back.....(Y)          (Yes/No)
File Allocation Auto Test.....(Y)      (Yes/No)
Allow Maintenance Mode.....(Y)        (Yes/No)
Default Language Code.....(0 )
Adjust Daylight Savings Time..(N)      (Yes/No)
Standard Time Zone Abbrev.....(PST)   (Optional)
Daylight Time Zone Abbrev.....(PDT)   (Optional)
Hours from UTC.....(8 ) (Optional)
Maximum Number of Tasks.....(20 ) (1 - 254)
Input Date Format.....(2) (0,1,2)
Output Date Format.....(0) (0,1,2)
Ignore Scroll Lock Key.....(N)        (Yes/No)

```

Date Format. Sets the format for date entry and display.

Format:	A	E	I
Meaning:	American	European	International
Format:	mm/dd/yy	dd-mm-yy	yy.mm.dd
Example:	07/04/97	04-07-97	97.07.04

This date format is the initial format used when the system is booted. It can be changed after logging onto an account by setting the [DATEFORM](#) environment variable (see “[Set](#)” on page 474). When it is changed, all users use the new value.

Four-digit year number input and display are controlled by the two fields [Input Date Format](#) and [Output Date Format](#) at the bottom of this screen.

System Identification. This is a 16-character text field that can be used to identify this computer system. If THEO+Net is installed and the Login Server is active, the computer name identified in the [Setup NET](#) configuration is used instead of the value in this field.

Save History Information. This field specifies whether or not command history information is saved and how frequently the file is “rotated.” When this field is set to “N” no command or boot history is maintained.

With this field set to “S,” “D,” “W” or “M” records are written to the boot history file every time that the system is booted and records are written to the system history file every time that a user logs on or off the system, and when a command starts or finishes. For more information about these files, refer to “[SYSTEM.HISTORY](#)” on page 722.

History File Path. This field is used only when the previous field is not set to “N.” It specifies the path or location of the boot and system history files.

Note that, if the path specified here does not exist when the system is next booted, history logging is not performed.

Math Processor Enabled. This yes/no field specifies whether or not the math coprocessor is used for floating point number calculations. When a math coprocessor is not present or is disabled with this field, THEOS uses software routines to perform floating point arithmetic.

If a math coprocessor is present on the system, it should be enabled to provide faster calculations.

COM=CON Default Bypass. This yes/no field enables or disables (default) a mechanism that might be needed by legacy applications. To bypass class code translations and session manager control, you may attach the logical name COM n to the console and then output characters to the comm device.

```
>attach com1 con
```

Data sent to the COM n will be sent to the console device. How THEOS interprets this data depends upon the setting in this field.

If COM=CON Default Bypass is NO (default setting), the data is examined by the Session Manager to ensure that the proper session is active but it is not translated by the class code or transformed in any other way.

If COM=CON Default Bypass is YES, the data is completely ignored by the Session Manager. It is assumed that the data is being sent to the console device for forwarding by it to some other device connected to the console’s auxiliary port.

Refer to “[Console Bypass:](#)” on page 68 for more information about bypassing class code translations and session manager control.

Disk Cache Size. This field specifies the amount of memory used by the disk cache system for caching data from physical disks. It is specified in kilobytes or megabytes. Use a value of zero to disable disk caching.

If sufficient memory is available, 2,000 or more kilobytes should be specified here to make the most efficient access to the disk system.

As a convenience, large amounts of cache memory can be specified in megabytes by terminating the value with the character “M.” For instance, to specify 16,384K of cache memory, enter “16M”.

Cache Write Delay. This yes/no field tells the disk caching system whether or not to delay disk writes. With this feature of the cache disabled, all disk writes are performed immediately. With it enabled, a disk write may be delayed for several milliseconds until the cache system writes the information to disk.

For most efficient operation, this field should be “Y” because the cache system can then accumulate several disk writes and write them to disk at one time in the most efficient sequence.

The performance increase with both disk caching and write delay enabled is considerable. However, using disk caching with write delay enabled poses a slight risk to data integrity. If a power failure occurs or if the system is reset between the time that a program requests a disk write operation and when the disk caching software actually performs the write to the physical disk, the data is not written to disk.

If this risk is unacceptable, use disk caching with write delay enabled and also utilize an on-line uninterruptable power supply (UPS) for your computer system.

Directory Write Back. This yes/no field controls whether or not the disk cache software writes changes to a file’s directory immediately or only when the file is closed. For highest performance, this feature should be enabled. Although slower, this feature can be disabled for safest operation.

File Allocation Auto Test. This yes/no field controls whether or not a [Disk SHOW](#) is performed when the computer is booted. See “[Booting and Rebooting the System](#)” on page 24 for a description of this automatic file allocation test.

Allow Maintenance Mode. This yes/no field controls whether or not you are allowed to remain in maintenance mode when the computer is booted. See “[Maintenance Mode](#)” on page 27 for a description of this special single-user mode capability.

This single-user mode is a diagnostic and maintenance mode and should not be used if you want normal operation for a single-user.

Default Language Code. THEOS has the ability to have and use more than one set of files for help text, messages, keywords and menus with each set of files being in a different language. The numeric code entered here specifies the set of files used during the initial bootup process and during logons. The [Account](#) command can specify a different language code for each account name (environment switch [LANG](#)).

Use a language code of zero if your system has only one set of files, even if that set is not English.

Language code 0 is used at system startup when the language code specified here has no matching files.

Adjust Daylight Savings Time. This yes/no field specifies whether or not the operating system will automatically adjust the clock and timezone abbreviation used when daylight savings time starts and stops. In order for this automatic adjustment to occur you must fill in the next three fields with the timezone names and your timezone offset from the primary meridian (UTC). If these three fields are not set, then the automatic adjustment field is set to “N.”

When automatic adjustment is requested, it is performed if appropriate in three situations: When the system is booted, when you log onto an account, and when the CSI prompt is displayed.

For a brief description of how and when this automatic adjustment occurs, refer to Appendix I: “[System Date and Time](#),” on page 755.

Standard Time Zone Abbrev. Specify the three-letter abbreviation for your time zone when daylight savings time is not in effect. For instance, PST is used for Pacific Standard Time in the western states of the United States. This abbreviation is displayed when you log on or off the system and when you display the current time with the [Show](#) command.

Daylight Time Zone Abbrev. Specify the three-letter abbreviation for your time zone when daylight savings time is in effect. For instance, PDT is used for Pacific Daylight Time in the western states of the United States. This abbreviation is displayed when you log on or off the system and when you display the current time with the [Show](#) command.

If daylight or summer time is not used in your area, leave this field blank. Do not set it to be the same as the “Standard Time Zone Abbrev” field.

Hours from UTC. Enter the number of hours your local time zone is from the primary meridian (zero longitude). UTC is also referred to as GMT and is the time in Greenwich, England, which is on the prime meridian. This number can be a signed number and you should use a negative value if your time zone is east of UTC but west of the International Date Line (Europe, Asia and most of Africa, *etc.*).

If your time zone is not a whole number of hours from UTC, then use a fraction. Some UTC adjustments for common locations in the world are shown in the following table.

Location	Hours from UTC	Location	Hours from UTC
England, Iceland, Greenwich, Western Africa	0	Solomon Islands, New Caledonia	-11
Azores, Cape Verde Islands	1	Eastern Australia, Guam	-10
		Central Australia	-9.5
Mid-Atlantic	2	Japan	-9
Greenland, Brasilia, Uruguay, Argentina	3	Western Australia, China, Philippines, Indonesia, Hong Kong, Taipei	-8
Newfoundland	3.5		
Atlantic time zone, Venezuela, Bolivia, Paraguay, Chile	4	Sumatra, Jakarta, Southeast Asia, Central Russia	-7
Eastern time zone, Cuba, Columbia, Ecuador, Peru	5	Tibet	-6
		India	-5.5
Central time zone, Central America, Mexico	6	Islamabad, Karachi	-5
		Afghanistan	-4.5

Table 29: Time Zones

Location	Hours from UTC	Location	Hours from UTC
Mountain time zone, Western Mexico	7	Iran, Kazan, Volgograd	-3.5
Pacific time zone, Arizona	8	Moscow, St. Petersburg, Saudi Arabia, Iraq, Kuwait, Nairobi, Syria, Jordan	-3
Alaska	9	Finland, Israel, Turkey, Egypt, Sudan, South Africa, Eastern Europe	-2
Hawaii	10		
Bering time zone, Midway Island, Samoa	11		
New Zealand, Marshall Islands	-12	Norway, Sweden, Switzerland, Poland, Germany, France, Italy, Spain, Algeria, Angola	-1

Table 29: Time Zones

Maximum Number of Tasks. This field controls how many memory partitions are defined. Every started user or session, every background task and each program subtask require a separate memory partition. In addition, the disk cache program uses one memory partition, the print spooler uses a memory partition plus an additional memory partition for each spooled printer, and the network login server uses a memory partition.

Input Date Format. Sets the format used by MultiUser BASIC applications for interpreting two-digit year dates. Refer to Appendix I: “[System Date and Time](#)” for information about this and the next field.

Output Date Format. Sets the format used by MultiUser BASIC applications for outputting dates that are specified with two-digit years.

Ignore Scroll Lock Key. This yes/no field disables or enables the Scroll Lock key on all consoles. The default value is “N,” which enables the Scroll Lock key. With this field set to “Y” users must use **[Break]**, **[S]** to stop and start the display on their console.

Disk Drive Attachments

THEOS® 32 SYSGEN Version 4					
Disk Drive Attachments					
Drive	Name	UCB	DEV	Unit	Options
S	DISK1	3	2	0	
F	FLOPPY1	1	1	0	
G	FLOPPY2	2	1	0	
I	IMAGE1	56	59	0	/MY.IMAGEDRV:S
M	RAM2MB	64	64	129	

This second screen allows you to define the disk drives that are publicly attached and available to all users on the system.

To add a new drive, position to the first empty line and enter the drive letter and device name just as if it were an [Attach](#) attachment. Specify any options that might apply but, unlike the [Attach](#) syntax, do not use the left parenthesis before the first option.

You can delete an existing drive attachment by positioning to the start of the line and pressing **Ctrl+N** or **F5**. You can change an existing attachment by positioning to the character with the **←** or **→** keys and then using **Del** to delete undesired options or merely type in the additional options. **Home** and **End** position to the beginning or end of the current line.

You must have at least one drive specified in this screen and it must be the S drive. Normally, the S drive is the boot drive. It is possible for the S drive to be a drive other than the boot drive.

You may specify hard disk drives, removable hard disk drives, floppy disk drives, CD-ROM drives, image drives and RAM disk drives.

Image disks are specified with the image disk file name or with the physical name of IMAGE n . When IMAGE n is used the image disk file name must be specified as an option or a matching default file must exist when the

system is booted (DISK.IMAGEN). Also, the image file name must exist on one of the hard disks specified in this screen and it must be owned by the system account.

You need only specify the name of the device. The UCB, DEV, Unit and Options are copied from the device names file (SYSTEM.THEOS32.DEV NAMES). All disks specified in this screen are attached as public devices.

Other Device Attachments

THEOS® 32 SYSGEN Version 4				
Other Device Attachments				
Device Name	UCB	DEV	Unit	Options
TAPE1 TAPE600	11	34	3	
COM1 SI01	6	5	0	B38400,CTS,W8
PRT1 CENTLP1	8	6	0	L80,P58,HPLASER,W8
PRT2 CENTLP2	9	6	1	L80,P58,HPLASER,W8
PRT3 CENTLP3	10	6	2	L80,P58,CANON2
PRT4 MULTI3	25	16	2	L80,P58,B38400,DTR,W8,EPSON
PRT5 MULTI4	26	16	2	L80,P58,B38400,DTR,W8,EPSON
SLAVE9 CON				L80,P58,HPLASER
VDI1 CON				DISPLAY

This third screen is used to specify public devices other than disks. These other devices may be printers (PRT1–PRT64), comm ports (COM1–COM16), tapes (TAP1–TAP4), slave printers (SLAVE1–SLAVE nnn) and graphics devices (VDI1–VDI nnn). Use the same methods for adding, deleting and changing entries as described for the “[Disk Drive Attachments](#)” screen.

Spooled Printers. All printers specified in this screen (except slave printers) will become spooled printers if the spooler is enabled in the “[Print Spooler Settings](#)” screen. If the spooler is not enabled in this configuration file, the printers specified in this screen are merely public printers.

Slave Printers. A slave printer is a printer that is physically connected to a user’s console. It shares the communications cable used to connect the console to the computer. Because the console is always a private device, a slave printer is also a private device.

A slave printer is specified with the following syntax:

```
SLAVE $nnn$  CON options
```

The nnn refers to the user number specified in the next Sysgen screen. For *options*, specify only line length, page depth and printer class code.

If there are public printers (spooled or not spooled) and a slave printer specified for a user, that user's slave printer will be attached as the first available printer number between 1–64. If all 64 printers are defined, then that user's slave printer is attached as PRT64 and that user does not have access to the public printer defined as PRT64. (If the spooler is enabled, the spooler will have access to that printer.)

Slave printers for a network user are not specified with Sysgen. Use the client's network setup program to define a slave printer.

Graphics Devices. A graphics or VDI (Virtual Device Interface) device may be specified in this screen. VDI devices may only be attached as an alternate way of using one of the printers specified in this screen or as a console defined in the “[User Console Attachments](#)” screen.

A VDI device is specified with the following syntax:

```
VDI $nnnn$  PRT $nn$  options
```

or

```
VDI $nnnn$  CON options
```

The nn in PRT nn refers to the number of the public printer defined in this screen. The $nnnn$ in VDI $nnnn$ refers to the user number specified in the “[User Console Attachments](#)” screen. For *options*, specify only the VDI driver number, line length, page depth and printer class code.

When a VDI is associated with a user's console, it will be attached as VDI1 for that user.

User Console Attachments

THEOS® 32 SYSGEN Version 4					
User Console Attachments					
User	Name	UCB	DEV	Unit	Options
1	CRT	5	3	0	L80,P29,PCTERM
2	CRT:2	12	3	0	L80,P29,PCTERM
3	CRT:3	13	3	0	L80,P29,PCTERM
4	CRT:4	14	3	0	L80,P29,PCTERM
5	MULTI1	23	16	0	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
6	MULTI2	24	16	1	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
7	MULTI5	27	16	4	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
8	MULTI6	28	16	5	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
9	MULTI7	29	16	6	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
10	MULTI8	30	16	7	L80,P24,ANSIPCT,B38400,XON/XOFF,W8
11	MULTI9	31	16	8	L80,P24,ANSIPCT,B9600,XON/XOFF,W8, ACCOUNT=LETTERS

Commands

The fourth screen defines the users and their consoles that are started when the system boots. Use the same methods for adding, deleting and changing entries as described for the “[Disk Drive Attachments](#)” screen.

Be sure to specify all options that apply to the user’s console attachment. The class code specification is particularly important.

Similar to the [Start](#) command described on page 527, you may specify that a user is automatically logged onto an account. Use the option `ACCOUNT=account` to enable this feature. For an example, refer to user 11 in the above screen.

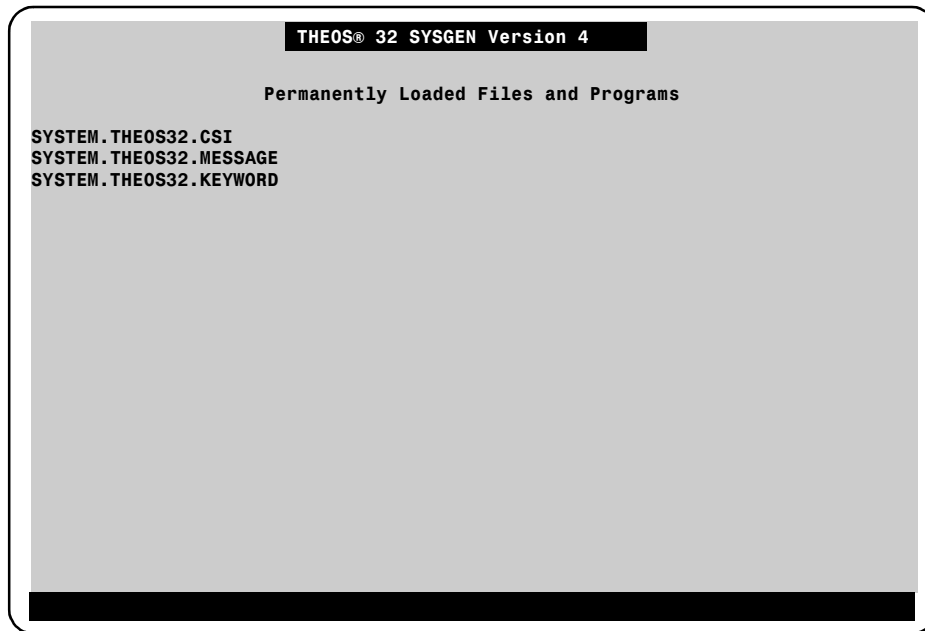


When the ACCOUNT is specified, you may also specify `PASSWORD=password`. Specifying an account’s password in this configuration file compromises the protections provided by password-protected accounts. The password in this file is “clear text” and may be viewed by anybody with read access to this file. To reduce the risk, never remove the shared read protection for this file.

Users 2–4 in the above screen define multiple sessions for user one. Each user may define their own sessions after they are started. Refer to “[Session Management](#)” on page 60 for more information about starting and using multiple sessions.

The MODEM option may be specified for users that are connected via remote modems. Refer to the [Start](#) command description of this mode.

Permanently Loaded Files and Programs



The fifth screen in Sysgen defines the programs and files that are permanently loaded into memory at system startup time. Use the same methods for adding, deleting and changing entries as described for the “[Disk Drive Attachments](#)” screen.

Merely list the files and programs that you want loaded, as in the example above. If the drive code is not specified, the file is loaded from the S drive.

Besides commonly used system programs, you may load application programs. For instance, the menu program used by your applications could be loaded because it is accessed so often by users.

Only programs and files owned by the system account may be loaded during system startup.

Print Spooler Settings

THEOS® 32 SYSGEN Version 4			
Print Spooler			
Enable Print Spooler.....	(Y)	(Yes/No)	
Drive Assignment.....	(S)		
Begins Started.....	(Y)	(Yes/No)	
Check at Boot.....	(N)	(Yes/No)	
Secure Mode.....	(N)	(Yes/No)	
Default Size.....	(116)	(32 - 32767)	

Prt Form	Prt Form	Prt Form	Prt Form
1 A	17	33	49
2 B	18	34	50
3 C	19	35	51
4 D	20	36	52
5 E	21	37	53
6	22	38	54
7	23	39	55
8	24	40	56
9	25	41	57
10	26	42	58
11	27	43	59
12	28	44	60
13	29	45	61
14	30	46	62
15	31	47	63
16	32	48	64

Commands

The sixth and last screen of Sysgen enables the print spooler at system start time and assigns the initial form to each of the started printers. Use the same methods for adding, deleting and changing entries as described for the “[Disk Drive Attachments](#)” screen.

Enable Print Spooler. This yes/no field controls whether or not the print spooler is initialized at system start time. A “Y” entry here enables the print spooler and all non-slave printers defined in this system configuration file (“[Other Device Attachments](#)” screen) become spooled printers.

Drive Assignment. Specify the drive letter for the drive that contains the SYSTEM.SPOOLER library. If the spooler is enabled and this library cannot be found on that drive, it is automatically created the next time that the system is booted.

You may only specify a drive code that is defined in the “[Disk Drive Attachments](#)” screen described on page [538](#).

Begins Started. This yes/no field controls whether or not the print spooler is initialized in the started or stopped condition. A “Y” means that all spooled printers are started when the system boots.

Check at Boot. This yes/no field controls whether or not the print spooler's queue file is checked for integrity when the system is first started.

Secure Mode. This yes/no field specifies whether or not spooled report files are secure. When enabled, spooled reports are created with hidden, erase and read protections enabled. Only the despooler and the Spooler, Archive and TBackup commands can access these files. Normal file operations can not read them (list), copy them or delete them.

Default Size. This field controls the size of the spooler's queue and SYSTEM.SPOOLER library if it does not exist and must be created during system startup.

Form. Enter the paper form letter or letters for each of the spooled printers defined in the second screen of Sysgen. Unlike the [Attach](#) and [Spooler](#) commands, form characters are interpreted exactly as entered.

The form letter specified in this screen is used as the initial or default queue for each user's printer attachment when they are started and when they log onto an account (not lateral logons).

Note

If printers are defined in the “[Other Device Attachments](#)” screen and the spooler is enabled in the “[Print Spooler Settings](#)” screen, when the system is rebooted the spooled printers are assigned consecutively. That is, if you defined PRT3, PRT6 and PRT10, they become spooled printers 1, 2 and 3.

Cautions

This program defines the configuration of your system the next time it boots. Because this information is critical to the proper operation of the system, you should enable the “Allow Maintenance mode” capability on the first screen until you are sure that the configuration is proper.

If there is a serious error in the configuration that prevents proper operation of your system, you can reboot and respond with [Esc](#) when the bootup process says to “Press ESC now to enter maintenance mode.” Then use Sysgen to modify the configuration until it is correct.

Only when you are satisfied that the configuration is proper should you disable the “Allow Maintenance Mode” capability.

Restrictions

The Sysgen command may only be used when you are logged onto the SYSTEM account (user number 0) with a privilege level of five.

See also

[Attach](#), [Load](#), [Setup](#), [Spooler](#), [Start](#)

System Command

The System command changes the system disk drive.

```
1  SYSTEM
2  SYSTEM  drive
_____
drive          »  optional drive letter code
```

Commands

Operation

Mode 1—This mode is used when the system disk is a removable disk drive and you want to change the disk volume in the drive. When the command is executed, you are prompted with:

```
Mount new system disk on drive S[1:1:0]
```

Change the disk volume to another valid THEOS system disk and press

.

Mode 2—This mode is used when you want to change the system disk to a different drive. For instance, after booting from an Emergency Boot Diskette, the System command is used to switch to the hard disk.

```
>system m
```

Caution

You may change to a non-system disk. That is, you may change to a disk that does not contain an operating system or its support files. Although this may be desirable, in this situation there will be some limitations to the commands that you may execute. Any command that requires a file in the SYSTEM.MENU32 or SYSTEM.TEOS32 libraries will not execute properly because these libraries are always assumed to be resident on the current system disk.

Notes

You must use this command to change the attachment of the system disk. The [Attach](#) cannot change the S disk assignment.

The System command does not load any programs or files from the new system disk. Unless they are loaded into memory, it does check for the presence of the following files: SYSTEM.TEOS32.CSI, SYSTEM.TEOS32.KEYWORD and SYSTEM.TEOS32.MESSAGE.

If [WORK](#) is S, all of the system work files are copied to the new system disk.

Restrictions

The `System` command requires a privilege level of five.

You must be in true single-user mode. That is, no users or background tasks started, including the print spooler or Network Login Server. The disk cache may be enabled.

Unless they are already loaded into memory, the files `SYSTEM.TEOS32.CSI`, `SYSTEM.TEOS32.KEYWORD` and `SYSTEM.TEOS32.MESSAGE` must be present on the new system disk.

See also

[Reboot](#), [Sysgen](#)

Tail Command Filter

The Tail command displays the ending of a text file on the standard output device.

- 1 TAIL *file...* (*options*
- 2 TAIL *file* (*options* FOLLOW

file » file name with optional path; may contain wild cards

options » +*nnn*
 -*nnn*
 CHARS
 FOLLOW
 LINE

Operation

Mode 1—The last lines or characters of *file* are output to the standard output device. When more than one *file* is specified, the last lines or characters of each of the files are output.

```
>tail system.theos32.config
USER1 5:3:0 L80,P24,C90
USER2 5:3:0:2 L80,P24,C90
USER3 5:3:0:3 L80,P24,C90
USER4 5:3:0:4 L80,P24,C90
USER5 5:3:0:5 L80,P24,C90
USER6 5:3:0:8 L80,P24,C90
LOAD SYSTEM.TEOS32.CSI
LOAD SYSTEM.TEOS32.MESSAGE
LOAD SYSTEM.TEOS32.KEYWORD
LOAD SYSTEM.TEOS32.SYNONYM
```

Mode 2—This mode outputs the tail of a file and then monitors the file for any growth in the file. When additional data is written to the file by another user or task, it is displayed and monitoring continues. You must use **[Esc]**, **[F9]** or **[Break]**, **[Q]** to terminate this command.

```
>tail system.history (-2 follow
09/08/96 15:55:06.579   1 End Program      RC: 0 CPU: 0.380
09/08/96 15:56:16.078  16 Start Program   Command: TAIL
SYSTEM.HISTORY ( -2 FOLLOW
```

When another user causes additional records to be written to the history file, they are displayed here.

Options	<u>C</u>HARS	Count characters instead of lines.
	<u>F</u>OLLOW	Use Mode 2 of the Tail command. If multiple <i>files</i> are specified, only the first <i>file</i> is output and followed.
	<u>L</u>INES	Count lines instead of characters. This is a default option.
	+nnnn	Begin output <i>nnnn</i> lines or characters from the start of the file.
	-nnnn	Begin output <i>nnnn</i> lines or characters from the end of the file. The default is -10.
Notes	<p>When multiple files are displayed, each file's output is identified with a line displaying the complete path to the file.</p> <p>A <i>file</i> specification can omit the file type if the environment variable FILETYPE is defined.</p> <p>For more information about the FILETYPE variable, see “Environment Variables” on page 102.</p>	
Defaults	LINES is a default option and -10 is the default count.	
See also	Head , List , More	

Tape Command

The Tape command initializes and manipulates a tape volume.

1 TAPE

2 TAPE *tape functions*

tape

» tape device name, such as TAP2 or TAP4

functions

» EJECT LABEL SHOW VERIFY
FORMAT REWIND TENSION WTM
INIT *label* RUN UNLOAD

Commands

Operation **Mode 1**—Using the TAP1 device, this mode rewinds to the beginning of the tape volume and reads the header information. The tape label, the first file name and its creation date are displayed.

```
>tape
Tape TAPE1 label "FRIDAY".
Archived from disk "PRODUCTN" on 12/06/96, at 17:35.
File name "ARCHIVE.DISKTAPE" created 12/06/1995.

>
```

Mode 2—The requested *function* is performed on *tape*. If *tape* is not specified, TAP1 is used.

```
>tape show
Tape TAPE1 label "FRIDAY".
Archived from disk "PRODUCTN" on 12/06/96, at 17:35.

>
```

Function	<u>EJECT</u>	If the tape device supports a programmable eject feature, the tape cartridge is ejected from the tape drive.
	<u>FORMAT</u>	<p>A TENSION function is performed on the tape and then new header labels are written. If the tape supports physical formatting, the tape is formatted instead of tensioned.</p> <pre> >tape format Enter tape label: Tuesday > </pre>
	<u>INIT</u> <i>label</i>	<p>A REWIND operation is performed and then it writes a new volume label without tensioning or formatting the tape.</p> <pre> >tape init "Monday" </pre> <p>Tape labels are one to eight characters in length.</p>
	<u>LABEL</u>	<p>A REWIND operation is performed and then all file labels are displayed and data blocks are counted.</p> <pre> >tape label VOL1TSC002 HDR1ARCHIVE.DISKTAPE 0001000100000098316 000000 HDR2F0409600128 ** Tape Mark ** Number of data blocks = 28917 ** Tape Mark ** EOF1ARCHIVE.DISKTAPE 0001000100000098316 028917 EOF2F0409600128 ** Tape Mark ** ** Tape Mark ** </pre>
	<u>REWIND</u>	Rewinds to the start of the tape.
	<u>RUN</u>	This function is synonymous with the EJECT function. If the tape device supports a programmable eject feature, the tape cartridge is ejected from the tape drive.
	<u>SHOW</u>	<p>Reads the next tape header. The header and the file name and creation date are displayed. If the tape is positioned to the end of the tape, then "Tape mark" is displayed.</p> <pre> >tape show Tape TAPE1 label "FRIDAY". Archived from disk "PRODUCTN" on 12/06/96, at 17:35. > </pre>

<u>TENSION</u>	A “fast forward” and a “fast rewind” are performed on the tape to ensure uniform tension throughout the tape volume.
<u>UNLOAD</u>	This function is synonymous with the EJECT function. If the tape device supports a programmable eject feature, the tape cartridge is ejected from the tape drive.
<u>VERIFY</u>	Verifies the readability of the entire tape by rewinding to the start and then reading every block on the tape. <pre>>tape verify Block: 40, length: 0 ></pre>
<u>WTM</u>	Writes a tape mark on the volume. Every file on the tape is automatically terminated with a tape mark. The end of a tape is indicated by two consecutive tape marks.

Notes	Before a tape can be used by THEOS, it must be initialized with a tape label.
Restrictions	The Tape command requires a privilege level of four.
See also	Archive , Attach , Backup , CopyFile , Eject , Restore , TBackup

TBackup Command

The TBackup command backs up files to another drive, or restores those files. It can also be used to verify or compare the backup files to their original source files.

Commands

- 1 **T**BACKUP *file-spec... backup* (**B**ACKUP *backup-options*
- 2 **T**BACKUP *backup* (**C**OMPARE *compare-options*
- 3 **T**BACKUP *backup file-spec... (RESTORE *restore-options**
- 4 **T**BACKUP *backup* (**V**ERIFY *verify-options*

<i>backup</i>	»	drive letter of disk or tape volume containing backup or the file name specification for the backup data set			
<i>file-spec</i>	»	drive letter of source drive for backup or specification of files for backup			
<i>backup-options</i>	»	ACCOUNT ASK COMPARE DIFFERENTIAL EJECT	ERROR FILES FULL INCREMENTAL LABEL	MULTIUSER NOASK NOEJECT SETNAME SUBDIR	VERIFY VOLUME WAIT <i>date1</i> <i>date2</i>
<i>compare-options</i>	»	ASK EJECT ERROR	LIST NOASK NOEJECT	PRT <i>nn</i> SETNAME WAIT	
<i>restore-options</i>	»	ACCOUNT ASK DISKMAP ERROR EJECT FROM	LIST NEWFILE NOASK NOEJECT NOQUERY NOSYSFILES	OLDER OLDFILE PRT <i>nn</i> QUERY REPLACE SETNAME	VOLUME WAIT <i>date1</i> <i>date2</i>
<i>verify-options</i>	»	ASK ERROR	EJECT LIST	NOASK NOEJECT	PRT <i>nn</i> SETNAME

Operation

Mode 1—Creates a backup data set on *backup* of the files specified by *file-spec*. Although no options are required, you should specify the **SETNAME** for the data set.

The *backup* destination may be specified in one of two ways:

1. With a drive code. When this is used, the backup data set is written to a file named TBACKUP.VOL00001:*backup*. The destination volume is cleared of all existing files before this data set is created.

```
>tbackup s tape (backup setname "Weekly backup")
```

If the data set does not fit on a single volume, subsequent volumes will be requested and the file names used on those volumes will be TBACKUP.VOLnnnnn:*backup*, with *nnnnn* being a sequential number incremented for each volume.

2. With a file name specification. When this is used, the destination volume is not cleared and the backup data set is written to the file name specified.

```
>tbackup s weekly.backup:d (backup differential)
```

The data set must fit in the available space on the destination drive. If the data set does not fit in the space remaining on that volume, an error message will display.

The *file-spec* may be a list of several file specifications. For instance:

```
>tbackup *.data *.program *.notes tape (backup)
```

With the above command, all files on all drives in the default search sequence with a file-type of “data,” “program” or “notes” are backed up to tape.

file-spec may be omitted, in which case it means that all files on all drives in the drive search sequence are candidates for the backup. The drive search sequence is defined in the account environment (see “[Account](#)” on page 156) or by the **SEARCH** environment variable (see “[Set](#)” on page 474).

Unless specifically identified in *file-spec* the following sets of files are not backed up with this command:

```
SYSTEM.CSI*
SYSTEM.CSISV*
SYSTEM.EXEC*
SYSTEM.PIPE*
SYSTEM.WORK*
```

Mode 2—Compares the contents of a backup data set specified by *backup* with the files that it was originally backed up from.

For instance:

```
>tbackup s tape (backup setname "Weekly Backup"
>tbackup tape (compare setname "Weekly Backup"
```

If the source files have been changed since the backup data set was created, this compare operation will fail. Remember, in a multiuser environment, other users may be changing the database during the backup or between the start of the backup and the comparison operation.

Mode 3—Restores the files from the backup data set to the original location of the files.

```
>tbackup tape s (restore
>tbackup tape *.data:s *.data:a program.source.*:s (restore
```

The **DISKMAP** option may be used to instruct TBackup to restore the contents of the data set to a drive other than the original source of the backup.

```
>tbackup s.file:s a.file:a tape (backup setname "Test"
>tbackup tape (restore diskmap s-a diskmap a-s setname "Test"
```

After the restore operation is complete, the A drive will have the file named S.FILE and the S drive will have the file named A.FILE.

Mode 4—Verifies the readability of the backup data set specified by backup. Verifying a data set does not test to see if the data set is an exact copy of the original file as the **Mode 2** compare operation does. Instead, it does read every byte of the data set, verifies that all of the checksums are correct, and that every file and “record” in the data set starts and ends where it should.

**Backup
Options**

ACCOUNT Only the files owned by the current account are candidates for the backup. This option sets **SUBDIR** option.

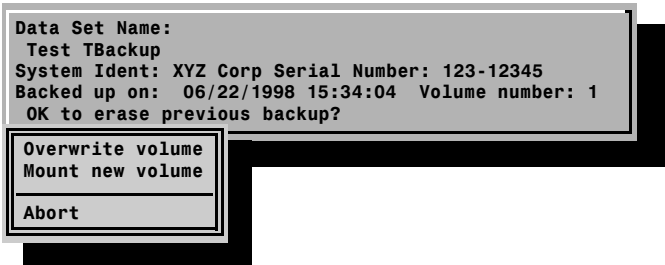
Use the **VOLUME** option to override this default.

ASK When TBackup is ready to begin writing the backup data, this option instructs TBackup to ask the operator to mount the destination volume and waits for confirmation that the proper volume is mounted.



Mount the first tape of the data set.

After you load the proper volume in the disk or tape drive and acknowledge this message, TBackup looks for an existing file named TBACKUP.VOLnnnnn. If there is an existing file, TBackup displays the following information about it:



Data Set Name:
Test TBackup
System Ident: XYZ Corp Serial Number: 123-12345
Backed up on: 06/22/1998 15:34:04 Volume number: 1
OK to erase previous backup?

Overwrite volume
Mount new volume

Abort

Select “Overwrite volume” to use this disk or tape as the backup volume. “Mount new volume” returns to the prior question, allowing you to insert a different disk or tape into the drive.

ASK is a default option unless *backup* is a file name specification, in which case this option is ignored. Use the **NOASK** option to override this default.

COMPARE Indicates that, after the backup data set is created, TBackup should compare the data set against the original files. The **NOASK** option is in effect when the comparison starts.

DIFFERENTIAL Only those files that have their modified bit set are included in the backup. This is the only option that prevents

TBackup from resetting the modified bit for files that it copies. See “[Differential Backups](#)” on page 566.

EJECT

This option applies when the *to-drive* is a tape drive or a removable hard drive. It specifies that the final volume is ejected from the drive after the backup is complete.

ERROR

Specifies that all errors detected during the backup are written to the indicated file. For instance:

```
>tbackup *.data:s tape (backup error backup.log:s
```

This command backs up all files with a file-type of DATA to the TAPE drive. Errors detected are logged to the file BACKUP.LOG:S.

FILES

Indicates that *file-spec* is the name of an ASCII stream file with each record in the file specifying a single file name or a wild-card file specification. The file name specifications in this file may include the account name and path to the file.

For instance, lines in the specification file might contain:

```
*.data:s
develop\custom\programs\program.source.sample:s
```

The SELECTED.FILES and SELECTED.EXEC files created by [FileList](#) and the FOUND.EXEC created by [Look](#) can be used for this specification file (see “[The EXEC and FILES Options](#)” on page 338). You may also create the file with an editor or application program.

For instance, [FileList](#) is used to create a list of files to be backed up:

```
>filelist *.data:a (exec
>filelist a not *.data:a (10/1/98 exec append
```

A SELECTED.EXEC file now exists that lists all of the “data” files and all files that have been changed since 10/01/1998. The following command backs up these files:

```
>tbackup selected.exec tape (backup file
```

FULL

The modified bit of a file does not affect whether or not it is included in the data set. This is a default option that can be overridden by using the [DIFFERENTIAL](#) or [INCREMENTAL](#) option.

The modified bit is always reset when this option is in effect and the file is backed up.

INCREMENTAL Tells TBackup to include only those files that have their modified bit set. The modified bit is reset for each file copied. See “[Incremental Backups](#)” on page 566.

LABEL Specifies that the *to-drive*’s volume label is set to *label*. For instance:

```
>tbackup s f (backup label "Monday1"
```

sets the label of the diskette in drive F to “Monday1.”

If additional disk volumes are required, the last character of the label is incremented. When this might happen, try to use a starting label name that ends with a sequence identifier, such as “1.” Or use a label name with seven or less characters.

MULTIUSER Allows TBackup to back up files from a public drive even though other users may be logged on and active. Normally, when TBackup is instructed to perform a full-volume backup ([VOLUME](#) option) on a public disk, it requires single-user mode. If other users are logged onto the system, it displays the message: “Must be single user or private volume.”

Using this option tells TBackup to not restrict the backup to single-user operation (the message is still displayed). **THIS CAN BE EXTREMELY DANGEROUS!** If another user changes some files while the backup is being created, the integrity of the backup is lost.

Use this option only if you are sure that all other users are inactive and will remain so while the backup is created.

NOASK Disables the destination volume operator confirmation at the beginning of the backup. This option is useful for unattended backups.

NOEJECT A default option that specifies that the last volume in *backup* is not ejected when the backup is complete.

SETNAME Specifies the backup data set name. This name may be up to 64 characters in length and it may include letters, digits, spaces and other punctuation characters. Enclose the data set name in quotation marks.

```
>tbackup s tape1 (backup volume setname "Full system backup"
```

The name for a backup data set can be used to ensure that the proper backup is being used later when it is restored. When the data set is used at a later time, this data set name is displayed (unless **NOASK** is in effect). You can specify that the backup must have the same data set name as specified when you created it by using the **SETNAME** option with the **Mode 3** command.

SUBDIR Tells TBackup that files in the current working directory and all of its subdirectories are included in the backup.

When not specified, only the files in the current working directory are included. None of the files in subordinate directories are included in the backup.

VERIFY Indicates that, after the backup data set is created, TBackup should verify the readability of the data set. The **NOASK** option is in effect when the verify operation starts.

Verifying a data set does not test to see if the data set is an exact copy of the original file as the **COMPARE** option does. Instead, it does read every byte of the data set, verifies that all of the checksums are correct, and that every file and “record” in the data set starts and ends where it should.

VOLUME Specifies that *file-spec* refers to files in all accounts, not just the current account. The default **ACCOUNT** option limits file-spec to files on the current account.

This option requires a privilege level of five, that you be currently logged onto the system account, and that the source disk be a private disk volume or, if it is a publicly attached disk, that all other users be logged off or that the **MULTIUSER** option be used.

WAIT Indicates that, after the backup data set is created and the status message is displayed, TBackup should wait for an operator response before clearing the status message and exiting.

date1 The first token that looks like a date is interpreted as a selection date. Only files with a date stamp greater than or equal to this date are candidates for the backup. For instance:

```
>tbackup s tape2 (backup 1/1/98
```

With this command only those files on the s drive that have been changed on or since January 1, 1998 will be backed up.

See also the [INCREMENTAL](#) and [DIFFERENTIAL](#) options.

date2 A second token that looks like a date is interpreted as a selection date. Only files with a date stamp less than or equal to this date are candidates for the backup.

