Prime Computer, Inc.

# FDR3108-190L PRIMOS Commands Reference Guide Revision 19.0

# PRIMOS Commands Reference Guide

# PRIMOS Commands Reference Guide

Sarah Lamb
and
Alice Landy

Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Massachusetts 01701

## COPYRIGHT INFORMATION

## PRINTING HISTORY – PRIMOS Commands Reference Guide

| Edition | Date | Number | Documents Rev. |
|---|---|---|---|
| *First Edition | January 1979 | FDR3108-101 | 16.3 |
| *Second Edition | May 1980 | FDR3108-101 | 17.2 |
| *Third Edition | January 1981 | FDR3108-101 | 18.1 |
| Fourth Edition | July 1982 | FDR3108-190 | 19.0 |

*These editions are out of print.

## HOW TO ORDER TECHNICAL DOCUMENTS

**U.S. Customers**

Software Distribution
Prime Computer, Inc.
1 New York Ave.
Framingham, MA 01701
(617) 879-2960 X2053, 2054

**Prime Employees**

Communications Services
MS 15-13, Prime Park
Natick, MA 01760
(617) 655-8000, X4837

**Customers Outside U.S.**

Contact your local Prime subsidiary or distributor.

**PRIME INFORMATION**

Contact your PRIME INFORMATION dealer.

## Summary of Rev. 19 Changes

### Changes to FDR3108

The following changes have been made to the **PRIMOS Commands Reference Guide** for Rev. 19:

- **Chapter 1** has been updated to reflect new commands and new command line features.
- **Chapter 2** contains many new and altered commands, as shown in the second half of this summary.
- **Chapter 3** has been reorganized so that it is now a dictionary of command functions. Material on global variables has been moved into Chapter 4.
- **Chapter 4** contains information on all old and new command features. Material on FUTIL, which used to be in Chapter 4, is in Appendix C.
- **Appendix A** is unchanged. It describes RVEC vectors.
- **Appendix B** now contains an explanation of the stack dump printed by the DMSTK command. Material on debugging, which used to be in this appendix, is now in the **Loading and Debugging Programmer's Companion.**
- **Appendix C** still describes old commands, FUTIL and PROTEC have been added to this appendix; CX, which is no longer supported, has been removed.
- **Appendix D** now contains the ASCII and EBCDIC character sets that were in Appendix E in previous editions.

### Changes in Commands

#### The following commands are obsolete at Rev. 19:

FUTIL
LISTF
MDL
PROTEC
PTCPY
RJ1004
RJ200UT
RJ7020
RJX80
RJGRTS
RJHASP

WS1004
WS200UT
WS7020
WSX80
WSGRTS
WSHASP

**The following commands are new at Rev. 19:**

ADD_REMOTE_ID
ATM
ATM_ADMIN
CHANGE_PASSWORD
COPY
EDIT_ACCESS
EDIT_PROFILE
EMACS
EVENT_LOG
FED
FIX_DISK
FTGEN
FTOP
FTR
HELP
LD
LON
LIST_ACCESS
LIST_GROUP
LIST_PRIORITY_ACCESS
LIST_QUOTA
LIST_REMOTE_ID
OAS
OA_ADMIN
OA_TERM
ORIGIN
PRIMOS
PROTECT
PTELE
REMOVE_PRIORITY_ACCESS
REVERT_PASSWORD
RJOP
RJQ
RWLOCK
SET_ACCESS
SET_DELETE
SET_PRIORITY_ACCESS
SET_QUOTA
USAGE
VISTA
VRPG
WP_ADMIN

**The following commands are altered at Rev. 19:**

ADDISK
ABBREV
ASSIGN
ATTACH
AVAIL
BATCH
CLOSE
COPY_DISK
CREATE
DATE
DEFINE_GVAR
DELETE
DELSEG
FIXRAT
JOB
LOGIN
LOGOUT
MAGNET
MAGSAV
MAKE
MESSAGE
RESUME
SIZE
SPOOL
STATUS
USAGE

# Contents

# PRIMOS Commands Reference Guide

# 1 Introduction

## To the Reader

This book is intended for the user who is working on a Prime computer and who needs detailed information on a particular PRIMOS command.

This is not an introduction to PRIMOS, Prime's operating system. Introductory material is supplied in the **Prime User's Guide**.

This book provides:

- Detailed information on most PRIMOS commands available to the user.
- A brief description of other PRIMOS commands (such as those available to operators, or those that invoke separately priced products) and references to detailed information about those commands.

The book is divided into four chapters.

Chapter 1, Introduction, provides a brief review of the PRIMOS command line, and explains our conventions for displaying command line formats. It also provides a summary that lists and defines all PRIMOS commands, grouped by function

Chapter 2, Dictionary of PRIMOS Commands, provides an alphabetical listing of commands, each entry containing either detailed instructions for the use of the command, or a reference to the book in which the detailed description may be found.

Chapter 3, Dictionary of Command Functions, describes the function calls available for use in the command line.

Chapter 4, Command Line Features, describes several new command line features.

In addition, four appendixes provide further details on miscellaneous matters.

## PRIMOS Command Line Format

The general format of the PRIMOS command line is:

**COMMAND** [**names**] [**–OPTION argument**]...[**–OPTION argument**]

For example, the format of the SPOOL command is:

**SPOOL** [**pathname**] [**options**]

In this example, **pathname** tells PRIMOS the name of the file to be printed, while **options** let the user specify how or where he or she wants the file printed. An example of a particular SPOOL command might be:

```
SPOOL MYFILE -FORM WHITE -AT BLDG2
```

Command lines may be up to 160 characters long. Command lines which exceed that length are rejected by the command processor, with the error message:

```
Command line longer than 160 characters. (listen_)
```

## Command Format Conventions

The conventions for PRIMOS command documentation are:

**WORDS-IN-UPPERCASE:** Capital letters identify command words or keywords. They are to be entered literally. If a portion of an uppercase word is in rust-colored letters, the rust-colored letters indicate the system-defined abbreviation.