```
>tbackup s tape2 (backup 2/1/98 2/28/98
```

This command backs up only those files on the s drive that were changed between and including the beginning and ending of February, 1998. Files changed before February or since February are excluded.

Compare Options

ASK

This option operates the same as the [ASK](#) option on page 555.

DISKMAP

Changes the drive codes used during the comparison. For instance:

```
>tbackup f (compare diskmap s-d
```

The specification “s-d” in the above command specifies that files originally backed up from the s drive are to be compared with the files on the current D drive.

Multiple DISKMAP options may be used to map files on multiple drives:

```
>tbackup s a b tape (backup
```

```
>tbackup tape (compare diskmap s-d diskmap a-e diskmap b-f
```

Here, the files backed up from s are compared to files on the current D drive, files backed up from A are compared to files on the current E drive, and files backed up from B are compared to files on the current F drive.

EJECT

This option operates the same as the [EJECT](#) option on page 556.

ERROR

This option operates the same as the [ERROR](#) option on page 556.

LIST

Lists the files in the backup to the specified file or device. For instance, to list the files in the backup to a disk file, use:

```
>tbackup tape1 (compare list backup.listing:s
```

To list the files to a printer you would use:

```
>tbackup tape1 (compare list prt5
```

- NOASK** This option operates the same as the **NOASK** option on page 557.
- NOEJECT** This option operates the same as the **NOEJECT** option on page 557.
- PRT*nn*** This option is a synonym to the **LIST PRT***nn* option.
- SETNAME** This option operates the same as the **SETNAME** option on page 557.
- WAIT** Indicates that, after the backup data set is compared and the status message is displayed, TBackup should wait for an operator response before clearing the status message and exiting.

Restore Options

- ACCOUNT** Restores files matching the *file-spec* that were backed up from the current account. Compare with the **VOLUME** option.

```
>logon data
```

```
>tbackup tape (restore account
```

In the above example, only files that were originally backed up from the DATA account are restored.

- ASK** This option operates the same as the **ASK** option on page 555.
- DISKMAP** This option operates the same as the **DISKMAP** option on page 559.
- EJECT** This option operates the same as the **EJECT** option on page 556.
- ERROR** This option operates the same as the **ERROR** option on page 556.
- FROM** Tells TBackup to only select those files on the backup data set that were owned by account name *account* at the time the backup was created. The originating account name is specified immediately after the FROM keyword.

```
>logon develop
```

```
>tbackup tape (restore from project4
```


The above command restores all of the files that were backed up from the account “PROJECT4” into the current account “DEVELOP.”

LIST This option operates the same as the [LIST](#) option on page 559.

NEWFILE Specifies that TBackup will only attempt to restore a file if it does not already exist on the destination drive. This option is mutually exclusive with the [OLDFILE](#) option (you may use one or the other, but not both).

NOASK This option operates the same as the [NOASK](#) option on page 557.

NOEJECT This option operates the same as the [NOEJECT](#) option on page 557.

NOQUERY A default option that tells TBackup to not ask for confirmation on each new or existing file being restored.

To disable this option use the [QUERY](#) option.

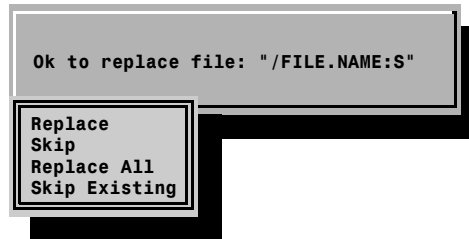
NOSYSFILES The standard, system-supplied files are omitted when files are restored. See “[NOSYSFILES Files](#)” on page 466 for a listing of the files skipped with this option.

OLDER Qualifying files are restored only if the backed-up file is older than the same file on the destination. In other words, files are restored only if they have been changed since the backup was made.

OLDFILE Specifies that TBackup will only attempt to restore a file if it does exist on the destination drive. This option implies the [REPLACE](#) option and is mutually exclusive with the [NEWFILE](#) option (you may use one or the other, but not both).

PRT*nn* This option is a synonym to the [LIST PRT](#)*nn* option.

QUERY Tells TBackup that the operator is to be “queried” or asked if each file matching the selection criteria is to be restored.



REPLACE This option tells TBackup that it is okay to attempt to restore a file even if it already exists on the destination drive. When this option is not used (and the **NOQUERY** option is not used), an attempt to restore an existing file causes you to be queried.

SETNAME This option operates the same as the **SETNAME** option on page 557.

VOLUME Specifies that all of the files on the data set may be restored if they match the *file-spec* and other options specified do not restrict them. Compare with the **ACCOUNT** option.

WAIT Indicates that, after the backup data set is restored and the status message is displayed, TBackup should wait for an operator response before clearing the status message and exiting.

date1 The first token that looks like a date is interpreted as a selection date. Only files in the backup data set with a date stamp greater than or equal to this date (new files) are candidates for restoring. For instance:

```
>tbackup tape a (restore 10/1/97
```

With this command only those files in the backup data set with a file change data of October 1, 1997 or later are restored to the A drive.

date2 A second token that looks like a date is interpreted as a selection date. Only files in the backup data set with a date stamp less than or equal to this date (old files) are candidates for the restore operation.

```
>tbackup tape a (restore 1/1/86 9/30/96
```

This command restores only those files from the data set with a file change date less than or equal to September 30, 1996, are restored to the A drive. The date 1/1/86 is the earliest date maintained by the THEOS file system and is interpreted as “from the earliest date.”

Verify Options	<u>ASK</u>	This option operates the same as the ASK option on page 555.
	<u>EJECT</u>	This option operates the same as the EJECT option on page 556.
	<u>ERROR</u>	This option operates the same as the ERROR option on page 556.
	<u>LIST</u>	This option operates the same as the LIST option on page 559.
	<u>NOASK</u>	This option operates the same as the NOASK option on page 557.
	<u>NOEJECT</u>	This option operates the same as the NOEJECT option on page 557.
	<u>PRT</u><i>nn</i>	This option is a synonym to the LIST PRTnn option.
	<u>SETNAME</u>	This option operates the same as the SETNAME option on page 557.
	<u>WAIT</u>	Indicates that, after the backup data set is verified and the status message is displayed, TBackup should wait for an operator response before clearing the status message and exiting.

Defaults	Mode 1 (BACKUP) defaults are: ASK , FULL , NOEJECT . ACCOUNT is a default when <i>file-spec</i> specifies a drive code only. NOASK is the default and cannot be overridden when <i>backup</i> is a file name specification.
	Mode 2 (COMPARE) defaults are: ASK and NOEJECT .
	Mode 3 (RESTORE) defaults are: ASK , NOEJECT , NOQUERY , VOLUME .
	Mode 4 (VERIFY) defaults are: ASK and NOEJECT .

Notes	Because of the compression algorithm used, some files may be larger than the original source file. For instance, a backup of a zipped file or a file that has already been compressed with the Compress command will be larger than the original file.
--------------	--

The compression percentage displayed refers to the difference in size between the original files and the resulting data set, including the data set

catalog information and other data set overhead. When the original size is small, this compression percentage may be greater than 100%.

Backup Data Set

The output of the TBackup command is a **backup data set**. This is a special stream file that contains the compressed forms of the files along with a catalog of the files on the data set. This data set can be manipulated like any other file, and it can even be opened and read like any other file. However, since it is a compressed form of the original files and it contains the directory entries for the files, it is not usable except by programs that know how to interpret this information. TBackup is the only supplied program that knows how to interpret the information and copy it back to its original, usable form.

A backup data set is a single file normally named TBACKUP.VOL00001. This single file contains all of the files included in this backup.

Frequently a file or set of files being backed up from hard disk to floppy or tape will be larger than a single diskette or tape. In that situation the TBackup command will ask you to mount another diskette or tape so that it can continue the backup of the file or files. In this situation each of the diskettes or tapes used will contain a single file named TBACKUP.VOLnnnnn.

Unless the **DIFFERENTIAL** option is used, the modified bit for each backed-up file is reset by the TBackup process.

TBackup Screens

TBackup, invoked with **Mode 1** (BACKUP) displays the following status screen while it is collecting information and backing up the files:

Backup Operation Status					
Time		Files	Bytes	Data Set Size	
Current: 15:45:27	Total:	6,902	131,849,179	Total:	0
Elapsed: 00:00:23	Current:	0	0	Volume:	0
Volume: 00:00:00	Skipped:	0	0	Total	Volume
			Compression:	0%	0%
Account: SYSTEM		Number: 0	Current Volume: 1		
File:					
Data Set Name					
Current Status: Selecting files... Please wait					
Progress: <div style="width: 0%; border: 1px solid black; display: inline-block;"></div> 0%					

The file counts do not include libraries or directories, only files.

TBackup, invoked with [Mode 2](#) (COMPARE), [Mode 3](#) (RESTORE) or [Mode 4](#) (VERIFY), displays screens similar to the following status screen while it is collecting information and comparing, verifying or restoring files:

Compare Operation Status						
Time		Files	Bytes	Data Set Size		
Current:	16:45:24	Total:	0	0	Total ECC:	0
Elapsed:	00:00:23	Current:	0	0	Volume ECC:	0
Volume:	00:00:00	Skipped:	0	0	Total Blks:	14,107
		Miscompare:	0	0	Volume Blks:	14,107
Account:						
File:						
Data Set						
Current Sta						
Mount the first disk of the data set in drive: "TAP1".						
Progress:						
0%						

The ECC counts refer to the number of errors detected using the error-correcting checksum fields in the data set.

Full Volume Backups

A full volume backup of a drive can be made easily with the TBackup command by using the [VOLUME](#) option and using a *file-spec* that specifies a drive-code only.

```
>tbackup s tape (backup volume
```

Full volume backups should be created on a frequent and periodic basis to assure yourself of having adequate protection in the case of disk or computer failure. Since a full volume backup contains a copy of every file, sub-directory and account on a disk, it is the only backup volume that needs to be accessed if you must restore a system.

Multivolume Backups

You can backup your entire system in one operation by performing a full volume backup for all drives on the system. For instance, a system with four hard drives can be archived with the command:

```
>tbackup s a b c tape (backup volume
```

An even more convenient method is to allow TBackup to backup all of the drives defined in the default search sequence. For instance:

```
>show search
```

```
SEARCH = SAB
```

```
>tbackup tape (backup volume
```

When the *file-spec* is omitted, TBackup uses the default search sequence as the *file-spec* specification. The above command is identical to:

```
>tbackup s a b tape (backup volume
```

Should the need every arise where you want to restore the files on this backup volume, use the restore mode of the TBackup command ([Mode 3](#)).

```
>tbackup tape (restore
```

When *file-spec* is omitted with the restore mode, TBackup restores all of the files in the backup volume to their original locations.

Differential Backups

A differential backup is a backup that includes only those files that have changed since the last full-volume backup. For instance, if a disk contains three files:

```
PROGRAM.COMMAND
DATA1.FILE
DATA2.FILE
```

A full-volume backup will create a backup data set that contains all of these files. If DATA1.FILE is changed and a differential backup is performed, it will create a backup data set that contains only that file. The other files are not included in this differential backup because they have not changed and there is a current copy of those files in the last full-volume backup.

If DATA2.FILE is then changed and another differential backup is performed, it will create a backup data set that contains both the DATA1.FILE and the DATA2.FILE because both of those files have been changed since the last full-volume backup was performed.

Full-volume	Differential 1	Differential 2
PROGRAM.COMMAND		
DATA1.FILE	DATA1.FILE	DATA1.FILE
DATA2.FILE		DATA2.FILE

Should a disk failure occur, the system could be restored by first restoring the full-volume backup and then the last differential backup.

Incremental Backups

An incremental backup is a backup that includes only those files that have changed since the last full-volume or incremental backup was created. For instance, in the example used for the differential backup, a full-volume

backup is performed and then DATA1.FILE is changed. When the incremental backup is performed, it creates a backup data set that contains only the DATA1.FILE. When DATA2.FILE is changed and another incremental backup is performed, it creates a backup data set that contains only the DATA2.FILE. The DATA1.FILE is not included in this backup because it has not changed since the last incremental backup was created.

Full-volume	Incremental 1	Incremental 2
PROGRAM.COMMAND		
DATA1.FILE	DATA1.FILE	
DATA2.FILE		DATA2.FILE

Should a disk failure occur, the system could be restored by first restoring the full-volume backup followed by the first incremental backup and then the second incremental backup.

Cautions

The [MULTIUSER](#) option tells the TBackup command to not check whether or not other users are logged on or are active. It does not prevent those other users from performing operations that change the database being backed up. If another user does change the database during the backup operation, the integrity of the backup data set is compromised.

For instance, a disk volume being backed up includes a customer master file with current balance fields and an invoice database. While the backup data set is being created, another user posts a transaction to the invoice database before it is included in the backup but after the customer master file is copied. If the backup data set is ever restored ([Mode 3](#)) or compared ([Mode 2](#)), the invoice database will not match the current balance fields in the customer master file.

Automated Backups

To use the TBackup command for automated backups, be sure that you have mounted the proper backup volume before the backup is initiated and use the following options:

- ▶ [COMPARE](#), to validate that the backup was accurate.
- ▶ [ERROR](#), to output the error messages to a file that can be checked after the backup is complete.
- ▶ [NOASK](#), to prevent the interruption of the backup process.
- ▶ [SETNAME](#), to put an identifying label in the backup data set.

Use the [ACCOUNT](#), [DIFFERENTIAL](#), [FULL](#), [INCREMENTAL](#), [SUBDIR](#) and [VOLUME](#) as appropriate.

Restrictions

When creating a backup data set ([Mode 1](#)), the destination media must be preformatted. If the backup requires more disks or tapes than anticipated, you can use another session or terminal to format the disk or tape while TBackup waits for you to confirm that the proper volume is mounted. To do this, the [ASK](#) option must be in effect and the destination drive must be publicly attached.

When creating a backup ([Mode 1](#)) with option [VOLUME](#) in effect and the drives specified in *file-spec* are publicly attached, all other users must be logged off or the [MULTIUSER](#) option must be specified. See “Caution” notes above regarding the [MULTIUSER](#) option.

The TBackup command requires a privilege level of four. To use the [VOLUME](#) option of the [Mode 1](#) (BACKUP) requires a privilege level of five.

See also

[Archive](#), [Backup](#), [Compress](#), [CopyFile](#), [Eject](#), [Expand](#), [Restore](#)

Tee Command Filter

The Tee command copies standard input to standard output and makes additional copies to a file.

TEE *file...* (*option*

file » file name with optional path

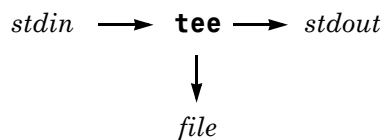
option » APPEND

Operation

The file from the standard input device is copied to both the standard output device and to *file*. If multiple *files* are specified, then multiple copies are made, one to each *file*.

```
>filelist s | tee file.list:s | more
```

The above command line creates a directory listing of the S drive. The listing is piped to the Tee command, which makes a copy of it in the file FILE.LIST:S and also pipes it to the More command, which displays the directory listing on the console.



Options

APPEND The output is appended to the end of *file*. Without this option, the output replaces *file*.

Notes

The *file* specified with this command may have a complete path specification, including the account name. However, when *file* is on another account, it is created using default attributes which include shared read and shared write protection. Therefore, the file is created but data cannot be written to it from non-owning accounts. Use the **CREATE** environment variable to set default attributes allowing shared file write access. See “**CREATE**” on page 102.

See also

[CopyFile](#)

THEO+COM Command

THEO+COM is a communications program that allows your computer to communicate with another computer by sending and receiving files and emulating a terminal to provide access to the other system as a console.

THEO+COM (options

options	»	ALF	HALF	SEND file	WYSE50
		ANSI	HANGUP	THEOS	WYSE60
		ASCII	MASTER	TRACE	XMODEM
		COMnn	NOEOT	TRACEFILE file	XMODEM-CRC
		CTL	NOLOGO	TVI	XMODEM-1K
		DIAL number	PCTERM	TYPE	YMODEM
		EOF=	RECEIVE file	VT100	
		EOT	REDIAL	VT220	
		FILES	SCRIPT file	VT220-8	

Command synonyms: COM, TERMINAL

Operation Invokes THEO+COM in terminal emulation mode. The remote system must be connected to the first communications port on your system.

Use command-line options or the built-in menu to perform any of the functions supported by THEO+COM or to change the configuration. The built-in menu is invoked by pressing **Break**, **M**.

Refer to the *THEO+COM Installation and User's Guide* for a complete description of the operation and usage of this command.

- Options**
 - ALF** Tells THEO+COM to add a line-feed at the end of every line transmitted. When this option is omitted, CR is used as the default End-of-Line character.
 - ANSI** Emulates an ANSI compatible terminal.
 - ASCII** Uses the ASCII protocol to send and receive files from the remote computer system.
 - C7** Synonym to the **TVI** option.
 - C55** Synonym to the **WYSE50** option.

<u>C58</u>	Synonym to the WYSE60 option.
<u>C90-C99</u>	Synonym to the PCTERM option.
<u>C180-C189</u>	Synonym to the PCTERM option.
<u>C100</u>	Synonym to the VT100 option.
<u>C210-C219</u>	Synonym to the PCTERM option.
<u>C220</u>	Synonym to the VT220 option.
<u>COMnn</u>	The currently attached <i>COMnn</i> is used for the communications port. When this option is not specified, the first attached COM device is used. Refer to the Attach and the Sysgen for information about attaching COM devices.
<u>CTL</u>	Sets control mode on. With control mode on every control character received in terminal emulation mode is displayed as two characters. For instance, Control-I is displayed as ^I.
<u>DIAL</u> <i>number</i>	Using the modem connected to the communications port, dials <i>number</i> and waits for a connection. After the connection is established, THEO+COM is exited, leaving the telephone connection open. DIAL must be the last option specified because characters following the word DIAL are treated as part of the number to dial. Refer to “ Dial ” on page 294 for information about <i>number</i> .
<u>EOF=</u>	Sets the character used to mark the end of a file transmitted with the ASCII protocol.
<u>EOT</u>	Used with the SEND file option to indicate that the end-of-transmission character is sent after all of the files are transmitted. This is a default option.
<u>FILES</u>	Can be used with the THEOS protocol and the SEND file option. It indicates that the file contains a list of files to be sent. This file is typically created by the FileList with its option FILES.
<u>HALF</u>	Uses half-duplex communications. Half-duplex is also called “Echoplex” because it causes every character transmitted to the remote system to be automatically echoed on your system. When this option is omitted, full-duplex communications is used.

<u>HANGUP</u>	Using the modem connected to the communications port, issue the hang-up command.
<u>MASTER</u>	Use master-mode duplex communications in terminal emulation. Each character typed is displayed on your screen. Each character received from the remote system is echoed back to that system.
<u>NOEOT</u>	Can be used with the THEOS protocol and the SEND file option. It indicates that no end-of-transmission character is sent after the last file is transmitted.
<u>NOLOGO</u>	Suppress the THEO+COM logo and copyright screen.
<u>NOTYPE</u>	Suppress the display of dialing messages and responses.
<u>PCTERM</u>	Emulate a PC Term or scan-code type keyboard and monitor. C90-C99 and C180-C189 may be used as synonyms to this option. C90-C99 selects a monochrome PC Term terminal emulation; C180-C189 selects a color PC Term terminal emulation.
<u>RECEIVE</u> <i>file</i>	Receive <i>file</i> from the communications port. The transmission protocol is specified with other options or it uses the default THEOS protocol.
<u>REDIAL</u>	Using the modem connected to the communications port, the last number dialed is redialed. After a connection is established, THEO+COM exits.
<u>SCRIPT</u> <i>file</i>	Execute the Modem Script Language file <i>file</i> . Refer to the <i>THEO+COM Installation and User's Guide</i> for a description of the Modem Script Language.
<u>SEND</u> <i>file</i>	Send <i>file</i> on the communications port. The transmission protocol is specified with other options or it uses the default THEOS protocol. <i>file</i> may contain wild cards.
<u>THEOS</u>	Use the THEOS protocol to send and receive files from the remote computer system.
<u>TRACE</u>	Enables file transfer tracing. A window is opened in the upper right corner of the display, showing the protocol activity during a transfer.
<u>TRACEFILE</u> <i>file</i>	Similar to the TRACE option, except the protocol activity is output to <i>file</i> .

<u>TVI</u>	Emulate a TeleVideo model 910 or model 925+ compatible terminal. C7 may be used as a synonym to this option.
<u>TYPE</u>	Display dialing and redialing messages and responses. This is a default option.
<u>VT100</u>	Emulate a DEC VT100 compatible terminal. C100 may be used as a synonym to this option.
<u>VT220</u>	Emulate a DEC VT220 compatible terminal in 7-bit mode. C220 may be used as a synonym to this option.
<u>VT220-8</u>	Emulate a DEC VT220 compatible terminal in 8-bit mode.
<u>WYSE50</u>	Emulate a Wyse model 50 compatible terminal. C55 may be used as a synonym to this option.
<u>WYSE60</u>	Emulate a Wyse model 60 compatible terminal. C58 may be used as a synonym to this option.
<u>XMODEM</u>	Use the XMODEM checksum, 256-byte protocol to send and receive files from the remote computer system.
<u>XMODEM-CRC</u>	Use the XMODEM CRC-16, 256-byte protocol to send and receive files from the remote computer system.
<u>XMODEM-1K</u>	Use the XMODEM-1K CRC-16, 1024-byte protocol to send and receive files from the remote computer system.
<u>YMODEM</u>	Use the YMODEM, CRC-16, 1024-byte protocol to send and receive files from the remote computer system.
Notes	The command name COM is a synonym to the THEO+COM command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file. If standard synonyms are disabled (see “ Account ” on page 156 and “ STDSYN ” on page 101), this synonym name may not be allowed.
Defaults	EOT , THEOS , TYPE
Restrictions	A communications port must be attached.

Touch Command

Sets the date and time-stamp on a file.

TOUCH *file* (*options*)

- file*» file name with optional path; may contain wild cards
- options*» DATE *file*₁
- NOTYPE
- TYPE
- date*
- time*

Operation The date and time-stamp on *file* is changed and the modified attribute is set.

If no options are used, the date is set to the current date and time.

```
>touch example.file
File "EXAMPLE.FILE:S" touched: 10/22/95 10:22:54
One file changed.
```

- Options** DATE *file*₁ Set the date and time of *file* to the date and time of *file*₁.
- NOTYPE Suppress the display of result messages and the summary line.

```
>touch example.* (notype
>
```

TYPE A default option that displays the results of each file changed and a summary line of the total number of files changed.

```
>touch example.*
File "EXAMPLE.BASIC:S" touched: 09/11/96 11:02:22
File "EXAMPLE.INDEXED:S" touched: 09/11/96 11:02:22
File "EXAMPLE.DIRECT:S" touched: 09/11/96 11:02:22
File "EXAMPLE.FILE:S" touched: 09/11/96 11:02:22
File "EXAMPLE.MAIL:S" touched: 09/11/96 11:02:22
File "EXAMPLE.DIRECT1:S" touched: 09/11/96 11:02:22
File "EXAMPLE.COMMAND:S" touched: 09/11/96 11:02:22
7 files changed.
```

date Set the date of *file* to *date*. If *time* is not specified, the time is set to 08:00:00.

```
>touch test.file (10/15/97
File "TEST.FILE:S" touched: 10/15/97 08:00:00
One file changed.
```

The earliest date that you can use in the THEOS file system is 1/1/86.

time Set the time of *file* to *time*. If *date* is not specified, date is set to the current date.

```
>touch test.file (15:32
File "TEST.FILE:S" touched: 09/11/96 15:32:00
One file changed.
```

Notes

A *file* specification can omit the file type if the environment variable [FILETYPE](#) is defined.

For more information about the [FILETYPE](#) variable, see “[Environment Variables](#)” on page 102.

The “modified” attribute for *file* is set with this command.

Defaults

The current date and time are used unless otherwise specified.

See also

[Change](#)

Tree Command

Displays the subdirectory tree.

Commands

1 **TREE** (*options*)

2 **TREE** *path* (*options*)

path » starting directory name; may not include account name

options » NOSORT
 PRT*nn*
 SIZE
 SORT

Operation **Mode 1**—Starting at the current working directory, the directory tree structure is displayed on the standard output device.

```
>pwd
/VERTICAL:S

>tree
/vertical
├── doc
│   └── programs
│       └── files
└── programs
```

Mode 2—Starting with *path*, the directory tree structure is displayed on the standard output device.

```
>pwd
/VERTICAL:S

>tree /package
/package
├── doc
└── programs
```


Options**NOSORT**

Causes the directory tree to be displayed in the sequence it is found on disk. No sorting of the directory names is attempted.

```
>tree (nosort
/
├── data
├── vertical
│   ├── doc
│   │   └── programs
│   │       └── files
│   └── programs
├── package
│   ├── doc
│   └── programs
└── misc
    ├── programs
    └── doc
```

Commands

PRT*nn*

Indicates that Tree is to print the current directory tree structure on the attached printer number *nn*.

The option keyword PRT may be specified as PRT, PRINT or PRINTER. As a convenience, PRINTER1 may be specified as P.

SIZE

Includes a count of the files in the subdirectory and the total disk space used by those files.

```
>tree (size
/ ..... (1186, 29012K)
├── data ..... (219, 17205K)
├── misc ..... (27, 77K)
│   ├── doc ..... (2, 16K)
│   └── programs ..... (23, 54K)
├── package ..... (248, 3162K)
│   ├── doc ..... (5, 4K)
│   └── programs ..... (240, 3150K)
└── vertical ..... (648, 8401K)
    ├── doc ..... (33, 62K)
    │   ├── files ..... (19, 27L)
    │   └── programs ..... (11, 27K)
    └── programs ..... (413, 4062K)
```

SORT

A default option that sorts the directory tree structure in alphabetical sequence.

```
>tree (size
/
├── data
├── misc
│   ├── doc
│   └── programs
├── package
│   ├── doc
│   └── programs
└── vertical
    ├── doc
    │   ├── files
    │   └── programs
    └── programs
```

Notes

The case mode used to display the directory tree is the same as your current CSI case mode as set in your account environment.

The line-graphics characters used to show the directory hierarchy are suppressed if the environment variable [LINEGRAPH](#) is set to "N."

Defaults

[SORT](#) is a default option.

Restrictions

You may only access directories in the current account.

Unique Command Filter

Unique copies standard input to standard output, or copies one file to another, omitting any duplicated lines.

1 **UNIQUE** *infile outfile* (*options*

2 **UNIQUE** *infile* (*options*

3 **UNIQUE** (*options*

infile » file name with optional path

outfile » file name with optional path

options » COUNT
 DUP
 UNIQUE
 +*nnn*
 nnn

Commands

The examples used in the descriptions that follow use an input file containing:

```
>list sample.file
```

```
The 1st line.
The 2nd line.
The 2nd line.
The 2nd line.
And the third line.
The last line.
The last line.
```

Operation

Mode 1—Each line in *infile* is examined and compared to the previous line. If it is different than the prior line, it is written to *outfile*.

```
>unique sample.file
```

```
The 1st line.
The 2nd line.
And the third line.
The last line.
```

Mode 2—This mode is identical to Mode 1 except that the output is written to the standard output device.

```
>unique sample.file > unique.lines
```

Mode 3—This mode is normally used in a pipe command-line because the input file comes from the standard input device and the output is written to the standard output device.

Options

COUNT Each line in the file is output preceded by a count of the number of consecutive instances of the line. The duplicated instances of a line are not output.

```
>unique sample.file (count
1 The 1st line.
3 The 2nd line.
1 And the third line.
2 The last line.
```

Option COUNT cannot be used in combination with the **DUP** or **UNIQUE** options.

DUP The **UNIQUE** command outputs only one copy of each duplicated line in the file. All unique lines in the input file are not copied to the output file.

```
>unique sample.file (dup
The 2nd line.
The last line.
```

UNIQUE This option causes the Unique command to output only the unique lines in the file. That is, only the lines that have no duplicate lines.

```
>unique sample.file
The 1st line.
The 2nd line.
And the third line.
The last line.

>unique sample.file (unique
The 1st line.
And the third line.
```

+nnn With this option the first *nnn* characters of each line are not used when testing for duplicate lines.

```
>unique sample.file (+7
The 1st line.
And the third line.
The last line.
```

nnn Specifies that the first *nnn* fields of each line are not used when testing for duplicate lines. A field is identified by a tab character or a space character.

```
>unique sample.file (2 count
4 The 1st line.
1 And the third line.
2 The last line.
```

Notes

An *infile* or *outfile* specification can omit the file type if the environment variable [FILETYPE](#) is defined.

For more information about the [FILETYPE](#) variable, see “[Environment Variables](#)” on page 102.

Restrictions

infile must be a stream file.

See also

[List](#), [Sort](#)

Unload Command

The Unload command unloads a program or data file from memory.

UNLOAD *program*

program » name of program or file to load into memory

Commands

Operation	<p>The <i>program</i> or data file is unloaded from memory. When <i>program</i> is a simple name, the following locations are searched:</p> <pre> SYSTEM.CMD32.<i>program</i> SYSTEM.TEOS32.<i>program</i> <i>program</i>.COMMAND </pre>
Notes	<p>If <i>program</i> is not currently loaded into memory, no message displays.</p> <p>If <i>program</i> is in memory but was not loaded during system startup or by Load, it is not unloaded by this command and no message displays.</p>
Restrictions	<p>The Unload command requires a privilege level of five.</p> <p>You may only Unload a program or data file from memory if it was loaded with Load or during system startup.</p> <p>If <i>program</i> is in use by another user, it is not unloaded at this time.</p>
See also	Load , Sysgen

Unnumber Command Filter

Unnumber copies a file to the standard output device, removing any line number from each line as it is copied.

1 UNNUMBER *file...*

2 UNNUMBER

file » file name with optional path

Commands

Operation

Mode 1—Each *file* is copied to the standard output device and unnumbered as it is copied. Specifying multiple *files* causes the second and remaining *files* to be appended to the *first* file copied to the standard output device.

```
>unnumber program1.basic
! Program: JULIAN Compute Julian date
! Programmer: Jane Doe
OPTION VERSION 1.1,"Copyright 1995 by ABC Software."
IF CMDARG$(1)=" "
GOSUB COMPUTE.JULIAN
ELSE GOSUB COMPUTE.DATE
IFEND
...

>unnumber program1.basic program2.basic
! Program: JULIAN Compute Julian date
! Programmer: Jane Doe
OPTION VERSION 1.1,"Copyright 1996 by ABC Software."
IF CMDARG$(1)=" "
...
! Program2: Compute late charge.
INPUT "beginning balance",AMOUNT
INPUT "month number",M%
CUR.MONTH% = VAL(LEFT$(DATE$(0),2))
TOT.TOT.INT = TOT.TOT.INT+TOT.INT
```

This command is frequently used in a pipe:

```
>unnumber numbered.text | tee some.text | more
```

Mode 2—Copies the file from the standard input device to the standard output device, unnumbering each line as it is copied.

Notes

A line is considered numbered if the first non-space character is a digit. A line is unnumbered by removing all leading spaces and digits. If the resulting line is blank, it is not output.

When a line contains multiple “line numbers,” only the first line number is removed. For instance:

```
10 10 This is line number 10.
```

The above line is unnumbered to:

```
10 This is line number 10.
```

Restrictions

file must be a stream file.

Compressed or encoded files, such as those used by MultiUser BASIC, cannot be unnumbered with this command.

See also

[Number](#)

Upcase Command Filter

Upcase copies a file to the standard output device, converting all letters to uppercase.

1 UPCASE *file*

2 UPCASE

file » file name with optional path; wild cards are not allowed

Command synonym: UC

Commands

Operation

Mode 1—*file* is copied to the standard output device with all lowercase letters converted to uppercase. The original *file* is unchanged.

```
>upcase system.theos32.crtcfg
SCREENSAVER=
MOUSE=2
KEYPADMODE=1
REPEATRATE=30
STATUSLINE=1
YESSCANCODE=0X15
...
```

```
>upcase system.theos32.crtcfg > crt.config
```

Mode 2—Copies standard input to standard output, converting all lowercase letters to uppercase. This form is normally used only in a pipe. However, if standard input is the console, records are copied until a **Ctrl+D** is encountered, signaling the end of the input file.

```
>filelist *.* | upcase > file.list
```

Notes

The command name UC is a synonym to the Upcase command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file. If standard synonyms are disabled (see “[Account](#)” on page 156 and “[STDSYN](#)” on page 101), this synonym name may not be allowed.

Restrictions

file must be a stream file.

See also

[Lowcase](#)

WhereIs Command

This command searches the directory tree of the current account looking for all instances of a file name.

WHEREIS *file...* (*options*)

file » file name with optional path; may contain wild cards

options » *date1*
 date2

Operation

Starting with the root directory of the current account, a search is made for all files that match *file*. If *file* specifies a path, the search then starts at that path. All subdirectories subordinate to the starting search directory are examined.

```
>tree /
/
├── data
├── abc
│   ├── doc
│   │   └── programs
│   │       └── files
│   └── programs
├── package
│   ├── doc
│   └── programs
└── misc
    ├── programs
    └── doc

>whereis *.*.check*
/ABC/ABC.CMD32.CHECKREG:S      Continue (Yes,No,Go)? Y
/ABC/PROGRAMS/ASCII.SOURCE.CHECKREG:S  Continue (Yes,No,Go)? Y
/ABC/PROGRAMS/PROGRAM.SOURCE.CHECKREG:S Continue (Yes,No,Go)? Y
/ABC/PROGRAMS/PROGRAM.SOURCE.CHECKMNT:S Continue (Yes,No,Go)? Y

>
```

When a file is found that matches *file*, the complete path to the file is displayed and you are asked: "Continue (Yes,No,Go) ?" The only valid responses accepted to this question are **Y**, **N** and **G**.

A **Y** response means that you want to continue the search; an **N** response means that you do not want to continue searching and you want to exit WhereIs without changing anything.

A **G** response means that you want to discontinue the search and you want to set your current working directory to the path of the file found.

Options

date1 Includes a file only if the file's last change date is greater than or equal to this date. That is, if the file was changed on or after this date.

This option may be used with the *date2* option.

```
>whereis *.*:s f (10/15/96
```

The above command will include in the search only those files that have been created or changed since October 14, 1996.

date2 Includes a file only if the file's last change date is less than or equal to this date. That is, if the file was changed on or before this date. May only be specified by first specifying the *date1* option.

```
>whereis *.*:s f (10/15/96 10/30/96
```

This command includes only those files that have been created or changed since October 14, 1996, but not any files that were created or changed after October 30, 1996.

To specify a *date2* when you don't care about *date1*, use a date of 1/1/86 for the *date1* option. This is the earliest date maintained by the THEOS file system.

```
>whereis *.*:s f (1/1/86 11/20/96
```

Here, since the *date1* specification is 1/1/86, only files created or changed prior to November 21, 1996, are included.

Defaults

Unless a path is specified with *file*, the search starts with the root directory of the current account.

Restrictions

You may only search for files in the current account.

See also

[ChDir](#)

Who Command

WhoAmI Command EXEC

The Who command displays all started user partitions and who is logged onto them. The WhoAmI command displays which account you are logged onto and other information about yourself.

- 1

WHO
- 2

WHO AM I
- 3

WHOAMI

Operation **Mode 1**—The Who command shows all partitions that have been started with consoles attached. For instance, the partitions used by the print spooler and disk cache programs are not displayed.

>who

THEOS® 32 Who List				
PID	Account	Program	Terminal	Attach Options
1	SYSTEM	HELP	CRT1:1	C90,L80,P24
2	SYSTEM	WHO	CRT1:2	C90,L80,P24
3	SAMPLES	CSI	CRT1:3	C90,L80,P24
4	PRIVATE	WINWRITE	CRT1:4	C90,L80,P24
5		LOGON	MULTI1	C180,L80,P24,B38400,PCXON/XOFF
6		LOGON	MULTI2	C180,L80,P24,B38400,PCXON/XOFF
10	JUDITH	CHECKREG	NET1	C210,L80,P24,REMOTE=Accounting

Mode 2—The Who Am I command displays the following information about yourself:

Account Name: SYSTEM
Logon Date: 20 June 1998, 09:52 AM
Pid: 10
Console: NET1 C210,L80,P24,REMOTE=Accounting
Server Name: ABC Corporation
Server Address: 192.168.100.1
Client Name: Accounting
Client Address: 192.168.100.3

The above information was produced when the user was using a network client as a user to the THEOS system.

Mode 3—This mode is synonymous with [Mode 2](#). It is actually an EXEC language program that invokes the Who Am I command. You could modify this EXEC program to supply other information.

Notes

If the environment variable [LINEGRAPH](#) is defined to be “N,” the line graphics are suppressed. For instance:

```
>who
 1 SYSTEM  HELP      CRT1:1    C90,L80,P29
 2 SYSTEM  CSI       CRT1:2    C90,L80,P29
 3 SAMPLES CSI       CRT1:3    C90,L80,P33
 4 SYSTEM  CSI       CRT1:4    C90,L80,P29
 5 SYSTEM  CSI       CRT1:5    C90,L80,P33
 6         LOGON     CRT1:8    C90,L80,P24
16 PRIVATE WHO       NET1      C210,L80,P28,REMOTE=DocSystem

>
```

Commands

See also

[Show](#)

Window Management Commands

These 18 commands are primarily used by EXEC language programs to provide window and session management control.

Commands

1	<u>wBypass</u>	<i>mode</i>
2	<u>wClear</u>	<i>window char</i>
3	<u>wClip</u>	<i>window clip</i>
4	<u>wClose</u>	<i>window</i>
5	<u>wClose</u>	ALL
6	<u>wFinish</u>	
7	<u>wFrame</u>	<i>window frame shadow attribute</i>
8	<u>wInvert</u>	<i>window invert</i>
9	<u>wMenu</u>	<i>Count Col Row Title Invert Invert Color Fg Bg Rfg Rbg</i> KEEP <i>window</i> HOT
10	<u>wMove</u>	<i>window col row</i>
11	<u>wOpen</u>	<i>window col row columns rows</i>
12	<u>wRefresh</u>	<i>window</i>
13	<u>wRemove</u>	<i>window</i>
14	<u>wRestore</u>	<i>window file</i>
15	<u>wSave</u>	<i>window file</i>
16	<u>wSelect</u>	<i>window update display</i>
17	<u>wStat</u>	
18	<u>wStat</u>	<i>window</i>
19	<u>wStat</u>	?
20	<u>wSwitch</u>	<i>switch</i>
21	<u>wSwitch</u>	<i>session</i>
22	<u>wTitle</u>	<i>window title top-bottom align attribute</i>

<i>align</i>	» LEFT CENTER RIGHT
<i>attribute</i>	» bit-mapped color and attributes <i>fg bg</i>
<i>bg</i>	» background color code or name (black, blue, green, cyan, red, magenta, yellow and white)
<i>char</i>	» character or character value
<i>clip</i>	» OFF ON
<i>col</i>	» leftmost column number
<i>columns</i>	» width of window (1–255)
<i>count</i>	» number of items in menu list
<i>display</i>	» TOP HIDDEN
<i>fg</i>	» foreground color code or name (black, blue, green, cyan, red, magenta, yellow and white)
<i>file</i>	» file name with optional path
<i>frame</i>	» NONE SINGLE DOUBLE RAISED SUNKEN
<i>invert</i>	» OFF ON
<i>mode</i>	» OFF ON
<i>row</i>	» topmost row number
<i>rows</i>	» height of window (1–255)
<i>session</i>	» session number (1–8)
<i>shadow</i>	» NONE LEFT RIGHT
<i>switch</i>	» OFF ON
<i>title</i>	» text for window title
<i>top-bottom</i>	» BOTTOM TOP
<i>update</i>	» OFF ON
<i>window</i>	» window number

wBypass The wBypass command enables or disables the window manager's bypass mode.

>wbypass off

Normally, window manager bypass mode is off, meaning that all characters sent to the console are intercepted by the window manager. Window manager saves the character and its attributes in the appropriate window in memory and transmits the character to the console if the window is selected and its update status is enabled.

>wbypass on

When window manager bypass mode is enabled, characters sent to the console are not intercepted by window manager. The character is displayed immediately on the console and it is not saved in any window in memory.

Invoking wBypass with no parameter displays the current window manager bypass mode.

>wbypass
Bypass mode is not set

wClear The wClear command clears the interior of an open window to a specified character or to spaces.

>wclear 5

>wclear 8 176

The first command clears window five to spaces. The second clears window eight, filling it with the “stipple pattern” character. Refer to Appendix J: “[THEOS Character Sets](#),” starting on page 761 for a list of characters supported by THEOS.

The window must be selected or refreshed to see the effects of this command.

wClip The wClip command changes the text clip attribute of a window. The clip attribute of a window controls whether text is truncated (clipped) or wrapped when it flows beyond the right edge of the window.

The initial or default clip attribute is ON, meaning that text is truncated.

wClose The **wClose** command closes and removes the specified window. If that window is the active window, then the next lower window or window zero is selected as the active window.

```
>wclose 4
```

The **wClose ALL** command selects window zero and then closes and removes all other open windows.