**Words-in-lowercase:** Lowercase letters identify arguments. The user substitutes an appropriate numerical or text value.

**Braces** { }: Braces indicate a choice of arguments and/or keywords. At least one choice must be selected.

**Brackets** [ ]: Brackets shown in brown indicate that the word or argument enclosed is optional. Brackets shown in rust indicate function calls. They must be entered literally.

**Hyphen** –: A hyphen identifies a command line option, as in: SPOOL –LIST. Hyphens must be included literally.

**Parentheses** ( ): When parentheses appear in a command format, they must be included literally.

**Ellipsis** ...: An ellipsis indicates that the preceding argument may be repeated.

**Angle Brackets** < >: Angle brackets are used literally to separate the elements of a pathname. For example:

```
<FOREST>BEECH>BRANCH537>TWIG43>LEAF4
```

**options:** The word **options** indicates that one or more keywords and/or arguments can be given, and that a list of options for the particular command follows.

**Spaces:** Command words, arguments, and parameters are separated in command lines by one or more spaces. In order to contain a literal space, an argument must be enclosed in single quotes.

## Conventions in Examples

In all examples, the user's input is rust-colored, and the system's output is not. For example:

```
OK, attach *>examples
OK, ed seginfo
EDIT
```

User input usually may be either in lowercase or in UPPERCASE. The rare exceptions will be specified in the commands where they occur.

## Advanced Command Line Functionality

At Rev. 19, the Primos command line supports the following features:

- User-defined abbreviations
- Command line syntax suppression
- Multiple commands on one line
- User-defined global variables
- PRIMOS command functions
- Command iteration
- Wildcard names
- Treewalk pathnames
- Name generation patterns

User-defined abbreviations are explained in the discussion of the ABBREV command in Chapter 2. A Dictionary of Command Functions appears in Chapter 3. The other features are discussed briefly in Chapter 4, and are discussed in depth in the **Prime User's Guide.**

The command processor recognizes these features by looking for special characters on the command line. These special characters, in the order they are searched for, are given in Table 1-1. (Note that user-defined abbreviations, which are processed first, use no special characters.)

Table 1-1
Special Characters in the Command Line

| Feature | Special Character | Comments |
|---|---|---|
| Abbreviations | | No special characters |
| Syntax suppressor | | In first position on line only |
| Command separator | ; | |
| Global variables | % % | |
| Functions | [ ] | |
| Iteration | ( ) | |
| Treewalking | @, @@, +, ^ | In any intermediate position of pathname |
| Wildcarding | @, @@, +, ^ | In final position of pathname |
| Name generation | =, = =, ^=, ^= =, + | |

In general, we can note that when abbreviations, global variables, and command functions appear on the command line, the command line processor substitutes the *value* of the item for the item itself. This creates a one-to-one substitution.

However, when an iteration list, a wildcard name, or a treewalk name is found, the command processor creates one command for each item it finds or matches on the list. In iteration lists, the user supplies each item explicitly. In wildcard and treewalk names, the user sets the pattern, and the command processor searches the specified directory or directories for all file system objects that "match" that pattern. Thus, these features can be thought of as creating "many-to-one" matches.

A name generation pattern can be used to create a matching name either for a single filename or for whatever number of filenames result from a wildcard or treewalking pattern.

**Note**

Not all commands support all the features listed above. The general rule is as follows: if a feature is not useful in connection with a particular command, that command will not recognize it.

## Summary of PRIMOS Commands

The following Summary of User Commands lists all user commands and command functions commonly recognized by PRIMOS. In this summary, commands are grouped by function: file-handling commands, data base commands, etc. All commands in this summary are discussed in more detail in Chapter 2. Command functions shown in this summary are discussed in more detail in Chapter 3.

A list of operator commands follows the Summary of User Commands. Operator commands are discussed in detail in the **System Operator's Guide.**

## Internal and External Commands

At the end of each definition in the Summary of User Commands, you will find the word *(Internal)* or *(External)*. This indicates whether the command is an internal command or an external command.

**Internal commands** are part of PRIMOS itself. **External commands** are actually programs that are stored in a special UFD named CMDNC0. Some external commands invoke separately priced software products. Not all systems obtain all these products. Moreover, System Administrators may add or remove external commands to meet the needs of their particular systems. For these reasons, not every system recognizes all the external commands listed in this book.

If you need to know what external commands are available on your system, ATTACH to CMDNC0 and list its files with the LD command. (This works only if the System Administrator allows users to look at CMDNC0.) An example of this procedure is:

```
OK, A <FOREST>CMDNC0
OK, LD

<FOREST>CMDNC0 ()

Files= 57.

$$.SAVE            AVAIL.SAVE         BATCH.SAVE         BATGEN.SAVE
CMPF.SAVE          CONCAT.SAVE        COPY.RUN           COPY_DISK
CPMPC.SAVE         CRMPC.SAVE         DELETE.RUN         ED.SAVE
EDB.SAVE           EDIT_PROFILE.SAVE                     EVENT_LOG.SAVE
FILMEM.SAVE        FILVER.SAVE        FIX_DISK.SAVE      FUTIL
HELP.RUN           HPSD.SAVE          JOB.SAVE           LABEL.SAVE
LATE.SAVE          LD.RUN             LOAD.SAVE          MAGNET.SAVE
MAGRST             MAGSAV             MAKE               MRGF.SAVE
NSED               PHYRST             PHYSAV             PMA.SAVE
PRIMOS             PRMPC.SAVE         PROP.SAVE          PROTECT.RUN
PRSER.SAVE         PRVER.SAVE         PSD.SAVE           PSD20.SAVE
REVERT_PASSWORD.RUN                   RUNOFF.SAVE        RWLOCK.RUN
SEG.SAVE           SET_DELETE.RUN     SIZE.SAVE          SLIST.SAVE
SORT.SAVE          SPOOL.SAVE         TERM.SAVE          TRAMLC.SAVE
UPCASE.SAVE        VPSD.SAVE          VPSD16.SAVE
OK,
```

You may notice that the files CMDNC0 have various suffixes. The suffix .SAVE is used by both Prime software and user-written software The suffix .RUN, on the other hand, is reserved for use by Prime software. Commands using the .RUN suffix behave like internal commands. At Rev. 19, these commands are: COPY, DELETE, HELP, LD, PROTECT, REVERT_PASSWORD, RWLOCK, SET_ACCESS, and SET_DELETE.