```
>wclose all
```

wFinish The **wFinish** command is synonymous with a **wClose ALL** command. It selects window zero and closes and removes all other windows.

wFrame The **wFrame** command changes an existing window's frame and shadow style. The window's frame and shadow are set to the indicated styles with the specified colors and attributes.

```
>wframe 3 single right
```

```
>wframe 4 double none white red
```

```
>wframe 30 single right 0x70
```

Both *frame* and *shadow* must be specified but the *attribute* or color of the frame can be omitted, in which case it is left unchanged. For a description of the attribute specification, refer to “[Frame & Title Attributes](#)” on page 149.

The changed frame and shadow styles and attributes are not displayed until the window is selected or refreshed, unless it is the current window.

wInvert The **wInvert** command changes the normal video/reverse video status of the indicated window. Although this command can be used on color or monochrome displays, it is only effective on monochrome displays. Windows should be defined with color attributes and invert attributes. The color attributes are ignored on monochrome displays and the invert attributes are ignored on the color displays.

```
>winvert 14 on
```

```
>winvert 23 off
```

The initial or default invert mode for a window depends upon its window number. Odd numbered windows default to invert ON, even numbered windows default to invert OFF.

wMenu

The **wMenu** command can only be used in an EXEC program because it requires the menu item text to be placed in the EXEC program's stack (**&BEGSTACK** or **&STACK** statements) before executing the command.

This command displays a menu or choice list on the screen and lets the operator select one of the items. The *count*, *col* and *row* parameters must be supplied but the other fields are optional.

```
&begstack
Item 1
Item 2
Item 3
&end

wmenu 3 5 10
```



Blank items in the list of menu choices display as a horizontal line. For instance:

```
&begstack
Item 1
Item 2
Item 3

Last Item
&end

wmenu 5 5 10
```



The window used by **wMenu** is always double-line framed and has a drop shadow on the right, space permitting.

col and *row* specify the upper-left corner of the interior of the window.

The width and height of the menu depend upon the items displayed. The width of the window is the larger of the longest item in the list and the length of the menu *title*. The height of the window is the smaller of the remaining screen depth minus 4 and *count*. When the window height is

less than *count*, a scroll bar is displayed on the right side of the menu frame.

INVERT Parameter. Use this parameter if the EXEC program may be used on a monochrome display. It has the same effect on the menu window as the [wInvert](#) command has on user-defined windows.

```
wmenu 10 3 3 invert on color white blue white red
```

COLOR Parameters. This parameter defines the colors used for the frame, title, menu item text and the selection highlight bar. It should be used if the EXEC program may be used on a color display.

```
wmenu 10 3 3 invert on color white blue white red
```

The *fg* and *bg* colors are used for the menu item text, the frame and the title text. *rvfg* and *rvbg* colors are used for the selection highlight bar.

When the COLOR phrase is not used and the display is capable of colors, the default colors are used for the menu window. Refer to “[Window Colors and Invert Status](#)” on page 148.

KEEP Parameter. This parameter performs two functions: It specifies the window number used for the menu and it tells wMenu to not close or remove the window when a selection is made.

When this parameter is not used the menu window is automatically closed and removed when the operator selects a menu item.

Note: The menu window is always opened or reopened when wMenu starts, even if it is the same window number and menu item list used the last time that wMenu executed.

HOT Parameter. The HOT parameter tells wMenu that menu items can be selected using “hot-keys.” When this parameter is used, each item in the menu list must specify the position of the character used for hot-key selection of the item. For instance:

```
&begstack
6 Item 1
6 Item 2
6 Item 3
&end

wmenu 3 5 10 HOT
```



In this list, the sixth character of each item is the hot-key character. Character positions are counted starting with the first non-space character following the hot-key character number. In the above example, the hot keys are the characters 1, 2 and 3, respectively.

Item Selection. When the menu displays, the selection highlight bar is positioned on the first item in the menu. You may use the **↑** and **↓** keys to move the highlight bar up and down. The **Home** and **End** keys move the highlight bar to the first and last items in the list.

The **Space** key advances the highlight bar to the next item. Unlike the **→** key, the **Space** key wraps from the last item to the first.

Pressing **Enter** selects the highlighted item.

You may also position to and/or select an item using a hot-key or soft hot-key. Hot-keys are enabled with the HOT parameter and are indicated with underlined characters in the item list. Soft hot-keys are enabled when the HOT parameter is not used.

When hot-keys are enabled, you may press an underlined character. This causes the highlight bar to move to the first item containing the underlined character. If there is only one item with that hot-key, the item is automatically selected just as if you had pressed **Enter**.

When soft hot-keys are enabled, pressing a character positions the highlight bar to the next item that starts with that character. If there are no more items starting with that character, then the highlight bar moves to the first item starting with the character.

Hot-key and soft hot-key selection is case-insensitive.

Selecting an item sets the return code (EXEC `&RETCODE` variable) to the selected item's number. Pressing `[Esc]` selects no item but exits `wMenu`, setting the return code to zero.

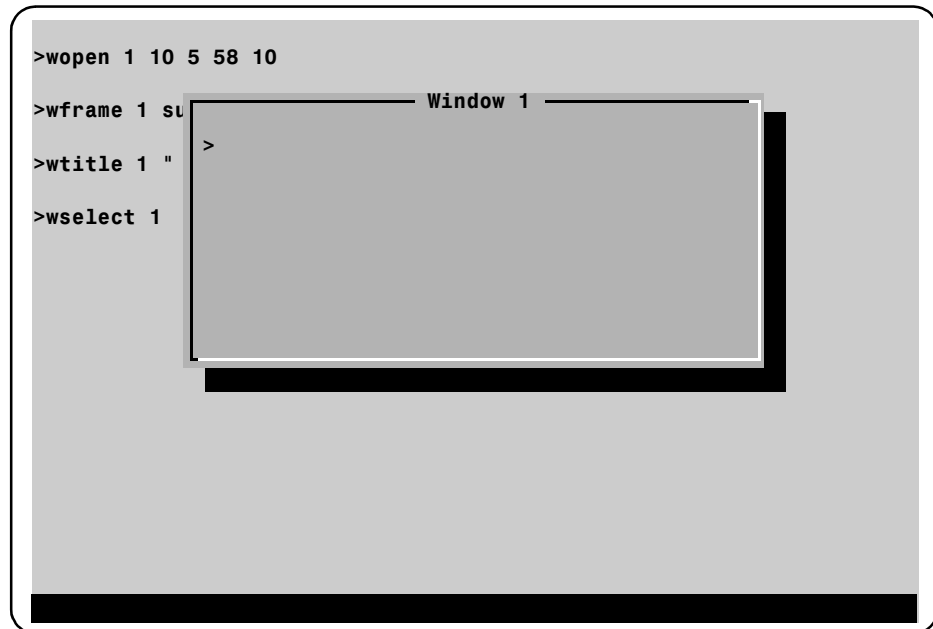
wMove

The `wMove` command moves a window to a new location on the screen. *col* and *row* are the column and row numbers of the new location for the upper-left corner of the interior of the window.

wOpen

The `wOpen` command defines a new window or redefines an existing window. *window* refers to the window number of the new or existing window. *col* and *row* are the column number and row number of the upper-left corner of the interior of the window (excluding the frame and shadow). *columns* and *rows* refer to the width and depth of the interior of the window.

```
>wopen 1 10 5 58 10
>wframe 1 sunken right
>wtitle 1 " Window 1 " top center
>wselect 1
```



The *window* may be specified with a question mark. When this is done, the next available window number is used for the new window. The window number is returned as the return code of the `wOpen` command.

This command does not select the new window, it only creates it in memory.

The initial attributes of a new or reopened window are: double-line frame with no shadow or title; clip on and update on. It has neither the hidden nor top attribute set (see [wSelect](#) command on page 600). If there is insufficient space around the interior of the window for a frame, then the window has no frame. The interior of a new or reopened window is empty with the cursor location at the upper-left corner.

The color and invert status of the window and its frame are dependent upon its window number, the capabilities of the console and the setup for session colors for this console. Refer to “[Window Colors and Invert Status](#)” on page 148 for more information about default colors and invert status.

wRefresh

The `wRefresh` command updates the display of a window on the console without making it the active window.

This command updates the display of the window interior, frame, title and shadow. It does not change the order of the window. That is, if the window is covered by another window, then the portion that is covered is not displayed. The [wSelect](#) command changes the order of display for a window.

wRemove

The `wRemove` command erases the display of a window from the screen. Any and all windows previously covered by *window* become visible.

This command does not close the window.

wRestore

The `wRestore` command retrieves a window definition and contents that were previously saved to disk with the [wSave](#) command.

```
wrestore 30 sample.copyright
```

The above command restores the window saved in the file `SAMPLE.COPYRITE` (see [wSave](#) command example).

A restored window has all of the attributes of the saved window including:

- ▶ Window location
- ▶ Window size
- ▶ Cursor location
- ▶ Frame style, attributes and color
- ▶ Shadow position
- ▶ Title position, alignment, attributes and color
- ▶ Clip status
- ▶ Invert status
- ▶ Display type (hidden, top or neither)
- ▶ Interior contents including text, attributes and color

The window is restored with this command but not displayed. You must use the [wRefresh](#) command or the [wSelect](#) command to display the new window on the screen.

wSave

The **wSave** command saves a window description and contents to a disk file.

```
wopen 1 46 4 30 3
wframe 1 double right 0x17
wclip 1 on
wselect 1 top
color white blue
&crt pon
&crt 7 1
&type SAMPLE Version 1.0 \
&crt 2 2
&type by ABC Software Corporation \
&crt 7 3
&type All rights reserved \
&crt poff
wselect 1 on top

wsave 1 sample.copyright
```

The preceding EXEC statements and commands create a window and fills it with a copyright notice for the program SAMPLE. This window definition is saved in the disk file SAMPLE.COPYRITE.

A window definition saved with the **wSave** command can be used by the [wRestore](#) command, the BASIC language WINDOW RESTORE statement or the C language **wRestore** function.

wSelect

The `wSelect` command selects a window as the active window for text display and operator input, and it defines the display order of the window.

For instance:

```
wselect 3 on
```

This command specifies that window number three is the active window, that it is refreshed and that all text written to the window will appear on the screen if the window is not overlaid by windows with the TOP attribute.

Selecting a window always makes it the active window. It also restores the display attributes in effect the last time that the window was selected. Specifically, the following attributes are restored: the cursor location, cursor on/off mode, video attributes (blink, underline, etc.) and normal and reverse video colors.

The first time that a window is selected, its cursor location is 1,1.

Selecting a window places it on top of all other windows except those that were last selected with the TOP display mode.

Window zero is always open and can be selected with this command but it cannot be specified as TOP or HIDDEN.

Update ON. Selecting a window with update mode ON means that the window is refreshed on screen and subsequent text written to this active window is displayed on the screen. This is the default mode when selecting a window.

Update OFF. Selecting a window with update mode OFF means that the window is not refreshed on screen. Text written to the window is not displayed on the screen but it is written to the window and will appear the next time that the window is refreshed or selected with update ON status.

Display TOP. Selecting a window with the TOP display attribute causes the window to be displayed on the screen on top of all other windows, including other windows marked as TOP. TOP implies an update ON status.

```
>wselect 3 top
```

```
>wselect 3 off top
```

The above two commands have identical effects: window three is selected as the active window, its display order is on top of all other windows, it is

refreshed on screen and subsequent text written to the window appears on the screen.

Display HIDDEN. Selecting a window with the HIDDEN display attribute causes the window to be removed from the screen. HIDDEN implies update OFF status. Although it is the active window, it does not appear on the screen until it is refreshed or reselected with update ON status.

A hidden window has all the properties of non-hidden windows except it is not visible on the screen until it is refreshed or selected without the HIDDEN attribute.

wStat

The wStat command displays the general status for all windows, the complete status for a specific window or it returns the currently active window number.

General Window Status. Using wStat with no parameters displays a brief summary of all open windows.

```
>wopen 1 2 2 70 20
>wopen 2 4 4 68 18
>wopen 3 6 6 66 16

>wstat
0* 80x24 beg=0,0 cur=0,11 order=0 update=1
1 70x20 beg=1,1 cur=0,0 order=-1 update=0
2 68x18 beg=3,3 cur=0,0 order=-1 update=0
3 66x16 beg=5,5 cur=0,0 order=-1 update=0
```

Each open window is listed with the following information:

- ▶ Window number. The active window is denoted with an asterisk.
- ▶ Window interior size using number base one.
- ▶ Window interior origin (upper-left corner) using number base zero.
- ▶ Current cursor location within the window using number base zero.
- ▶ Display order. An order number of -1 indicates that the window is not currently displayed on the screen. Either it has never been selected or refreshed, or it is HIDDEN.
- ▶ Window update status. A code of one means that text written to the window is displayed on the screen.

Display Current Window Number. When `wStat` is invoked with a question mark, it sets the return code to the current window number. This return code is easily used in an EXEC program by referencing the `&RETCODE` variable.

wstat ?

&type The current window number is &retcode
 &curr_window = &retcode

Specific Window Status. When `wStat` is given a window number, the complete status of that window is displayed.

>wselect 2

>wstat 2

```
status:          1
begin col,row:   3,3
width:           68
height:          18
curr col,row:    0,6
cursor:          0
frame-type:      2
shadow-type:     1
title-type:      2
title-align:     2
title-attr:      0x1C5F
frame-attr:      0x1C5F
clip:            1
update:          0
curr-attr:       0x1C5F
order:           0
color:           7,5,7,0
invert:          0
```

Status Element	Meaning	Codes Used
status	Window status	0 = Not open 1 = Open, not active 2 = Open, active
begin col,row	Window interior origin, base zero	
width	Window interior width, base one	
height	Window interior height, base one	
curr col,row	Cursor location, base zero	
cursor	Cursor display type	0 = Not displayed 1 = Blinking underline 2 = Blinking block 3 = Steady underline 4 = Steady block
frame-type	Frame style	0 = None 1 = Single line 2 = Double line 3 = Raised 4 = Sunken
shadow-type	Shadow style	0 = None 1 = Right 2 = Left
title-type	Title position	0 = None 1 = Top 2 = Bottom
title-align	Title alignment	0 = Center 1 = Left 2 = Right
title-attr	Bit-mapped value indicating the foreground and background title colors and the attributes.	
frame-attr	Bit-mapped value indicating the foreground and background frame colors and the attributes.	
clip	Interior text clipping	0 = Off 1 = On
update	Update and display status	0 = Update OFF 1 = Update ON 2 = Update ON, TOP 4 = Update OFF, HIDDEN

Status Element	Meaning	Codes Used
curr-attr	Bit-mapped value indicating the current window interior foreground and background colors and the attributes.	
order	Window display sequence	Lowest or bottom-most window is 0. A -1 means window is not displayed (hidden or has never been displayed).
color	Window interior color codes for fg, bg, rvfg, rvbg.	
invert	Monochrome invert status	0 = Off 1 = On
The bit-mapped values used for the title, frame and current attribute fields are not useful in an EXEC program. This same information is available by using the appropriate functions in a program. Refer to the language's reference manual for a description of these codes.		

wSwitch

The wSwitch command can enable or disable session-switching, or it can switch to another session on your console.

Session-switching is normally enabled. The wSwitch command can disable it if, for some reason, you do not want the operator switching to a different session while your EXEC program continues to execute.

```
wswitch OFF
```

This command disables session-switching.

```
wswitch ON
```

This command enables session-switching.

The wSwitch command can also switch to a different session running on your console. Merely enter the session number desired.

```
wswitch 4
```

Note that if you use wSwitch to switch to a session other than the one that this EXEC program is running on, you will not be able to see any display from this EXEC. It will continue to execute on its own session.

To determine which session you are using, use the wSwitch command with a question mark argument. This causes wSwitch to set the return code to the session number in use by this program.

```
wswitch ?  
&type My session is &retcode
```

wTitle

The wTitle command defines the title for a window. With this command you may either specify that a window has no title, or specify the title and its position and attributes. A window must have a frame in order to have a title.

```
>wtitle 3
```

The above command removes any title that window three might have.

```
>wtitle 2 " Window Two " bottom right
```

When a title is defined without the *top-bottom* specified, the default of TOP is used. When defined without the *align* parameter specified, the default of CENTER is used. The *attribute* for the title can be omitted, in which case it uses the default attributes of the frame for the title text. For a description of the attribute specification, refer to “[Frame & Title Attributes](#)” on page 149.

The changed title and attributes are not displayed until the window is selected or refreshed, unless it is the current window.

Restrictions [wOpen](#) and [wRestore](#) are the only window management commands that can create or define a new window. All other commands except [wStat](#) operate on existing, open windows.

See also Chapter 9 “[Windows](#),” starting on page [145](#)

WinWrite Command

The WinWrite command invokes the WindoWriter program, which is a general purpose, full-screen text editor.

WINWRITE *file... (options*

file » file name with optional path; may contain wild cards

options » NOBACKUP
 NOSHELL

Command synonym: WW

For a complete description of the operation and usage of this program, refer to the *WindoWriter User's Guide*.

Operation WindoWriter is loaded and the first *file* is opened as the current text file to edit.

The file specification may contain wild cards and there may be more than one file specified on the command-line. In either case, the first file is opened and, when you close that file, the next file is opened, and so on.

Options NOBACKUP This option specifies that any and all files saved during this session will not have a backup version created. This [NOBACKUP](#) option also applies to automatic file-saves performed by WindoWriter.

The [NOBACKUP](#) option, although it takes away a safeguard, is invaluable when disk space is at a premium. Editing a large file or a file that resides on a floppy diskette can use up large amounts of disk space, particularly when more than one level of backup is maintained.

NOSHELL This option disables the **CSI Shell...** item of the **File** menu of WindoWriter. This option would normally be used when WindoWriter is invoked from an application program. By specifying NOSHELL, the user is prevented from using WindoWriter to gain easy access to system commands that might cause problems for the application.

Note: The NOSHELL option only removes the **CSI Shell...** item from the **File** menu. The CSI Shell capability is still available with the shortcut keys.

Notes	<p>The command name WW is a synonym to the WinWrite command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file. If standard synonyms are disabled (see “Account” on page 156 and “STDSYN” on page 101), this synonym name may not be allowed.</p> <p>A <i>file</i> specification can omit the file type if the environment variable FILETYPE is defined.</p> <p>For more information about the FILETYPE variable, see “Environment Variables” on page 102.</p> <p>The keys used while in WindoWriter may differ from the keys used in other programs. See “Function Key Remapping” in the <i>WindoWriter User’s Guide</i>.</p>
Defaults	<p>The default configuration of WindoWriter is determined by the configuration file for WindoWriter. Each account may have their own configuration file. Refer to the <i>WindoWriter User’s Guide</i> for more information about this configuration file.</p>
Restrictions	<p><i>file</i> must be a stream text file.</p>
See also	<p>LineEdit, <i>WindoWriter User’s Guide</i></p>

WordCount Command

This command counts lines, words and characters in a stream file.

- 1 WORDCOUNT *file...* (*options*
- 2 WORDCOUNT *file...* (FILES *options*

<i>file</i>	»	file name with optional path; may contain wild cards
<i>options</i>	»	BYTES CHARS FILES LINES WORDS

Command synonym: WCOUNT

Commands

Operation

Mode 1—The file is analyzed and the total number of characters or bytes, words and lines is counted. The totals of these counts are displayed on the standard output device.

```
>wordcount system.theos32.devnames
```

Lines	Words	Chars	File-name
456	12,497	25,922	system.theos32.devnames

Omitting *file* means that the standard input device supplies the text of the file. This would normally be used in a pipe command-line. When the text comes from the console keyboard, it is terminated with a **Ctrl+D**.

```
>wc
```

```
The quick brown fox jumped over the lazy gray dog.
Now is the time for all good men to come to the aid
of their party.
```

```
Ctrl+D
```

Lines	Words	Chars	File-name
3	26	119	

Multiple files may be specified on the command-line. Each one of the files is analyzed separately and the totals are displayed. The total for all of the files is then displayed at the end.

```
>wordcount first.file second.file third.file
```

Lines	Words	Chars	File-name
55	1,127	5,635	first.file
120	2,500	10,389	second.file
32	600	1,357	third.file
207	4,227	17,381	Totals 3

Mode 2—In this mode *file* is an ASCII stream file containing one file description per line. Each file description in *file* is counted. The file descriptions may contain wild-card specifications.

This mode of the WordCount command is convenient when one or more sets of files are repetitively being counted. Merely edit a file containing the file description, such as:

```
>list daily.counts
system.history:s
*.log:s
```

```
>wc daily.counts (file
```

Lines	Words	Chars	File-name
13,398	200,690	862,356	system.history
250	2,435	10,389	private.log
3,029	4,398	25,942	user.log
16,677	207,523	898,687	Totals 3

Options

BYTES Count and display the total number of bytes in the file. This is synonymous with the **CHARS** option.

CHARS Count and display the total number of characters in the file. This is synonymous with the **BYTES** option.

```
>wordcount system.theos32.devnames (chars
```

Chars	File-name
25,922	system.theos32.devnames

LINES Count and display the total number of lines or records in the file.

```
>wordcount system.theos32.devnames (lines
```

Lines	File-name
456	system.theos32.devnames

WORDS Count and display the total number of words in the file.

```
>wordcount system.theos32.devnames (words
```

Words	File-name
12,497	system.theos32.devnames

Notes

The command name WCount is a synonym to the WordCount command. It is not a separate program but only an entry in the SYSTEM.TEOS32.SYNONYM file. If standard synonyms are disabled (see “[Account](#)” on page 156 and “[STDSYN](#)” on page 101), this synonym name may not be allowed.

A *file* specification can omit the file type if the environment variable [FILETYPE](#) is defined.

For more information about the [FILETYPE](#) variable, see “[Environment Variables](#)” on page 102.

Defaults

Unless a specific option is specified, [CHARS](#), [LINES](#) and [WORDS](#) are displayed by default.

Restrictions

file must be a stream file.

11 EXEC Job Control Language

The THEOS EXEC Job Control Language is a batch processing language with text display, data input and decision-making capabilities. As a batch processor, any command that can be executed from the command line, can be executed from within an EXEC program. This includes the commands described in Part II of this manual, along with programs provided by third parties or user-written programs.

■ Introduction

With an EXEC program, you can:

- ▶ Execute any command
- ▶ Display a menu and accept operator selection
- ▶ Display text on the screen
- ▶ Accept operator input
- ▶ Test variables or conditions and conditionally execute statements
- ▶ Use subroutines to make the EXEC program modular
- ▶ Use “for,” “while” and “until” looping structures
- ▶ Pause the program for awhile or wait for a specified time-of-day
- ▶ Check for the existence of a file
- ▶ Determine the size of a file or the amount of disk space remaining
- ▶ Access and use user environment variables

One of the EXEC processor’s most powerful features is the ability to perform command-line argument substitution. This capability is used extensively with EXEC programs that can be automatically created by the commands: [FileList](#), [GetFile](#) and [Look](#). For a description of this capability, refer to “[The EXEC and FILES Options](#)” on page [338](#).

Although the principal function of an EXEC program is to execute a command program, there are many features of the language that make an EXEC language program much more than a simple batch processor. The variables, functions and expressions that can be defined and used in an EXEC, along with its decision-making capabilities, make EXEC a true programming language, and not just a batch processor.

■ EXEC Language Concepts

Although powerful, the EXEC language is simple and straightforward.

■ EXEC Program Lines

An EXEC program is comprised of EXEC *statements*, EXEC *remarks* and CSI command lines. Each line of an EXEC program may contain one statement or one command or an EXEC remark.

Spaces and tab characters at the beginning of a line are ignored and do not affect the meaning of the statement or command.

The following is a simple EXEC program:

```

; PRINT.EXEC program

; This program uses the LIST command with the PRINT option to
; print a file on the primary printer.

; EXEC command syntax:
;
;     PRINT fn...

LIST &1 &2 &3 &4 &5 &6 &7 &8 &9 (PRINT
```

• EXEC Statements

All EXEC keywords and EXEC variables start with the ampersand character (&). Any line in an EXEC program whose first non-space character is an ampersand is processed as an EXEC statement.

• EXEC Remarks

An EXEC remark starts with the semicolon character (;). This is the same character used for remarks in the CSI. A line in an EXEC program whose first non-space character is a semicolon is treated as a remark and the remainder of the line is ignored.

Blank lines in a program are also treated as remarks. Blank lines are useful when trying to make the program source more readable.

Remarks may be added to the end of an EXEC statement or a CSI command in the program. All characters from a semicolon to the end of the line are ignored.

```

; This is a remark

&type Some text           ; remark text here

show users                 ; remark text here

```

In the above example, all of the remarks in the program are shown with boldface characters.

• CSI Commands

Any line in an EXEC program that does not start with an ampersand character or a semicolon character is assumed to be a CSI command line and is executed by the CSI.

■ Statement Labels

A label may be added to the start of any EXEC statement. A *label* is an alphanumeric string of characters, starting with a letter and ending with the colon character (:). Labels may contain letters, digits and underline characters. Label names are case-sensitive.

A label is used as an identifier for the [&CALL](#) and the [&GOTO](#) statements. Labels must be unique within a program.

```

display_args: ; Display command-line arguments

    &type Current index count: &index

    &for &i = 0, &index
        &type #&i = &&i
    &next

    &return

```

When referencing a label in a [&CALL](#) or [&GOTO](#) statement, the label is used exactly as it was defined, but without the terminating colon character.

```

&call display_args

```

■ Constants, Variables, Functions and Keywords

Every statement in an EXEC program is made up of constants, variables, functions and keywords. These elements are called *tokens*. In general, a token is a series of non-space characters

• Constants

An EXEC constant may be either numeric or alphanumeric. They are specified without delimiters (that is, they are not enclosed within parentheses).

A numeric constant is any series of digits with an optional leading minus sign and possibly one decimal point. Numeric constants may also be specified with “scientific notation.”

Any constant that does not start with a digit, minus sign or a period is an alphanumeric or string constant.

```
Company
12345
654.321
-23
123E2      same as      12300
```

Alphanumeric constants may be enclosed within a pair of double quotation marks. Doing so causes all of the characters between the quotation marks to be treated as a single string literal, including any embedded spaces.

• Variable Names

EXEC supports variables, either user-defined or command-line argument variables. All variable names start with an ampersand character (&).

User-defined variable names start with a letter and contain letters, digits and underline characters. User-defined variable names are case-sensitive. That is, the variable name &name is different than the variable name &Name or &NAME. A user-defined variable name cannot be the same as a function name or an EXEC keyword.

Some typical variable names include:

```
&name
&Address
&City_State_Zip
&total
&GRAND_TOTAL
```


• **Functions**

EXEC provides 13 predefined functions. These are described on page 663.

&CAT	&LEN
&DISKSIZE	&LIT
&ENV	&NULL
&EXIST	&RETCODE
&FILESIZE	&SUB
&INDEX	&SYSTEM
	&TYP

These function names are case-insensitive and can be used with uppercase or lowercase names. EXEC functions can be used in EXEC statements and CSI commands. However, the value of a function cannot be changed with the assignment statement described on page 629.

• **Keywords**

All EXEC keywords start with an ampersand character. EXEC keywords include all of the function names and all of the statement names.

&BEGSTACK	&EXIST	&REPEAT
&BEGTYPE	&FILESIZE	&RETCODE
&BREAK	&FOR	&RETURN
&CALL	&GOTO	&SKIP
&CAT	&IF	&SLEEP
&CONTINUE	&IFEND	&SPACE
&CONTROL	&INDEX	&STACK
&CRT	&LEN	&SUB
&DISKSIZE	&LIT	&SYSTEM
&ELSE	&LOOP	&TYP
&ELSEIF	&MENU	&TYPE
&END	&NEXT	&UNTIL
&ENV	&NULL	&VARY
&ERROR	&QUIT	&WAIT
&ESC	&READ	&WEND
		&WHILE

An EXEC keyword cannot be used as a variable name.

Command-line argument variable names are numbers in the range of 0 to 256. The value of command-line argument variables is initialized by the command line invoking the EXEC program or by the &READ ARGS statement. Their values may be changed by EXEC assignment statements and EXEC input statements (&READ).

When an EXEC is invoked, the command-line arguments are initialized to the values of the command line. For instance:

Note: When the CSI invokes an EXEC program, each of the parameters of the command line is changed to uppercase. Thus, the above command line is assigned to the command-line variables as:

After this initialization, the command-line variables may be reassigned mixed-case values. To initialize command-line variables with mixed-case values, enclose the arguments in quotation marks.

These references are only useful when used in the conditional-expression of the `&IF` statement. For instance:

618 *Command-Line Arguments*

■ Expressions and Operators

EXEC variables, constants and functions may be combined into expressions by using various operators. The terms of an expression may be surrounded by parentheses to indicate grouping and precedence. Terms within parentheses are analyzed before those terms outside of parentheses.

There are four types of expressions: string, numeric, relational and logical. The simplest expression is a constant.

• String Expressions

A string expression is an expression whose value is a string of characters. It uses only string variables, constants or functions as terms in the expression. There are three string operators that can be used in string expressions:

<code>\</code>	Concatenation
<code>[field]</code>	Subfield
<code>[from to]</code>	Substring

String Concatenation

The string concatenation operator appends one string onto another. For instance:

<code>abc \ def</code>	produces: abcdef
<code>abc \ def \ ghijkl</code>	produces: abcdefghijkl

When one string is concatenated onto another, no spaces are added.

Subfield

The subfield operator is used on strings that have the syntax of a file name. For instance:

<code>filename.filetype.mbrname[2]</code>	produces: filetype
---	--------------------

The general syntax for this operator is:

string[*field*]

The *string* may be a string constant or a variable containing a string. *field* is a numeric expression.

EXEC

The subfield operator may only be used in the right side of assignment statements.

When encountered, a string expression containing a subfield operator causes the *string* and *field* to be evaluated. Fields in the string are delimited by periods or colons.

For instance:

```
&filename = data.library.one:s

&fn = &filename[1]
&ft = &filename[2]
&mn = &filename[3]
&fd = &filename[4]

&type &fn, &ft, &mn, &fd
```

Displays:

```
data, library, one, s
```

Another example using the same string field:

```
&type The components of &filename are:
&for &field = 1, 4
    &a = &filename[&field]
    &type &field = &a
&next
```

Displays:

```
The components of data.library.one:s are:
1 = data
2 = library
3 = one
4 = s
```

Note that the above examples do not use the subfield operator in a **&TYPE** statement. When subfield operators are used in the **&TYPE** statement, they are treated as literals:

```
&type &filename[3]
```

Displays:

```
data.library.one:s[3]
```

Substring

The substring operator extracts a portion of a string. For instance:

 abcdefghijklmnop[2 5] produces: bcde

The general syntax for this operator is:

string[*from to*]

The *string* may be any expression, string or numeric. The *from* and *to* values are numeric values, either constants, variables or numeric expressions. The *from* and *to* values must be separated by one or more spaces. The value of *to* must be equal to or greater than the value of *from*.

The substring operator may only be used in an expression appearing on the right side of an assignment statement.

```
&month = &system date[1 2]
&type Today's date is &system date
&type The current month number is: &month
```

Displays:

```
Today's date is 02/04/1996
The current month number is: 02
```

• **Numeric Expressions**

A numeric expression is an expression whose value is a number. Numeric expressions use terms that are either numeric constants or variables containing numbers. There are six numeric operators that may be used in a numeric expression:

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo or remainder operation
-	Unary negative

Each term in the expression is evaluated. Only numeric values are accepted. A term that contains non-numeric characters is evaluated up to the first non-numeric character.

Using a string constant or variable in a numeric expression does not cause an error. However, the value of a non-numeric string is zero. For instance:

```
23 + item      produces: 23
23 + 12item    produces: 35
```

Numeric values may be either integers or floating point values.

The addition, subtraction, multiplication and division operators perform the same operations as expected. Unless parentheses are used to change the order of evaluation, multiplication and division are performed first, and then addition and subtraction (see “[Expression Evaluation and Operator Precedence](#)” on page 625).

The rules of arithmetic state that dividing a number by zero is invalid. When attempted, the result is 9.999999999999999E+126, which is the largest number that can be produced.

The modulo operator (%) produces the remainder of one integer divided by another integer. For instance:

```
15 % 4          produces: 3
```

The modulo operator only operates on two integers. When either of the values is a floating point value, the result is always the value of the second field. For instance:

```
2.3 % 2          produces: 2
```

• Relational Expressions

A relational expression is an expression that compares two values, producing either a true result or a false result, depending upon the relationship tested. There are six relational operators that can be used in an expression:

=	or	EQ	equality
!=	or	NE	inequality
>	or	GT	greater than
<	or	LT	less than
>=	or	GE	greater than or equal
<=	or	LE	less than or equal

The operators are case-insensitive. That is, EQ is the same as eq, Eq or eQ.

Two strings can be compared or two numeric values can be compared. Although you can compare a string to a number, the number is treated as a string of characters. For instance:

```
&field1 = text
&field2 = 34

&relationship = &field1 > &field2           produces:   true
```

In the above expression, **&field1** is greater than **&field2** because the character “t” is greater than the character “3.”

Relational expressions produce true or false results. Numerically, a false result has the value zero; a true result has a value of one.

A relational expression can be a single term. For instance:

```
&if &index
    &type Command arguments present
&else
    &type No command arguments
&ifend
```

The **&index** term in the above example is a simple, relational expression. In this situation, the numeric value of the term is the value of the relational expression. Non-zero values are true; a zero value is false.

Unlike the BASIC language, when the single term contains a non-numeric string, EXEC treats it as the number zero.

Relational expressions are normally only used in the **&IF**, **&ELSEIF**, **&UNTIL** and the **&WHILE** statements. Less often, they are also used in assignment statements.

A logical expression performs a Boolean or logical evaluation of one or more terms. This evaluation is performed on the binary, integer value of the terms.

~	Bitwise NOT
&	Binary AND
	Binary OR
^	Binary exclusive OR (XOR)

```
23 = 000000000000000000000000010111  
15 = 000000000000000000000000001111
```

23 & 15 = 00000000000000000000000000000000111 = 7

NOT	AND	OR	XOR
$\sim 1 = 0$	$0 \ \& \ 0 = 0$	$0 \ \ 0 = 0$	$0 \ \wedge \ 0 = 0$
$\sim 0 = 1$	$1 \ \& \ 0 = 0$	$1 \ \ 0 = 1$	$1 \ \wedge \ 0 = 1$
	$0 \ \& \ 1 = 0$	$0 \ \ 1 = 1$	$0 \ \wedge \ 1 = 1$
	$1 \ \& \ 1 = 1$	$1 \ \ 1 = 1$	$1 \ \wedge \ 1 = 0$

Table 30: Logical Operator Truth Tables

The result of a logical expression is a numeric, integer value. This value is frequently used as a simple, relational expression.

■ Expression Evaluation and Operator Precedence

When an expression contains more than one term, the expression is evaluated in a predefined order that is called precedence. The rules of evaluation of expressions are:

- Any variables in the expression are evaluated recursively, in a right-to-left sequence. That is, when a term contains an ampersand character, the rightmost portion of the term is expanded. After the term is expanded, the term is examined again for any ampersand characters. If any are found, the term is expanded again, *etc.* For instance:

<code>&n = 2</code>		
<code>&array2 = text</code>		
<code>&array&n</code>	expands to:	<code>&array2</code>
<code>&array2</code>	expands to:	<code>text</code>

- When the operators are of equal precedence, EXEC performs the operations in the order of their occurrence, from left to right. Otherwise, the operations are performed in order of their precedence.

For example, in the expression `3 * 2 / 4.5`, the multiplication and division operators have equal precedence. This expression is evaluated in a left-to-right manner: the product of `3×2` is divided by `4.5`.

- Parentheses are assigned the highest precedence. Any term or expression inside of the parentheses is evaluated first, before any term or expression which is not within the parentheses. When nested parentheses are encountered, EXEC works from the inside out: the terms inside the innermost pair of parentheses are evaluated first, then those in the next outward layer, *etc.*

For example, in the expression `2 * (3 + (2 / 4.5))`, the quotient of `2÷4.5` is added to `3` and this sum is then multiplied by `2`.

EXEC

The operators and their precedence are:

Operator	Description
<i>variable</i>	Variable
()	Grouping
[<i>field</i>] [<i>from to</i>]	Subfield and substring
functions	Predefined EXEC functions
- ~ +	Unary negative, unary positive, complement
\	String concatenation
* /	Multiplication and division
+ -	Addition and subtraction
= != >	Relational operators
< >= <=	
~	Bitwise NOT
&	Binary AND
^	Binary eXclusive OR
	Binary OR

Table 31: EXEC Operator Precedence

■ EXEC Input Stack

The EXEC language is primarily a batch processing language. The general goal of a batch processor is to automate the execution of one or more commands. Since some commands are designed to request data supplied by an operator, an EXEC program needs some method of providing that data as if it had come from the operator. The EXEC stack provides this capability.

Although most THEOS-supplied commands have a command-line option (NOASK, NOQUERY) that can suppress operator interaction, programs provided by third parties or user-supplied programs may not. For instance, a program that prints a check register may need to know the starting and ending dates for the register report.

```
>checkreg

Enter starting date: 01/01/1996

Enter ending date: 12/31/1996
```

To automate this type of program in a batch-processing EXEC program, the replies to these questions need to be supplied by the EXEC program input stack.

Data placed on the stack is always accessed by a program when it requests input from the standard input device (stdin). Only when there is no data present on the stack does a program wait for input to come from the standard input device.

To automate the previous example, the EXEC program needs to place the two responses to the input requests on the stack:

```
&stack 01/01/1996
&stack 12/31/1996
checkreg
```

The EXEC input stack is a “first-in, first-out” buffer (FIFO). This means that the first item stacked is the first item read by the next standard input device request.

The EXEC input stack is always cleared upon the completion of any program. To execute two programs in sequence, the responses for the second program must be stacked after the first program terminates.

For instance, to execute the checkreg program twice:

```
&stack 01/01/1996
&stack 06/30/1996
checkreg
&stack 07/01/1996
&stack 12/31/1996
checkreg
```

There are two statements that can add data to the EXEC input stack: the [&BEGSTACK](#) statement described on page 629 and the [&STACK](#) statement described on page 656. An EXEC program can access the input stack just like any other program. The [&READ](#) statement requests input from the standard input device. If the EXEC program has placed data on the stack, the [&READ](#) statement will read it.

Note: Data on the EXEC input stack is accessed by any program requesting input from the console keyboard. When the standard input device is the console keyboard (default), programs requesting input from the standard input device will read the data from the stack. If there is data on the EXEC input stack, there is no way for a program to bypass this stacked data and get data from the keyboard.

EXEC

■ Creating an EXEC Program

EXEC programs are created either automatically, by the [FileList](#), [GetFile](#) or the [Look](#) commands, or manually by using a text editor. Although there is no editor designed specifically for creating EXEC language programs, the WinDoWriter text editor (see “[WinWrite](#)” on page 607) can perform syntax coloring when it knows that the file is an EXEC program. Refer to the *WinDoWriter User’s Guide* chapter on “Formatting Documents” for information about enabling this feature.

The normal name for an EXEC program is a name with a file-type of EXEC. However, you may make the program a member of the SYSTEM.COMD32 library or a private command library.

EXEC programs that are not a member of a library do not have to have a file-type of EXEC. However, if an EXEC program does not have a file-type of EXEC, then it is only executed when both the file-name and the file-type are specified. With a file-type of EXEC, only the file-name is necessary as the operating system searches for programs with the EXEC file-type in its default search sequence. See “[Command Search Sequence](#)” on page 48 for a complete description of the search rules and defaults.

■ EXEC Statements

All EXEC statements are lines of text whose first non-space character is an ampersand character (&). Any line that does not start with the ampersand character is either a remark (starts with a semicolon), a label (token terminated with a colon) or a command line to be executed.

Assignment

The assignment statement defines a value for a variable.

```
&variable-name = expression
```

Every assignment statement contains an equal sign (=) with a single variable name to the left and an expression to the right of the equal sign.

variable-name may be any user-defined variable name as described on page 616 or it may be a command-line argument variable name as described on page 618. The *variable-name* may not be an EXEC function. No error is reported when the *variable-name* is invalid, but the statement is ignored.

expression may be a constant, a variable-name reference, an EXEC function, or any combination of these. For instance:

```
&3 = 1
&end = &start
&start_time = &system tod
&end_time = &system tod - &start_time
```

&BEGSTACK

The &BEGSTACK statement defines the start of one or more lines of text that is to be placed on the EXEC input stack.

```
&BEGSTACK
text...
&END
```

There must be a matching &END statement for each &BEGSTACK statement. All lines of text between a &BEGSTACK statement and its matching &END statement are placed in the EXEC input stack. Every line of *text* is stacked with a terminating carriage-return character.

The lines of *text* are treated as literal text and any variable-name references or expressions are not evaluated. Variables and expressions may be added to the stack with the &STACK statement described on page 656.

Any leading or trailing spaces in the line are part of the text. However, when the stack is accessed by the **&READ** statement, the leading spaces are ignored.

The data placed in this EXEC input stack is accessed by **&READ** statements in this EXEC program or by programs invoked by this EXEC program. Data placed on the stack is accessed by a program that requests input from the console keyboard.

The stack and its usage are discussed on page 627.

```
&begstack
input
First line of new file
some text to be added to file
last line of text

file
&end
lineedit new.file
```

The above program stacks some text and then invokes **LineEdit**. The **LineEdit** command expects its input to come from the standard input device. If the standard input device is the console keyboard, and since there is data stacked, the command retrieves that data as if it had come from the keyboard. The data tells **LineEdit** to begin input mode, and gives it three lines of text to add to the file. The stack has a blank line that tells **LineEdit** to terminate input mode. The last stacked line tells **LineEdit** to save the file to disk and to exit.

See also “**&STACK**” on page 656.

&BEGTYPE

The **&BEGTYPE** statement operates similarly to the **&BEGSTACK** statement just described, except that the lines of text are displayed on the console instead of being added to the EXEC input stack.

```
&BEGTYPE
text...
&END
```

The lines of text between the **&BEGTYPE** and the matching **&END** statement are treated as literal text and displayed on the console. Each line of text is displayed on a separate line. Any leading spacing in text is preserved and displayed. The text is not evaluated. Variables and expressions may be displayed with the **&TYPE** statement.

&begtype

```
Sample Program
(c) 1996 by ABC Software
```

&end

```
&type Enter command \
&read &cmd
```

Displays:

```
Sample Program
(c) 1996 by ABC Software

Enter command ?
```

See also “&TYPE” on page 657.

&BREAK

The &BREAK statement is used in a looping control structure to exit the loop. It does so by transferring control to the statement following the loop-terminating statement.

&BREAK

A &BREAK statement inside of a looping structure transfers control to the statement following the loop-terminating statement, thus causing the loop to be exited. For instance:

```
&for &i = 1, &index
  &if &i = (
    &call get_option
    &break
  &ifend
  &call save_filename
&next
```

In this example, a loop structure examines each command-line option. If the argument is an open parenthesis character, the argument is processed by the `get_option` subroutine and the loop is exited with the &BREAK statement. If the argument is an open parenthesis character, it is processed with the `save_filename` subroutine and the loop is repeated.

&CALL

The **&CALL** statement invokes a subroutine in the EXEC program.

&CALL *label*

The **&CALL** statement transfers control to the subroutine starting with the identifier *label*. This label must exist somewhere in this EXEC program.

During execution of the subroutine, when the **&RETURN** statement is encountered, control is transferred back to the statement following this **&CALL** statement.

Subroutines may use “nested calls.” That is, one subroutine may call another subroutine which calls another subroutine, *etc.* As each **&RETURN** statement is encountered, the current subroutine is exited with control returning to the statement following the **&CALL** statement that invoked it.

EXEC

```

&call display_args
&type Accept new command line
&read args
&skip 1
&call display_args
&quit

display_args: ; Display command-line arguments

&type Current index count: &index

&for &i = 0, &index
    &type #&i = &&i
&next

&return

```

The above program performs two calls to the subroutine `display_args`. The subroutine is terminated with the **&RETURN** statement. After the subroutine executes the first time, control returns to the **&TYPE** statement following the call to the subroutine. After executing the second time, control returns to the **&QUIT** statement following the call to the subroutine.

&CONTINUE The &CONTINUE statement is used in a looping control structure to cause the remainder of the statements in the loop to be skipped and for the next iteration of the loop to be started.

&CONTINUE

A &CONTINUE statement inside of a looping structure transfers control to the loop-terminating statement, thus causing the loop to be repeated. For instance:

```
&for &i = 1, &index
  &if &&i[1] = -
    &call get_option
    &continue
  &ifend
  &call save_filename
&next
```

In this example, a loop structure examines the first character of each command-line option. If that first character is a minus sign character, the argument is processed by the `get_option` subroutine and the remainder of the loop is skipped with the `&CONTINUE` statement. If the argument does not start with the minus sign, it is processed with the `save_filename` subroutine.

EXEC

&CONTROL The &CONTROL statement defines or changes various operating parameters for the EXEC program.

&CONTROL <i>option...</i>			
<i>option</i>	»	AUTO CR <i>nn</i>	OFF
		CASE M	ON
		CASE U	PROMPT <i>string</i>
		NOSTACK	STACK
		NOTRACE	TRACE

The &CONTROL statement is an executable statement. This means that the control options specified take effect only when the statement is executed.

You may specify multiple control options on the same line.

For instance:

```
&control on stack case U trace prompt Data:
```

The above statement is the same as:

```
&control on
&control stack
&control case U
&control trace
&control prompt Data:
```

AUTO CR When **&READ** statements are executed, input is automatically accepted as if a carriage-return were pressed, when *nn* characters are typed.

```
&control autocr 3
&type Enter text \
&read &reply
```

Displays:

```
Enter text ? xxx
```

As soon as the third character is typed, the text is accepted and assigned to the variable **&reply**.

The automatic carriage-return can be disabled by specifying a zero or blank for the *nn* value.

CASE M Sets the input case mode for **&READ** statements to mixed case. Characters are accepted in the same case mode as typed.

CASE U Sets the input case mode for **&READ** statements to uppercase. Characters typed are always accepted as their uppercase equivalent. This is the default or initial status for each EXEC program.

NOSTACK Specifies that text read from the EXEC input stack is not to be displayed on the console. This is the opposite of the default **STACK** control.

NOTRACE Specifies that the EXEC program source lines are not displayed as they are executed. This is the default or initial status for each EXEC program.

OFF

Specifies that when commands are executed by this EXEC program, the command line is not displayed on the console. This is the default or initial status for each EXEC program.

&control off

```
&type Ready to execute command.
show who
```

When executed, displays:

```
Ready to execute command.
ACCOUNT  = SAMPLE
USERNUM  = 9
PORT     = 1
PRIVLEV  = 5
LOGON    = 12:29:01 01-30-96
```

Compare this display with the **&CONTROL ON** example.

ON

Specifies that when commands are executed by this EXEC program, the command line is displayed on the console preceded by the current CSI prompt string (normally, this is the greater-than symbol).

&control on

```
&type Ready to execute command.
show who
```

When executed, displays:

```
Ready to execute command.

>show who
ACCOUNT  = SAMPLE
USERNUM  = 9
PORT     = 1
PRIVLEV  = 5
LOGON    = 12:29:01 01-30-96
```

Compare this display with the **&CONTROL OFF** example.

PROMPT The text following the keyword **PROMPT** is set as the prompt string displayed when a **&READ** statement executes. The default prompt string is a single colon character (:).

```
&type Using default prompt string
&read &ans
&control prompt Data?
&type Using custom prompt string
&read &ans
```

Displays:

```
Using default prompt string
:first
Using custom prompt string
Data?second
```

You can define an empty prompt string:

```
&control prompt
&type Enter reply now \
&read &reply
```

STACK Specifies that text read from the EXEC input stack is displayed on the console, just as if it were typed. This is the default or initial status for each EXEC program.

TRACE Specifies that the EXEC program source lines are displayed as they are executed. The source lines are distinguished from the normal program display by enclosing each source line in angle brackets.

```
&control trace
&a = 1
&b = 3 + &a
&type &b
```

When executed, displays:

```
<&a = 1>
<&b = 3 + &a>
<&type &b>
4
```

&CRT

The &CRT statement performs cursor movement and screen editing on the console.

1	&CRT	col row		
2	&CRT	video-attribute		
3	&CRT	cursor-control		
<hr/>				
	video-attribute	»	BON FON KON MON PON RVON SON ULON	BOFF FOFF KOFF MOFF POFF RVOFF SOFF ULOFF
	col	»	screen column number	
	cursor-control	»	BELL CLEAR DC DL DOWN	EOL EOS EU HOME IC IL LEFT RIGHT TAB UP
	row	»	screen row number	

EXEC

Mode 1—Positions the cursor to the indicated screen location. *col* and *row* are specified relative to the upper-left corner of the screen. The upper-left corner is addressed with 0 0.

Mode 2—Sets the video attribute for subsequent text display. Some of the *video-attributes* may not be supported on your console. If they are not, the class code for the console is probably coded to ignore the request.

The meaning and display of the various *video-attribute* codes are described in Chapter 4 “[Consoles, Keyboards and Mice](#)” in [Table 7: “Console Display Attributes”](#) on [page 73](#).

Mode 3—Positions the cursor or performs the indicated screen edit command.

<i>cursor-control</i>	Meaning
BELL	Sounds the terminal's alarm.
CLEAR	Clears the entire screen and positions the cursor to the upper-left corner.
DC	Deletes the character under the cursor, shifting all of the remaining characters on that line to the left.
DL	Deletes the line that the cursor is on, shifting all of the remaining lines on the screen up one line.
DOWN	Moves the cursor down one line without changing its column position.
EOL	Erases all characters from the cursor position to the end of the line.
EOS	Erases all characters from the cursor position to the end of the screen.
EU	Erases all unprotected characters on the screen.
HOME	Positions the cursor to the upper-left corner of the screen.
IC	Inserts a space character at the cursor location. The characters to the right of the cursor location are shifted right one character position. The cursor location does not change.
IL	Inserts a blank line before the current line. The lines below the current line are shifted down one line with the last line scrolling off of the screen.
LEFT	Moves the cursor to the left one column position.
RIGHT	Moves the cursor to the right one column position.
TAB	Moves the cursor to the next eight-column tab stop.
UP	Moves the cursor up one line.

Table 32: EXEC &CRT Video-controls

&ELSE

The **&ELSE** statement is used in a multiline **&IF** statement.

&ELSE

A multiline **&IF** statement may have an **&ELSE** statement clause. When it does, the **&ELSE** statement marks the end of the statements that are executed when the **&IF** statement *conditional-expression* is true and marks the beginning of the statements executed when the *conditional-expression* is false.

There may be only one **&ELSE** statement defined between an **&IF** statement and its matching **&IFEND** statement.

Refer to the “**&IF**” on page 644 for an example.

&ELSEIF

An **&ELSEIF** statement is used in a multiline **&IF** statement.

&ELSEIF *condition-expression*

A multiline **&IF** statement may have one or more **&ELSEIF** statements. Each **&ELSEIF** statement defines a condition to be tested if the preceding condition tested false.

The **&ELSEIF** statements must be defined prior to any **&ELSE** statement used in the **&IF** statement structure.

Refer to the “**&IF**” on page 644 for an example.

&END

The **&END** statement marks the end of text literal lines used by the **&BEG-STACK**, **&BEGTYPE** and **&MENU** statements.

&END

See “**&BEGSTACK**” on page 629, “**&BEGTYPE**” on page 630 and “**&MENU**” on page 648.

&ERROR

The **&ERROR** statement specifies an action to be taken if any command executed by this EXEC program completes with a non-zero return code.

- 1 **&ERROR** *statement*
- 2 **&ERROR**

Whenever an EXEC program executes a command, the return code set by that command is saved in the **&RETCODE** function. Most commands will set their return code to zero when they complete with no errors and to some non-zero value when they detect an error.

A program can test the **&RETCODE** function after each command executes to see if an error was detected by the command:

```
copy one.file to.another
&if &retcode &call error_process
copy another.file to.newfile
&if &retcode &call error_process
```

An easier method of “trapping” these error conditions is to use the **&ERROR** statement. It allows you to define a statement that is executed anytime that a command returns with a non-zero return code.

```
&error &call error_process
copy one.file to.another
copy another.file to.newfile
```

There are two forms of the **&ERROR** statement.

Mode 1—This form of the statement defines a statement that is executed whenever a command returns with a non-zero return code. This statement will be effective until another **&ERROR** statement is executed.

The *statement* defined after the **&ERROR** keyword may be any valid, single-line EXEC statement (statements such as **&FOR**, **&WHILE**, **&BEGIN**, **&STACK**, etc. are not allowed).

If *statement* is a **&CALL** statement, the return location after the subroutine completes execution is the statement following the command-line statement that caused the **&RETCODE** to be set to a non-zero value.

Mode 2—The second form of the statement disables automatic error detection for command return codes. This is the default condition when an EXEC program first starts executing.

&ESC

The &ESC statement invokes the console output controls just as if they had been requested from the operator via the keyboard.

1	&ESC	control
2	&ESC	control on-off
<hr/>		
	control	» O P W
	on-off	» OFF ON

This statement has two basic modes of operation:

Mode 1—With this first mode of operation, the output control is “toggled.” That is, if its current state is ON, it is set to OFF; if its current state is OFF, it is set to ON.

This form of the statement is not recommended because you will never know precisely the current state of the control. It is possible that, after setting a specific state, the operator used a keyboard key to change the state of the control (see “[System Control Keys](#)” on page 71).

Mode 2—In this second mode, the output control is set to a specific state, either ON or OFF.

- O** Suppress Output. Controls the display of text on the console. When set ON, text is not displayed; when set OFF, text is displayed.
- P** Print Echo. Controls output to the current echo file or device (see “[Echo](#)” on page 312). When set ON, text output to the console is also echoed to the echo file; when set OFF, text is not echoed to the echo file.
- W** Page Wait. Controls the screen page wait feature of THEOS. When set ON, and when a page wait is performed on the console (&WAIT statement in an EXEC or a complementary function in a BASIC or C language program), an “up-carot” character (^) displays in the lower left corner of the screen and processing stops until the operator acknowledges the page wait. When set OFF, no page wait is performed.

EXEC

&FOR

The &FOR statement defines the start of and the parameters for a looping structure.

```
&FOR variable = start-expression, end-expression
      statement
      ...
&NEXT
```

The &FOR statement is very similar to the FOR statement available in the BASIC language. It marks the start of one or more statements that are executed repetitively while an index variable increments from its initial value to its ending value.

The end of the repetitive statements is marked by a loop-terminating statement. Traditionally, this terminating statement is the &NEXT statement. However, any of the four loop-terminating statements may be used because they are all synonyms to each other. These four synonymous terminating statements are: &NEXT, &LOOP, &REPEAT and &WEND statements.

When executed, *variable* is assigned the value of *start-expression*. *variable* is then compared with the value of *end-expression*. If *variable* is greater than *end-expression*, control transfers to the statement following the loop-terminating statement. For instance:

```
&for &i = 1, 3
      &type &lit &i = &i
&next
&type Loop done. Terminating value for &lit &i is &i.
```

In this simple example, the variable **&i** is set to one. It is compared to the terminating value three and, since it is not greater than three, the statement between the &FOR statement and the &NEXT statement is executed. When the &NEXT statement is encountered, control returns to the &FOR statement where **&i** increments from one to two. Since two is not greater than three, the loop is repeated. When **&i** is incremented to four, it is greater than three so the loop terminates and the statement following the &NEXT statement is executed.

```
&i = 1
&i = 2
&i = 3
Loop done. Terminating value for &i is 4.
```

Notice that, when a loop is terminated normally, the *variable* has the value that caused the loop termination. In the above example, the *variable* was

incremented to four. Since four is larger than the *end-expression* value of three, the loop was terminated.

A &FOR statement loop always increments the index variable and it always performs a test to determine if the value of *variable* is greater than the value of the *end-expression*. If *end-expression* is equal to or less than the value of *start-expression*, the statements in the loop are not executed.

The start-expression and the end-expression must be numeric expressions. Although their values may be floating-point, only the integer portion is used in the loop. For instance:

```
&for &i = 1.35, 3.48
      &type &lit &i = &i
&next
```

Displays:

```
&i = 1
&i = 2
&i = 3
```

The value of *variable* may be changed by statements executed in the loop.

You may quickly jump to the loop-terminating statement by using the [&CONTINUE](#) statement inside the loop. See “[&CONTINUE](#)” on page 633.

The loop can be terminated early by executing a [&BREAK](#) statement in the loop (see page 631).

&GOTO

The &GOTO statement jumps to a different location in the program.

&GOTO *label*

The *label* must exist in the current EXEC program. The &GOTO statement transfers control to the statement following the location identified by *label*.

&IF

The **&IF** statement provides a means of conditional execution of EXEC statements.

```

1  &IF conditional-expression statement
2  &IF conditional-expression
   statement
   ...
   &IFEND
3  &IF conditional-expression
   statement
   ...
   &ELSE
   statement
   ...
   &IFEND
4  &IF conditional-expression1
   statement
   ...
   &ELSEIF conditional-expression2
   statement
   ...
   &ELSEIF conditional-expression3
   statement
   ...
   &ELSE
   statement
   ...
   &IFEND

```

The **&IF** statement operates similar to “if” statements in other languages.

Except for Mode 1, every **&IF** statement defines a “statement structure” that comprises the **&IF** statement itself, the statements to be executed when the condition is true, the statements to be executed when the condition is false, and the terminating **&IFEND** statement. The true and false statements are optional.

All forms of the **&IF** statement are given a *conditional-expression* to evaluate. The expression is either true or false. A *conditional-expression* may be an expression using conditional operators or it may be a numeric expression. The *conditional-expression* is true when the condition specified is true or the numeric expression evaluates to a non-zero value. It is false when the condition specified is false or a numeric-expression evaluates to zero.

Mode 1—This form of the &IF statement is a simple, single-line &IF statement that allows you to conditionally execute a single statement. The *conditional-expression* is evaluated and, if it is true or non-zero, the *statement* is executed. If the condition is false or zero, the *statement* is not executed.

```
&if &index &call process_arguments
```

The above statement performs the [&CALL](#) only when [&INDEX](#) is not zero. This simple form of the *conditional-expression* is only valid when the variable is null or empty or it contains a numeric value. Variables containing strings can be tested, but only with relational operators.

```
&if &name <> &null &type The name is &name
```

Here, the variable [&name](#) is compared to the null or empty string. It is displayed on the console if it is not an empty string.

Mode 2—The second form of the &IF statement is similar to the first form, except that it allows multiple statements to be executed when the *conditional-expression* is true. The *statement(s)* may be any valid EXEC statement, including another &IF statement or a multiline statement such as [&FOR](#) or [&BEGSTACK](#). If it is a multiline statement, all of the lines comprising the statement or statement structure must be defined prior to the [&IFEND](#) statement terminating this &IF statement.

```
&if &name = &null
    &type Please enter your first name \
    &read &name
&ifend

&if &index
    &for &i = 1, &index
        &type Arg #&i = &i
    &next
&ifend
```

Mode 3—This third form of the &IF statement is similar to the second form with the addition of a clause that is executed when the *conditional-expression* is false.

The [&ELSE](#) statement marks the end of the statements that are executed when the *conditional-expression* is true and marks the start of one or more statements that are executed when the expression is false. The statements following the &IF statement may be omitted. (The statements following the [&ELSE](#) statement may also be omitted. However, it is better to use Mode 2 instead.)

When the *conditional-expression* is true or non-zero, the statements following the &IF statement are executed. When the &ELSE statement is encountered, control transfers to the statement following the &IFEND statement. When the *conditional-expression* is false or zero, the statements following the &IF statement are skipped until the &ELSE statement is encountered. The statements following this &ELSE statement are then executed.

```
&if &name = &null
    &type Please enter your first name \
    &read &name
&else
    &type Your first name: &name
&ifend
```

In this example, the &name variable is tested. If it is undefined the operator is asked to specify its value. It is not displayed again after the operator enters the new value. When the &name variable is defined, it is merely displayed without asking the operator to reenter it. Only one of the set of statements is executed.

Mode 4—This form provides a multiple-choice capability. Instead of the either-or capability of Mode 3, this mode allows you to execute one set of statements from multiple sets of statements. For instance:

```
&if $menuix = 1          ; Selected menu choice 1?
    &call procedure_1
&elseif &menuix = 2      ; Selected menu choice 2?
    &call procedure_2
&elseif &menuix = 3      ; Selected menu choice 3?
    &call procedure_3
&elseif &menuix = 4      ; Selected menu choice 4?
    &call procedure_4
&elseif &menuix = 5      ; Selected menu choice 5?
    &call procedure_5
&elseif &menuix = 6      ; Selected menu choice 6?
    &call procedure_6
&else
    &call choice_error    ; Invalid choice!
&ifend
```

In the above example, the variable &menuix is tested to see if it has a value of one. If it does, procedure_1 is performed and the remaining tests and procedures are skipped. If &menuix is not one, it is tested to see if it is two. If so, procedure_2 is performed and the remaining tests and procedures are skipped. This process is continued until it is determined that &menuix is not any of the values tested for. In that situation, the “else clause” is executed which performs the choice_error process.

Only one of the sets of statements is executed.

Although the above example used *conditional-expressions* that were similar, it is not necessary. Each *conditional-expression* in the **&IF** statement and the **&ELSEIF** statements can be unrelated to the other *conditional-expressions* used. For instance:

```
&if &exist some.file
    ; already exists, don't do anything
&elseif &disksize s > 6000
    ; use regular, blank file
&elseif &filesize default.file < 6000
    ; use default.file
&else
    ; no room
&ifend
```

In this example, each of the *conditional-expressions* tests completely different conditions.

An **&IF** statement is either a single-line **&IF** statement (Mode 1) or a multiline **&IF** statement. A multiline **&IF** statement must have one and only one matching **&IFEND** statement. A multiline **&IF** statement may have one, but only one, **&ELSE** statement defined within it. A multiline **&IF** statement may have one or more **&ELSEIF** statements defined within it. If there is one or more **&ELSEIF** statements in the **&IF** statement, any **&ELSE** statement must be defined last. The statements in each of the sections of the **&IF** statement are optional and may be omitted.

&IFEND

The **&IFEND** statement marks the end of a multiline **&IF** statement.

&IFEND

The **&IFEND** statement may only be used as the terminating statement in a multiline **&IF** statement structure. Every multiline **&IF** statement must have one, and only one, **&IFEND** statement.

Refer to the “**&IF**” on page 644 for an example.

&LOOP

The &LOOP statement is a looping statement repeat and terminator.

&LOOP

The &LOOP statement is synonymous with the [&NEXT](#), [&REPEAT](#) and [&WEND](#) statements and may be used as the terminating statement for the [&FOR](#), [&UNTIL](#), [&VARY](#) or [&WHILE](#) statements.

Traditionally, the &LOOP statement is used only as the terminator for the [&VARY](#) statement.

&MENU

The &MENU statement defines a list of menu items, displays the menu and allows the operator to select one of the items.

```
1  &MENU
    menu-title
    menu-item
    ...
    &END

2  &MENU variable-name
    menu-title
    menu-item
    ...
    &END
```

There must be a matching [&END](#) statement for each &MENU statement. The first line of text following the &MENU statement is used as the menu title. All lines after the menu title, up to the [&END](#) statement, are used as the menu items. Blank lines in the list of menu items display as item separators.

The &MENU statement determines the maximum width of the menu by using the length of the longest menu item and the menu title. The menu is displayed on the screen in a framed window at the approximate center of the screen.

The operation of the &MENU statement is the same as any menu displayed by a THEOS program and is described on page [75](#). When the operator makes a selection, the number of the menu item selected is saved in a variable.

Mode 1—With this mode, the item selected is saved in the variable named &MENUIX.

Mode 2—The second mode specifies the name of the variable used to save the item number selected.

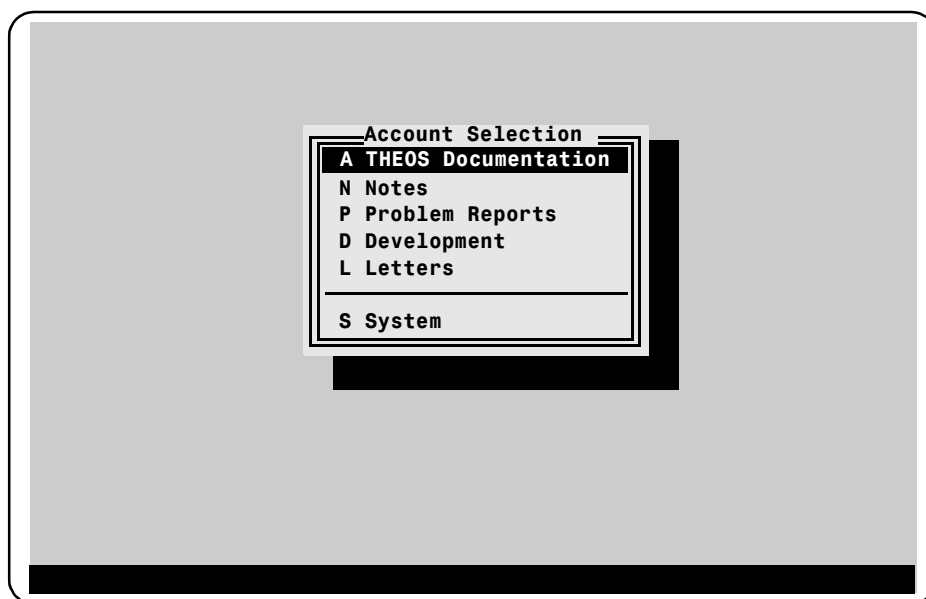
When no item is selected (operator pressed **Esc**), the item number returned is zero.

```
&menu
Account Selection
A THEOS Documentation
N Notes
P Problem Reports
D Development
L Letters

S System
&end

&if &menuix = 1
    &quit logon manuals
&elseif &menuix = 2
    &quit logon notes
...
&ifend
```

Displays as:



The return value is set to -2 if the menu cannot be displayed. This could happen if the current window is too small to display the menu.

If you need more control of the menu appearance, use the [wMenu](#) command described on page 594. It is designed to be used by EXEC programs and sets the [&RETCODE](#) function to the value of the menu item selected by the operator.

&NEXT

The **&NEXT** statement is a looping statement repeat and terminator.

&NEXT

The **&NEXT** statement is synonymous with the [&LOOP](#), [&REPEAT](#) and [&WEND](#) statements and may be used as the terminating statement for the [&FOR](#), [&UNTIL](#), [&VARY](#) or [&WHILE](#) statements.

Traditionally, the **&NEXT** statement is used only as the terminator for the [&FOR](#) statement.

Refer to the [&FOR](#) statement, described on page 642, for a description and example usage.

&QUIT

The **&QUIT** statement exits the current EXEC program.

- 1 **&QUIT**
- 2 **&QUIT** *numeric-expression*
- 3 **&QUIT** *string-expression*

The **&QUIT** statement always exits the current program. What happens after the program is exited depends upon the form of the **&QUIT** statement used.

Mode 1—The current program is exited and the prior program or environment continues execution. For instance, if this EXEC program was invoked by another EXEC program, then that other program continues execution. If this program was invoked by a BASIC language program or a compiled command, then that program or command continues execution. If this program was invoked from the CSI command line, then control returns to the command line.

The return code for this EXEC program is set to zero.

Mode 2—Identical in operation to Mode 1 except that the return code for this EXEC program is set to the value of *numeric-expression*.

Mode 3—The current program is exited but control does not return to the prior program. Instead, the string-expression is evaluated and the resulting string is executed as a command-line.

```
&if &menuix = 1
    &quit
&elseif &menuix = 2
    &quit 255
&elseif &menuix = 3
    &quit show users
&else
    &quit 1
&ifend
```

Although literals are used in the above example for the numeric-expression and the string-expression terms, any valid expression can be used:

```
&quit &retcode
```

This statement quits the current program, setting the return code to the return code of the last program executed by this EXEC.

```
&quit &2 (&option1
```

This statement quits the current program, transferring control to the program name specified as the second command-line argument to this program.

&READ

The &READ statement accepts data from the keyboard or EXEC input stack, and assigns it to one or more variables.

- 1 **&READ** *variable-name...*
- 2 **&READ**
- 3 **&READ ARGS**

There are three forms of the &READ statement:

Mode 1—This is the normal form of the &READ statement. It displays the current input prompt on the screen and accepts data from the operator. If there is data in the EXEC input stack (see “&BEGSTACK” on page 629 and “&STACK” on page 656), the data is read from the stack. When there is no

data in the stack, processing stops until the operator types some data and presses **Enter ↵**.

This form of the statement allows you to specify one or more variable names after the keyword **&READ**. Each variable name is assigned the value of the next word or token from the input stream (EXEC input stack or keyboard). A word or token is defined as a series of alphanumeric characters separated by one or more spaces. The leading and trailing spaces around a token are discarded and are not assigned to the variable.

The variable-names used may be simple variable names (i.e., **&name**, **&address**, *etc.*), command-line argument names (i.e., **&1**, **&2**, *etc.*), or indirect variable names (i.e., **&name&x**). Indirect variable names are evaluated to their simple name equivalent before text assignment is performed.

```
&read &a &b &c
&type &lit &a = &a
&type &lit &b = &b
&type &lit &c = &c
```

Displays:

```
:THE QUICK BROWN FOX JUMPED OVER
&a = THE
&b = QUICK
&c = BROWN
```

Unused words and tokens from the input stream are ignored. In the above example, the words “JUMPED” and “OVER” are discarded. When fewer words or tokens are present in the input stream than there are variable names, the extra variable names are assigned the null or empty string.

Mode 2—When no variable names are specified after the **&READ** keyword, the **&READ** statement accepts one line of text from the keyboard or the EXEC input stack and executes the line as if it were part of the EXEC program.

```
&read
&read
&read
```

Displays:

```
:&TYPE THE VALUE OF &LIT &A IS &A
THE VALUE OF &A IS
:&A = SAMPLE
:&TYPE THE VALUE OF &LIT &A IS &A
THE VALUE OF &A IS SAMPLE
```

This form of the **&READ** statement can accept and execute any valid EXEC statement except those that require multiple lines, such as **&BEGSTACK**, **&BREAK**, **&END**, **&IF**, **&FOR**, **&WHILE** and **&UNTIL**.

Mode 3—With this mode, all of the command-line arguments are redefined with the data read from the keyboard or EXEC input stack. The function **&INDEX** is reset to reflect the new command-line arguments.

```
&call display_args
&type Accept new command line
&read args
&skip 1
&call display_args
&quit

display_args:

&type Current index count: &index

&for &i = 0, &index
    &type #&i = &i
&next

&return
```

Displays:

```
>sample one two three ( four
Current index count: 5
#0 = SAMPLES\SAMPLE.EXEC
#1 = ONE
#2 = TWO
#3 = THREE
#4 = (
#5 = FOUR
Accept new command line
:A B C D E F G H

Current index count: 8
#0 = SAMPLES\SAMPLE.EXEC
#1 = A
#2 = B
#3 = C
#4 = D
#5 = E
#6 = F
#7 = G
#8 = H
```

Notice that command-line argument zero (program name) is not changed.

Text read from the keyboard with any form of the **&READ** statement is accepted in the case mode specified with the **&&CONTROL CASE** statement (default is U). The text is displayed on the console unless **&ESC O OFF** is in effect (see page 641).

Text read from the stack is accepted with the case mode unchanged and it is displayed on the console unless **&ESC O OFF** is in effect or **&CONTROL NOSTACK** is in effect (see page 634).

&REMARK

A line of text whose first non-space character is a semicolon (;) is a remark statement and is skipped by the EXEC processor.

; any text

Remarks may also be added to the end of any statement in the program by using the semicolon character. All characters from the semicolon to the end of the line are ignored by the EXEC processor.

A line that is completely blank is also treated as a comment and ignored.

```
; This is a sample comment
;
    ; This is an indented comment
;
&crt clear          ; clear the screen (this is a comment)
```

&REPEAT

The **&REPEAT** statement is a looping statement repeat and terminator.

&REPEAT

The **&REPEAT** statement is synonymous with the **&NEXT**, **&LOOP** and **&WEND** statements and may be used as the terminating statement for the **&FOR**, **&UNTIL**, **&VARY** or **&WHILE** statements.

Traditionally, the **&REPEAT** statement is used only as the terminator for the **&UNTIL** statement.

Refer to the **&UNTIL** statement, described on page 659, for a description and example usage.

&RETURN

The **&RETURN** statement exits a subroutine invoked by the **&CALL** statement.

&RETURN

Every subroutine must have at least one **&RETURN** statement. The **&RETURN** statement is used to exit a subroutine and to return control to the statement following the **&CALL** statement used to invoke the subroutine.

Refer to “**&CALL**” on page 632 for example usage.

&SKIP

The **&SKIP** statement skips one or more of the following lines in the EXEC program.

&SKIP *count*

The **&SKIP** statement is similar to the **&GOTO** statement in that it performs an unconditional transfer to another location in the program. Unlike the **&GOTO** statement, the **&SKIP** statement performs its transfer to a location that is relative to the **&SKIP** statement.

When *count* is omitted, the **&SKIP** statement merely skips the next statement in the program. When *count* is a positive number, that number of statements following the **&SKIP** statement is skipped. When *count* is negative, that number of statements before the **&SKIP** statement is skipped.

&SLEEP

The **&SLEEP** statement pauses execution of the EXEC program for a period of time or until a specific time.

- 1 **&SLEEP** *seconds*
- 2 **&SLEEP** *time-of-day*

Mode 1—This first mode of the **&SLEEP** statement pauses execution for the number of *seconds* specified.

```
&crt bell
&crt bon
&type System backup in 2 minutes!
&crt boff
&sleep 120
archive s tape (incremental
```

The above code displays a blinking message on the screen and waits for two minutes before executing the ARCHIVE command line.

Mode 2—The second form of the &SLEEP statement pauses execution until a specific time of day.

```
&type System archive scheduled for 8:00 p.m.
&sleep 20:00
archive s tape (incremental
```

You must specify the time with at least one colon character (:) and you must use 24-hour notation. Use two colon characters when specifying a time that includes seconds (i.e., 20:15:30).

&SPACE

The &SPACE statement displays blank lines on the console.

&SPACE *count*

count is the number of blank lines output to the console. When *count* is omitted, the default value of one is used.

&STACK

The &STACK statement adds one line of data to the EXEC input stack.

```
1  &STACK token...
2  &STACK token... \
```

The &STACK statement operates similarly to the &TYPE statement described next, except that the tokens are not displayed on the screen. Instead, they are added to the input stack. The stack and its usage are discussed on page 627.

Each *token* is evaluated and converted to its text equivalent. The *tokens* are then output to the EXEC input. Each evaluated *token* is output, one after another, with one space character following each *token*.

The two forms of the &STACK statement are identical except that the second form terminates the list of *tokens* with a backslant character (\). This backslant character causes the carriage-return character to be left off, thus allowing multiple &STACK statements to be used to build one input line on the stack. When the backslant character is not used, the *tokens* are output and terminated with a carriage-return character.

Tokens whose value is an integer, are output with comma separators at the thousands position. Floating-point values are output without comma separators.

The *tokens* that may be specified in a **&STACK** statement may be text constants, numeric constants, simple variable names, complex variable names and EXEC functions. You may not specify an expression that uses operators. When you do, the operators are treated as text constants and are merely output.

```
&stack input
&stack Program name: &prgname
&stack Last edit date: &system date
&stack
&stack file
lineedit &prgname
```

The above example is similar to the example used for the **&BEGSTACK** statement on page 629. It differs by adding the contents of variable names to the stack. The **&BEGSTACK** statement can only add text constants to the stack.

&TYPE

The **&TYPE** statement displays text on the console.

```
1 &TYPE token...
2 &TYPE token... \
```

Each *token* is evaluated and converted to its text equivalent. The *tokens* are then displayed on the console starting at the current cursor position. Each evaluated *token* is displayed, one after another, with one space character following each *token*. If the end of the display line is encountered, the display wraps to the start of the next display line.

The two forms of the **&TYPE** statement are identical except that the second form terminates the list of *tokens* with a backslant character (\). This backslant character causes the cursor to remain at the end of the display line. When the backslant character is not used, the *tokens* are displayed and the cursor is moved to the start of the next line.

Tokens whose value is an integer, are displayed with comma separators at the thousands position. Floating-point values are displayed without comma separators.

The *tokens* that may be specified in a &TYPE statement may be text constants, numeric constants, simple variable names, complex variable names and EXEC functions. You may not specify an expression that uses operators. When you do, the operators are treated as text constants and are merely displayed.

```
&type Sample EXEC Program
&type
&type Today's date is &system today
&type Number of seconds since 1/1/1900: &system tod
&tod = &system tod
&days = &tod / 24 / 60 / 60
&type Number of days since 1/1/1900: &tod/24/60/60
&type Number of days since 1/1/1900: &days
```

Displays:

```
Sample EXEC Program

Today's date is January 31, 1996
Number of seconds since 1/1/1900: 823,120,644
Number of days since 1/1/1900: 823120644/24/60/60
Number of days since 1/1/1900: 9526.8605902777777
```

Using the second form of the &TYPE statement:

```
&type Please enter your name \
&read &name
&type Please enter your address
&read &address
```

In the following display, notice that the second &TYPE statement causes the cursor to move to the next line before the &READ statement accepts input.

```
Please enter your name :Rudolph
Please enter your address
:North Pole
```

See also “&BEGETYPE” on page 630

&UNTIL

The **&UNTIL** statement defines the start of and the conditions for a looping structure.

```
&UNTIL conditional-expression
        statement
        ...
&LOOP
```

The **&UNTIL** statement marks the start of one or more statements that are executed repetitively until a specified condition is true.

The end of the repetitive statements is marked by a loop-terminating statement. Traditionally, this terminating statement is the **&REPEAT** statement. However, any of the four loop-terminating statements may be used because they are all synonyms to each other. These four synonymous terminating statements are: **&NEXT**, **&LOOP**, **&REPEAT** and **&WEND** statements.

When the **&UNTIL** statement is executed, the *conditional-expression* is tested. If it is false, or zero, the following statements are executed until the loop-terminating statement is encountered, at which point the loop is repeated by testing the value of the *conditional-expression* again. If the value of the *conditional-expression* is true, control transfers to the statement following the loop-terminating statement.

Unlike the **&FOR** statement looping structure, there is nothing in the **&UNTIL** statement that changes the conditions being tested. Therefore, there must be some variable assignment or other statement in the body of the loop that changes the conditions of the **&UNTIL** statement *conditional-expression*. For instance:

```
&until &exist &filename
    &type The file "&filename" does not exist.
&repeat
```

This example produces an infinite loop. The condition being tested (**&exist &filename**) never changes! This code can be corrected by adding some statements that change the results of the condition or that use the **&BREAK** statement to terminate the loop.

```

&until &exist &filename
    &type The file "&filename" does not exist.
    &type Do you wish to create it? \
    &read &reply
    &if &reply = Y
        create &filename (indexed keyl 10 recl 10 file 100
    &else
        &error_status = 253
        &break
    &ifend
&repeat

```

In the above example, the result of the *conditional-expression* is changed by creating the file. If the operator chooses to not create the file, the loop is terminated with the **&BREAK** statement. Notice that, after the **CREATE** command is used, the loop is not terminated. In this situation, the loop is repeated and the *conditional-expression* is tested again. If the file creation was successful, the loop terminates naturally. If the file creation was unsuccessful, the operator is informed that it still does not exist.

You may quickly jump to the loop-terminating statement by using the **&CONTINUE** statement inside the loop. (see page 633).

The loop can be terminated early by executing a **&BREAK** statement in the loop (see page 631).

&VARY

The **&VARY** statement is a synonym to the **&FOR** statement.

&VARY *variable = start-expression, end-expression*
statement
&LOOP



Although either **&VARY** or **&FOR** can be used, the traditional statement name is **&FOR**.

&WAIT

The &WAIT statement performs a page wait operation.

&WAIT

A page wait is used when the program is about to either clear the screen or display another page of information. Page waits are automatically performed by many of the THEOS commands that display multiple pages of text (FILELIST, LIST, *etc.*).

The &WAIT statement performs a page wait by displaying an up-caret (^) in the lower left corner of the screen. It then waits for the operator to press  to acknowledge the request. When  is pressed, processing continues.

Page waits are ignored when **&ESC W OFF** is in effect (see page 641).

&WEND

The &WEND statement is a looping statement repeat and terminator.

&WEND

The &WEND statement is synonymous with the **&NEXT**, **&LOOP** and **&REPEAT** statements and may be used as the terminating statement for the **&FOR**, **&UNTIL**, **&VARY** or **&WHILE** statements.

Traditionally, the &WEND statement is used only as the terminator for the **&WHILE** statement (WEND stands for “While End”).

Refer to the **&WHILE** statement, described next, for a description and example usage.

&WHILE

The &WHILE statement defines the start of and the conditions for a looping structure.