## Summary of User Commands

### Establishing Your Identity and Accessing the System

| | |
|---|---|
| **LOGIN** | Begins a terminal session. It identifies the user to the system and establishes the initial contact between system and user. *(Internal)* |
| **LOGOUT** | Ends a terminal session. It closes all files and releases the PRIMOS process to another user. *(Internal)* |
| **ADD_REMOTE_ID** | Specifies a user id to be used for remote file access. *(Internal)* |
| **LIST_REMOTE_ID** | Displays your current set of remote ids. *(Internal)* |
| **CHANGE_ PASSWORD** | Replaces any login password you may have with a new login password of your choice. *(Internal)* |
| **LIST_GROUP** | Lists the access groups to which you belong. *(Internal)* |
| **ATTACH** | Moves your location from one directory to another. *(Internal)* |
| **ORIGIN** | Moves your location back to your origin directory (the directory to which you were attached when you logged in). *(Internal)* |

### Directory Handling

| | |
|---|---|
| **CREATE** | Creates and names a new directory. *(Internal)* |
| **CNAME** | Changes the name of a directory. *(Internal)* |
| **DELETE** | Deletes (removes) files, directories, or segment directories. *(External)* |
| **LD** | Lists the contents of a directory. *(External)* |
| **SET_QUOTA** | Defines the maximum number of records a directory and its subtree may use. *(Internal)* |
| **LIST_QUOTA** | Lists current disk quota and storage information for a specified directory. *(Internal)* |

### Protecting Directories

WITH ACCESS CONTROL LISTS:

| | |
|---|---|
| **SET_ACCESS** | Sets access control protection on files, directories, or segment directories. Will convert a password-protected directory to an ACL-protected directory. *(Internal)* |

| | |
|---|---|
| **EDIT_ACCESS** | Changes protection on ACL-protected objects by modifying existing ACLs (access control lists). *(Internal)* |
| **LIST_ACCESS** | Displays access protection for a specified file or directory. *(Internal)* |
| **LIST_PRIORITY _ACCESS** | Displays the priority access in effect on a given disk partition. *(Internal)* |

WITH PASSWORDS:

| | |
|---|---|
| **PASSWD** | Replaces any existing passwords on the current directory with new passwords. *(Internal)* |
| **PROTECT** | Defines rights of others to access the user's directory. *(External)* |
| **REVERT_ PASSWORD** | Converts the current directory from an ACL-protected directory to a password-protected directory. *(External)* |

**File Handling**

| | |
|---|---|
| **CNAME** | Changes the name of a file. *(Internal)* |
| **COPY** | Copies file system objects *(External)* |
| **DELETE** | Deletes file system objects. *(External)* |
| **SIZE** | Displays the size of a file; displays the number of entries in a directory, segment directory, or access category. *(External)* |
| **SLIST** | Prints the contents of a file on your terminal. *(External)* |
| **SORT** | Sorts up to 20 files into a single output file. *(External)* |
| **SPOOL** | Queues a file for printing or plotting on the system printers. *(External)* |

OPENING AND CLOSING FILES:

| | |
|---|---|
| **OPEN** | Opens a file unit for reading, writing, or updating. *(Internal)* |
| **CLOSE** | Closes a file. *(Internal)* |
| **BINARY** | Opens a file for writing on PRIMOS file unit 3, usually as a binary output file for use by a compiler or assembler. *(Internal)* |
| **INPUT** | Opens a source file for reading on file unit 1. *(Internal)* |
| **LISTING** | Opens a file for writing on file unit 2. *(Internal)* |
| **COMINPUT** | Opens (or closes) a Command Input File, usually on file unit 6. *(Internal)* |

| | |
|---|---|
| **COMOUTPUT** | Opens (or closes) a file on file unit 127 for recording interactive terminal input and output. *(Internal)* |

COMPARING AND MODIFYING FILES:

| | |
|---|---|
| **CMPF** | Compares two to five ASCII files; prints out any discrepancies it finds. *(External)* |
| **FILVER** | Compares runfiles, notifying the user of any differences between them. *(External)* |
| **MRGF** | Merges two to five ASCII files, allowing the user to resolve conflicts among them. *(External)* |
| **NUMBER** | Numbers or renumbers statements in a BASIC program. *(External)* |
| **UPCASE** | Creates an uppercase-only file from a file containing both upper- and lowercase characters. *(External)* |
| **CONCAT** | Combines a number of ASCII files into one; accepts formatting commands for spooling the file. *(External)* |

PROTECTING FILES:

| | |
|---|---|
| **SET_ACCESS** | Creates an access control list to protect a file system object. *(Internal)* |
| **EDIT_ACCESS** | Modifies the access control list protecting a file system object. *(Internal)* |
| **LIST_ACCESS** | Lists the access control list protecting a file system object. *(Internal)* |
| **SET_DELETE** | Sets the delete switch on a file; used to guard against accidental deletion. *(External)* |
| **RWLOCK** | Sets the read/write concurrency lock on a file system object. *(External)* |
| **PROTECT** | Sets user rights on a file (effective only on files in password-protected directories). *(External)* |

## Editors and Text Handlers

| | |
|---|---|
| **ED** | Invokes EDITOR, Prime's most commonly used text editor. *(External)* |
| **EDB** | Invokes Prime's binary editor (used primarily for building and maintaining subroutine libraries). *(External)* |
| **EMACS** | Invokes EMACS, a character-oriented screen editor. *(External)* |

| | |
|---|---|
| RUNOFF | Invokes RUNOFF, Prime's text-formatting utility. *(External)* |