```
&WHILE conditional-expression
      statement
      ...
&WEND
```

The &WHILE statement marks the start of one or more statements that are executed repetitively while a specified condition is true.

The end of the repetitive statements is marked by a loop-terminating statement. Traditionally, this terminating statement is the **&WEND** statement. However, any of the four loop-terminating statements may be used because they are all synonyms to each other. These four synonymous terminating statements are: **&NEXT**, **&LOOP**, **&REPEAT** and **&WEND** statements.

When the **&WHILE** statement is executed, the *conditional-expression* is tested. If it is true, or non-zero, the following statements are executed until the loop-terminating statement is encountered, at which point the loop is repeated by testing the value of the *conditional-expression* again. If the value of the *conditional-expression* is false or zero, control transfers to the statement following the loop-terminating statement.

Unlike the **&FOR** statement looping structure, there is nothing in the **&WHILE** statement that changes the conditions being tested. Therefore, there must be some variable assignment or other statement in the body of the loop that changes the conditions of the **&WHILE** statement *conditional-expression*.

```
&control prompt >
&while &command <> EXIT
    &cr1 down
    &read &command
&wend
```

This example simulates the CSI by displaying the default CSI prompt and then accepting a “command line.” To exit this loop, the operator must type the word **EXIT**. Otherwise, the command line entered is merely displayed and not executed.

You may quickly jump to the loop-terminating statement by using the **&CONTINUE** statement inside the loop (see page 633).

The loop can be terminated early by executing a **&BREAK** statement in the loop (see page 631).

■ EXEC Functions

EXEC functions are predefined keywords that can be used in expressions to access various system-defined values or other special values.

&CAT

The &CAT function is used to combine two strings together.

&CAT *token1 token2*

token1 and *token2* are both evaluated if they contain the ampersand character. The resulting string values are then combined by concatenating the string value of *token2* onto the end of the string value of *token1*. No spaces are added during this concatenation.

The tokens used by this function may not be other functions. If this is desired, assign the value of the function to a variable and then use that variable as an argument to the &CAT function.

```
&name = &cat THEOS Software
&type &name
&who = &system account
&pid = &system pid
&me = &cat &who &pid
&type &me
```

This program displays the following when it is executed while logged onto the account TEST from partition number one:

```
THEOSSoftware
TEST1
```

&DISKSIZE

The &DISKSIZE function returns the number of bytes available on an attached disk.

&DISKSIZE *drive*

If *drive* is the letter of an unattached disk drive, zero is returned.

```
&type Space available on S drive is &disksize s
```

This statement might display:

```
Space available on S drive is 126,833,152
```

&ENV

The &ENV function returns the current value of an environment variable.

&ENV *environment-name*

If the *environment-name* is invalid or it is not currently defined in the environment, a null or empty string is returned.

```
&type DIALDIR is &env dialdir
show dialdir
```

Displays:

```
DIALDIR is system\dialdir:s
DIALDIR=system\dialdir:s
```

For information about environment variables, refer to Chapter 6 “[User Account Environments](#)” starting on page 95.

&EXIST

The &EXIST function determines if a file exists and the file’s type.

&EXIST *filename*

The *filename* may be any valid file description and may include the complete or partial path specification. When *filename* does not start with a forward slash character (/), the path is assumed to start with the current directory.

If *filename* does not specify a disk drive letter, then the currently attached drives in the current drive search sequence are searched. The first file found that matches *filename* is used.

The possible values returned by the &EXIST function are:

Return Value	Meaning
0	file does not exist
1	file is an executable program
2	file is a keyed data file
4	file is an indexed data file
8	file is a direct data file
16	file is a stream data file
64	file is a subdirectory
128	file is a library


```

&if &exist some.new.file = 0
    &type some.new.file does not exist.
    &type Do you wish to create? \
    &read &reply
    &if &reply = Y
        create some.new.file (direct file 100 reclen 10
    &ifend
&ifend

```

&FILESIZE

The **&FILESIZE** function returns the number of bytes used by an existing file.

&FILESIZE *filename*

Similar to the **&EXIST** function, the *filename* may be any valid file description and it may include the complete or partial path specification. When *filename* does not start with a forward slash character (/), the path is assumed to start with the current directory.

If *filename* does not specify a disk drive letter, the currently attached drives in the current drive search sequence are searched. The first file found that matches *filename* is used. Wild cards are not allowed.

If *filename* does not exist, **&FILESIZE** returns a zero.

```
&type SYSTEM.TEOS32.CSI size: &filesize system.theos32.csi
```

Displays:

```
SYSTEM.TEOS32.CSI size: 129,884
```

&INDEX

The **&INDEX** function returns the number of command-line arguments used when this EXEC program was invoked.

&INDEX

Command-line arguments are referenced within an EXEC program with the special variable names **&n**, where *n* is the number of the command-line argument.

For instance:

```
>test.exec one two three
```

The above command line invokes the program TEST.EXEC, giving it three command-line arguments. These command-line arguments are referenced with the variables &1, &2 and &3, respectively. The &INDEX function, when used in this program, returns the value three, indicating that there are three command-line arguments available.

If TEST.EXEC contained the following lines:

```
&type Number of command-line arguments: &index

&for &x = 1, &index
    &type Argument &x is: &&x
&next
```

It would display:

```
Number of command-line arguments: 3
Argument 1 is: ONE
Argument 2 is: TWO
Argument 3 is: THREE
```

Notice that the arguments are displayed in uppercase letters. All command-line arguments are changed to uppercase when they are received by a program unless the argument is enclosed within a pair of quotation marks.

For additional information about command-line arguments, refer to “[Command-Line Arguments](#)” on page 618.

&LEN

The &LEN function computes the length of a string and returns that value.

&LEN *token*

When *token* is a string literal, the number of characters in the string is returned. When *token* contains a reference to a variable, the *token* is evaluated and expanded to its full plain-text version and the length of that text is returned.

```
&city = Sacramento
&citylen = &len &city
&type The length of "&city" is &citylen
```

Displays:

```
The length of "Sacramento" is 10
```

&LIT

The **&LIT** function is used in statements to specify that the following string is not to be expanded. That is, it is to be treated as a literal.

&LIT *string*

The **&LIT** function is used when a string might contain an ampersand character (**&**). It tells the EXEC processor to treat the ampersand character as a regular character and not as a variable name indicator.

&LIT cannot be used as the argument to another function.

```
&city = Vancouver
&type &lit &city contains &city
```

Displays:

```
&city contains Vancouver
```

&NULL

The **&NULL** function is a reserved keyword used for assigning or comparing to an empty text string.

&NULL

The **&NULL** function represents an empty or zero-length text string. It is normally used in conditional-expressions to test a variable.

```
&if &city = &null
  &type Please enter the city name \
  &read &city
&ifend
```

This code only asks the operator to enter a city name if the **&city** variable is currently undefined. If **&city** contains any value, the **&TYPE** and **&READ** statements are skipped because the conditional-expression in the **&IF** statement is false.

&RETCODE The &RETCODE function returns the value of the return code set by the last program executed by this EXEC program.

&RETCODE

Each time that an EXEC program executes a command (not an EXEC statement), the command sets a return code which is saved by the EXEC processor. The &RETCODE function retrieves that return code.

The value of &RETCODE is reset every time that a command executes. If you need to test a program's return code after other commands have executed, you should save it immediately after the command execution.

```
change &file (private ewnxnr notype
&changerc = &retcode
...
&if &changerc
    &type The file &file could not be changed.
&ifend
```

Generally, command return codes are zero for successful operation and non-zero for error reports. The value of the return code will normally indicate the specific error detected.

See also “&ERROR” on page 640.

&SUB The &SUB function extracts a substring from a string.

- 1 **&SUB** *token from*
- 2 **&SUB** *token from to*

Both modes return a substring of *token*.

Mode 1—The substring returned is the string of characters starting with the character at position *from* to the end of the *token* string.

```
&var = abcdefghijklmnopqrstuvwxyz
&var1 = &sub &var 20
```

In the above example, &var1 will contain “tuvwxyz.”

Mode 2—The substring returned is the string of characters starting with the character at position *from* through and including the character at position *to*.

```
&var = abcdefghijklmnopqrstuvwxyz  
&var2 = &sub &var 4 8
```

In the above example, &var2 will contain “defgh.”

This form of the **&SUB** function operates similar to the substring operator. For instance, the following two statements produce the same results:

```
&var2 = &sub &var 4 8  
&var2 = &var[4 8]
```

The only difference between the **&SUB** function and the substring operator is that the substring operator can only be used in expressions while the **&SUB** function can be used anywhere that a token can be used.

&SYSTEM The &SYSTEM function returns various system parameters that are external to the EXEC program.

EXEC

- 1 **&SYSTEM ACCOUNT**
- 2 **&SYSTEM CWD**
- 3 **&SYSTEM DATE**
- 4 **&SYSTEM DAY**
- 5 **&SYSTEM FTIME** *format time*
- 6 **&SYSTEM PID**
- 7 **&SYSTEM PORT**
- 8 **&SYSTEM PRIV**
- 9 **&SYSTEM SERIAL**
- 10 **&SYSTEM TASKS**
- 11 **&SYSTEM TIME**
- 12 **&SYSTEM TOD**
- 13 **&SYSTEM TODAY**
- 14 **&SYSTEM UID**
- 15 **&SYSTEM USERS**
- 16 **&SYSTEM WEEKDAY**

Each of these forms of the &SYSTEM function returns a specific value.

ACCOUNT Returns the account name that you are currently logged on to.

```
&curr_account = &system account
&type Current account is &curr_account
```

Displays:

```
Current account is SAMPLES
```

CWD Returns the current working directory. If the current directory is the root, a null string is returned.

&type Current directory is: **&system cwd**

Displays:

Current directory is: /SAMPLE/EXAMPLE:S

DATE

Returns the current system date using the current date format.

&type Date format is &env dateform
&type The current date is **&system date**

Displays:

Date format is I
Current date is 1996.01.30

DAY

Returns the day number of the week for the current date. Day numbers range from zero for Sunday through six for Saturday.

&type The current date is &system date.
&type It is &system weekday.
&type The weekday number is **&system day**.

Displays:

The current date is 1996.01.30.
It is Tuesday.
The weekday number is 2.

EXEC

FTIME Returns the date formatted according to the format specified in *format*. The format codes for the date are the same as those used by MultiUser BASIC's STRTIME\$ function or THEOS C's strftime function.

The date value is specified by omitting it (means use current date and time) or with a literal or variable whose contents is consistent with the value returned by the [&SYSTEM TOD](#) function. That is, the number of seconds since January 1, 1970.

```
&type The current date is &system ftime "%A, %B %d, %Y."
```

Displays:

```
The current date is Monday, July 4, 1998.
```

PID This is a synonym to the [&SYSTEM PORT](#) function.

PORT Returns the current port or partition number. This number is the user number assigned to this console or task.

```
show who
&type The current port is &system port
```

Displays:

```
ACCOUNT = SAMPLES
USERNUM = 9
PORT     = 1
PRIVLEV  = 5
LOGON    = 12:29:01 96.01.30
The current port is 1
```

PRIV Returns the current privilege level for this session.

```
&type Command privilege level is &system priv.
```

Displays:

```
Command privilege level is 5.
```

SERIAL Returns the operating system serial number for your system.

```
&type THEOS serial number: &system serial
```

Displays:

```
THEOS serial number: 102-12345
```


TASKS Returns the total number of tasks that are running. This number includes all user foreground tasks, user background tasks, the disk cache, printer spooler and despoolers, the network login server, *etc.*

```
show users
&type Number of tasks: &system tasks
```

Displays:

THEOS® 32 Show Users							
Pid	Username	Programe	Status & Info	Pid	Username	Programe	Status & Info
1	SYSTEM	SHOW	*	11			Stopped
2	SYSTEM	CSI	I	12			Stopped
3	TEST	CSI	I	13			Stopped
4	MANUALS	WINWRITE	I	14			Stopped
5		LOGON	I	15			Stopped
6		LOGON	I	16	ACCTNG	MENU	I
7	ACCTNG	MENU	I	17	TCP/IP	NETLOGIN	R
8	ORDERS	INVOICE	I	18	SPOOLER	DESP00L	I Task of 19
9	ORDERS	ORDENTRY	I	19	SPOOLER	DESP00L	I
10	LETTERS	WINWRITE	I	20	CACHE	SYNC	ZN

Number of tasks: 15

Compare this function with the &SYSTEM [USERS](#) function.

TIME Returns a string containing the current time-of-day using 24-hour notation.

```
&type The time is: &system time
```

Displays:

The time is: 13:37:30

TOD Returns the current time calculated as the number of seconds since January 1, 1970.

```
&type The time is: &system time on &system date
&type Which is &system tod since 1/1/1970
```

Displays:

The time is: 13:41:59 on 01/16/1996
Which is 821,828,519 since 1/1/1970

The &SYSTEM TOD is useful for computing elapsed times.



TODAY Returns the current system date using month name format.

```
&type Today is &system today
```

Displays:

```
Today is 16 January 1996
```

Note that the sequence for the year, month and day depends upon the current date format. Date format one displays as Month-name day-number, year. Date formats two and three display as day-number month-name year, with no punctuation.

The actual names used for the months is read from the SYSTEM.TEOS32.MESSAGE file, described on page 733.

UID Returns the account number of the account that you are currently logged on to.

```
&curr_acc = &system uid
&type Current account number is &curr_acc
```

Displays:

```
Current account number is 9
```

Compare this function with the &SYSTEM ACCOUNT function.

USERS Returns the total number of users that have been started. This number includes only users that are started, whether or not they are logged onto the system. Each user with multiple sessions started is counted as one user.

```
show users
&type Number of users: &system users
```

Displays:

THEOS® 32 Show Users							
Pid	Username	Programe	Status & Info	Pid	Username	Programe	Status & Info
1	SYSTEM	SHOW	*	11			Stopped
2	SYSTEM	CSI	I	12			Stopped
3	TEST	CSI	I	13			Stopped
4	MANUALS	WINWRITE	I	14			Stopped
5		LOGON	I	15			Stopped
6		LOGON	I	16	ACCTNG	MENU	I
7	ACCTNG	MENU	I	17	TCP/IP	NETLOGIN	R
8	ORDERS	INVOICE	I	18	SPOOLER	DESPool	I Task of 19
9	ORDERS	ORDENTRY	I	19	SPOOLER	DESPool	I
10	LETTERS	WINWRITE	I	20	CACHE	SYNC	ZN

```
Number of users: 5
```

This example was used on a system where the first four partitions are multiple sessions on the main console (counts as one user), and the next six partitions are two sessions each for three different terminals (counts as three users). The fifth user reported is a network login from a remote computer system.

Compare this function with the `&SYSTEM TASKS` function.

WEEKDAY Returns the current weekday name for today's date.

```
&type The current date is &system date.
&type It is &system weekday.
&type The weekday number is &system day.
```

Displays:

```
The current date is 1996.01.30.
It is Tuesday.
The weekday number is 2.
```

The actual names used for the weekdays is read from the `SYS-TEM.THEOS32.MESSAGE` file, described on page [733](#).

&TYP

The `&TYP` function tests a token for its content type: numeric or alphanumeric.

&TYP *token*

A token that evaluates to numeric characters only (digits, decimal point and minus sign character) returns an N; otherwise, an A is returned.

EXEC

A: Contacting THEOS

Support for THEOS 32 Version 4 and other THEOS products is provided to authorized dealers and reseller's of THEOS products. End users should contact their THEOS dealer regarding questions or problems relating to installation or operation of the THEOS operating system and other THEOS products.

THEOS Support Services for Resellers and Distributors are designed to provide the type of assistance best suited to your needs. Support options range from no-cost / low-cost information services on the World Wide Web to THEOS on-site training classes, fee-based direct support or an annual support contract. Depending upon your needs and budget, you may choose any one of these options or combine several into a custom program suitable to your requirements.

When contacting Technical Support, include or be prepared to provide the following information:

- ▶ Product name and version number.
- ▶ Product patch level (use the `SHOW VERSION` command).
- ▶ Operating system serial number (displayed on bootup or use the `SHOW SERIAL` command).
- ▶ Operating system version number (displayed on bootup or use the `SHOW VERSION` command).
- ▶ Type of hardware being used.
- ▶ What happened and what you were doing when the problem occurred.
- ▶ The exact wording of any messages that appeared on the screen(s).
- ▶ How you tried to solve the problem.

Dealers and THEOS resellers may contact THEOS Technical Support by mail, fax, telephone or the Internet.

Contacting THEOS

Mail: THEOS Technical Support
THEOS Software Corporation
1801 Oakland Boulevard, Suite 315
Walnut Creek, CA 94596-7000

Fax: 925-935-1177

Telephone: 925-935-1118 ext 211
Hours: Monday-Friday,
8:30am - 4:30pm, Pacific Time

Internet E-mail: support@theos-software.com

Internet WWW: <http://www.theos-software.com>

B: LineEdit Text File Editor

The **LineEdit** command is a utility program that edits ASCII text files using a line-editing mode. Unlike WindoWriter, which uses a full-screen editing technique, LineEdit displays lines of text and uses commands entered as lines to manipulate the file.

LineEdit is not intended to be a general-purpose editor for letters, memos or program source. Use WindoWriter for that purpose because it is not only a full-screen editor, but it has many features that make it well adapted for general and special purpose editing.

LineEdit, on the other hand, is well suited to editing a text file without operator input. That is, editing a text file with preprogrammed commands from an application program or utility or an EXEC language program.

This chapter describes the editing capabilities and internal commands of LineEdit. Refer to “**LineEdit**” on page 371 for a description of the LineEdit command-line and its options.

■ **Introduction**

The LineEdit command edits an existing text file or creates a new text file. When the command is first entered, a message displays indicating whether the text file is new or an existing file:

```
>lineedit sample.text
New file "SAMPLE.TEXT".
Edit:
*
```

OR

```
>ledit system.theos32.devnames
Old file: "SYSTEM.TEOS32.DEVNAMES:S".
Edit:
*
```

The asterisk (*) shown above is the **LineEdit prompt**. LineEdit uses this symbol to indicate that it is ready for you to enter one of the LineEdit commands described on page 683.

■ **LineEdit Command-lines**

LineEdit commands are entered by simply typing the command and any parameters or modifiers, followed by **Enter ↵**. Commands may be entered using either uppercase or lowercase characters. A command with its following text and modifiers makes up a **command-line**.

While entering a command-line, before pressing **Enter ↵**, you can make changes because **command-line edit mode** is used by LineEdit to accept commands from the console keyboard. In the command edit mode, you can move the cursor to the beginning or end of the command-line, move one character left or right, delete, insert or replace characters, or delete the entire line.

The specific keys that you use to edit a command-line are dependent upon the class code in use for your console. Refer to the “ClassGen” on page 225 and the “CRT” on page 276 for information about class code definitions and how you can test the keyboard with your class code. In this manual the normal or default keys are used to describe the features of LineEdit.

In the following table, references to control character sequences (i.e., **Ctrl+H**) describe keys that can be used independently of the class code definition.

LINEEDIT

Key	Function
Enter ↵	Tells LineEdit to accept the command as typed.
← or Ctrl+H	Moves left one character position.
→ or Ctrl+L	Moves right one character position.
F8 or Ctrl+G	Moves to the beginning of the command-line.
⇧ Shift+F8 or Ctrl+E	Moves to the end of the command-line.

Table 33: LineEdit Command-line Editor Functions

Key	Function
<div>F5</div> or <div>Ctrl+N</div>	Erases all characters from the cursor position to the end of the line.
<div>Delete</div> or <div>Ctrl+Z</div>	Deletes the character under the cursor.
<div>← Backspace</div>	Deletes the character to the left of the cursor.
<div>Ctrl+X</div>	Cancels the entire line.

Table 33: LineEdit Command-line Editor Functions

There is no “replace mode” while entering commands. All characters typed are inserted into the line, shifting any following characters to the right.

■ The Current Line

LineEdit, as a line-oriented editor, operates on one line of text at a time. This one line that LineEdit manipulates is referred to as the *current line*. Although there are some commands that can change or display several lines, they all start with the current line. Most of the commands in LineEdit operate on the current line.

When LineEdit’s verify mode is on (see “Verify Command” on page 712), every command that changes the contents of the current line, or changes the current line pointer, displays the current line again before returning to LineEdit’s command mode.

```
*change /united states/United States/  
We, the People of the United States, in order to form a more  
*
```

If verify mode is off, or if you are ever unsure of where the current line is, use the Type Command described on page 710.

```
*type  
We, the People of the United States, in order to form a more  
*
```

LINEEDIT

■ Text Delimiters

With the exception of FIND, INPUT and REPLACE, all of the LineEdit commands that allow text to be specified on the command-line require that the text be delimited. A ***delimiter*** is any character that starts and stops a sequence of characters. LineEdit allows any non-alphanumeric character to be used as a delimiter.

The start-of-text delimiter must be the same as the end-of-text delimiter. For instance, in the previous example, the slash character (/) is used as the delimiter.

When a command, such as CHANGE, has a “from” and a “to” text argument, the same delimiter must be used for both arguments and only three delimiters are actually used.

***change /united states/United States/**

End of “to text”

End of “from text” and
start of “to text”

Start of “from text”

■ LineEdit Commands



The following is a description of every command available in LineEdit.

Remember, any references to keys refer to the default key definitions. If these do not work properly on your system, your class code definition may have reassigned the function to a different key. Refer to the “CRT” on page 276 to determine the current definitions for your console.

Again Command

Repeats the last LineEdit command executed.

1 AGIN

2 **F3** or **Ctrl**+**A**

***type 4**

```
SELF1 248:58:0 CPSIO B28400,W8,E2 ; Loop back 1a
SELF2 249:58:0 CPSIO B28400,W8,E2 ; Loop back 1b
SELF3 246:58:0 CPSIO B28400,W8,E2 ; Loop back 2a
SELF4 247:58:0 CPSIO B28400,W8,E2 ; Loop back 2b
```

***again**

```
;-----Disk Drive Definitions-----
;
; Floppy disks
*
```

If you are unsure about which command will be repeated with the AGAIN command, use the ? command described on page 712. It displays the last valid command.

Bottom Command

Move to the end of the file.

1 BOTTOM

2 **End** or **Ctrl**+**Y**

Sets the current line pointer to the last line in the file.

***bottom**

```
;-----End of DEVNAMES File-----
*
```

Case Command

Set the input case mode.

1	<u>C</u> ASE	
2	<u>C</u> ASE	<i>casemode</i>
<hr/>		
	<i>casemode</i>	» U L M

Mode 1—Displays the current input case mode.

Mode 2—Sets the input case mode to uppercase, inverted case or mixed case. The L or invert case mode means that all lowercase letters typed are accepted as uppercase letters and all uppercase letters typed are accepted as lowercase letters.

```
*case
M
*List
We, the People of the United States, in order to form a more
*case u
*LIST
We, the People of the United States, in order to form a more
*CASE L
*list
We, the People of the United States, in order to form a more
*
```

In each [List Command](#) above, it was typed as “List.”

Change Command

Change text in the file.

1 CHANGE

2 CHANGE /from-text/to-text/

3 CHANGE /from-text/to-text/ lines occurrences start

from-text

»

text

lines

»

number of lines to search

occurrences

»

number of times on each line to make change

start

»

number of the first instance on line to change

to-text

»

text

This command changes text in a line. It always operates starting with the current line. After the change or changes have been made to the line, it is displayed on the screen (if VERIFY is on).

Mode 1—Repeats the last CHANGE command.

Mode 2—Changes the first occurrence of *from-text* to *to-text* on the current line.

Mode 3—Changes *from-text* to *to-text*. If *lines* is specified, then the change is performed for that many consecutive lines, starting with the current line. When *lines* is not specified, the default value of one is used for *lines*, *occurrences* and *start*.

If *occurrences* is specified, then that many occurrences are changed on each line. If *occurrences* is not specified, then the default value of one is used for both *occurrences* and *start*.

When *start* is specified, then *start*-1 occurrences of *from-text* are left unchanged on each line. If *start* is not specified, the default value of one is used.

The asterisk (*) character can be used for both *lines* and *occurrences*. The asterisk means “all.”

LINEEDIT

For instance:

```
*type
CENTLP1 8:6:0 SPO W8      ; primary printer port
*change /printer/parallel/
CENTLP1 8:6:0 SPO W8      ; primary parallel port
*

*change /abc/xyz/ 10 2 3
```

This command changes the text “abc” to the text “xyz” for 10 lines, two occurrences per line, starting with the third occurrence on each line.

```
*change /abc/xyz/ *
```

This third example will change the text “abc” to “xyz” throughout the remainder of the file, but only the first occurrence on each line.

See also: [Modify Command](#), [Replace Command](#)

Column Command

Display a “line ruler.”

COLUMN

Displays a “ruler” of column numbers and then redisplay the current line underneath the ruler.

```
*column
          1          2          3          4          5          6
12345678901234567890123456789012345678901234567890
CENTLP1 8:6:0 SPO W8      ; primary printer port
*
```

This ruler is useful for counting spaces or aligning text columns.

Combine Command

Combines two lines into one.

COMBINE

Combines the current line with the next line by adding a single space to the end of the current line and then appending the next line to it.

```
*list 3
Four score and seven years ago
our fathers brought forth on
this continent a new nation,
*combine
Four score and seven years ago our fathers brought forth on
*
```

CSI Command

Execute another command and return to LineEdit.

CSI *command*

command » CSI command to execute, with parameters

This LineEdit command executes any system command and then returns back to the LineEdit program with the text file unchanged. Virtually any command can be executed with the CSI command, however, you should not use the [Logoff](#) command because it does not return back to the LineEdit environment.

```
*list 3
Four score and seven years ago our fathers brought forth on
this continent a new nation,
conceived in liberty,
*csi show who
ACCOUNT = EXAMPLE
USERNUM = 2
PORT = 5
PRIVLEV = 4
LOGON = 08:36:03 10/12/96
*list 1
Four score and seven years ago our fathers brought forth on
*
```

LINEEDIT

Delete Command

Delete one or more lines of text.

- 1 DELETE
- 2 DELETE *count*
- 3 DELETE */text/*

<i>count</i>	»	number of lines to delete
<i>text</i>	»	text to locate

Mode 1—The current line is deleted and the next line becomes the current line. In the following example, remember that the [List Command](#) does not change the current line pointer.

***list 4**

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defence, promote the general welfare, and secure the blessings of liberty to ourselves and our

***delete**

perfect union, establish justice, insure domestic tranquility,

*

Mode 2—Deletes *count* number of lines, starting with the current line.

***list 4**

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defence, promote the general welfare, and secure the blessings of liberty to ourselves and our

***delete 3**

and secure the blessings of liberty to ourselves and our

*

Mode 3—Performs a “conditional delete.” First, LineEdit scans the file starting with the line following the current line. It looks for the first occurrence of a line that contains *text*. This is a “case-sensitive” search. That is, specifying */The/* will search only for the three characters T, h and e and will not match on THE or the.

If LineEdit can find a line containing *text*, it then deletes all of the lines from the current line to the line containing *text*. It does not delete the line containing *text*, and that line becomes the current line.

If LineEdit cannot find any line containing *string*, then it displays the message “String not found.” and does not delete any lines.

***list 4**

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defence, promote the general welfare, and secure the blessings of liberty to ourselves and our

***delete /defence**

provide for the common defence, promote the general welfare,

*

In the above example, LineEdit is told to delete lines down to the line containing “defence.” Since it finds the first instance of “defence” on the third line, it deletes two lines and makes that third line the current line.

***list 4**

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defence, promote the general welfare, and secure the blessings of liberty to ourselves and our

***delete /xxx**

String not found



*

In this second example, LineEdit is told to delete lines down to the line containing “xxx.” LineEdit cannot find any lines that contain “xxx” so it reports this, deletes no lines and leaves the current line unchanged.

LINEEDIT

Down Command

Move down one or more lines in the file.

- 1 DOWN
- 2  or **Ctrl** + **J** or **Enter** 
- 3 DOWN *count*
- 4 DOWN */text/*
- 5 DOWN ?

count » number of lines to move down
text » text to locate

Mode 1—Moves the current line pointer down one line.

Mode 2—Moves the current line pointer down one line. This is merely a set of shortcut keys to Mode 1.

Mode 3—Moves the current line pointer down *count* lines.

Mode 4—Performs a “conditional move.” First, LineEdit scans the file starting with the line following the current line. It looks for the first occurrence of a line that contains *text*. This is a “case-sensitive” search. That is, specifying */The/* will search only for the three characters T, h and e and will not match on THE or the.

If LineEdit can find a line containing *text*, it moves the current line to the line containing *string*. If LineEdit cannot find a line containing *text*, it displays the message “String not found.” and does not move the current line pointer.

Mode 5—Displays the last DOWN */text* or LOCATE */text* command.


```
*down /and
and secure the blessings of liberty to ourselves and our
*d ?
/and
*
```

In the following examples, remember that the [List Command](#) displays one or more lines without moving the current line pointer.

***list 8**

The Constitution of the United States

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defence, promote the general welfare, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America.

*

The Constitution of the United States

down**down**

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, posterity, do ordain and establish this Constitution for the

***down /representative**

No person shall be a representative who shall not have

*

When two or more single-line DOWN commands are used consecutively, the cursor is moved up one line and the command-line is erased from the screen. This allows the display to be more readable. For instance, after the initial LIST 8 command, the actual display in the previous example is:

The Constitution of the United States

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, posterity, do ordain and establish this Constitution for the

***down /representative**

No person shall be a representative who shall not have

*

LINEEDIT

See also: [Locate Command](#), [Next Command](#)

Dup Command

Duplicates the current line one or more times.

```
1  DUP
2  DUP  count
```

count » number of lines to duplicate

Mode 1—Duplicates the current line one time and moves the current line pointer to the duplicated line.

Mode 2—Duplicates the current line *count* times and moves the current line pointer to the last duplicated line.

Note: Because the content of the current line is the same as the prior current line, it is not redisplayed, even if verify is on.

File Command

Save the file to disk and exit LineEdit.

```
1  FILE
2  [F10] or [Ctrl]+[F]
3  FILE  filename
```

filename » file name with optional path

Mode 1—Save the file to disk with the current file name and exits LineEdit.

Mode 2—Save the file to disk with its current file name and exits LineEdit.

Mode 3—Save the file to disk as *filename* and exits LineEdit

```
*name
TEST.FILE:S
*file sample
"SAMPLE.FILE:S" filed.
```

See also: [Name Command](#), [Quit Command](#), [Save Command](#)

Find Command

Locates the next line that starts with indicated text.

```
1  _FIND  text
```

```
2  _FIND
```

```
3  _FIND  ?
```

```
text          »  starting text to locate
```

Mode 1—Searches the remainder of the file, starting with the line following the current line, for a line that starts with *text*. This is a “case-sensitive” search. That is, specifying */The/* will search only for the three characters T, h and e and will not match on THE or the.

If a line is found, the current line pointer is set to that line and the line is displayed (if verify mode is on). If a line is not found, the message “String not found.” is displayed and the current line pointer is not changed.

```
*list 8
```

```
    The Constitution of the United States
```

```
We, the People of the United States, in order to form a more
perfect union, establish justice, insure domestic tranquility,
provide for the common defence, promote the general welfare,
and secure the blessings of liberty to ourselves and our
posterity, do ordain and establish this Constitution for the
United States of America.
```

```
*find United
```

```
United States of America.
```

```
*
```

In the above example, the line starting with “United” is located, not just a line containing “United” as the [Locate Command](#) or [Down Command](#) would have done.

Mode 2—Repeat the last FIND command.

Mode 3—Displays the *text* for the last FIND command.

```
*f ?
```

```
United
```

```
*
```

See also: [Down Command](#), [Locate Command](#)

Get Command

Perform a “Paste” operation, or an import text operation.

- 1 GET
- 2 GET *filename*
- 3 GET *filename from to*

<i>filename</i>	»	optional file name with optional path
<i>from</i>	»	optional first line number to get or text in first line to get
<i>to</i>	»	optional last line number to get or text in last line to get

Mode 1—“Pastes” the lines of text from the “clipboard” into the file. All lines added to the file are added or inserted after the current line. To add lines to the beginning of the file, be sure to use the **TOP** command first.

If nothing has been copied to the clipboard, this command reports that “File “SYSTEM.WORK $nnna$ ” not found. (The $nnna$ in the file name represents your partition number and work file sequence letter.) You must use the [Put Command](#) or the [Putd Command](#) to copy text to the clipboard.

Mode 2—Imports the entire contents of *filename* into the current file by adding the lines from *filename* after the current line.

When verify mode is on, the lines imported are displayed.

Mode 3—Imports line number *from* through *to* of *filename* into the current file by adding the lines after the current line. If *to* is not specified, all lines of *filename* from line number *from* to the end of the file are imported.

See also: [Dup Command](#), [Put Command](#), [Putd Command](#)

Help Command

Display a brief summary of all the LineEdit commands.

1

HELP

2

HELP *command*

3

 or  + 

Mode 1—Displays a summary of all commands available.

*help		
?		Query last command
;		Comment
Again		Re-execute last "string" command
Bottom		Go to bottom of file
CASE	[U L M]	Set case mode Upper, Lower (inverted), Mixed
Change	[/frstr/tostr[/ [#lines [#occur [#start]]]]]	Global change
Column		Display column ruler
COMbine		Append next line to current line
CSI	command	Execute THEOS command, return to LINEEDIT
DElete	[#lines /str/]	Delete lines
Down	[#lines]	Go down
DUP	[#lines]	Duplicate current line
FILE	[filename]	Save the file, then exit
Find	[text]	Anchored locate
Get	[filename][frct /frstr/ [toct /tostr/]]	Merge from another file
Help		Ask for help (this display)
Input	[text]	Insert a line or Insert mode
LEngth		Ask for size of file
LIst	#lines	Display lines
Locate	[/str[/]]	Locate string or Locate again
Modify	[#lines]	Modify lines
^		

LINEEDIT

Mode 2—Displays the syntax for a specific command.

*help file		
FILE	[filename]	Save the file, then exit
*		

Mode 3—This is merely a shortcut to the Mode 1 form of the HELP command.

Input Command

Add one or more new lines of text.

```
1  _INPUT  text
2  _INPUT

_____
text          »   text line to insert into file
```

Mode 1—Adds *text* as a new line after the current line. To add a line to the beginning of the file, be sure to use the [Top Command](#) first. To add a blank line, enter INPUT followed by a single space before pressing **Enter ↵**.

```
*top
Top of file.
*f .fill
.fill
*i Dear Sir,
*
```

The line is not displayed after input, even if verify mode is on.

Mode 2—Enters “Input mode.” LineEdit displays “Input:” and then allows you to type one or more lines of text. These lines are added after the current line. You may input as many lines as you like. To terminate input, enter a blank line (press **Enter ↵** without any other text or spaces on the line). To add a blank line to the file, enter one or more spaces and then press **Enter ↵**. The **Esc** key can also be used to terminate the INPUT command.

```
*i
Input:
As one of our longstanding and valued customers we areEnter ↵
making available our new widget. This is a prereleaseEnter ↵
of the product and is made only on a limited basis.Enter ↵
Enter ↵
*
```

The above example shows **Enter ↵** being used at the end of each display line. If you do not use **Enter ↵**, LineEdit adds a carriage return when it reaches the right side of the screen. However, this may be in the middle of a word. If you forget, use the [Combine Command](#) to combine the lines together and then the [Split Command](#) to divide them at an appropriate place.

See also: [Get Command](#), [Replace Command](#)

Length Command

Display the current length of the file.

LENGTH

Displays the length of the file in number of lines and characters.

```
*length
50 lines, 1618 bytes, 202402319 available, at byte 319.
*
```

List Command

Display one or more lines of the file without moving the current line pointer.

1 LIST
2 LIST *count*

count » number of lines to display

Mode 1—Displays the current line.

Mode 2—Displays *count* number of lines, starting with the current line. *count* may be an asterisk meaning “all.” Unlike the [Type Command](#), the current line pointer is not changed.

```
*list 6
We, the People of the United States, in order to form a more
perfect union, establish justice, insure domestic tranquility,
provide for the common defence, promote the general welfare,
and secure the blessings of liberty to ourselves and our
posterity, do ordain and establish this Constitution for the
United States of America.
*list 3
We, the People of the United States, in order to form a more
perfect union, establish justice, insure domestic tranquility,
provide for the common defence, promote the general welfare,
*
```

Compare the above display with the [Type Command](#) example.

LINEEDIT

LIST and TYPE display lines of text even if verify mode is off.

See also: [Down Command](#), [Next Command](#), [Page Command](#), [Type Command](#)

Locate Command

Locates the next line that contains the specified text.

```

1  _LOCATE  /text
2  _LOCATE
3  [F2] or [Ctrl]+[W]
4  _LOCATE  ?

```

text » text in file to locate

Mode 1—Similar to the [Down Command](#), this mode performs a “conditional move.” First, LineEdit scans the file starting with the line following the current line. It looks for the first occurrence of a line that contains *text*. This is a “case-sensitive” search. That is, specifying /The/ will search only for the three characters T, h and e and will not match on THE or the.

If LineEdit can find a line containing *text*, it moves the current line to the line containing *string*. If it cannot find a line containing *text*, it displays the message “String not found.” and does not move the current line pointer.

Mode 2—Repeats the last DOWN /*text* or LOCATE /*text* command.

Mode 3—Repeats the last DOWN /*text* or LOCATE /*text* command. These are merely shortcut keys to Mode 2.

Mode 4—Displays the last DOWN /*text* or LOCATE /*text* command.

```

*locate /and
and secure the blessings of liberty to ourselves and our
*loc ?
/and
*

```

See also: [Down Command](#), [Find Command](#), [Next Command](#)

Modify Command

Edit an existing line or lines.

```

1  MODIFY
2  MODIFY  count

```

count » number of lines to modify

Mode 1—Enters “modify mode” for the current line only.

Mode 2—Enters “modify mode” for *count* number of lines.

In modify mode, an existing line of text is displayed and the cursor is positioned to the first character of the line.

Any control characters embedded in the line are displayed with two characters: an “up carrot” character followed by the noncontrol version of the character. For instance, a tab character is actually a CTRL,I and is displayed as ^I. Although these control characters display as two characters, they represent only one character and will be handled as such if the character is replaced or deleted.

The modify mode has its own insert and replace modes with insert mode being the initial mode. This means that, initially, any characters that you type will be inserted into the line at the cursor position. A replace mode is also available that allows you to type characters that replace the existing characters in the line. The cursor shape indicates the insert or replace mode: insert mode is indicated by a blinking block cursor; replace mode is indicated by a blinking underline cursor.

While in modify mode, there are several special editing keys that can be used. These are very similar to the keys described in [Table 34](#): “LineEdit Modify Command Editor Functions” on [page 700](#), and the same notes apply regarding the class code definitions of keys on your terminal.

The table on the next page describes all of the editing keys available in modify mode.



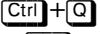


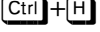









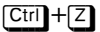




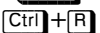


Key	Function
	End modify mode. LineEdit saves the line in the file. If <i>count</i> was specified for the Modify command, then the next line of text is modified.
  	Exit modify mode without saving any of the changes made.
 	Move left one character position.
 	Move right one character position.
  	Move to the beginning of the text line.
  	Move to the end of the text line.
 	Delete the character under the cursor.
	Delete the character to the left of the cursor.
 	Invoke a search operation. The next character typed is the character that Modify will search for, starting at the current cursor location. If the character exists in the remainder of the line, then the cursor is positioned to it. If there is no matching character, then the cursor is positioned to the end of the line.
 	Switch between insert mode and replace mode or between replace mode and insert mode.
 	Change the case of the character under the cursor. That is, if the character is lowercase, then change it to uppercase and vice versa. After the character is changed, the cursor advances to the next character.

Table 34: LineEdit Modify Command Editor Functions

See also: [Change Command](#), [Replace Command](#)

Name Command

Define the name of the text file.

1	<u>NAME</u>
2	<u>NAME</u> <i>filename</i>
<hr/>	
<i>filename</i>	» file name with optional path

Mode 1—Displays the current name of the file.

```
*name
TEXT.FILE:S
*
```

Mode 2—Changes the name of the file to *filename*. If only a file-name is provided (no file-type), then the new name of the file depends upon the prior name of the file. A “flat file” is changed to use the new file-name with the prior file-type. A library member file is changed to use the *filename* specified as the new member-name.

The new name of the file is always displayed.

```
*name sample.letter
SAMPLE.LETTER:S
*name example
EXAMPLE.LETTER:S
*name letter.library.latenote
LETTER.LIBRARY.LATENOTE:S
*name sample
LETTER.LIBRARY.SAMPLE:S
*
```

See also: [File Command](#), [Save Command](#)

Next Command

Move down one or more lines in the file.

1 NEXT

2 NEXT *count*

count » number of lines to move down

Mode 1—Moves the current line pointer down one line.

Mode 2—Moves the current line pointer down *count* lines.

The **Next** command is synonymous with the [Down Command](#) except it does not have the ability to search for text.

See also: [Down Command](#), [Locate Command](#)

Page Command

Display one full screen of text.

1 PAGE

2 PageDown or Ctrl+P

Mode 1—Display one screen of text. The number of lines displayed depends upon the attached page size of the console.

The current line pointer is always set to the last line displayed.

Mode 2—This is a shortcut to the **Page** command.

Put Command

Perform a “Copy” to the “clipboard” or an export to a file.

```

1  PUT
2  PUT  count
3  PUT  /text
4  PUT  filename
5  PUT  filename count
6  PUT  filename /text

```

<i>count</i>	»	number of lines to copy
<i>filename</i>	»	file name with optional path
<i>text</i>	»	text in last line to copy

Mode 1—Copies the current line to the clipboard.

Mode 2—Copies *count* number of lines to the clipboard.

Mode 3—Copies lines of text to the clipboard until a line is copied that contains *text*. If there are no lines containing text, then the remainder of the file is copied.

Mode 4—Copies the current line to the file *filename*. Any existing file with that name is replaced by this copy operation.

Mode 5—Copies *count* number of lines to the file *filename*. Any existing file with that name is replaced by this copy operation.

Mode 6—Copies lines of text to the file *filename* until a line is copied that contains *text*. Any existing file with that name is replaced by this copy operation. If there are no lines containing text, then the remainder of the file is copied.

In all cases, the copy starts with the current line. After the copy, the current line pointer is set to the last line copied.

See also: [File Command](#), [Get Command](#), [Putd Command](#), [Save Command](#)

Putd Command

Performs a “Cut” operation. That is, a copy with delete operation.

```

1  PUTD
2  PUTD  count
3  PUTD  /text
4  PUTD  filename
5  PUTD  filename count
6  PUTD  filename /text

```

<i>count</i>	»	number of lines to copy
<i>filename</i>	»	file name with optional path
<i>text</i>	»	text in last line to copy

The operation of this command is identical with the [Put Command](#) except that any and all lines that are copied to the clipboard or *filename* are deleted from the file. The current line pointer is set to the line following the last line copied.

By using the [Get Command](#) and this Putd command, “Cut and Paste” operations can be performed that allow you to move lines of text from one location in the file to another location in the file. The [Get Command](#) and the [Put Command](#) provides a “Copy and Paste” operation that allows you to duplicate lines of text in other locations in the file.

Note: The “clipboard” used by the Put and Putd commands exists only during the current LineEdit session. When LineEdit is exited, the contents of this clipboard are erased and cannot be used by other programs (not even WindoWriter).

See also: [File Command](#), [Get Command](#), [Put Command](#), [Save Command](#)

Quit Command

Exit LineEdit without saving the file.

- 1 QUIT
- 2 **F9** or **Ctrl+Q**
- 3 QUIT *retcode*
- 4 QUIT *command*

<i>command</i>	»	CSI command to execute, with parameters
<i>retcode</i>	»	return code to exit with

Mode 1—Quits LineEdit and sets the return code to 0.

Mode 2—Quits LineEdit and sets the return code to 0. These are merely shortcut keys to Mode 1.

Mode 3—Quits LineEdit and sets the return code to *retcode*.

Mode 4—Quits LineEdit and executes the CSI *command*.



If a Quit is done and there have been some changes made to the file that have not been saved, LineEdit will ask “OK to forget changes? (Y/N).” Respond with **N** to cancel the Quit operation and return to LineEdit. Respond with **Y** to exit LineEdit without saving the changes.

```
*quit
>lineedit sample.file
Old file "SAMPLE.FILE:S".
Edit:
*delete 3
*quit
OK to forget changes? (Y/N) Y
>
```

See also: [File Command](#)

Replace Command

Replace the existing current line with one or more new lines of text.

1 REPLACE *text*

2 REPLACE

text » text to replace current line

Mode 1—Replaces the current line of text with *text*. To replace the current line with a blank line, enter Replace followed by a single space before pressing **Enter ↵**.

Mode 2—Similar to the [Input Command](#) except that, instead of adding the lines of text after the current line, the current line is replaced with the new lines of text.

***top**

Top of file.

***list 3**

This is the first line of text.

And this is the second line.

This is the last line of text.

***replace**

Input:

This is a new first line of text.

This is an additional line added after the first line.

***top**

Top of file.

***list ***

This is a new first line of text.

This is an additional line added after the first line.

And this is the second line.

This is the last line of text.

*

See also: [Input Command](#)

Save Command

Save the current file onto disk.

1 SAVE

2  +  or  + 

3 SAVE *filename*

filename » file name with optional path

Mode 1—Saves the file to disk with the current file name.

Mode 2—Saves the file to disk with the current file name. These are merely shortcut keys to Mode 1.

Mode 3—Saves the file to disk as *filename*. Changing the name of the file with *filename* follows the same rules as described for the [Name Command](#) described on page 701.

```
*name
TEST.FILE:S
*save
"TEST.FILE:S" saved.
*save sample
"SAMPLE.FILE:S" saved.
*name
SAMPLE.FILE:S
*
```

See also: [File Command](#), [Name Command](#)

Split Command

Divide the current line into two lines.

```
1  SPLIT  /text/
```

```
2  SPLIT
```

```
3  SPLIT  ?
```

```
text                »   text to end the current line
```

Mode 1—Splits the current line after the first occurrence of *text* in the line.

***list 6**

```
We, the People of the United States, in order to form a more
perfect union, establish justice, insure domestic tranquility,
provide for the common defence, promote the general welfare,
and secure the blessings of liberty to ourselves and our
posterity, do ordain and establish this Constitution for the
United States of America.
```

***split /States,**

```
We, the People of the United States,
in order to form a more
*
```

Mode 2—Repeats the last Split command using the current line.

Mode 3—Displays the last Split command text.

***split ?**

```
/States,
*
```

See also: [Combine Command](#)

Tabset Command

Define the tab stop columns.

```
1  TABSET
2  TABSET  column-list
```

column-list » space-separated list of tab-set column numbers

Mode 1—Displays the current tab stop settings.

```
*tab
9 17 25 33 41 49 57 65 73 81 89 97 105 113 121
*
```

Mode 2—Sets the tab stop columns to *column-list*.



The default tab settings are every eight columns. It is not advised that these settings be changed because all other programs in THEOS (except Script) use these same settings.

Top Command

Moves to the top of the file.

```
1  TOP
2  [Home] or [Ctrl]+[T]
```

Mode 1—Moves the current line pointer to the position before the first line of the file. Any [Input Command](#) operations performed will add lines before the first existing line in the file.

Mode 2—Moves the current line pointer to the position before the first line of the file.

See also: [Bottom Command](#)

Type Command

Display text lines and move the current line pointer.

```
1  _TYPE
2  _TYPE  count
```

count » number of lines to display

Mode 1—Displays the current line.

Mode 2—Displays *count* number of lines, starting with the current line. Unlike the [List Command](#), the current line pointer is changed and always points to the last line displayed.

***type 6**

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defence, promote the general welfare, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America.

***type 3**

United States of America.

ARTICLE I.

*

List and Type display lines of text even if verify mode is off.

Compare the above display with the [List Command](#) example.

See also: [Down Command](#), [List Command](#), [Next Command](#), [Page Command](#)

Up Command

Move up one or more lines in the file.

1

UP

2

↑

 or

Ctrl

+

K

3

UP *count*

count

»

number of lines to move up

Mode 1—Moves the current line pointer up one line.

Mode 2—Moves the current line pointer up one line. These are merely shortcut keys to Mode 1.

Mode 3—Moves the current line pointer up *count* lines.

*type 8

The Constitution of the United States

We, the People of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defence, promote the general welfare, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America.

*up

posterity, do ordain and establish this Constitution for the

*up 3

perfect union, establish justice, insure domestic tranquility,

*

LINEEDIT

See also: [Down Command](#), [Locate Command](#), [Next Command](#), [Top Command](#)

Verify Command

Turns the current line verify mode on or off.

```
1  VERIFY
2  VERIFY  status

status          »  ON
                  OFF
```

Mode 1—Displays the current status of the verify mode.

```
*verify
ON
*
```

Mode 2—Sets the verify mode either on or off. When verify is on, all other commands that either move the current line pointer or change the current line, will display the current line before returning to the LineEdit prompt.

Query (?) Command

Display the last LineEdit command executed.

```
?
```

This command is particularly useful in two situations: When a multiple command macro has failed or before using the [Again Command](#).

When a macro command finishes execution, the ? command can display the last command executed in the macro.

Comment (;) Command

The comment command is ignored.

```
; comment
```

```
comment      »   any text
```

Comment commands are useful when LineEdit is invoked with stacked EXEC commands. .

Macro (X, Y or Z) Commands

The macro commands allow one or more commands to be stored and executed as a single command set.

1

X

command(s)

2

X

"filename" line

3

X

4

X

count

5

X

?

<i>count</i>	»	number of times to execute
<i>filename</i>	»	file name with optional path
<i>line</i>	»	line number in file to assign to macro
<i>command</i>	»	one or more LineEdit commands combined with ampersand character (&)

Macro names may be X, Y or Z

LINEEDIT

Mode 1—Defines the macro X to be the *command* specified. See “Macro Definitions” on the following page.

Mode 2—Defines the macro X with the first line of the file *filename* or, if *line* is specified, to be record number *line* of the file *filename*. The pair of quotation marks surrounding *filename* is required.

```
*x "quick.macro"
*y "macro.defs" 1
*z "macro.defs" 4
*
```

Mode 3—Executes the macro one time.

Mode 4—Executes the macro *count* times.

Mode 5—Displays the current definition of a macro.

```
*z ?
up 1 & locate /, / & split /, / & up 1 & change /, /,/
*
```

Macro Definitions

Macros defined with Mode 1 or Mode 2 may contain one or more LineEdit commands:

```
*x change /, and/ and/ 1 1 1
*y top & x
*z up 1 & locate /, / & split /, / & up 1 & change /, /,/
```

The macro X is defined with a single command. This type of macro definition is used as a shortcut to some command that is frequently used.

The macro Y is defined with two commands: The first command (*top*) moves the current line pointer to the top of the file while the second command (*& x*) executes the current definition of the macro X. Note that a macro command name in a macro definition is not executed as a “subroutine call.” That is, the macro name should only be used at the end of the macro definition because it will not return back to this macro. A definition of *x top & y & z* will only execute the *top* and the *y* macro. The *z* macro is not executed.

The Z macro contains several commands and, when executed, performs the following operations: Move the current line pointer up one line (*up 1*), locates the next line with a comma followed by a space (*& locate /, /*), splits that line following the comma, space (*& split /, /*), moves back to the first of the two resulting lines (*& up 1*) and then changes the comma, space to just a comma without the trailing space (*& change /, /,/*).

Multiple commands in one macro definition must be separated with the ampersand character (*&*). To indicate that the ampersand character is part of a text string and not a command connector, it must be doubled (*&&*).



```
*y locate /Bill && Judy/ & type
```

The above macro contains two commands that locate the next line containing the string “Bill & Judy” and displays it.

Spaces between multiple commands in a definition are ignored during execution and may be used freely for better readability.

When macro definitions are stored in a file for usage by Modes 3 and 4, do not include the macro name. For instance, the three macro definitions used at the beginning would be stored as:

```
change /, and/ and/ 1 1 1
top & x
up 1 & locate /, / & split /, / & up 1 & change /, /,/
```

Special Macro Commands

There are two special commands that may be used in a macro definition. These are the Error and Skip commands.

ERROR *skip-number*

skip-number » number of subcommands to skip on error

SKIP *skip-number*

skip-number » number of sub-commands to skip

LINEEDIT

The Error command is used in a macro immediately before a command that might fail. It tells the macro processor how many of the following commands to skip when that command does fail.

Although the Skip command can be used in any macro, it should only be used in a macro that uses the Error command. Skip is an unconditional jump in a macro definition; Error is a conditional jump.

For instance:

```
*x error 2 & locate /abc/ & change /abc/def/ & skip -4
```

When using the Error or Skip command, remember that the macro processor has already incremented its subcommand pointer by the time it analyzes the Error or Skip command.

When the example executes:

1. The `error 2` statement: If the next command is unsuccessful, skip the two following commands. The next command is `& locate /abc/` and the two following commands are `& change /abc/def/` and `& skip -4`.

When `& locate /abc/` fails, the next command executed will be the end of the macro definition and the macro execution will terminate.

2. The `& locate /abc/` command: Attempts to locate the next line containing the string “abc.” If successful, the following command is executed; otherwise, the prior `error 2` command tells the macro processor what to do.
3. The `& change /abc/def/` command: Executed only when the `locate` was successful.
4. The `& skip -4` command: Executed only when the `locate` was successful. Tells the macro processor to jump back four commands and continue executing. Remember, when the macro processor is analyzing this command, its command pointer or number has already been incremented. So, the `-4` value tells it to skip back to the `error 2` command.

If the Error command is not used before a command and that command fails, the execution of a macro terminates.

The LineEdit commands that might fail or be unsuccessful include:

```
DELETE /text/
DOWN /text/
FIND
LOCATE
SPLIT
UP /text/
```



Caution: Be very careful when using Error and Skip commands because it is possible to create infinite loops in a macro. If an infinite loop does occur, the only way to terminate the loop is by using the [Force](#) command on another user’s terminal and forcing this user to execute some other command. Respond with **[Y]** when the [Force](#) asks if you want to try harder.

C: EXEC Language Summary

■ EXEC Statements

<i>; any text</i>	Remark statement
&variable-name = <i>exp</i>	Assign value to variable
&BEGSTACK <i>text...</i>	Add data to stack
&END	
&BEGTYPE <i>text...</i>	Display text on screen
&END	
&BREAK	Terminate loop structure
&CALL <i>label</i>	Invoke subroutine
&CONTINUE	Repeat current loop
&CONTROL <i>option...</i>	Change operating controls
&CRT <i>col row</i>	Move cursor
&CRT <i>video-attribute</i>	Change display attribute
&CRT <i>cursor-control</i>	Move cursor or edit screen
&ELSE	Begin false part of IF
&ELSEIF <i>condition-exp</i>	Begin false part of IF
&END	Terminate stack, type or menu
&ERROR <i>statement</i> &ERROR	<i>statement</i> executed on error
&ESC <i>control</i>	Toggle system display control
&ESC <i>control on-off</i>	Set system display control
&FOR <i>variable</i> = <i>start-exp</i> , <i>end-exp</i> <i>statement</i> ...	Automatic incrementing looping structure
&NEXT	
&GOTO <i>label</i>	Jump to <i>label</i>
&IF <i>conditional-exp statement</i>	Execute <i>statement</i> if true

&IF <i>conditional-exp</i> <i>statement</i> ...	Execute <i>statements</i> if true
&IFEND	
&IF <i>conditional-exp</i> <i>statement</i> ...	Execute <i>statements</i> if true
&ELSE <i>statement</i> ...	Execute <i>statements</i> if false
&IFEND	
&IF <i>conditional-exp1</i> <i>statement</i> ...	Execute <i>statements</i> if <i>exp1</i> true
&ELSEIF <i>conditional-exp2</i> <i>statement</i> ...	Otherwise, execute these statements if <i>exp2</i> true.
&ELSEIF <i>conditional-exp3</i> <i>statement</i> ...	Otherwise, execute these statements if <i>exp3</i> true.
&ELSE <i>statement</i> ...	Otherwise, execute these statements
&IFEND	
&IFEND	End of multiline IF statement
&LOOP	End of looping structure
&MENU <i>menu-title</i> <i>menu-item</i> ...	Display menu; response to &MENUIX
&MENU <i>var-name</i> <i>menu-title</i> <i>menu-item</i> ...	Display menu; response to <i>var-name</i>
&END	
&NEXT	End of looping structure
&QUIT	Exit program
&QUIT <i>numeric-token</i>	Exit program; set return code
&QUIT <i>string-token</i>	Exit program and execute command

&READ <i>variable-name...</i>	Accept data, assign to variable
&READ	Accept data, execute as statement
&READ <i>arguments</i>	Accept data as command-line args
&REPEAT	End of looping structure
&RETURN	End of subroutine
&SKIP <i>count</i>	Skip <i>count</i> number of statements
&SLEEP <i>seconds</i>	Suspend execution for <i>seconds</i>
&SLEEP <i>time-of-day</i>	time
	Suspend execution until
&SPACE <i>count</i>	Display <i>count</i> blank lines
&STACK <i>token...</i>	Add <i>token</i> to stack, with CR
&STACK <i>token...</i> \	Add <i>token</i> to stack, without CR
&TYPE <i>token...</i>	Display <i>token</i> with CR
&TYPE <i>token...</i> \	Display <i>token</i> without CR
&UNTIL <i>conditional-exp</i> <i>statement</i>	Execute <i>statements</i> until true
...	
&LOOP	
&VARY <i>variable = start-exp, end-exp</i> <i>statement</i>	Same as FOR statement
&LOOP	
&WAIT	Perform page wait
&WEND	End of looping structure
&WHILE <i>conditional-exp</i> <i>statement</i>	Execute <i>statements</i> while true
...	
&WEND	

■ EXEC Functions

&CAT *token1 token2*

Concatenate two items

&DISKSIZE *drive*

Get available disk space

&ENV *environment-name*

Get environment variable

&EXIST *filename*Test existence of *filename***&FILESIZE** *filename*Get file size of *filename***&INDEX**

Get number of cmd-line args

&INDEX *token*Get length of *token* value**&LEN** *string*Treat *string* as literal**&NULL**

Empty string

&RETCODE

Return code of last program

&SUB *token from*

Get ending substring of token

&SUB *token from to*

Get substring of token

&SYSTEM ACCOUNT

Get account name

&SYSTEM CWD

Get current subdirectory

&SYSTEM DATE

Get current system date

&SYSTEM DAY

Get day number of week

&SYSTEM PID

Get partition number

&SYSTEM FTIME *fmt time*

Format time and date

&SYSTEM PORT

Get partition number

&SYSTEM PRIV

Get privilege level

&SYSTEM SERIAL

Get serial number

&SYSTEM TASKS

Get number of tasks

&SYSTEM TIME

Get current time-of-day

&SYSTEM TOD

Get number of seconds

&SYSTEM TODAY

Get current system date

&SYSTEM UID

Get account number

&SYSTEM USERS

Get number of consoles

&SYSTEM WEEKDAY

Get weekday name

&TYP *token*

Determine type of item

D: System Files

■ *account.COMMAND* and *account.EXEC*

As described in “[Logoff](#)” on page [386](#), when you log onto an account the Logon command searches for a program with the same name as the account name. This program may be in the SYSTEM.CMD32 library, the user-defined command library or as a flat file named *account.COMMAND* or *account.EXEC*. When this program is found, it is automatically executed.

These *logon EXEC* programs are very useful for automatically starting users in the appropriate application.

■ *account.LOGON*

See “[SYSTEM.LOGON](#)” on page [723](#).

■ *account.REMINDER*

See “[SYSTEM.REMINDER](#)” on page [726](#).

■ SYSTEM.BOOTLOG

This file is created or appended to when the “[Save History Information](#)” field of the system configuration is set to “S” (see “[Sysgen](#)” on page [531](#)). When the system is booted, records are written to this file logging the date and time of the boot and logging the various actions performed by the boot process. These actions include loading device drivers, attaching disks, printers, tapes, communications devices and consoles.

Other file names may be used for this log depending upon the setting of the “[Save History Information](#)” field. Refer to Appendix E: “[System Log Files](#)” starting on page [737](#) for a description of these names and the contents of the log files.

■ CDROM.CATALOG

This file is created and maintained by the [CDPlayer](#) command. It is an indexed file containing the artist, title and track titles for audio CDs.

■ **SYSTEM.CHIST***nnn:work-drive*

This set of files resides on the system disk and they are marked as “hidden” files. That is, they are not displayed by [FileList](#) unless you use the **HIDDEN** option. These files are created and maintained by the CSI and contain the command history for each user. There is one file for each user, with the *nnn* being the user’s port number.

Each file contains the last 16 commands executed by the user. These files are used when the operator uses the **command history recall** features of the CSI. Refer to “[Command-Line Recall and History](#)” on page 41 for additional information.

■ **SYSTEM.CMD32 Library**

This library contains all of the THEOS-provided commands described in this manual. As described in “[Command Search Sequence](#)” on page 48, this library is always searched first when a requested program name contains only a program name without specifying the complete file name of the program.

■ **SYSTEM.HELP32 Library**

This library contains the text files used by the [Help](#) command described on page 356. It also contains the text files used by the internal help commands available with the BASIC language products, LineEdit, Link32, More, Patch and WinWrite commands.

You may add your own help text files to this library. If you do add your own files to this library, be sure to use member names that do not conflict with files provided by THEOS.

■ **SYSTEM.HISTORY**

This is an ASCII text file that shows the activity of all users. It is created or appended to when the “[Save History Information](#)” field of the system configuration is set to “S” (see “[Sysgen](#)” on page 531).

It is maintained by the Logon and Logoff commands and the CSI, but only if “[Save History Information](#)” is enabled by [Sysgen](#).

Other file names may be used for this log depending upon the setting of the “[Save History Information](#)” field. Refer to Appendix E: “[System Log Files](#)”

starting on page [737](#) for a description of these names and the contents of the log files.

■ **SYSTEM.LOGON**

As described in “[Logon](#)” on page [386](#), this is an optional, ASCII text file that, if it exists, is displayed every time a user logs onto the system. This file is useful for displaying company or application names and logos, as long as the information can be displayed with the THEOS character set described on page [761](#).

This file may have “escape characters” embedded in it.

Character	Meaning
\f	Clear the screen.
\n	Start a new line.
\r	Move to the start of this line.
\t	Advance to the next, eight-column tab stop.
\\	Output a single backslant character.
\xnn	Output the character whose hexadecimal value is <i>nn</i> .

Table 35: Logon Banner File Escape Characters

There may be files named *account.LOGON* that are specific to a particular account name.

■ **SYSTEM.MAILBOX**

This is the file maintained by the [Mailbox](#) command described on page [398](#).

■ **SYSTEM.MENU32 Library**

This library contains the menu files specific to each command with each of the member files containing text specific to a program. The format of the records in each of the files is determined by the application using it.

■ **LOGON**

This member file is used by the [Logon](#) command. Similar to the `SYSTEM.LOGON` file described on page 723, this is an optional, ASCII text file that, if it exists, is displayed every time a user logs onto the system. This file is useful for displaying company or application names and logos, as long as the information can be displayed with the THEOS character set described on page 761.

This file differs from `SYSTEM.LOGON` in that `SYSTEM.MENU32.LOGON` displays before the “Logon please” prompt and `SYSTEM.LOGON` displays after the user has entered the account name and password. Both types of files may exist and, if they do, they will each be displayed.

This file may use the “escape characters” described in [Table 35](#) on page 723.

■ **MSG**

This member file is used by the [Msg](#) command. It is an editable text file and, as distributed, has the message “Press ESC to clear MSG Window.”

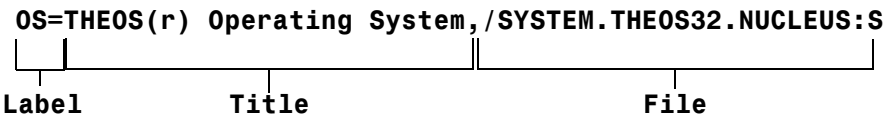
Only the first line of this file is displayed by the [Msg](#) command.

■ **SHOW_VER**

This member file is used by the [Show](#) command to display the `VERSION` information. The records in the file specify which products or programs are reported by [Show](#).

The file is composed of a series of text records, each one identifying a specific product. The sequence of the records does not matter, but this sequence is used by the [Show](#) command when displaying product versions.

Each record is composed of three parts:



Label The label is an identifier for the record. It is used by installation programs to find an existing entry in the file. The label section is composed of letters, digits and the underscore character. All leading characters, up to and including the terminating equal sign, are considered to be the record label.

Although it is ignored by the [Show](#) command, every record must have a label.

Title The title is the text displayed by the [Show](#) command. It includes all characters between the label's equal sign and the terminating comma.

The title text may include the literal (r), as in the above example. When the [Show](#) command is executed on a VGA display, this sequence of characters is translated into the registered trademark symbol.

File The last portion of the record starts after the terminating comma of the title field. This specifies the complete path to a program containing the version, version date and patch level information.

When [Show](#) cannot find a file matching this file specification, no information or error message is displayed.

Note: The sequence of characters “(r)” and “(c)” will display as ® and © respectively, when the console is a VGA display.

■ **SYSTEM.MODEM Library**

This library of files is used by the system startup process and by the [Start](#) command. This library contains member files that contain the modem initialization strings specific to each user started. The member names are the user partition numbers. For instance, the name of the file used to start partition number eight is SYSTEM.MODEM.8.

A member file for a specific user may not exist or it may be an empty file. In this situation, no modem initialization string is sent to that user when it starts.

■ **SYSTEM.REMINDER**

As described in “[Logoff](#)” on page [386](#), when you log onto an account, the Logon command searches for reminder messages specific to the account. These *account.REMINDER* files are maintained by the [Reminder](#) command described on page [452](#).

The file SYSTEM.REMINDER is a general reminder message file that is maintained by [Reminder](#) and displayed by the [Logon](#) command.

These reminder files are keyed access files with a key of the reminder date in *yymmdd* format. Repeating reminders use ** for the year portion of the key.

■ **SYSTEM.SPOOLER Library**

This library contains the files used by the print spooler. The library is created either by the [Spooler](#) command with the BUILD option or during system bootup if the library does not exist and the system configuration specifies that the print spooler is enabled.

There is always at least one file in this library with a member name of QUEUE. This file is the print spooler’s queue and contains information about each report that the spooler is maintaining. In addition to this file, there may be other member files if there are any reports that either have not been printed or have been printed but have their HOLD status enabled. These report files use member names like *Fnnnnn* where *nnnnn* is the spooled report number.

■ **SYSTEM.TEOS32 Library**

This library contains members that comprise the THEOS operating system. With few exceptions, they are all necessary for the proper operation of your system.

■ **ACCOUNT**

This file contains the definition and environment settings for each account. It is maintained by the [Account](#) command described on page [156](#) and is read protected for security reasons. To display the information in this file use the [Account](#) command.

■ B32RT*, B3220* and B386*

These sets of files are the run-time modules used by MultiUser BASIC Version 1.0, MultiUser BASIC Version 2.0 and BASIC386 compiled programs.

■ BOOTER1

This is a copy of the “bootstrap loader” that is written to the first four sectors of any disk that might be used to boot the system. It is a generic program that uses BIOS INT 13 to perform disk reads and BIOS INT 10 to write to the main console.

This file is written to a disk by the [Disk](#) FORMAT, BUILD or BOOT option and by the [Setup DISK](#) command.

■ BOOTER2

This file is a “real mode” program loaded by BOOTER1. It is responsible for loading the operating system nucleus, the console device driver, the console class codes and the disk drivers. It also initializes memory maps and displays the first part of the welcome screen, up to the date and time display. It loads and executes the BOOTER3 file.

■ BOOTER3

This file is a “protected mode” program loaded by BOOTER2. It is responsible for correcting the system’s time-of-day clock for daylight savings time adjustment, and for displaying the remainder of the welcome screen, performing any disk analysis if requested by the system configuration and asking if you wish to remain in maintenance mode if requested by the system configuration. It starts other users, the session manager, the Network Login Server, and any publicly attached devices.

■ BOOTMSG

This member file is used during system bootup. This is an optional, ASCII text file that, if it exists, is displayed every time that the system is booted. This file is useful for displaying system configuration information, company logos, *etc.*, as long as the information can be displayed with the THEOS character set described on page [761](#).

This file may use the “escape characters” described in [Table 35](#) on page [723](#).

■ CLASS_{nnn} (Class Codes)

These are the class code files. This group of files is used for input and output character translation for consoles and printers. The individual files correspond to a specific terminal or printer type.

Over 90 console class codes and 11 printer class codes are distributed with THEOS 32. These class codes cover most of the popular terminals and printers.

Many terminals, particularly PC Term compatible terminals, have variations due to international keyboard options. The standards used for assigning class code numbers to these variations of the basic terminal are:

#	Language
0	American English
1	United Kingdom
2	French
3	German
4	Italian
5	Spanish
6	Swiss
7	Latin American
8	Canadian English/French
9	Belgian

Class code numbers currently reserved for and used by PC Term compatible terminals are:

Class Code Range	Terminal Type
90–99	Monochrome PC Term
170–179	IBM 3151 PC Term
180–189	Color PC Term
190–199	ANSI PC Term
200–209	Color Wyse 370 PC Term
210–219	Color NetTerm and THEO+COM user and PC Term with white background

One of the above class code numbers must be used when a PC Term terminal is used.

Although you may redefine one of the existing class code files for a new terminal, it is best if you use a new number. Numbers 65–89 are reserved for your usage for custom or new terminal class codes. Reusing an existing number causes problems when a system is re-installed or updated with a later version of the operating system because the installation process will replace the class code definitions with their default configurations.

Refer to the `SYSTEM.THEOS32.DEV NAMES` file for a current listing of console and printer class code numbers and meanings.

■ CLOCK

This file is loaded during the system bootup process and operates as an integral part of the operating system nucleus. Its responsibilities are to maintain and control device and timing interrupts on the system.

■ COLORCFG

An ASCII text file maintained by [Setup COLOR](#). It specifies the screen colors used for each session on a console along with the preferred colors for standard windows.

■ CONFIG

This is the “system configuration file” maintained by [Setup Sysgen](#). It is an ASCII text file with all of the parameters defined during the [Sysgen](#) process. It is loaded by the `SYSTEM.THEOS32.BOOTER2` program and used by the `SYSTEM.THEOS32.BOOTER3` program during a system’s initial bootup.

■ CRTCFG

This file is the main console configuration file maintained by the [Setup CRT](#) command. It is an ASCII text file with all of the parameters defined during the Setup process.

■ CSI

This is the Command String Interpreter, the program responsible for accepting command lines from the console and dispatching the request to the appropriate command. This program incorporates the EXEC language processor.

■ DESPOOL

This is the printer spooler, a program responsible for redirecting all printer data sent by programs to a disk file and subsequently sending that disk file to the actual printer.

■ DEV_{nnn}

These are the various device drivers supported by your system. The device driver numbers and their functions are:

Driver	Device
DEV1	Floppy disk drive.
DEV2	Hard disk drive (IDE and SCSI).
DEV3	Main console monitor/keyboard.
DEV5	SIO1, SIO2, SIO3 and SIO4 serial devices.
DEV6	LPT1, LPT2 and LPT3 parallel devices.
DEV16	Multiport serial driver.
DEV34 [†]	Non-SCSI, streaming tape drive.
DEV44–DEV55	Reserved for third party, vendor-supplied, drivers.
DEV58	NULL com pseudo-device (SELF1–4).
DEV59	Image disk pseudo-drive.
DEV64	RAM disk pseudo-drive.
DEV101	SCSI and IDE/ATAPI tape drive.
DEV102	CD-ROM disk drive.
DEV103	THEO+Fax printer interface.
DEV104	TCP/IP network virtual ports for TWS and NetTerm.
DEV105	TCP/IP Telnet virtual ports.
[†] The DEV34 device driver is created with the Setup TAPE command. It is a modified copy of one of the following files: ARCHTAP1, ARCHTAPE, EVERTAPE and SPERTAPE.	

Table 36: Device Driver Numbers

■ DEVNAMES

This is the “device names file” used by the [Attach](#), the [Sysgen](#) and other commands to translate the mnemonic names of devices, class codes and VDI types into the numbers used by THEOS.

Although there are many types of records in this file, they all use the same syntax:

SI01	6:5:0	CPSIO		; first com port
Name	Number	Type	Options	Remark/Comment

Name This is the mnemonic name used when attaching a device. It is also the name used when *Attach* displays the current attachments.

Number This is the number used by THEOS for the device or name. For devices, the number is composed of three parts: *ucb:device-driver-number:seq.*

Type The first letter of this field identifies the device or entry type and may be one of six codes:

Type	Meaning
D	Disk device
T	Tape device
C	Console/Communication device. This type code is followed by other letters indicating the type of communications allowed. These additional type letters tell <i>Attach</i> and <i>Sysgen</i> what options are allowed when the device is attached.
	I Device accepts input O Device performs output P Device may be used as a printer S Device uses serial communications
N	Class code name record.
V	VDI name record.

Options The default options applied when this name is used for a new device attachment. For instance: *B38400,W8,E2.*

Comment The semicolon marks the beginning of a comment. All characters from the semicolon to the end of the line are ignored.

A record may be composed of only a comment if it starts with a semicolon character.

There may be multiple records using the same number. For example, a standard entry of “SIO2” and another entry of “MODEM” could use the same number and type codes. These are called synonym entries and they provide convenient methods of attaching commonly used devices. For instance, the “MODEM” entry might define the default options that are used when attaching the modem device.

The sequence of the records in this file does not matter except when *Attach* is displaying the current attachments. For each device attached, *Attach* searches the file sequentially, looking for the first entry matching the attached device’s *ucb:device-driver-number:sequence*. When a match is found, *Attach* uses the name on that record. Thus, you should have your preferred device name entries defined before other synonym entries.

■ DTIME

This file is used in conjunction with the “Adjust Daylight Savings Time” capability of the system. The existence of this indicates that the system is using daylight savings time. This file is erased and the file *STIME* is created when the system switches to standard time.

■ FIXDEV

This program is used during installation of THEOS 386/486 Plus Paks to make them comply with THEOS 32 requirements.

■ FORM_{nnn}

These programs are the disk formatters for floppy and hard disk drives.

■ HOSTS

This is a file maintained by the *Setup NET* command that associates node names with their respective IP addresses. It contains comments explaining the content and format for each record.

■ KEYWORD

This file is used by THEOS and many third-party utility programs to decode any command-line options used. The file specifies the spelling and the minimum abbreviation for these “words.”

A current numerical or alphabetical list of the words can be displayed or printed with the [Keyword](#).

■ MESSAGE

This message file contains the messages displayed by programs. These messages can be translated to another language to make the operating system more useful in areas where English is not the primary language.

A current numerical list of these messages is displayed or printed with the [Message](#).

■ MODEM

This file contains the ASCII text sent to each user when it is first started. The data in this file is used if the library SYSTEM.MODEM does not exist (see “[SYSTEM.MODEM Library](#)” on page 725). This text should be the modem commands required by modems on your system to be initialized in the proper state. The default contents of this file are:

```
AT E0 Q1 S0=1
```

These are “Hayes-compatible” commands that tell a modem to turn command echo off, to not display result codes and to answer the phone on the first ring. Edit this file so that it contains the codes required by your modems for similar operation.

To disable this feature for all consoles started on your system, rename or erase this file and the SYSTEM.MODEM library. To disable or to customize this feature for specific consoles, create a SYSTEM.MODEM library and add member files that are specific to each user partition.

■ NUCLEUS

This is the THEOS operating system.

■ PATCHDOS

This is a program that applies corrections to the THEO+DOS® Plus Pak so that it can comply with the directory attributes used by THEOS 32.

■ SESSMAN

This is the session or task manager.

■ SETaaaaa

These files are the programs used by the [Setup](#) for the various setup functions. For instance, SETCRT is the CRT function of the [Setup](#) command; SETSIO is the SIO function of the [Setup](#) command; *etc.* Only the members present here are offered as [Setup](#) functions.

■ SMCFG

This is the configuration file used by the session manager.

■ STIME

This file is used in conjunction with the “Adjust Daylight Savings Time” capability of the system. The existence of this indicates that the system is using standard time. This file is erased and the file [DTIME](#) is created when the system switches to daylight savings time.

■ SYNONYM

This file contains the standard command synonyms and abbreviations. This file must be kept in sorted order. There are two types of records in the file: one for command abbreviations and another for command synonyms. The format for these records is:

command-name minimum-spelling

command-name synonym synonym-minimum-spelling

The *command-name* must be the fully spelled out name of the command. *Minimum-spelling* is the number of characters required for recognition. For instance:

ACCOUNT 3

Here, the *command-name* is ACCOUNT and the *minimum-spelling* is 3 meaning that the command is recognized if the first three letters are specified.

Synonym is the fully spelled synonym for the *command-name* and *synonym-minimum-spelling* is the number of characters required for recognition of the synonym name. For instance:

```

B3220 3
B3220 B20 2
B3220 BASIC32 1

```

These records state that the MultiUser BASIC, Version 2.0 product can be invoked with an abbreviation of B32 (the first three letters of B3220). Alternately, the synonym B20 can be used and it can be abbreviated to B2; and the synonym BASIC32 can be use with an abbreviation of B.

■ **VDInnn**

These files are the graphics translators for VDI devices.

■ **System Files That May be Deleted**

After installation of your system and after you have assured yourself that it is operating properly, some of the system files may be deleted. Any unused device driver, class code or VDI driver may be deleted. In case you should ever need them in the future, it is best to copy them to diskette before deleting them.

The run-time programs for BASIC386 and BASIC286 may be deleted if you do not have any programs written with those versions. The current version of BASIC is MultiUser BASIC and uses different run-time program names. The files that may be deleted include:

```

SYSTEM.TEOS286.B286IO
SYSTEM.TEOS286.B286LINK
SYSTEM.TEOS286.B286MATH
SYSTEM.TEOS286.B286PLOT
SYSTEM.TEOS286.B286STR
SYSTEM.TEOS286.B286TRIG
SYS.TEOS386.B386IO
SYS.TEOS386.B386LINK
SYS.TEOS386.B386MATH
SYS.TEOS386.B386PLOT
SYS.TEOS386.B386STR
SYS.TEOS386.B386TRIG
SYSTEM.TEOS32.B386IO
SYSTEM.TEOS32.B386LINK
SYSTEM.TEOS32.B386MATH
SYSTEM.TEOS32.B386PLOT
SYSTEM.TEOS32.B386STR
SYSTEM.TEOS32.B386TRIG
SYSTEM.TEOS32.PATCHDOS

```


E: System Log Files

The operating system and many of the Plus Paks that may be added to it have the capability of generating log files recording important events. These log files are stream or sequential text files that may be viewed or printed with various system utilities. They may also be analyzed by application programs.

There are two types of log files that may be generated by the operating system:

- ▶ Boot Log
- ▶ Command History Log

The boot log contains a history of the dates and times that the system was booted and the important events relating to the boot process. The command history log contains a history of every command that each user started and the time that the command completed its execution.

■ Log File Location

By default, history logs are maintained in files written to the root directory of the system disk. However, you may specify a different location for the system log files by specifying a path in the “History File Path” field of the system configuration file (see “[Sysgen](#)” on page 531).

■ Log File Names

The name of a log file is determined by the “[Save History Information](#)” field of the system configuration (see “[Sysgen](#)” on page 531). When this field is set to “N” no log files are created. When the field is set to any other value the name of file is determined by the specific setting and possibly the date that the log file was created. A setting of “D,” “W” or “M” causes the name of the log file to change as it is rotated by day, week or month.

The names used for the system log files are:

Code	Rotated	Boot Log File Name	History Log File Name
S	No	/logs/SYSTEM.BOOTLOG	/logs/SYSTEM.HISTORY
D	Daily	/logs/BLyymmdd.LOG	/logs/SHyymmdd.LOG
W	Weekly	/logs/BLyymmdd.LOG	/logs/SHyymmdd.LOG
M	Monthly	/logs/BLyymm01.LOG	/logs/SHyymm01.LOG

/logs/ is defined by the “[History File Path](#)” field in [Sysgen](#) menu.

yy is the year number.

mm is the month number.

dd is the day number. For code W, the day number is always the day number of the Monday of the week that the log file was started.

The format for each record in the system log files is:

YYYY/MM/DD HH:MM:SS Message text

Date and time of event

■ System Boot Log File

The system boot log file contains a history of each time that the system was booted and the events involved in that boot process. Specifically, a record is written to the log file when the boot process:

- ▶ Starts
- ▶ Device drivers are loaded
- ▶ Disk drives are attached
- ▶ Disk caching is started
- ▶ File system is checked
- ▶ Spooler file system is checked
- ▶ Users are started

■ **System Command History Log File**

The system command history log file contains a history of each time that a user executed or finished a command.

Each record in the file has the following fields in the locations indicated:

Columns	Content
1–10	Event date in the yyyy/mm/dd format
12–23	Event time to the nearest millisecond
25–27	User partition number
29–44	Event
46–end	Additional information about event

The types of events that are recorded and the additional information recorded with each event type include:

Event	Additional Information Field
Boot	Operating system version
Logon	Account name and user number
Logoff	CPU time used
Start Program	Command line executed
End Program	Return code and CPU time used.

Logs

F: Command Privilege Levels

This table shows the commands distributed by THEOS that can be executed when your privilege level is set to the various levels. These are the THEOS defaults, which can be modified with the [Change](#) command described on page [216](#).

Command	Can Execute with Privilege Level					
	5	4	3	2	1	0
Account						
Archive						
ASM32						
Attach						
B32						
Backup						
BASIC386						
Boot32						
Cache						
Calendar						
Cat						
CC32						
CDPlayer						
CGITest						
Change						
ChDir						
Classgen						
Clear						
Color						
Compare						
Compress						
CopyFile						
Create						
CRLF						
CRT						
Date						

Command	Can Execute with Privilege Level					
	5	4	3	2	1	0
Debug32						
Dial						
DialNet						
Disk						
Echo						
Edit						
Eject						
EmailChk						
EmailGet						
Erase						
Exit						
Expand						
Fax						
File						
FileList						
FileType						
Find						
Finger						
FontEdit						
Force						
FTP						
GetFile						
Head						
Help						
Ident						
IXDiag						
Keyword						
Line						
LineEdit						
Link32						
List						
Load						
LogName						
Logoff						

Command	Can Execute with Privilege Level					
	5	4	3	2	1	0
Logon						
Look						
Lowcase						
Mailbox						
Make						
MakeBoot						
Message						
MkDir						
More						
Mount						
Msg						
Net						
NetTerm						
NSLookup						
Number						
Password						
Patch						
Peek						
Ping						
Printer						
PutFile						
PWD						
Quote						
Reboot						
Receive						
Reminder						
Rename						
Repeat						
Restore						
RmDir						
Script						
See						
Send						
SendMail						

Command	Can Execute with Privilege Level					
	5	4	3	2	1	0
Set						
Setup						
Shell						
Show						
Sleep						
Sort						
Split						
Spooler						
Start						
Stop						
Sysgen						
System						
Tail						
Tape						
TBackup						
Tee						
Telnet						
Terminal						
THEO_COM						
THEO_DOS						
TheoMail						
Tinstall						
Touch						
TraceRT						
Tree						
TWS						
Unique						
Unload						
Unnumber						
Upcase						
wBypass						
wClear						
wClip						
wClose						

Command	Can Execute with Privilege Level					
	5	4	3	2	1	0
wFinish						
wFrame						
WhereIs						
Who						
WhoAml						
wInvert						
WinWrite						
wMenu						
wMove						
wOpen						
WordCount						
wRefresh						
wRemove						
wRestore						
wSave						
wScroll						
wSelect						
wStat						
wSwitch						
wTitle						

G: Command Return Codes

All commands, programs and EXEC procedures set a return code when they exit. A **return code** is a numeric value indictating either the success or failure of the command's operation. Most programs indicate successful operation by setting a return code of zero with non-zero return codes indicating a failure of some type.

A program that detects an error during its operation or a condition that prevents it from performing the requested operation, displays a message describing the error or condition and sets the return code to the number of the message. For example:

```
>erase some.file
File "SOME.FILE" not found.
No files erased, 0 bytes recovered.