## Compilers, Translators, and Interpreters

| | |
|---|---|
| BASIC | Invokes the Prime BASIC interpreter. *(External)* |
| BASICV | Invokes a virtual-memory BASIC subsystem (BASIC/VM). *(External)* |
| COBOL | Compiles a COBOL program. *(External)* |
| DBASIC | Invokes an interpretive BASIC with double-precision arithmetic capabilities. *(External)* |
| F77 | Compiles a FORTRAN IV or FORTRAN 77 source program, using the FORTRAN 77 compiler. *(External)* |
| FTN | Compiles a FORTRAN IV program. *(External)* |
| NCOBOL | Compiles a COBOL program, using the nonshared COBOL compiler. *(External)* |
| PASCAL | Compiles a Pascal program. *(External)* |
| PL1G | Compiles a PL/I, subset G, program *(External)* |
| PMA | Assembles a program written in the Prime Macro Assembler language (PMA). *(External)* |
| RPG | Compiles an RPG II program, using the R-mode RPG compiler. *(External)* |
| VRPG | Compiles an RPG II program, using the V-mode RPG compiler. *(External)* |

## Loading and Executing Programs

| | |
|---|---|
| DELSEG | Frees previously used segments. *(Internal)* |
| FILMEM | Fills memory locations 100 to top of 32K with zeroes (used before invoking LOAD). *(External)* |
| LOAD | Invokes Prime's loader for R-mode object files. *(External)* |
| REN | Reenters a subsystem following a QUIT or an error condition. *(Internal)* |
| RESTOR | Restores a runfile from disk to memory. *(Internal)* |
| RESUME | Executes a CPL program or an R-mode program. *(Internal)* |
| SAVE | Saves the contents of a specified location in memory. *(Internal)* |
| SEG | Loads and/or executes a V-mode or I-mode program. *(External)* |

|  |  |
|---|---|
| **START** | Starts a program loaded by the RESTOR command, or restarts a program halted by a QUIT or an error condition. *(Internal)* |

### Debuggers

|  |  |
|---|---|
| **DBG** | Invokes the source level debugger, for debugging high-level language programs. *(External)* |
| **PSD**<br>**HPSD**<br>**PSD20** | Invoke versions of the Prime Symbolic Debugger, for debugging R-mode and S-mode assembler programs (or object code from high-level language programs). *(External)* |
| **VPSD**<br>**VPSD16** | Invoke versions of the Prime Symbolic Debugger, for debugging V-mode and I-mode assembler programs or object code. *(External)* |

### Job Processors

|  |  |
|---|---|
| **COMINPUT** | Executes a Command Input File. *(Internal)* |
| **COMOUTPUT** | Creates a record of command input and output. *(Internal)* |
| **CPL** | Executes a CPL program. *(Internal)* |
| **JOB** | Executes a CPL program or COMINPUT file as a Batch job. *(External)* |
| **PHANTOM** | Executes a CPL program or COMINPUT file as a phantom. *(Internal)* |
| **$$** | Used inside CPL or COMINPUT files to pass JOB information to the Batch monitor. *(External)* |

### Setting Your Terminal Characteristics

|  |  |
|---|---|
| **DELAY** | Defines a time function that delays the printing of a character after a carriage return (CR) has been output to a terminal. *(Internal)* |
| **LATE** | Forces the terminal to ignore commands until a specified time. *(External)* |
| **TERM** | Defines terminal characteristics such as the erase character, BREAK key, duplex and half-duplex operation. *(External)* |

### Defining Your Command Environment

|  |  |
|---|---|
| **ABBREV** | Defines abbreviations for PRIMOS commands and their arguments. *(Internal)* |
| **DEFINE_GVAR** | Defines a global variable file. *(Internal)* |

| | |
|---|---|
| DELETE_VAR | Deletes one or more variables from a global variable file. *(Internal)* |
| LIST_VAR | Lists variables from a global variable file. *(Internal)* |
| RDY | Selects the prompt message you want displayed at your terminal. *(Internal)* |
| RLS | Frees space by discarding unwanted stack history. *(Internal)* |
| SET_VAR | Adds a variable to a global variable file. *(Internal)* |
| LON | Enables or disables phantom logout notification. *(Internal)* |

### System Information

| | |
|---|---|
| AVAIL | Provides information on amount of disk space being used. *(External)* |
| BATGEN | Provides information about Batch queues. *(External)* |
| DATE | Prints the current date and time at your terminal. *(Internal)* |
| HELP | Provides online information about PRIMOS commands. *(External)* |
| PROP | Provides information about system printers and/or plotters. *(External)* |
| STATUS | Provides general information: what users are logged in, what disks are running, what nodes on a network are available, etc. *(Internal)* |
| USERS | Prints the number of users currently logged in. *(Internal)* |
| USAGE | Meters system usage. *(External)* |

### Information on Your Current Environment

| | |
|---|---|
| STATUS ME | Lists your user id, line number, disk, and assigned magnetic tape devices. *(Internal)* |
| LIST_REMOTE_ID | Lists your current set of remote ids. *(Internal)* |
| LIST_GROUP | Lists the access groups to which you belong. *(Internal)* |
| LIST_VAR | Lists the contents of your global variable file, if you have one active. *(Internal)* |
| LIST_ACCESS | Lists access on your current directory. *(Internal)* |
| LIST_PRIORITY _ACCESS | Lists priority ACL on your current disk. *(Internal)* |

| | |
|---|---|
| **LIST_QUOTA** | Gives information on records used and maximum records available in your current directory. *(Internal)* |
| **AVAIL** | Lists records used and available on your current disk. *(External)* |
| **SIZE** | Displays size of files, number of entries in directories or segment directories. *(External)* |

### Information on Program and Command Execution

| | |
|---|---|
| **BATCH** | Provides information on progress of user's Batch jobs. *(External)* |
| **DMSTK** | Produces a call/return trace of the user's stacks. *(Internal)* |
| **PM** | Prints the contents of the RVEC vector. *(Internal)* |
| **PRERR** | Prints locations and messages from PRIMOS's error vector, ERRVEC. *(Internal)* |
| **TIME** | Prints accounting information: time since login, CPU time used, and I/O time used. *(Internal)* |

### Message Facilities

| | |
|---|---|
| **MESSAGE** | Sends message from one terminal to another. *(Internal)* |

### Terminal I/O Handling

| | |
|---|---|
| **RSTERM** | Empties the terminal's input and/or output buffers. *(Internal)* |
| **TYPE** | Prints text at the terminal or into a COMOUTPUT file. *(Internal)* |

### I/O Device Handling

GENERAL COMMANDS:

| | |
|---|---|
| **ASSIGN** | Gives the user at the terminal control of a magnetic tape unit or other peripheral device. *(Internal)* |
| **UNASSIGN** | Releases control of a previously assigned peripheral device. *(Internal)* |

MAGNETIC TAPES:

| | |
|---|---|
| **LABEL** | Creates either an ANSI COBOL level 1 volume label or an IBM 9-track EBCDIC or 7-track BCD label on a magnetic tape. *(External)* |
| **MAGNET** | Writes files or directories to tape or restores them from tape to disk. *(External)* |
| **MAGRST** | Restores files or directories from tapes created by MAGSAV. *(External)* |
| **MAGSAV** | Copies files or directories from disk to tape. *(External)* |

PAPER TAPES:

| | |
|---|---|
| **BASINP** | Loads a BASIC source program from paper tape. *(External)* |

PUNCHED CARDS:

| | |
|---|---|
| **CPMPC** | Punches a file onto a card deck. *(External)* |
| **CRMPC** | Reads cards into a file. *(External)* |

PRINTERS:

| | |
|---|---|
| **PRMPC** | Prints a file on an MPC parallel interface printer. *(External)* |
| **PRSER** | Prints a file on a serial interface printer. *(External)* |
| **PRVER** | Prints a file on an assigned printer/plotter. *(External)* |

## Communications

| | |
|---|---|
| **FTR** | Invokes the File Transfer Request utility to transfer files to or from a remote site. *(External)* |
| **NETLINK** | Establishes a connection to any system on the Public Data Network. *(Internal)* |
| **OWLDSC** | Allows an OWL-1200 terminal to emulate an IBM 3277 model 2 display station on systems where DPTX/DSC is running. *(External)* |
| **PRTDSC** | Invokes the printer emulation program on systems where DPTX/DSC is running. *(External)* |
| **PT45DSC** | Invokes PT45 interface program on systems where DPTX/DSC is running. *(External)* |
| **RJQ** | Sends utility command for the Remote Job Entry system; used to submit jobs to remote computer sites. *(External)* |

| | |
|---|---|
| **TCF** | Invokes DPTX/TCF on a system where DPTX/TSF and DPTX/OSC are running. *(External)* |
| **TRAMLC** | Transmits or receives a file over an assigned AMLC line between two Prime computer systems. *(External)* |
| **RJOP** | Controls Remote Job Entry workstations. *(External)* |

**Data Management**

DBMS SUBSYSTEMS:

| | |
|---|---|
| **CDML** | Invokes the COBOL data manipulation language. *(External)* |
| **CLUP** | Invokes the DBMS cleanup processor. *(External)* |
| **CSUBS** | Invokes the COBOL DBMS subschema. *(External)* |
| **DBACP** | Invokes the data base administrator command process. *(External)* |
| **DBUTL** | Invokes a data base dump utility that allows you to monitor the contents of a data base schema and shared user table. *(External)* |
| **FDML** | Invokes the FORTRAN DML preprocessor. *(External)* |
| **FSUBS** | Invokes the FORTRAN DBMS subschema. *(External)* |
| **SCHDEC** | Invokes the DBMS Schema Decompiler. *(External)* |
| **SCHED** | Invokes the schema editor (SCHED). *(External)* |
| **VISTA** | Invokes DBMS/QUERY, the DBMS query language and report writer. *(External)* |

MIDAS:

| | |
|---|---|
| **CREATK** | Invokes a program to build a template for a MIDAS file. *(External)* |
| **KBUILD** | Invokes a program to build a keyed-index or direct access MIDAS file. *(External)* |
| **KIDDEL** | Invokes a program to delete all or part of the records in a MIDAS file. *(External)* |
| **MCLUP** | Invokes the MIDAS cleanup utility. *(External)* |
| **MPACK** | Invokes a utility that packs and restructures MIDAS files. *(External)* |

POWER:

| | |
|---|---|
| **POWER** | Invokes the PRIME/POWER data management facility. *(External)* |

### Forms

| | |
|---|---|
| **FAP** | Invokes the FORMS Administrative Processor. *(External)* |
| **FDL** | Invokes the FORMS Definition Language. *(External)* |
| **FED** | Invokes the FORMS Editor. *(External)* |

### OAS (Office Automation System)

| | |
|---|---|
| **ATM** | Logs you into OAS Advanced Text Management Option Selection Menu. *(External)* |
| **OAS** | Logs you into OAS Master Function Selection. *(External)* |
| **OA_TERM** | Downline loads the PT65 Terminal for OAS. *(External)* |
| **PTELE** | Invokes the OAS Telephone Inquiry function. *(External)* |

### System Settings

| | |
|---|---|
| **ASRCWD** | Directs output stream to terminal or peripheral device. *(Internal)* |
| **SVCSW** | Controls handling of SVC instructions. *(Internal)* |
| **VRTSSW** | Sets the virtual sense switches. *(Internal)* |

### Command Functions

ARITHMETIC FUNCTIONS:

| | |
|---|---|
| **CALC** | Evaluates arithmetic or logical expressions. |
| **HEX** | Converts a hexadecimal number to its decimal equivalent. |
| **MOD** | Divides one number by another and returns the remainder. |
| **OCTAL** | Converts an octal number to its decimal equivalent. |
| **TO_HEX** | Converts a decimal number to its hexadecimal equivalent |
| **TO_OCTAL** | Converts a decimal number to its octal equivalent. |