RC = 19, 09:13:58, ET = 0:06, CPU = 0.352

>message 19
File "... " not found.

RC = 0, 09:14:19, ET = 0:00, CPU = 0.120

>
```

Notice that the message displayed by the Erase command is, in fact, message number 19 from the SYSTEM.TEOS32.MESSAGE:S file, as confirmed with the Message command.

One significant exception to this rule about return codes being message numbers is operator cancellation. If the operator terminated operation of the program by entering **Break**, **Q** the return code is always set to 254 which is not a message number.

Another exceptions to the success/fail and message number return codes is illustrated by the Start command:

```
>start daily.backup
Task started as process #29.

RC = 1029, 09:23:19, ET = 0:00, CPU = 0.144

>
```

Although the **Start** command successfully performed the operation, the return code is not set to zero. Instead it is set to 1,000 plus 29 which is the number of the task that it started. For the **Start** command, return codes less than 1,000 indicate failure and provide the message number of the reason for the failure. Return codes over 1,000 indicate success and provide the number of the started task.

■ Displaying Return Codes

The return code set by a program or procedure can be displayed by enabling the **RDYMSG** environment switch. This switch is set by the account environment definition (see “**Account**” on page 156) or by the **Set** command (see page 479).

```
>set rdymsg on

RC = 0, 08:15:06, ET = 0:00, CPU = 0.096

>
```

Normally, this “ready message” is not displayed except in a program development and testing environment.

■ Commands Using Wildcard Arguments

Many commands allow you to use wild cards. For instance:

```
>list *.text

>copyfile *.text:s a
```

A return code of zero from these commands indicates success. However, a nonzero return code will generally only reflect the result of the last file operated on or it may reflect the status of the last file that had an error during the operation of the command. For instance, if you performed the following command:

```
>copyfile *.text:s a (noq
File "TIMEZONE.TEXT:S" is protected.
"/MESSAGE.TEXT:S" not copied to "MESSAGE.TEXT:A" because file exists.
"/MSG.TEXT:S" copied to "MSG.TEXT:A".

RC = 99, 12:20:54, ET = 0:00, CPU = 0.032

>
```

Here, the return code of 99 indicates that there was at least one file that it could not copy and that the last file that it had problems with was a “file exists” error.

A nonzero return code in this situation does not indicate that the copy failed completely...one or more files may have been copied successfully.

■ **Setting and Testing the Return Code**

The return code for a program or procedure is set in different ways, depending upon the language used. When a program does not specifically set a return code the return code is set to zero.

■ **Return Codes in EXEC Procedures**

In an EXEC language procedure, the return code is set with the **&QUIT** statement.

1. **&Quit**

When nothing is specified with the statement, the return code is set to zero.

2. **&Quit *numeric***

When a numeric literal or a variable containing a number is specified with the statement, the return code is set to that value.

3. **&Quit *string***

When a string literal or a variable containing a string is specified with the statement, the return code is set to zero.

The return code of a program or command invoked by an EXEC procedure can be tested in the procedure by referencing the **&RETCODE** variable.

```
erase my.workfile:s (notype
&If &RetCode > 0
    ; error on erase command
&IfEnd
```

■ **Return Codes in MultiUser BASIC Programs**

In a BASIC language program, the return code is set with the **QUIT** statement.

1. **Quit**

When no argument is specified with the statement, the return code is set to zero.

2. Quit *numeric*

When a numeric literal or a variable containing a number is specified with the statement, the return code is set to that value.

3. Quit *string*

When a string literal or a variable containing a string is specified with the statement, the return code is set to zero.

The return code of a program or command invoked by the BASIC program can be tested in the program by using the `CSI.RETURN.CODE` function.