FILE SYSTEM FUNCTIONS:

| | |
|---|---|
| **ATTRIB** | Returns information (type, length, or date last modified) about a specified file or directory. |

| | |
|---|---|
| **DIR** | Returns the directory portion of a pathname. |
| **ENTRYNAME** | Returns the entryname portion of a pathname. |
| **EXISTS** | Determines whether or not a file system object exists and whether its file type matches the type specified. |
| **GVPATH** | Returns the pathname of your global variable file, if you have one active. |
| **OPEN_FILE** | Opens a file for reading or writing. |
| **PATHNAME** | Returns a full pathname. |
| **READ_FILE** | Reads a line from an ASCII file. |
| **WILD** | Produces a blank-separated list of entrynames representing the file system objects that match the specifications provided. |
| **WRITE_FILE** | Writes a line of text into an ASCII file. |

STRING-HANDLING FUNCTIONS:

| | |
|---|---|
| **AFTER** | Returns the portion of a string that appears after some specified character(s). |
| **BEFORE** | Returns the part of a string that precedes some specified character(s). |
| **INDEX** | Returns the starting position of a specified substring within a string. |
| **LENGTH** | Returns the number of characters in a given string. |
| **NULL** | Tests for null strings. |
| **QUOTE** | Places a pair of quotes around a string, and doubles any quotes appearing within the string. |
| **SEARCH** | Compares two strings; returns the position of the first character in the first string that matches any character in the second string. |
| **SUBST** | Replaces text in one string with text from another. |
| **SUBSTR** | Returns a substring (specified by length and starting position) of a string. |
| **TRANSLATE** | Replaces character(s) in one string with character(s) from another. |
| **TRIM** | Removes characters from the left, right, or both sides of a specified string. |
| **UNQUOTE** | Removes outer quotes from around a specified text string and changes double quotes within string to single quotes. |
| **VERIFY** | Compares two strings; returns the position of the first character in one that does not match any character in the other. |

MISCELLANEOUS FUNCTIONS:

| | |
|---|---|
| **ABBREV** | Expands the value of an abbreviation. |
| **CND_INFO** | Allows a CPL condition handler to examine the condition information of the most recent condition on the stack. |
| **DATE** | Returns the current date and/or time in a variety of formats. |
| **GET_VAR** | Returns the value of a named variable. |
| **QUERY** | Prints the contents of specified text on the terminal screen and waits for a YES or NO reply. |
| **RESCAN** | Removes one level of quotes from a specified string and evaluates any function calls or variable references that no longer appear in quotes. |
| **RESPONSE** | Prints a specified prompt at the terminal and waits for any reply. |

## Summary of Operator Commands

Below are the PRIMOS commands used primarily by the System Operator/Administrator. Many of these commands can be issued only from the supervisor terminal. These commands are explained briefly in Chapter 2; for a full explanation, see the **System Administrator's Guide** or the **System Operator's Guide**.

| | | |
|---|---|---|
| ADDISK | FTGEN | PROP |
| AMLC | FTOP | REMOTE |
| ATM_ADMIN | LOGPRT | REMOVE_PRIORITY_ACCESS |
| BATCH | LOOK | REPLY |
| BATGEN | MAKE | RJOP |
| CHAP | MAXSCH | SET_PRIORITY_ACCESS |
| CONFIG | MAXUSR | SETIME |
| COPY_DISK | MCLUP | SETMOD |
| DISKS | NET | SHARE |
| DPTCFG | NETCFG | SHUTDN |
| DPTX | OA_ADMIN | STARTUP |
| ELIGTS | OPRPRI | USRASR |
| EVENT_LOG | PHYRST | WP_ADMIN |
| FIX_DISK | PHYSAV | |
| FIXRAT | PRIMOS | |

# 2 Dictionary of PRIMOS Commands

## ▶ ABBREV [pathname] [options]

The ABBREV command allows users to create their own abbreviations for PRIMOS commands and their arguments and to use these abbreviations at will during interactive sessions and in CPL programs. Abbreviations *cannot* be defined for subcommands. They may be used in CPL programs, if CPL's & EXPAND directive is used; but they may not be used in command input files. (System Administrators may disable the abbreviation preprocessor, thus preventing the use of ABBREV files at their installations.)

When an abbreviation file is active, PRIMOS calls its abbreviation preprocessor to scan each command line entered from the terminal. The abbreviation processor checks each token (that is, each word or numeral) and expands it to its full form before passing the expanded command to the standard command processor for execution. Abbreviations are not expanded when commands are read from a command input file.

### Using ABBREV

To use ABBREV:

1. Create an empty abbreviation file with the command **ABBREV new-pathname –CREATE.** (If the UFD in which the file is created has a password, the password must be part of **new-pathname.**) For example:

```
OK, abbrev 'beech secret'>abbrevs  -create
Creating new abbreviation file: <FOREST>BEECH SECRET>ABBREVS (ab_file_)
OK,
```

2. Define abbreviations within the file up to a limit of approximately 200 abbreviations.

3. At the start of a subsequent session, reactivate the abbreviation file by giving the command **ABBREV pathname.** If you deactivate the file during the session (via the **ABBREV –OFF** command), reactivate it with the command **ABBREV –ON.**

4. If you are in a multiuser environment, deactivate your abbreviation file at the end of your work session, either by giving the **ABBREV –OFF** command or by logging out.

**Note**

Certain error conditions, which return the message "User environment re-initialized." wipe out your active abbreviations. Restore them by giving the command **ABBREV pathname.**

## Rules for Defining Abbreviations

Each abbreviation has two parts: its **name** and its **value.** Rules for each are as follows:

**Names** Names may be up to eight characters in length. They may contain any ASCII character except spaces, quotes ('), commas, greater-than symbols (>), and vertical bars (|). They may be given as lowercase or uppercase characters; the abbreviation preprocessor converts all lowercase characters to uppercase.

**Note**

Do not begin abbreviation names with the hyphen (–). If you do begin an abbreviation name with a hyphen, you must enclose the name within quotes in any ABBREV command lines. For example:

```
OK, abbrev -delete '-xyz'
OK,
```

**Values** The value of an abbreviation is the character string represented by the abbreviation. All ASCII characters are legal within values, including spaces. Leading spaces and/or tabs are removed during definition. The percent sign (%) functions as an escape character and is used to define variables. The token %**i**% in a value stands for the *ith* token found after the abbreviation in the command line. Currently, **i** can range from 1 to 9. The use of variables permits the repetition and reordering of tokens from the command within the expanded value. For example, defining the abbreviation:

```
CMPL %1% %2%.%1% -L %2%.LIST -B %2%.BIN -64V -DEBUG -EXPLIST
```

would allow the command:

```
CMPL PL1G FLAME
```

to be expanded to:

```
PL1G FLAME.PL1G -L FLAME.LIST -B FLAME.BIN -64V -DEBUG -EXPLIST
```

If a command line contains fewer variables than the value calls for, the missing variables will be taken to be null strings. If a command line provides more variables than the value expects, the unexpected variables are added to the end of the command line. Thus, if the value of X is %1% %4%, then the command X A B C D E is expanded to A D E.

Abbreviations may be used as arguments to other abbreviations. When this happens, the abbreviations are treated as though they were nested. The innermost abbreviation is expanded first and any variables needed by it are used. The expanded expression is then available to the next abbreviation, together with any other (unused) tokens in the command line, and so on.

Here is an example of creating and using two abbreviations. (We will work in VERIFY mode, explained below, to make ABBREV echo the expanded abbreviations before using them.)

```
OK, abbrev abbrevs -create
Creating new abbreviation file: <FOREST>BEECH>ABBREVS (ab_file_)
OK, abbrev -add u <FOREST>BEECH>BRANCH5>%1%
OK, abbrev -add sp spool %1% -at 1 %2% %3% -list
OK, abbrev -list
Abbreviation file: <FOREST>BEECH>ABBREVS
Abbreviations: 2

    SP       spool %1% -at 1 %2% %3% -list
    U        <forest>beech>branch5>%1%

OK, ab -verify
OK, sp u squirrel -defer 2200
(listen_) "spool <forest>beech>squirrel -at 1 -defer 2200 -list"
[SPOOL rev 19.0.0]
PRT003 spooled,  records:    1, name: SQUIRREL

System HQA
 user  prt  time  name              size  opts/#  form  defer  at: PICKUP
-----  ---  ----- ----------------  ----  ------  ----  -----  -----------
HARRY  001 10:27  FROM.ALAN2           3          PINK             2
NUTTY  002 13:49  MAIL                 1          WHITE
BEECH  003 13:51  SQUIRREL             1                22:00  1
OK,
```

Since U is the inner abbreviation in this example, it is expanded first. It wants one variable, %1%; so it takes SQUIRREL as that variable. When SP is expanded, it takes the expanded value of U (<FOREST>BEECH>BRANCH5>SQUIRREL) as its %1%, then takes –DEFER and 2200 as its %2% and %3%.

Abbreviations may be defined for ABBREV commands. The abbreviation preprocessor checks all command lines after expanding the first token. If the first token expands to "ABBREV", the rest of the line is read without expansion. For example, .**A** is a handy abbreviation for ABBREV –ADD COMMAND, and **.L** is a handy abbreviation for ABBREV –LIST.

```
OK, ab -verify
OK, ab -ac .a abbrev -ac
(listen_) "ab -ac .a abbrev -ac"
OK, .a .l abbrev -list
(listen_) "abbrev -ac .l abbrev -list"
OK, .l
(listen_) "abbrev -list"
Abbreviation file: <FOREST>BEECH>ABBREVS
Abbreviations: 4

(C) .A       abbrev -ac
(C) .L       abbrev -list
    SP       spool  -at 1   -list
    U        <forest>beech>branch5>%1%

OK,
```

### Using Abbreviations in Command Lines

- If a user-defined abbreviation is identical to a PRIMOS-defined abbreviation, the user's abbreviation takes precedence when the user's abbreviation file is activated. The PRIMOS abbreviation is therefore unavailable to the user during that time, unless it is enclosed in quotes and typed in uppercase.
- Tokens in a command line (i.e., abbreviations, commands, or arguments) may be separated by spaces, tabs, commas, or >.
- Any characters placed in quotes within a command line are not expanded by the abbreviation preprocessor, but are handed on literally — quotes and all — to the command processor.
- Expansion of any command line may be suppressed by the –EXECUTE option, explained below.

### Options

The ABBREV command supports the options listed below. In each case, supplying the pathname within the command activates the file as well as performing the option-related task. If the file is already active, the pathname is not needed.

To create a file:

| | |
|---|---|
| **–CREATE** | Creates and activates an empty abbreviation file. A pathname must be supplied with this option. If file named by the pathname already exists, the command activates that file. |

To get information on ABBREV:

| | |
|---|---|
| **-HELP** | Displays a short summary of the ABBREV command, explaining the various options you can use. |

To activate and deactivate files:

| | |
|---|---|
| **ABBREV pathname** | Activates an existing abbreviation file. Naming a nonexistent file, or a file that is not an abbreviation file, produces an error message. |
| **-OFF** | Turns off abbreviation expansion. |
| **-ON** | Turns on abbreviation expansion. If no **pathname** is given, the command reactivates the file the user was previously using. If **pathname** is given, activates that file. |
| **-VERIFY** | Turns on verify mode. This prints each expanded command line at the terminal before executing it. |
| **-NO_VERIFY** | Turns off verify mode (default). |
| **-EXECUTE rest-of-line** | Passes **rest-of-line** to command processor for execution without expanding it. |
| **-EXPAND rest-of-line** | Expands **rest-of-line** and prints it out on terminal, but does not execute expanded line. |
| **-EXPAND_EXECUTE** | Expands **rest-of-line** and then passes it to the command processor for execution. |