```
SYSTEM "erase my.workfile:s (notype"
IF CSI.RETURN.CODE > 0
    ! error on erase command
IFEND
```

H: File Systems

The term *file system* refers to both the layout of information on a disk and to the software used to access that information. THEOS 32 uses its own file system called the THEOS/4G file system.

■ THEOS/4G File System

■ Disk Layout

The layout of any disk using the THEOS/4G file system is always the same. The only variable is the size of the main directory.

Start	Sectors	Contents
0	3	Bootstrap loader
4	1	Disk Volume Label (VLB)
5	n	Main Directory
$5+n$	1	Free Space Map
$6+n$	1	Bad Sector Map (not always present)
$7+n$	nn	Subdirectories and files
Available sectors		

Table 37: THEOS/4G File System, Disk Layout

The disk volume label sector is described in the VLB.H file included with the THEOS C language add-on product. This file also describes the layout of a free space map sector and a bad sector map sector.

The smallest piece of data accessed on a THEOS/4G file system is the sector, which is 256 bytes of data. This is true no matter how small or large the disk is. The drive and its controller may use a larger block size but, even then, it rarely exceeds 4K. Thus, there is no penalty in disk utilization when large disks are used.

As the name implies, the THEOS/4G file system supports disks up to 4GB in size. Drives larger than 4GB must be partitioned into two disks, one as a primary partition, the other as an extended partition. The extended partition will have one or more logical disks defined within it.

■ Main Directory

The main or root directory of a THEOS/4G disk is a preallocated, fixed size, keyed access file containing 64-byte records. This file is not accessed by programs as a file but the operating system uses the same algorithms to access it as if it were a file. The records in this directory file contain the descriptions of files, libraries and subdirectories that exist on the disk.

Since the main directory uses the keyed access file structure, it provides very fast access to the directory entries.

The layout of each record in the main directory is described in the FDB.H file included with the THEOS C language product. In general, each record contains the following information:

Field	Meaning
filename	Eight-character file-name
filetype	Eight-character file-type
status	Type of file (indexed, keyed, stream, etc.)
owner	Owning account number
protect	Access protection codes and the hidden and modified file status
date	Date and time file was last changed
size	Size of file, in bytes
reclen	Length of each record in the file, in bytes
keylen	Length of the key for each record in the file, in bytes
ext_lev1	Five, level one extents
ext_lev2a	Pointer to sector containing 51 extents
ext_lev2b	Pointer to sector containing 51 extents
grow	File growth factor

Table 38: File Directory Entry Contents

■ Library and Subdirectory Directories

Libraries and subdirectories also use the same record layout as the main root directory. The access method for libraries are identical to the main directory because libraries are also preallocated, fixed size, keyed access files. Libraries have the same, fast access as the root directory.

Subdirectories are not preallocated keyed access files. They are stream files that grow as additional space is needed. Because they are stream files, their access is not as efficient as the main directory, especially for subdirectories containing a large number of members. To access a file in a subdirectory, the subdirectory is searched sequentially from the beginning until the file member is found.

■ THEOS/4G Files

The THEOS/4G file system supports many types of files including:

- ▶ Subdirectories
- ▶ Libraries
- ▶ Stream
- ▶ Relative
- ▶ Keyed
- ▶ Indexed
- ▶ Program (both 16-bit and 32-bit)

These file types are described in Chapter 8 “[Directories and Files](#)” starting on page [129](#).

As indicated in Table 38 on page 752, the directory entry for a file may contain as many as 107 extents or fragments. Each extent of a file is a contiguous sequence of up to 65,535 sectors of disk space allocated to the file. Thus, the maximum size of any one file, assuming that each extent has used its maximum size, is 1,798,134,720 bytes (107 extents of 65,535 sectors of 256 bytes each). The smallest size for a file is zero bytes using zero sectors.

I: System Date and Time

THEOS uses a **system date** and a **system time** common to all users of the system. This date and time are maintained in the computer's clock circuitry and is normally maintained by a battery so that the time is accurate even when the computer system is powered off.

The system date and time are used as a “date/time stamp” when any file is created or modified. Each time that a file is closed that has been changed in any way, the current system date and time are written to the file's directory entry. By using this time stamp, you can easily determine which files have changed recently and which file is newer than another. The **Make** utility program supplied with the language compilers relies upon the accuracy of this time stamp when determining whether or not a program must be updated.

■ Date and Time Display Formats

THEOS supports three display formats for dates: American, European and International.

1. American: *mm/dd/yyyy*
2. European: *dd-mm-yyyy*
3. International: *yyyy.mm.dd*

The international format is the recommended format to use for several reasons:

- ▶ Easily readable and writable by software.
- ▶ Easy to compare two dates.
- ▶ Consistent with time notation where larger units are written before smaller units.
- ▶ Strings containing both a date and time are easily compared (“2001.12.31 22:45:15”).
- ▶ The format is widely used in many countries such as Japan, Korea, Hungary, Sweden, Finland, Denmark, *etc.*
- ▶ It is the format recommended by the International Standards Organization, ISO8601.

Dates before 1900 or after 1999 are always displayed with a four-digit year. Note: A file's date/time stamp displays with only a two-digit year due to space limitations in a directory listing.

The date format chosen applies to all users on the system and is defined in the Setup Sysgen configuration. Although the date format can be changed with the [Set](#) command, you should do so only as a temporary change.

Time is normally displayed in the same format: **hh:mm:ss**. It can be displayed in several different formats with the [Date](#) command.

■ What is UTC?

Coordinated Universal Time (UTC) is the name of the international time standard. It is the current term for what was commonly referred to as Greenwich Meridian Time (GMT). Zero hours UTC is midnight in Greenwich, England, which lies on the zero longitudinal meridian.

Universal time is based on a 24-hour clock, therefore, afternoon hours such as 4 p.m. UTC are expressed as 16:00 UTC (normally pronounced with words: sixteen hours, zero minutes).

■ Time Zones

Since a day is 24 hours long, the world may be split into 15-degree-wide longitudinal bands (360 degrees ÷ 24 hours). Each band represents one hour. As an example, Portland, Oregon is located at approximately 122 degrees west longitude. Local time in Portland lags UTC time by eight hours (122 ÷ 15 = 8.13). So, if UTC or Greenwich time is 14:30, the time in Portland is 06:30 or 6:30 a.m.

The 15-degree-wide bands are referred to as **time zones**. The exact boundaries of the time zones are determined by the local governments, taking into consideration the population density and geography. For instance, Ireland and the United Kingdom span two 15-degree-wide bands but only one time zone is used for both countries.

There is no international standard which specifies abbreviations for local time zone names such as MET (Middle European Time), EST (Eastern Standard Time), *etc.* Sometimes the same abbreviation is even used for two very different time zones. In addition, politicians modify the rules for local time zones, especially for daylight savings time, every few years. The only reliable method of describing a local time zone is to specify the numerical difference between the local time zone and UTC.

THEOS supports both the local time zone abbreviations and the specification of “Hours from UTC” designation for the local time zone. Use the [Setup SYSGEN](#) command to define the abbreviations used and your distance from the zero meridian.

■ **Daylight Savings Time and Automatic Adjustment**

THEOS supports automatic adjustment of the system time-of-day clock when daylight savings time is started or stopped in your area. Unfortunately, the rules for determining the date that daylight savings time goes into effect and when it stops vary from country to country. THEOS uses two sets of rules to determine when to adjust the clock:

- ▶ North America (Canada, United States, Mexico): Daylight savings time starts in the morning of the first Sunday in April and stops in the morning of the last Sunday in October.
- ▶ Europe: Daylight savings time (summer time) starts in the morning of the last Sunday in March and stops in the morning of the last Sunday in September.

Although it is recognized that there are many users of THEOS outside of these areas, THEOS only supports the above two rules.

THEOS automatically adjusts the time-of-day clock on the appropriate day if:

1. You have requested automatic adjustment. (Specify “Y” in the SYSGEN configuration field “Adjust Daylight Savings Time.”)
2. Both the standard and daylight time zone names are defined. (Fill in the “Standard Time Zone Abbrev” and the “Daylight Time Zone Abbrev” fields in the SYSGEN configuration.)
3. The “Hours from UTC” has been set to a non-zero value.
4. The time has not already been adjusted for this time-change.

The clock is adjusted by the system boot process, the account logon process and by the CSI when the CSI prompt displays. When the time is adjusted, the rules for North America are used if the “Hours from UTC” specify an offset between five and nine, inclusive. All other offset values cause the rules for Europe to be used.

If the above automatic adjustment rules are not suitable for your location, do not request automatic adjustment.



Do not request automatic adjustment if your system is used with another operating environment such as Windows® 95 because that system may also perform automatic adjustment, which would adjust the clock too much. Also, some computers have a BIOS that performs automatic adjustment.

Even when automatic adjustment is not requested, THEOS changes to the daylight time zone name or standard time zone name on the appropriate day if both of these abbreviations are defined (the clock is not adjusted). If your area does not observe daylight savings time, then do not specify an abbreviation for that field. If you do not want the time zone name shown with the current time, then do not define an abbreviation for the “Standard Time Zone Abbrev” field.

■ **21st Century**

Although the 21st century does not start until January 1, 2001, four-digit year number considerations must be addressed long before that. All date fields in records, displays and reports should be formatted for a four-digit year number. If you don’t, programs and displays will not be able to interpret dates after 1999 accurately. If the year number is used in disk and tape labels, it must also be formatted for a four-digit year number.

The THEOS 32 Version 4 operating system does not have a problem with the 21st century because it can generate and maintain date fields accurately with dates to the year 2100. BASIC language programs can validate dates in the range 1/1/1900 through 12/31/2100 and THEOS recognizes that the year 2000 is a leap year.

A file’s date/time stamp, which reflects the date that the file was last modified, is accurate through the year 2049.

■ The Year 2000 Problem

The year 2000 problem talked about in the media and on the Internet refers to programs that maintain dates with only a two-digit year number. This problem is serious and must be addressed by all programmers and system designers. If not addressed now, dates in files will not be interpreted properly by programs and comparing dates in the different centuries will produce erroneous results.

For a simple example that illustrates the problem, consider a program that needs to determine the age of a person but it has recorded dates with a two-digit year number. The person's birth date is recorded in a file as 11.02.06 (meaning February 6, 1911). When the system's date is 02.02.17 (February 17, 2002), the difference between the two date values should be 31,788 days but will be computed as -4,737 days. The person would not qualify for Social Security or any other retirement program, *etc.*

One solution to the year 2000 problem is to always record dates with a four-digit year number. This requires potentially massive database updates with record expansions and program modifications for all programs that access the database.

The THEOS operating system provides two special environment variables that, when set to appropriate values, allow two-digit year number dates in existing databases and programs to be interpreted as your application requires. These environment variables are **DATEIN** and **DATEOUT**. They control how two-digit year number dates are interpreted on input and how they are formatted for output.

These environment variables are global in nature and affect all users and all programs that use the standard date manipulation functions provided with the operating system and its language support libraries.

DATEIN. This environment variable controls how two-digit year number dates are interpreted by BASIC applications. It may have one of three different values:

- 0 Two-digit year numbers are always interpreted as twentieth-century dates.
- 1 Two-digit year numbers are always interpreted as current-century dates.
- 2 Two-digit year numbers are interpreted temporally. If the difference between today's date and the date interpreted as a twentieth-century date is greater than the difference between

today's date and the date interpreted as a twenty-first-century date, then the date is treated as a twenty-first-century date. Otherwise, it is treated as a twentieth-century date.

Current Date	Date Value	DATEIN		
		0	1	2
1998.7.4	98.2.25	1998.2.25	1998.2.25	1998.2.25
2002.7.4	98.2.25	1998.2.25	2098.2.25	1998.2.25
2050.7.4	98.2.25	1998.2.25	2098.2.25	2098.2.25
1998.7.4	12.2.25	1912.2.25	1912.2.25	2012.2.25
2002.7.4	12.2.25	1912.2.25	2012.2.25	2012.2.25
2050.7.4	12.2.25	1912.2.25	2012.2.25	2012.2.25

DATEOUT. This environment variable controls how two-digit year number dates are displayed by BASIC applications. It may have one of three different values:

- 0** Twentieth-century dates are output with two-digit year numbers, other dates are output with four-digit year numbers.

When a two-digit year date is input to a date-display function, the operation of this mode depends upon the [DATEIN](#) setting and the value of the date that results from that interpretation.

- 1** Dates are always output with four-digit year numbers.
- 2** Dates are always output with two-digit year numbers.

Interpreted Date Value	DATEOUT		
	0	1	2
1998.2.25	98.2.25	1998.2.25	98.2.25
2020.2.25	2020.2.25	2020.2.25	20.2.25

To assure continued operation of your applications far into the future, all new records written to a database should use four-digit year numbers (DATEOUT 1).

































































J: THEOS Character Sets

The table on the following pages lists all of the characters that can be displayed by THEOS.

ANSI #	ANSI name	Input	PC Term Display ¹	Display	Printer	Fax
000	NUL					
001	SOH		☺ ²	HOME		
002	STX		☹ ²	FON	compress on	compress on
003	ETX		♥	FOFF	compress off	compress off
004	EOT		♦	PON	shade on	shade on
005	ENQ		♣	POFF	shade off	shade off
006	ACK		♠	Right		
007	BEL		•	Bell		
008	BS		■	BS		
009	HT		○	Tab		
010	LF		●	LF		
011	VT		♂	ULON	underline on	underline on
012	FF		♀	CLEAR	eject page	eject page
013	CR		♪	CR		
014	SO		♪	RVON	bold on	bold on
015	SI		⚙	RVOFF	bold off	bold off
016	DLE		▶			
017	DC1		◀			
018	DC2		⤴			
019	DC3		!!			
020	DC4		1			
021	NAK		β	KOFF	tall on	Head font
022	SYN		▬	ULOFF	underline off	underline off
023	ETB		⤵	EOL	wide on	Subhead font
024	CAN		↑	EOS	wide off	Subhead off
025	EM		↓	KON	tall off	Head off
026	SUB		→	UP		
027	ESC		←			
028	FS		┐	EU		
029	GS		↔	BON	italic on	italic on
030	RS		▲	BOFF	italic off	italic off
031	US		▼			

¹ These symbols are displayed on PC Terms when monitor mode is in effect.

² On VGA displays, these two characters appear as © and ® respectively.


















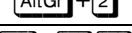












ANSI #	Input	Display	ANSI #	ANSI name	Input	Display
032			064			@
033		!	065			A
034		"	066			B
035		#	067			C
036		\$	068			D
037		%	069			E
038		&	070			F
039		'	071			G
040		(072			H
041	)	073			I
042		*	074			J
043		+	075			K
044		,	076			L
045		-	077			M
046		.	078			N
047		/	079			O
048		0	080			P
049		1	081			Q
050		2	082			R
051		3	083			S
052		4	084			T
053		5	085			U
054		6	086			V
055		7	087			W
056		8	088			X
057		9	089			Y
058		:	090			Z
059		;	091			[
060		<	092			\
061		=	093		]
062		>	094			^
063		?	095			_

ANSI #	ANSI name	Input	Display	ANSI #	ANSI name	PC Term Display ³	Display
096			`	128		Ç	
097			a	129		Ü	
098			b	130		é	SON
099			c	131		â	SOFF
100			d	132		ä	
101			e	133		à	
102			f	134		å	
103			g	135		ç	
104			h	136		ê	
105			i	137		ë	
106			j	138		è	
107			k	139		ï	
108			l	140		î	
109			m	141		ì	
110			n	142		Ä	
111			o	143		Å	
112			p	144		É	
113			q	145		æ	
114			r	146		Æ	
115			s	147		ô	
116			t	148		ö	
117			u	149		ò	
118			v	150		û	
119			w	151		ù	
120			x	152		ÿ	
121			y	153		Ö	
122			z	154		Ü	
123			{	155		ø	
124				156		£	
125			}	157		¥	
126			~	158		₪	
127	DEL			159		f	

³ These symbols are displayed on PC Terms by using the lead-in.

ANSI #	ANSI name	PC Term Display ³	Display	ANSI #	Input	PC Term Display ³	Display
160		á	┐	192	"	À	À
161		í	┑	193	"	Á	Ä
162		ó	┑	194	^	Â	Â
163		ú	┐	195	`	Ã	À
164		ñ	┑	196	`	—	Á
165		Ñ	┑	197	`	+	É
166		ª	┑	198	"	ƒ	ë
167		º	┑	199	^		ê
168		¿	┑	200	`	ℓ	è
169		┐	—	201	`	ℓ	é
170		┑		202	"	≡	ï
171		½	┐	203	^	≡	î
172		¼	┐	204	`		ì
173		ì	┐	205	`	=	í
174		«	┐	206	"	≡	ö
175		»	ℓ	207	"	≡	ö
176			┑	208	^	≡	ô
177			┑	209	`	≡	ò
178			ℓ	210	`	≡	ó
179			≡	211	"	ℓ	Ü
180		┑		212	"	ℓ	ü
181		┑		213	^	'	û
182			≡	214	`	ℓ	ù
183		≡	≡	215	`	≡	ú
184		┑	=	216	,	≡	Ç
185				217	,	┑	ç
186				218	~	┐	Ñ
187		┑		219	~	■	ñ
188		┑		220	E	■	Æ
189		┑		221	e	■	æ
190		┑		222	o	■	Å
191		┑		223	o	■	å

³ These symbols are displayed on PC Terms by using the lead-in.

ANSI #	Input	PC Term Display ³	Display
224	 	α	β
225	 	β	ζ
226	 	Γ	i
227	 	π	\emptyset
228	 	Σ	\pounds
229	 	σ	\pounds
230	 	μ	P_t
231	   	τ	ϵ
232	 	Φ	$\frac{1}{4}$
233	 	Θ	$\frac{1}{2}$
234	 	Ω	\ddot{y}
235	 	ö	§
236	 	∞	\cdot
237	 	\emptyset	2
238		ε	
239		\cap	
240		\equiv	
241		\pm	
242		\geq	
243		\leq	
244		\lceil	
245		\rfloor	
246		\div	
247		\approx	
248		\circ	
249		\bullet	
250		\cdot	
251		$\sqrt{\quad}$	
252		n	
253		2	
254		\blacksquare	
255			

Index

! 42
wild card **137**
Break **71**, 475
Break key, defining 98
Break, B 71
Break, C **71**
Break, D 71
Break, P 72
Break, PrtScreen 72
Break, Q **72**, 347, 479
Break, T 72
Break, Tab 64
Break, W 72
Break, function-key 64
Break, n 64
Caps Lock 64
Ctrl+Alt+Del 72, 487
Ctrl+D 375
Esc 39, 75
F1 **38**, 356, 371
F2 45
F3 42
Num Lock 64, 486
Scroll Lock 64
* (LineEdit prompt) 371, **679**
* wild card **137**
; 37, 614
< 52
<\$endtrange 568
> (CSI prompt) 33
> (i/o redirection) 52
>& 52
>> 52
>>& 52
? wild card **137**
@ wild card **137**
^ (page wait symbol) 72
| 53

A

ABBREV environment switch 40, 49, **98**, 475, 503
Abbreviation
Command name 110
See also: ABBREV environment switches
Account
Name 58, 385
Password 58
Changing 420
System 95, 140, 329, 405
File ownership 96, 142, 332
See also: Accounts, user
account.COMMAND **721**
account.EXEC 388, 389, **721**
account.LOGON **721**
account.REMINDER 388, 452, **721**
Accounts
Backing up 553
Restoring 554
Accounts, user 95
Adding 157, 158, 161
Copying files owned by another 268
CSI prompt 158
Deleting 157, 162, 165
Displaying 156–158, 163–164, 166–167
on printer 157
on screen 158
Environment 95–111
Clearing 475
Default 389
Displaying 474
Setting 474
Switches 40, 98–102, 162, 388
Variables 102–111, 162, 388
File ownership 95, 130, 327, 331, 351, 456
Home subdirectory 159
Logging off of 386
Logging onto 58, 386
Maintaining 156, 162
Name **95**, 387

- Number **95**, 359
- Password 528
- Passwords **96**, 158, 162, 387
 - Changing 420
- Privilege level **97**, 158
 - Display 507
 - five 329, 331, 348, 369, 416, 476, 477, 480, 511, 530, 544, 546, 582
- Renaming files owned by another 456
- Restoring files 460
- Synonyms **96**, 159, 161
- System, *See*: Account, system
- See also*: Commands, ACCOUNT
- ANSI characters 761
- ANSI forms control **377**
- Archive
 - File name 172
 - Incremental 171, 176
 - Label 171, 462
 - Volume 176, 458, 465
 - Displaying 459
- Archiving
 - Differential 171, 175
 - Displaying contents of 173, 174
 - Drives 168, 174, 552
 - Files 168, 552
 - Current account 170
 - Current subdirectory 174
 - by date 170
 - Multivolume 175
 - See also*: Backup and Restore
- Arguments, command, *See*: Command lines, arguments
- ASCII
 - File transfer protocol 451, 472
- Assembly language, *See*: Patching commands, assemble
- Attaching devices 178–193
 - CD-ROMs 191
 - Changing attachment 179
 - Class codes 181
 - COM ports 191, 539
 - Consoles 186, 527
 - Detaching 179, 193, 386
 - Displaying existing attachments 179
 - Drives 188, 538

- Floppy 188
- Hard 188
- Image 189–190
- RAM 188–189
- Flow control 180, 181, 182, 183, 184
- Graphics 539
- Printers 186, 539
 - Class code 186
 - Slave 187, 539
 - Spoiled 184, 187, 519
- Public 183, 386, 527, **539**
- Serial devices 191
- Tapes 191, 539
- Word length 184
- Audio CD-ROMs 212

B

- Background user, *See*: User, background
- Backing up
 - Accounts 553
 - Differential 555
 - Directories 553
 - Disks 553
 - Displaying contents of 559, 560, 561, 563
 - Drives 552, 558, 565
 - Files 552, 553
 - Current account 555
 - Current subdirectory 558
 - by date 558, 559, 562
 - Multivolume 565
- Backup
 - Buffered 195, 196
 - Disk drive 194
 - Image drive 190
 - Incremental 557
 - Library
 - BACKLIB 373
 - Restore from tape 194
 - Stream file 373
 - to floppy 194
 - to tape 194
 - Verify 196
- Backups
 - Comparing 554, 555, 559
 - Creating 553

- Differential 566
- Ejecting media 556, 559, 560, 563
- Error log 556, 559, 560, 563
- Incremental 566
- Restoring 554, 560
- Verifying 554, 558, 563
- Bad sectors, *See*: Disk drives, hard, bad sectors
- Baud rate 180
- Boot drive **24**
- Booting Operating System **23**, 198–199, 449
 - Ctrl**+**Alt**+**Del** 72
 - in maintenance mode 27
 - Initial screen 24
 - THEOS 32 198
 - THEOS 386/486 198
 - with Emergency Boot Diskette 29
- Bootstrap loader 24, 198, 298, 301, 305, 308, 727
- BREAK environment switch **98**, 475, 503
- BREAK key 98
- Break signal
 - Transmitting 71
- Break,Q
 - Disable 100
- Browsing **77**
 - LIST command display 380
 - MORE command display 413
- Bypass
 - window, *See*: Window, bypass
- Bypass, console
 - See*: Console, bypass
- Bypass, printer
 - See*: Printer, bypass

C

- Cache SYNC **200**
 - REBOOT 449
- Calculator
 - CSI 46–47
 - Patch 428
- Calendar display 206
- CASE environment switch **99**
- CD-ROM
 - Attaching 191
 - Audio 212

- Drive 118
 - Ejecting 314
- CDROM.CATALOG 214, **721**
- Character codes 761
- Characters
 - ANSI 761
 - control 761
 - inputting 761
 - international 761
 - line-drawing 761
 - PC-Term 761
 - special symbol 761
- Class codes **67**, 728
 - Attaching devices 181
 - Changing 225
 - Console 67, 186, 225
 - Character sets 281
 - IBM PC 284
 - THEOS 283
 - Cursor addressing 227, 278
 - Display attributes 228, 277
 - Input functions 233, 290, 680
 - International symbols 235, 283
 - Line graphics 234, 282
 - Screen editing 231, 280
 - Scrolling 279
 - Set alpha color 231, 243, 277
 - Testing 276–290
- console 226–236
- Creating 225
- Defining
 - Control character entry 238
- Name 68
- Number 68
- Printer 81, 186, 225
 - Defining 236–237
 - Printer initialization 237
 - Testing 440
- Clean
 - Disk drive 298
 - File space 316
- Clear type-ahead key 72
- Clock
 - Adjusting 34, 388, 535, **757**
 - Limitations 757

- Daylight savings time 34, 388, 732, 734, **757**
- System 34, 755
- Time-of-day 34, 503, 755
- See also:* Time
- Color
 - Names 242
 - Numbers 242
 - Setup 484
 - See also:* Window, color
- Color display attributes
 - Defining 231–232
- COM ports 191, 294
 - Console 191
 - Listing to 376
 - Receiving files 450, 451, 572
 - Same as console 191
 - Self test 191
 - Sending files 471, 472, 572
- Command
 - Forcing user to execute 347
 - Help 356
 - Search sequence 48–51, 479
- Command line **33**, 34–38
 - Arguments 36, 41, 618
 - Case mode 35, 37, 99
 - Comments **37**
 - Editing 38
 - History 45, 533, 722
 - Quotation marks 36, 396, 416
 - Recall 41–45, 722
 - Last 42
 - Prior 43
 - Token 42
 - Repeating 457
 - Tokens 37
- Command name **33**
- Command privilege levels 741
- Command String Interpreter
 - See:* CSI
- Commands
 - Account 156–167
 - Archive 168–177
 - Attach 178–193
 - Backup 194–197
 - Boot32 198–199
 - Boot40 198–199
 - Cache 200–205
 - Calendar 206–208
 - Cat 209–211
 - CD, *See:* CHDIR
 - CDPlayer 212–215
 - Change 216–221
 - ChDir 222–224
 - Classgen 225–239
 - Clear 240
 - Color 241–243
 - COM, *See:* THEO_COM
 - Compare 244–247
 - Compress 248–251
 - CopyFile 252–268
 - Create 269–273
 - CRLF 274–275
 - CRT 276–290
 - Date 291–293
 - Dial 294–295
 - Disk 296–311
 - Echo 312–313
 - Eject 314
 - Erase 315–317
 - Expand 319–323
 - File 324–325
 - FileList 326–341
 - FileType 342–343
 - Find 344–346
 - Force 347–348
 - GetFile 349–353
 - Head 354–355
 - Help 356–357
 - Ident 359
 - Install 360
 - IXDiag 362
 - Keyword 367–369
 - LC, *See:* Lowcase
 - LE, *See:* LineEdit
 - LEdit, *See:* LineEdit
 - Line 370
 - LineEdit 371–374, 679–716
 - List 375–382
 - Load 383–384
 - Logname 385
 - Logoff 386–389

- Logon 386–389
- Look 390–396
- Lowcase 397
- Mailbox 398–400
- MakeBoot 401–405
- MD, *See*: Mkdir
- Message 406–409
- Mkdir 410–411
- More 412–413
- Mount 414
- MSG 415–417
- Number 418–419
- Password 420
- Patch 421–438
- Peek 439
- Printer 440–444
- PutFile 445–447
- PWD 448
- RD, *See*: Rmdir
- Reboot 449
- Receive 450–451
- Reminder 452–453
- Rename 454–456
- Repeat 457
- Restore 458–466
- Rmdir 467–468
- See 469–470
- Send 471–473
- Set 474–481
- Setup 482–498
- Shell 499–500
- Show 501–511
- Sleep 512
- Sort 513–516
- Split 517
- Spooler 518–526
- Start 527–530
 - Return code 747
- Stop 527–530
- Sysgen 531–544
- System 545–546
- Tail 547–548
- Tape 549–551
- TBackup 552–568
- Tee 569
- THEO_COM 570–573

- Touch 574–575
- Tree 576–578
- UC, *See*: Upcase
- Unique 579–581
- Unload 582
- Unnumber 583–584
- Upcase 585
- wBypass 592
- WC, *See*: WordCount
- wClear 592
- wClip 592
- wClose 593
- wFinish 593
- wFrame 593
- WhereIs 586–587
- Who 588–589
- WhoAmI 588–589
- Window Management 590–606
- wInvert 593
- WinWrite 607–608
- wMenu 594
- wMove 597
- wOpen 597
- WordCount 609–611
- wRefresh 598
- wRemove 598
- wRestore 598
- wSave 599
- wSelect 600
- wStat 601
- wSwitch 605
- WW, *See*: WinWrite
- Comparing files 244–247
- Compressing files 248–251
- Compression library file **251**, 319, 325
 - Deleting entries in 249
 - Displaying contents of 249
- Connecting devices, *See*: Attaching devices
- Console 67–77
 - Bypass 68
 - Class code, *See*: Class codes, Console
 - Clearing screen 58, 240
 - clearing screen 762
 - Clearing window 592
 - Colors
 - Display current 241

- Names 242
- Numbers 242
- Setting 241
- Setup 484
- Configuration 485, 729
- Display attributes 73–74, 228
 - Blink 73
 - Cursor on/off 73
 - EXEC language control 637
 - Format 73
 - Monitor mode 73
 - Protect 73
 - Reverse video 74
 - Status line 74
 - Underline 74
- display attributes
 - blink 762
 - cursor on/off 762
 - format 762
 - protect 762
 - reduced intensity 762
 - reverse video 762
 - status line 764
 - underline 762
- Displaying text 313
- Echoing text to file or device 312, 314
- Escape character 68
- Files
 - Terminating 375
- Input functions
 - See:* Class codes, Console
- International symbols
 - See:* Class codes, Console
- Keypad mode 486
- Line graphics
 - See:* Class codes, Console
- Main **67**
 - Mouse 78
- Menu, *See:* Menus
- Monitor mode 73, 277
- Mouse
 - Setup 486
 - Testing 288
 - See also:* Mouse
- Multiple-page browsing 77
- Network
 - Slave printer 83
- Page-wait 72, 412, 661
- Printers 82, 187
- Repeat rate 486
- Same as COM 191
- Screen editing 231
 - Erase unprotected 73
- screen editing
 - erase to end-of-line 762
 - erase to end-of-screen 762
 - erase unprotected 762
- Screen saver 486
- Screen size **70**, 182, 183, 285
- Sessions 60–66
- Special keys 71
- Status line 74, 486
 - Clearing 240
- status line 764
- Testing class code 276–290
- Throughput performance 286
 - to another system 570
- Type-ahead 72
- Contacting THEOS 677
- Control characters 761
 - Displaying 469
- Control keys, *See:* System control keys
- Converting end-of-record marks 274
 - CR to CR,LF 274
 - CR to LF 274
 - CR,LF to CR 274
 - LF to CR 274
- Coordinated Universal Time, *See:* UTC
- Copies
 - Spooler 92
 - Default 91
- Copying files, *See:* Files, Copying
- COPYLIB environment variable 476, 503
- CREATE environment variable **102**, 272
- Creating files, *See:* Files, Creating
- CSI 34–54
 - and Echo command 37
 - Calculator 46–47
 - Case mode
 - Setting 99
 - Command history 722
 - Editing command lines 38

- Help key 38
- History 45, 533, 722
- Prompt **33**
 - CRT attributes 109
 - Setting 107, 158, 479
 - Variables 108
- Recall command line 41–45
- Recall last command line 42
- Recall prior command line 43
- Recall tokens 42
- Shell 499, 687
 - Disable during edit 607
 - Exiting 499
- Current working directory 109, **132**, 327, 448, 476
- Cursor on/off 73, 762

D

- Data files, *See*: Files, direct, indexed, keyed, and stream
- Date 755–758
 - 21st century 758–760
 - Current 25, 503
 - Day name 292
 - Abbreviation 292
 - Display 291–293, 503
 - See also*: DATEOUT environment variable
 - Entering
 - See*: DATEIN environment variable
 - Formats 291
 - American **103**, **755**
 - European **103**, **755**
 - International **103**, **755**
 - Gregorian 207
 - Julian 293
 - Limitations 207, 758
 - Month name 292
 - Abbreviation 293
 - Reminder 452
 - Setting 476
 - System 25, 291, 476, 503, 671, **755**
 - Weekday number 293
 - Year number 756
 - Four-digit 291, 292
 - Two-digit 293
- DATEFORM environment variable **103**, 453, 476, 503
- DATEIN environment variable 503, 759
- DATEOUT environment variable 504, 760
- Daylight savings time, *See*: Clock
- Debug break-point 71
- DECIMAL environment switch **99**
- Decimal is Comma, *See*: DECIMAL environment switch
- Default image file 123
- Default library **134**
- Device
 - Attaching, *See*: Attaching devices
 - Connecting, *See*: Attaching devices
 - Detaching, *See*: Attaching devices, Detaching
 - Driver 178
 - Logical name 178
 - Number 180
 - Sharing 55
- Device names file
 - See*: SYSTEM.THEOS32, .DEVNAMES
- Device,IDE 505
- Device,PCI 507
- Device,SCSI 507
- DIALDIR environment variable **104**
- Differential archive
 - See*: Archiving, Differential
- Differential backups
 - See*: Backing up, Differential
- DigiBoard
 - Configuration 488–490
- Direct files, *See*: Files, direct
- Directories 129–134
 - Backing up 553
 - Default 106, **109**
 - DOS 349
 - Clearing 446
 - Creating 447
 - Erasing 467
 - Libraries 131
 - Listing 326–341, 344
 - Sorted 329, 333, 334, 345, 578
 - Optimization 302, 308
 - Path **132**
 - Renaming 454

- Restoring 554
- Root **129**, 299, 301, 752
 - Clearing 461
 - Size 299, 303, 306, 308, 492
- Searching 344, 586
- Size of 577
- Subdirectories 130
- Tree **132**, 576, 586
 - See also*: Subdirectories
- Disable Break-Q, *See*: QUIT, disable
- Disk caching 124, 200–205
 - and loaded files 384
 - Block size 203
 - Cache size 203, **534**
 - Directory write back 200, 203, **534**
 - Enabling 200
 - Hit rate 203
 - Statistics 202
 - Display 204
 - Status 200, 201, 202
 - Display 203
 - Synchronization 200
 - Write delay 200, 203, **534**
- Disk Drives
 - Hard
 - Configuration 490–493
 - Formatting 491–493
- Disk drives 113–126
 - Allocation 28, 299, 300, 301, 307, 308, 534
 - Map 301
 - CD-ROM 118
 - Changing 414
 - Clean 298
 - Clear 299, 308
 - Cylinders 305
 - Density code 305
 - File system 751–753
 - THEOS/4G **751**
 - file system
 - THEOS 303
 - THEOS/4G 303
 - Floppy 119, 188
 - Formatting 119, **309**, 494–495
 - Quick 119, **310**, 495
 - Unconditional 495
 - Formatting 297, 305–306, 308, 493
 - Fragmentation 299, 308
 - Free space 300, 307
 - Hard 113–116, 188
 - Bad sectors 116
 - Formatting 116
 - Logical disk 115
 - Partition 114, 491
 - Active 114, 493
 - DOS directory 353, 447
 - Extended 114, 115
 - Logical 492
 - Primary 114, 492
 - Removing 493
 - Spanned 115, 493
 - Surface analysis 116, 492
 - Heads 305
 - Image 122–123, 189, 190
 - Creating 122
 - Increment 305
 - Label 28, **300**, 301, 306, 308, 492
 - Listing directory 326–341
 - Mapping 504
 - Misallocation 299, 302, 303, 308
 - Modifying contents of, *See*: Patching, Disk sectors
 - Optimization 302, 308
 - Private 125, 128
 - Public 125, 127
 - RAM 120–121, 188, 189
 - Removable, hard 117
 - Sectors 306
 - Displaying 431
 - Patching 438
 - Reading 431
 - Writing 435
 - Status 296, 298, 299, 303, 307
 - System 538
 - Changing 545
 - Tracks, *See*: Disk drives, Cylinders
- DISK.IMAGE n 122, 123, 189
- Disk-letter, *See*: Files, Disk-letter
- Disks
 - Backing up 553
 - Restoring 554, 560
- Disks, *See*: Disk drives

Displaying

- Another user's console 439
 - Archive volume contents 459, 460
 - Attached devices 179
 - a calendar 206
 - Console connection 588
 - Current working directory 448
 - Date 503
 - Directory tree 576
 - Disk sector 430
 - Environment 474, 501
 - File 430
 - File record 430
 - Hard disk mapping 491
 - Logon information 588
 - Memory usage 506
 - Network connection 588
 - Nonprintable characters 469
 - Program version 510
 - Serial number 508
 - Size of directories 577
 - Spooled report list 525
 - Started users 588
 - System clock 503
 - System configuration file 531
 - System time 508
 - Tape label 549, 550
 - Tasks 508, 509, 510
 - Text on console 313
 - Time-of-day 503
 - User partition 511
 - Version 510
 - Window bypass mode 592
 - Window status 601–604
 - Word count 609
 - See also:* Listing
- Driver, physical device 178
- Drives
- Backing up 552

E

- Echo file
- Closing 312
 - Defining and opening 312
 - Device, defining 312
 - Start/stop 72, 312
- Editing files 679–716
- See also:* Files, Stream, Editing; Line-Edit; Patching
- Ejecting
- CD-ROMs 314
 - Tape or disk 314, 556, 559, 560, 563
- Emergency Boot Diskette 29
- Adding files to 404
 - Creating 401
 - Files included 403
- End-of-file
- DOS 446
 - Standard input is console 210
- Environment, *See:* Accounts, user, environment
- Environment switches
- ABBREV 40, 49, **98**, 475, 503
 - BREAK **98**, 475, 503
 - CASE **99**
 - Changing 474
 - Clearing 475
 - DECIMAL **99**
 - Displaying 474, 501
 - LANG **99**
 - MSG **100**, 478, 506
 - QUIT 72, **100**, 479, 507
 - RDYMSG 65, **101**, 479, 507
 - STDSYN 40, 49, **101**
 - User account 162
 - WORK **102**, 480, 511
- Environment variables
- Changing 474
 - Clearing 475
 - COPYLIB 476, 503
 - CREATE **102**, 272
 - DATEFORM **103**, 453, 476, 503
 - DATEIN 503, 759
 - DATEOUT 504, 760
 - DIALDIR **104**

- Displaying 474, 501
- EXEC language access 664
- FILETYPE **105**, 381, 608
- HISTORY **105**, 504, 533
- HOME **106**
- LIBRARY **110**, 134, 327, 478, 505
- LINEGRAPH **107**
- LINKLIB **111**, 478, 505
- OBJLIB **111**, 478, 506
- PATH 50, **111**, 478, 506
- PROMPT **107**, 158, 479, 507
- SEARCH 479, 507
- SUBDIR 109, 159, 476, 479, 503, 508
- SYNONYM 49, **110**, 479, 508
- User account 162
- User-defined 511
- Erasing files 315
 - After compression 250
 - See also:* Files, Erasing
- Error log
 - Backup 556, 559, 560, 563
- Errors
 - Color 485
 - Command not found 49, 51
 - See also:* Standard error device; Messages
- Escape character 68
- EXEC language 613–675
 - &Begstack 374, 594, **629**, 651, 657
 - &Begtype **630**
 - &Break **631**, 643, 660, 662
 - &Call **632**, 640, 655
 - &Cat function **663**
 - &Continue **633**, 643
 - &Control **633**
 - Autocr **634**
 - Case **634**, 654
 - Nostack **634**
 - Notrace **634**
 - Off 331, **635**
 - On **635**
 - Prompt **636**
 - Stack **636**
 - Trace **636**
 - &Crt **637**
 - &Disksize function **663**
 - &Else **639**, 645
 - &ElseIf **639**, 639, 646, 647
 - &End 629, **639**
 - &Env function **664**
 - &Error **640**
 - &Esc **641**, 654
 - O **641**
 - P **641**
 - W **641**, 661
 - &Exist function **664**
 - &Filesize function **665**
 - &For **642**, 648, 659, 660, 661
 - &Goto 615, **643**, 655
 - &If 639, **644–647**, 647
 - &IfEnd 639, **647**
 - &Index function **665**
 - &Len function **666**
 - &Lit function **667**
 - &Loop 642, **648**, 650, 654, 659, 661, 662
 - &Menu **648**
 - &MenuIX variable 649
 - &Next 648, **650**, 650, 654, 659, 661, 662
 - &Null function **667**
 - &Quit **650**, 749
 - &Read 628, 630, 634, 636, **651–654**
 - &Repeat 642, 648, **654**, 659, 661, 662
 - &Retcode function 597, 640, **668**, 749
 - &Return **655**
 - &Skip **655**
 - &Sleep **655**
 - &Space **656**
 - &Stack 594, 628, 651, **656**
 - &Sub function **668**
 - &System function **670**
 - Account **670**
 - CWD **670**
 - Date **671**
 - Day **671**
 - PID **672**
 - Port **672**
 - Priv **672**
 - Serial **672**
 - Tasks **673**
 - Time **673**
 - TOD **673**
 - Today **674**

- UID **674**
- Users **674**
- Weekday **675**
- &Typ function **675**
- &Type **657**
- &Until 648, **659**, 661
- &Vary 648, **660**, 661
- &Wait 641, **661**
- &Wend 642, 648, 650, 654, 659, **661**, 662
- &While 648, 661, **661**
- Assignment 629
- Command-line arguments 618, 629, 652
 - Number of 665
 - Reassigning 653
- Conditional execution 639, 644
- Constants 616
- Displaying text 630, 657
- Environment variables 664
- Error control 640
- Executing input line 652
- Expressions 619
 - Evaluating 625
 - Logical 624
 - Numeric 621
 - Relational 622
 - String 619
 - String concatenation 619, 663
 - String subfield 619
 - Substring 621
- Functions 617
- Getting
 - Account name 670
 - Account number 674
 - Current working directory 670
 - Disk size 663
 - File size 665
 - Number of tasks 673
 - Number of users 674
 - Partition number 672
 - Privilege level 672
 - Serial number 672
 - System date 671, 674
 - System time 673
 - Time of day 673
 - Weekday name 675

- Weekday number 671
- Input stack **627**, 629, 634, 636, 651, 654, 656
- Keywords 617
- Label **615**
- LineEdit usage 374
- Looping control 631, 633, 642, 648, 650, 654, 659, 661
- Menu
 - with wMenu 594–597
- Operators 619
 - Modulo 622
 - Precedence 626
- Page-wait 661
- Pausing execution 655
- Program termination 388, 650
- Remarks 654
- Silent execution 635
- Subroutines **632**, 655
- Variables **616**, 629, 652
 - User-defined 616, 652
- Expanding files 319–323

F

- File attributes **142**
 - Access protections **143**
 - Default 102
 - Displaying 332
 - Erase **143**, 220, 317, 456
 - Execute **143**, 220
 - Private **144**
 - Changing 218
 - Private read 221
 - Read **143**, 220, 382, 456
 - DOS file 446
 - Shared **144**, 220, 317
 - Changing 219
 - Write **143**, 220, 456
 - Account ownership 96, 130, **142**
 - Changing 217, 218
 - Changing **216**
 - Checksum 330
 - Displaying 336
 - Growth Factor **144**
 - Growth factor 271, 331, 336
 - Changing 218

- Hidden 102, **143**, 220
 - DOS file export 446
 - Listing 331
- Last change date 258, 259, 260, 263, 264, 272, 321, 322, 335, 336, 346, 587, 755
 - Changing 219, 574
 - in listing heading 378
 - Limitations 758
- Modified 102, **143**, 220, 248, 250, 258, 260, 272
- System
 - DOS file 447
- File name
 - Disk-letter **135**, 336
 - File-name **135**, 336
 - File-type **135**, 336
 - Member-name **135**, 336
 - Path **136**
 - Search sequence 101, 140
 - Typeless **135**
 - Wild card **136**
 - See also:* Wild cards
- File-name, *See:* Files, File-name
- Files
 - Appending 209–211, 255, 418, 514, 569, 583
 - Archiving 168
 - Backing up 552, 553
 - Clearing contents 270
 - Closing 386
 - Comparing 244, 247
 - ASCII 246
 - Binary 247
 - Compressing 248–251
 - ASCII 251
 - Binary 251
 - Concatenating, *See:* Files, Appending
 - Copying 252–268
 - by record 255, 257, 262
 - direct 267
 - Entire subdirectory 262
 - Indexed 257, 262, 267
 - Keyed 257, 262, 267
 - Older 260
 - Owned by another account 268
 - Public 260
 - Stream 257, 262, 264–267
 - Text to standard output device 370
 - Translate to lowercase 258
 - Translate to uppercase 263
 - Truncating stream file records 263
 - Unique records 579
 - Using last change date 263, 264
 - Verify 259, 263
 - with manipulation of records 261, 263, 264–267
 - Creating 269–273
 - Contiguous allocation 271, 299
 - Direct 270, 271
 - in another account 272
 - Indexed 270, 271–272
 - Keyed 270, 271–272
 - Library 269, 270–271
 - Subdirectory 269, 410
 - Deleting records 429
 - Direct **141**, 325, 377, 379, 431, 435
 - Number of records 271, 330
 - Patching 437
 - Record length 271, 336
 - Disk-letter **135**, 336
 - Dividing 517
 - Duplicating 569
 - Erasing 315, 467
 - After compression 250
 - All 298, 308
 - Clean 316
 - Restrictions 317
 - Expanding 319–323
 - Exporting 256
 - DOS 445
 - File-name **135**, 336
 - File-type **135**, 336
 - Default 105
 - Flat **131**, 327
 - Fragmented 299, 308
 - Importing
 - DOS 349, 350, 352
 - THEOS 8 349, 350
 - Indexed **142**, 325, 377, 379, 431, 435
 - Key length 272, 336
 - Number of records 271, 330

- Patching 437
- Record length 272, 336
- Keyed **141**, 325, 377, 379, 431, 435
 - Key length 272, 336
 - Number of records 271, 330
 - Patching 437
 - Record length 272, 336
- Listing 375–382, 412
 - Beginning 354
 - Binary 376, 380–381
 - Ending 547
 - Hexadecimal 378
- Loading into memory 383, 542, 582
- Locating 344, 586
- Locking 386
- Member-name 336
- Modified attribute 171, 177, 555, 556, 557, 564
- Program, *See*: Program files
- Receiving, *See*: Transferring files
- Renaming 454
- Restoring 458, 554, 560
- Search sequence 140, 479
- Size of 336, 432
- Status 297
- Stream **141**, 325, 342, 354, 377, 379, 397, 418, 470, 579, 585
 - Converting end-of-record marks 274, 275
 - Counting words, etc. 609
 - Creating 371, 679
 - Editing 371, 607–608
 - See also*: LineEdit
 - Patching 437
 - Searching 390
 - Sorting 513–516
 - Splitting 517
- Transportable 342
- Typeless 105, **135**
- Types 336
 - Compression library 325
 - Direct 325
 - Directory 325
 - Displaying 324
 - Indexed 325
 - Keyed 325

- Library 325
- Program 325
- Program object 325
- Program source 325
- Script 325
- Stream 325
- FILETYPE environment variable **105**, 381, 608
- File-type, *See*: Files, File-type
- Filter **53**
- Flat files **131**
- Flow control 181
 - Bidirectional 180
 - Clear to send (CTS) 181
 - Data terminal ready (DTR) 181
 - End transmission (ETX) 181
 - Form feed delay 182
 - Line feed delay 182
 - Parity 182, 183
 - XON/XOFF 184
- Foreground user, *See*: User, Foreground
- Forms
 - Spooler **87**, 89–94, 525, 544
 - Extended 185, 526
 - spooler 520
- FOUND.EXEC
 - Appending to 391
 - Creating 391
 - with Change command 216
 - with Compress command 249
 - with Expand command 320
 - with File command 324
 - with List command 364, 376, 556
 - with Send command 471
- Four-digit year 759, 760
- Free space
 - Disk drive 300, 307
- Function keys
 - Defining 233

G

- GMT, *See*: UTC
- Graphics, *See*: Virtual Device Interface

H

Help 356–357
 Color 485
 LineEdit 695
 Hexadecimal
 File listing 378, 380–381
 Highlight bar 75
 HISTORY environment variable **105**, 504,
 533
 History file
 Boot 738
 Command **722**
 History files
 Boot log **721**
 Hold
 Spooler 92
 Default 91
 HOME environment variable **106**
 HOME subdirectory
 setting 159
 Hot-key **76**
 HTTP
 Log file 738, 739

I

I/O redirection **51**
 Cat command 209
 Head command 354
 Line command 318, 370
 Lowcase command 397
 More command 412
 Number command 418
 See command 469
 Sort command 513
 Split command 517
 Symbols 52
 Tail command 547
 Tee command 569
 Unique command 579
 Unnumber command 583
 Uppcase command 585
 See also: Standard input and output de-
 vices
 IBM PC character set 284

Image disk **122**, 122–123
 Creating 122
 Default file name 123
 Sysgen 538
 Using 123
 Image drives 189, 190
 Incremental archive
 See: Archive, Incremental
 See: Backups, Incremental
 Indexed files, *See:* Files, Indexed
 Inputting characters 761
 INSTALL.EXEC 360
 Installing software, *See:* Commands, Install
 International symbols 761
 Defining 235
 Printer 443
 Interrupt Request 505
 IRQ 505

K

Key words
 Listing 367
 Maintaining 367
 See also: SYSTEM.THEOS32, .KEY-
 WORDS
 Keyed files, *See:* Files, keyed

L

Label
 Backup volume 557
 LANG environment switch **99**
 Language code 535
 Key words 367
 See also: LANG environment switch
 Lateral logon 387
 Libraries 131
 Backup 373
 Changing size of 270
 Creating, *See:* Files, Creating
 Default 110, 327
 LINKLIB 111
 OBJLIB 111
 PATH 111
 Member files **131**
 Renaming 454

- Size of 336
- LIBRARY environment variable **110**, 134, 327, 478, 505
- Limits
 - Disk drive size 751
 - Number of consoles 66
 - Number of sessions 66
 - Number of tasks 66
 - Number of users 66
 - Reminder message length 452
 - Subdirectory levels 411
- Line-drawing graphics 107, 282, 443, 761
 - Defining 234
- LineEdit
 - Changing text 685
 - all 685
 - Delimiter 682
 - Clipboard 694, 703, **704**
 - Combining text lines 687
 - Command line
 - Editing 680
 - Command Summary 372
 - Commands
 - ; **713**
 - ? **712**
 - again **683**
 - Case **684**
 - Change **685**–686
 - Column **686**
 - Combine **687**
 - Comment **713**
 - CSI **687**
 - Delete **688**–**689**
 - Down **690**–691
 - Dup **692**
 - Error **715**
 - File **692**
 - Find **693**
 - Get **694**
 - Help **695**
 - Input **696**
 - Length **697**
 - List **697**
 - Locate **698**
 - Macros **713**
 - Modify 699–700

- Name **701**
- Next **702**
- Page **702**
- Put **703**
- PutD **704**
- Quit **705**
- Replace **706**
- Save **707**
- Skip **715**
- Split **708**
- Tabset **709**
- Top **709**
- Type **710**
- Up **711**
- Verify 685, **712**
- x **713**
- y **713**
- z **713**
- commands
 - Bottom **683**
- Comments 713
- Copy and paste 704
- Copy to clipboard 703, 704
- Copy to file 703, 704
- CSI shell 687
- Current line **681**
- Cut and paste 704
- Deleting text 688, 704
 - Conditional 688
- Delimiter **682**
- Displaying
 - Changes 681, 685, 687, 712
 - Help information 695
 - Input case mode 684
 - Last command 712
 - Last find text 693
 - Last locate string 698
 - Length of file 697
 - Line ruler 686
 - Lines 697, 702, 710
 - Locate string 690
 - Macro definition 714
 - Name of file 701
 - Screen full of text 702
 - Split line text 708
 - Tab stops 709

- Verify status 712
 - Dividing text line 708
 - Duplicating lines 692
 - Exiting 692, **705**
 - Input
 - Case mode 684
 - Lines 696
 - Multiple lines 696, 706
 - Locating text 693
 - Macros 713–716
 - Defining 713, **714**
 - Defining from file 713
 - Displaying definition 714
 - Error command 715
 - Executing 714
 - Executing multiple times 714
 - Skip command 715
 - Merge from file 694
 - Modify command keys 700
 - Modify mode 699
 - Modifying text 699
 - Naming file **701**
 - During save 692, 707
 - Paste operation, *See*: LineEdit, Commands, Get
 - Positioning
 - Conditional 690
 - Down 690, 702
 - to end-of-file 683
 - to line containing text 690, 698
 - to line starting with text 693
 - to top-of-file 709
 - Up 711
 - Prompt 679
 - Remarks 713
 - Repeating command 683
 - Replacing text lines 706
 - Return code 705
 - Saving file 692, 707
 - Splitting text line 708
 - Tab stops 709
 - LINEGRAPH environment variable **107**
 - LINKLIB environment variable **111**, 478, 505
 - Listing
 - Directory 326–341
 - Files 412, 418, 469
 - Beginning of 354
 - Binary 380–381
 - Display controls 380
 - Ending of 547
 - Log file
 - Boot 738
 - HTTP 739
 - Logical device name 178
 - Logical disk, *See*: Disk drives, Hard, Logical disk
 - Logon 387
 - Automatic 528, **541**
 - Lateral **387**
 - Prompt 387
 - Logon EXEC **721**
 - Lowercase
 - File translation 258, 397
-
- ## M
- Mailbox 34, 398
 - Purging 400
 - Reading 399
 - Removing messages 399
 - Main console
 - Reboot system 72
 - See also*: Console, Main
 - Maintenance mode 535, 544
 - Starting in 27
 - Manipulating records while copying 261, 263, 264–267
 - Math coprocessor, *See*: System configuration
 - Member-name, *See*: Files, Member-name
 - Menus 75–77, 594–597, 648–650
 - Color 485
 - Highlight bar 75
 - Horizontal line 594
 - Hot-keys 76, 596
 - Scroll bar 75, 79
 - Selecting with mouse 79
 - Selection keys 75, 596
 - Underlined letter 76
 - Messages
 - Command not found 49, 51
 - Disk drive changed, need "XXXX" 414

- Insufficient privilege 97
- Receive enable 100, 416, 478
- Reminder 452
 - Deleting 453
- Removing 399
- Retrieving 398, 399
- Sending 398, 415
 - Large 399
 - to all 416
- Modem
 - Dialing 294, 571, 572
 - Disconnect 572
 - Initialization 57, 529, 733
 - Receiving files 450
 - Sending files 471
- Monitor mode
 - See:* Console, Monitor mode
- Mouse 78–79
 - Click 79
 - Double click 79
 - Drag 79
 - Main console 78
 - Setup 78, **486**
 - Terminal 78
 - Testing 288
- MSG environment switch **100**, 478, 506
- MultiBoot, *See:* THEOS MultiBoot
- Multisessions 56, **60**, 537
 - Color setup 484
 - Configuration 734
 - Session manager 734
 - Session number 65
 - Slave printer 82
 - Starting 62, 541
 - Automatic 62
 - Manual 63, 527
 - Switching between 64, 65
 - Disabling 605
- Multitasking
 - See also:* Tasks
- Multitasking **56**
- Multiuser **55**, 537
 - Priority 478, 507
 - Restrictions 66
 - Archiving 172
 - Backing up 557

- Backup 196
- Disk drive formatting 302
- PowerStep 32
- Restoring 462
- Starting 56–59, 527, 541
- Stopping 59, 527, 529
- See also:* Network

N

- Network
 - Configuration file 732
 - Console
 - Slave printer 83
- Nucleus, operating system 733
- Number display
 - American 99
 - European 99
- Numbering lines 378, 418
 - See also:* Unnumbering lines

O

- OBJLIB environment variable **111**, 478, 506
- Operating system
 - Nucleus 733
 - Patch level **25**
 - Serial number **25**, 31, 508
 - Version 510

P

- Page-wait toggle 72
 - See also:* Console, Page-wait
- Parity, *See:* Flow control, Parity
- Parsing **35**
- Partition 672
 - Number **57**
 - User **56**, 527
 - See also:* Disk drives, Hard, Partition
- Password, *See:* Accounts, User, Password
- Patch level, operating system **25**
- Patching
 - Command mode 425
 - Commands 427–437
 - Assemble 427
 - Calculator 428

- Checksum 428
- Code 428
- Compare 429
- Data 429
- Delete 429
- Display 430
- End 430
- Fill 431
- Get 431
- Help 431
- Key 431
- Length 432
- Move 433
- Patch level 434
- Put 435
- Quit 435
- Replace 435
- Search 436
- Set 436
- Use 437
- Video mode 437
- Data files 421
 - binary 421
- Direct files 437
- Disk sectors 421
- Exiting 430, 435
- Expressions 425
- Full-screen mode 422
- Indexed files 437
- Keyed files 437
- Program files 421
- Sectors 438
- Stream files 437
- Video mode commands 423
- Path 49, 50, 111, **136**
 - Directory **132**
 - See also:* File name, Path
- PATH environment variable 50, **111**, 478, 506
- Pausing execution 512
 - in EXEC program 655
- PCI 507
- PC-Term
 - special symbols 761
- Physical device name 178
- Pipes 53–54, 354, 375, 397, 413, 418, 470, 580, 585
 - See also:* Filter
- Playing audio CDs 212
- PowerStep 25, 32, 66
- Print screen image 72
 - See also:* Echo file
- Printer 81–86
 - Automatic line-feed 180
 - Bypass 84–86
 - File 86
 - Multiple-character 85
 - Single-character 85
 - Class code, *See:* Class codes, Printer
 - Display attributes
 - Bold 441
 - Compress 441
 - Double size 441
 - Italic 441
 - Shade 441
 - Underline 441
 - Initialization 186, 237
 - International symbols 443
 - Line length 182
 - Line-drawing graphics 443
 - Page length 182, 183
 - Private 82–83
 - Slave 82, 187, 539
 - Enable 231
 - Public 84–94, 539
 - Spooler 87–94, 518–526, 539, 543, 730
 - Changing status 519
 - Copies 92, 524
 - Default 91
 - Forms **87**, 89–94, 520, 525, 544
 - Extended 185, 526
 - Hold 92, 524
 - Default 91
 - Initialization 521
 - Queue
 - Default 386
 - listing 525
 - Queues **88**, 89–94, 524, 525
 - Default 90, 544
 - Overriding 92
 - Extended 90, 92, 185, 526
 - Standard 89, 92

- Report
 - Aborting print 523
 - Alignment 523
 - Changing 519
 - Deleting 524
 - Printing 525
 - Reprinting 523
- Report name 92
 - Changing 524
 - Default 92
- Secure mode 94, 544
- Starting 521, 543
- StartingStarting spooler
 - See:* Printer, Spooler, Starting
- Status 525
- Stopping 522
- SYSTEM.SPOOLER library 520, 543, 726
- Terminating 521
- Verify 522, 544
- Verifying 544
- SpoolerStarting 543
- Testing 440
- Priority, *See:* Multiuser, Priority
- Privilege level, *See:* Accounts, User, Privilege level
- Program
 - Search sequence 48–51, 383, 479, 582
- Program cancel key 71
- Program files **142**
 - Creating EXEC 628
 - Key 331, 337
 - Loading into memory 383, 542, 582
 - Modifying 421
 - Patch level
 - changing 218, 434
 - displaying 337, 434
 - Patching 421, 438
 - Privilege level **97**
 - Changing 218
 - Displaying 332, 337
 - Serial number
 - Displaying 332, 337
 - Size 336
 - Version 334
 - Displaying 337

- Version date
 - Displaying 337
- Viewing 421
 - See also:* Patching
- Program filters, *See:* Filter
- PROMPT environment variable **107**, 158, 479, 507

- Prompts
 - Color 485
 - CopyFile command 255
 - CSI 33
 - Erase command 315
 - EXEC programs 636
 - Keyword command 367
 - LineEdit command 371, 679
 - Logon command 387
 - Mailbox command 398
 - Message command 406
 - More command 412
 - MSG command 415
 - Password command 420
 - Patch command 425
 - Reboot 487
 - Reboot command 449
 - Reminder command 452
 - Restore command 461
 - Rmdir command 467

- Protocol
 - ASCII file transfer 451, 472
 - THEOS file transfer 451, 472, 473
 - XMODEM file transfer 451, 473
 - YMODEM file transfer 451, 473
- Public account, *See:* Account, System

Q

- QUEUE=\$
 - See:* Queues, Spooler, Extended
- Queues
 - Spooler **88**, 89–94, 386, 524, 525
 - Default 90, 386
 - Overriding 92
 - Extended 90, 185, 526
 - Standard 89
- QUIT environment switch 72, **100**, 479, 507
- Quit key 71

R

RAM drives 188, 189
 Random files, *See*: Files, Direct, Indexed or Keyed
 RDYMSG environment switch 65, **101**, 479, 507
 See also: <DefaultParaFont> Return codes
 Ready message 101, 333
 Enabling display 101, 748
 Reboot, *See*: Booting Operating System
 Receive messages
 See: MSG environment switch
 Receiving files, *See*: Transferring files
 Records
 Counting duplicate 580
 Deleting 429
 Locking 386
 Omitting duplicate 579
 Reading 431
 Unique 579
 Writing 435
 Redirecting input or output
 See: I/O redirection
 Regular expressions 392–395
 Anchors 393
 Character sets 394
 Control characters 392
 Literal characters 392
 Metacharacters 392
 Repeat counts 395
 Wild cards 394
 Reminder messages, *See*: Messages, Reminder
 Report name
 Spooler 92
 Reports, spooled, *See*: Printer, Spooler, Report
 Restoring
 Accounts 554
 Backups 560
 Directories 554
 Disks 554
 Drives 458, 560
 Files 458, 459, 554, 560

Volume 465, 560
 Return codes **747**, 747–750
 and Start command 747
 and wildcards 748
 RDYMSG 748
 Setting 749
 in BASIC program 749
 in EXEC 749
 Root directory, *See*: Directories, Root

S

Screen editing commands
 Defining 231
 Screen saver 486
 Scroll bar 75
 SCSI 507
 SEARCH environment variable **101**, 479, 507, 553
 Search sequence
 Files 140
 Programs 48–51
 See also: SEARCH environment variable
 Searching files
 for text **390**
 Secure mode
 Spooler 94, 544
 SELECTED.EXEC
 Creating 330, 338–339, 351
 without &Control off 331
 with Change command 216
 with Compress command 249
 with Expand command 320
 with File command 324
 with FileList command 329
 with List command 364, 376, 556
 with Look command 391
 with Send command 471
 SELECTED.FILES
 Creating 330, 339
 with Change command 216
 with Compress command 249
 with Expand command 320
 with File command 324
 with FileList command 329
 with List command 364, 376, 556

- with Look command 391
 - with Send command 471
- SELF devices 192
- Sending files, *See*: Transferring files
- Sequential files, *See*: Files, Stream
- Serial number
 - Operating system **25**, 31
 - Program 31, 332
- Sessions, *See*: Multisessions
- Setup menu 482
- Shell, *See*: CSI, shell
- Single-user **55**
 - Returning to 30
 - See also*: Maintenance mode
- Slave printer, *See*: Printer, Private, Slave
- Sleep 512
- Sorting files, *See*: Files, stream, Sorting
- Spooler, *See*: Printer, Spooler
- Standard error device **51**
 - Redirect (>&) **52**
 - Redirect append (>>&) **52**
- Standard error display
 - Cat command 210
 - Disk command 304
- Standard input device **51**
 - Redirect (<) **52**
 - See also*: Pipes
- Standard output device **51**, 59
 - Copying text line to 370
 - Redirect (>) **52**
 - Redirect append (>>) **52**
 - See also*: Pipes
- Standard output display
 - Archive command 173
 - Attach command 179
 - Cat command 209
 - Change command 219
 - Compare command 245
 - Compress command 249
 - CopyFile command 263
 - CRLF command 275
 - Date command 291
 - Disk command 304
 - Echo command 312
 - Erase command 317
 - Expand command 322
- File command 324
- FileList command 343
- Find command 344
- Head command 354
- Ident command 359
- Line command 370
- List command 375
- Logname command 385
- Lowcase command 397
- Message command 407
- More command 412
- Number command 418
- PWD command 448
- Rename command 456
- Restore command 464
- RmDir command 468
- See command 469
- Sort command 513
- Spooler command 525
- Tail command 547
- Tee command 569
- Tree command 576
- Unique command 579
- Unnumber command 583
- Uppcase command 585
- WordCount command 609
- Standard synonym
 - See*: STDSYN environment switch
- Starting users, *See*: Multiuser, Starting
- stderr, *See*: Standard error device
- stdin, *See*: Standard input device
- stdout, *See*: Standard output device
- STDSYN environment switch 40, 49, **101**
- Stream files, *See*: Files, stream
- SUBDIR environment variable 109, 159, 476, 479, 503, 508
- Subdirectory **130**
 - Account ownership 130
 - Copying entire 262
 - Creating, *See*: Files, Creating
 - Current 132, 448, 476, 479, 576
 - Parent 133
 - Path
 - .. 222
 - Partial specification 223
 - Relative path 133, 222

- .. 133
- Support
 - Contacting THEOS 677
- Surface analysis, *See*: Disk drives, Hard,
 - Surface analysis
- Symbols, character 761
- Synonym
 - Command name 110
- SYNONYM environment variable 49, **110**, 479, 508
- SYS.CMD32 198
- SYS.CMD386 198, 466
- SYS.HELP32 198
- SYS.HELP386 198, 466
- SYS.MENU32 198
- SYS.MENU386 198, 466
- SYS.THEOS32 198
- SYS.THEOS386 198, 466
- System account, *See*: Account, system
- System cancel key 72
- System configuration 734
 - Adjust Daylight Savings Time 535, 757
 - Allow Maintenance Mode 535, 544
 - Cache Write Delay 534
 - Daylight Time Zone Abbrev 536, 757
 - Default Language Code 291
 - Devices 539
 - Directory Write Back 534
 - Disk cache 534
 - Disk drives 538
 - File 529, 531, **729**
 - File Allocation Auto Test **534**
 - Hours from GMT 536, 757
 - Initialization 25
 - Language code 535
 - Loading files into memory 542
 - Math processor 533
 - Memory usage 506
 - Number of tasks 66, 537
 - Printer spooler 88, 543
 - Save History Information 504, 533
 - Standard Time Zone Abbrev 291, 535, 757
 - Starting sessions 62, 541
 - Starting users 57
 - System Identification 505, 532
- System control keys 71
 - Break signal 71
 - Clear type-ahead 72
 - Debug break-point 71
 - Echo file toggle 72, 641
 - Output suppress 71, 641
 - Page-wait toggle 72, 641
 - Print screen 72
 - Program cancel 71
 - Reboot system 72
 - System cancel 72
- System date, *See*: Date, System
- System files
 - NOSYSFILES option 466, 561
- System name 505
- System setup 482–498
- System time, *See*: Time, System
- SYSTEM.BOOTLOG **721**
- SYSTEM.CHIST_{nnn} **722**
- SYSTEM.CMD32 198, 383, 466, 582, **722**
- SYSTEM.CMD386 198, 466
- SYSTEM.HELP32 198, 356, 466, **722**
 - ._COMMAND 356
- SYSTEM.HELP386 198, 466
- SYSTEM.HISTORY 106, 477, **722**
- SYSTEM.LOGON 388, **723**
- SYSTEM.MAILBOX 400, 466, **723**
- SYSTEM.MENU32 198, 466, **724**
 - .LOGON 387, **724**
 - .MSG 417, **724**
 - .SHOW_VER **724**
- SYSTEM.MENU386 198, 466
- SYSTEM.MODEM 57, 529, **725**
- SYSTEM.REMINDER 388, 452, **726**
- SYSTEM.SPOOLER 94, 466, 521, 522, 543, 544, **726**
 - Creating 520
- SYSTEM.THEOS32 198, 383, 466, 582, **726**
 - .ACCOUNT **726**
 - .B3220* **727**
 - .B32RT* **727**
 - .B386* **727**
 - .BOOTER1 **727**
 - .BOOTER2 **727**, 729
 - .BOOTER3 **727**
 - .BOOTMSG 25

- .CLASS **728**
- .CLASS_{nnn} 58, 225
- .CLOCK **729**
- .COLORCFG 242, **729**
- .CONFIG 531, **729**
- .CRTCFG **729**
- .CSI 383, 545, **729**
- .DESPOOL **730**
- .DEVNAMES 57, 68, 81, 180, 193, 383, 527, 729, **731**
- .DEV_{nnn} **730**
- .DTIME **732**
- .FIXDEV **732**
- .FORM_{nnn} **732**
- .HOSTS **732**
- .KEYWORD 383, 545, **733**
 - Displaying 367
 - Maintaining 367
- .MESSAGE 292, 383, 545, 674, 675, **733**, 747
 - Conditional expressions 408
 - Display codes 407
 - Displaying 406, 407
 - Maintaining 406
 - Record limit 407
 - Syntax of records 407
 - Variables 408
 - Video attributes 408
- .MODEM 57, 529, **733**
- .NUCLEUS **733**
- .PATCHDOS **733**
- .SESSMAN **734**
- .SET_{aaaaa} **734**
- .SMCFG **734**
- .STARTCFG 25
- .STIME **734**
- .SYNONYM 383, **734**
- .VDI_{nnn} **735**
- SYSTEM.TEOS386 198, 466

T

- Tape drives 127, 549–551
 - Eject 550, 551
 - Ejecting 314
 - Formatting 550
 - Header 549
 - Label 550
 - Rewind 549, 550
 - Tension 551
 - Using 127–128
 - Write tape mark 551
- Tape, Mapping 508
- Tasks
 - Active 508, 509
 - Number of 537
 - Priority 478, 507
- TBackup
 - Label 557
 - Volume 564
- Telephone dialing 294, 571, 572
- Terminal
 - Emulation 570
- Testing
 - COM port communications 191
 - Console class code 276
 - Console screen sizes 70
 - Disk drives 302, 304, 308
 - File allocation 28
 - File existence in EXEC 664
 - Printer class code 440
 - Tape readability 551
- TFS, *See*: THEOS File Save file
- THEOS
 - File transfer protocol 451, 472, 473
- THEOS 32
 - Booting 198
- THEOS 386/486
 - Booting 198
- THEOS File Save file
 - Convert from 342
 - Convert to 342
- THEOS MultiBoot **23**, 493
 - Removing 23
- THEOS Software Portal **31**
- THEOS/4G, *See*: Disk drives

File system
 Time 755–758
 12-hour 291, 293
 24-hour 292
 Current 25
 Display 291–293, 503, 508
 Format 291, **756**
 Setting 480
 System 25, 291, 480, 503, 508, 673, **755**
 UTC **756**
 Zones 291, 535, 536, 756
 See also: Clock
 Time stamp, *See:* File attributes, Last
 change date
 Token **35**
 Tokenization, *See:* Parsing
 Tokens 37, 616
 Case mode 36
 Recall 42
 Transferring files
 ASCII protocol 451, 472
 from another system 450, 572
 THEOS protocol 451, 472, 473
 to another system 471, 572
 XMODEM protocol 451, 473
 YMODEM protocol 451, 473
 Transporting files 342
 Tree, directory **132**
 Two-digit year 759, 760
 Type-ahead buffer 72

U

Unnumbering lines 583
 See also: Numbering lines
 Uppercase
 File translation 263, 585
 User **55**
 Automatic logon 58, 528, 541
 Background
 Starting 58, 527, 528
 Stopping 59, 527, 528, 529
 Command history 504
 Displaying 510
 Displaying active 508, 509
 Forcing 347
 Foreground **57**

Starting 57, 527, 541
 Stopping 59, 527, 529
 Number of 537
 Partition **56**, 95, 511, 541
 Starting 527
 Peeking 439
 See also: Multiuser
 User account, *See:* Accounts, User
 User-defined environment variable 511
 UTC 536, **756**

V

VDI, *See:* Virtual Device Interface
 Verify
 Spooler 544
 Verifying
 Archives 460
 Backups 196
 File copy 263
 Video display attributes
 ANSI X3.64 230
 Bit-mapped 229
 Defining 228–230
 in system message file 408
 See also: Console, Display attributes
 Virtual Device Interface
 Attaching 192, 539, 540
 Virtual screen **146**

W

Wild cards 136–??
 # **137**
 * **137**
 = 352
 ? **137**
 @ **137**
 Specification **136**
 Window **145**, 145–152, 590–606
 Attributes **145**
 Coded 150
 Frame 149, 593, 603
 Hidden 151, 601
 Initial 598
 Title 149, 605
 Top 151, 600

- Bypass 592
 - Current mode 592
- Clearing 499, 592
- Clip 603
- Clipping 592
- Closing 386, 593
- Closing all 593
- Color 148, 604
- Display order 151, 600, 604
- Displaying 598, 600
- Erasing 598
- Frame 593, 603
- Frame styles 147
- Invert 148, 593, 604
- Menu 594–597
 - Color 595
 - Hot-keys 596
 - Invert 595
 - Keep 595
- Moving 597
- New 597
- Number 597
 - Next available 597
- Opening 597
- Position 146, 597, 603
- Saved
 - Attributes 599
 - Reading 598
 - Writing 599
- Selecting 600
- Shadow 148, 593, 603
- Size 597, 603
- Size limits 147
- Status 601–604
- Stipple pattern 592
- Text truncation 592
- Title 603, 605
 - Color 605
 - Position 605
- Updating 600
- Virtual screen 146
- Zero 150, 593, 600
- Word length 184
- Work drive 120, 388, 545
 - See also:* WORK environment switch
- WORK environment switch **102**, 480, 511

X

XMODEM

File transfer protocol 451, 473

Y

Y2K

See: Year 2000

Year 2000 759

Year number display 291, 292, 293, 756

YMODEM

File transfer protocol 451, 473