To monitor file:

| | |
|---|---|
| **-LIST [name-1 ... [name-n]]** | Lists the specified abbreviations from the current abbreviation file. If no **names** are given, this option lists the complete file. |
| **-STATUS** | Prints out the name of the current abbreviation file and the number of abbreviations it contains. |

To add, change, or delete abbreviations:

| | |
|---|---|
| **–ADD name value** | Adds the abbreviation **name** to the current file and gives it the value **value.** If an abbreviation for **value** already exists, the user will be asked if he wants to replace it. |
| **–ADD_ARGUMENT name value** | Adds an abbreviation that will be expanded only when it occurs in the argument position of a command line. |
| **–ADD_COMMAND name value** | Adds an abbreviation that will be expanded only when it occurs in the command position of a command line. |
| **–CHANGE name -1 [...name-n]** | Changes the specified abbreviations so that they are expandable anywhere on the command line. |
| **–CHANGE_ARGUMENT name-1 [...name-n]** | Changes the specified abbreviations so that they are expandable only in the argument position on the command line. |
| **–CHANGE_COMMAND name-1 [...name-n]** | Changes the specified abbreviations so that they are expandable only in the command position of the command line. |
| **–CHANGE_NAME old-name new-name** | Changes the name of the abbreviation **old-name** to **new-name.** |
| **–DELETE name-1 [...name-n]** | Deletes the specified abbreviations from the abbreviation file. |
| **–NO_QUERY** | Replaces old abbreviation without asking. (Only useful if followed by one of the ADD commands.) |

A **wildcard** may be used with the ABBREV command. Wildcards provide an easy way of selecting a set of abbreviation names. When you use a wildcard in place of a name, ABBREV selects all abbreviation names that match the wildcard.

For example, if you had an abbreviation file containing the following abbreviations:

```
(C) .A      abbrev -ac
(C) .D      abbrev -delete
(C) .L      abbrev -list
    SP      spool %1% -at 1 %2% %3% -list
    U       <forest>beech>branch5>%1%
```

You could list all names beginning with the character ".", as follows:

```
OK, abbrev -list .@
(C) .A       abbrev -ac
(C) .D       abbrev -delete
(C) .L       abbrev -list
OK,
```

For further information on wildcards, see Chapter 4 of this book.

▶ **ADDISK [PROTECT] pdisk1 [ pdisk2...pdisk9 | –RENAME newname ]**

ADDISK starts up the disk specified by the physical disk (pdisk) parameters. For details, see the **System Operator's Guide.**

The method you use to add remote disks with ADDISK depends on whether you are using FAM I or FAM II to communicate with the remote system. For FAM I, the format is:

**ADDISK nodename pdisk1 [pdisk2...pdisk8]**

For FAM II, the format is:

**ADDISK packname-1 [...packname-9] –ON nodename**

At Rev. 19, packnames must be unique: that is, two disks with the same name cannot be added to the same system. To avoid duplication of names, use the –RENAME option. For details, see the **System Operator's Guide.**

▶ **ADD_REMOTE_ID user-id [password] –ON nodename [–PROJECT project-id]**

ADD_REMOTE_ID allows you to specify the id which a slave on a remote system will run under when doing work on your behalf. A slave normally uses your user id. When you need to access a remote system which does not recognize your own user id, you need to add a remote id for the slave. You may also do this if you want your slave to use an id different from your own. You must add the remote id before trying to access the remote system.

The arguments and options for this command are the same as those for the LOGIN command. For explanation of naming rules for the user id, password, nodename, and project id, see the LOGIN command.

You should specify a user id that exists on the remote system you name in the command line. If the id does not exist on that remote system, the remote access will fail. You must also give the appropriate password, unless the id you specify does not use a password.

The name you specify with "–ON nodename" identifies the system on which your slave will be using the remote id If you specify a nodename for which you have already set up a remote id, the new user id replaces the existing one. That is, you can only have one remote id on any remote system. At any one time, you may have remote ids for up to 16 systems.

The –PROJECT option allows you to specify an optional project id. A **project** is a group of users with similar attributes and system usage. If you do specify a project id, it identifies your project affiliation on the remote system. Project ids are required on some systems; the remote access may fail if you do not specify a project id for remote ids on those systems.

The following example shows how a user called ANNA on SYS.A might add a remote id GUEST on SYS.G, where GUEST uses the password TOWEL and logs into the project GRAND:

```
ADD_REMOTE_ID GUEST TOWEL -ON SYS.G -PROJECT GRAND
```

Note that remote access will only work if GUEST has already been defined on SYS.G, and TOWEL is the right password.

The id established by this command exists until you log out, or until you give a new ADD_REMOTE_ID command with the same nodename.

For further information on adding remote ids, see the **PRIMENET Guide.**

*See also:* **LIST_REMOTE_ID, LOGIN**

## ▶ AMLC [protocol] line [config] [lword] *Operator command*

The AMLC command starts up an AMLC line. The parameters are discussed in detail in the **System Administrator's Guide.**

## ▶ ASRCWD [number]

The chief use of ASRCWD is to allow a user with a serial line printer to recover from the following situation: Assume output is being sent to the serial line printer and a user program aborts or a CONTROL-P condition occurs. At this point, the user is not able to get output (from the Editor, for example) printed or displayed at the user's terminal. Issuing the command line: ASR 0 then allows the user to recover and get output at the terminal.

For further information, see ASRCWD in Appendix C, Old Commands.

▶ ASSIGN $\left\{ \begin{array}{l} \textbf{device [-WAIT]} \\ \textbf{DISK pdisk [-WAIT]} \\ \text{AMLC } \textbf{[protocol] amlc-line [config]} \end{array} \right\}$

The ASSIGN command obtains complete control over a disk or a peripheral device from the user terminal.

**device** is an available device. These are the assignable devices:

| Device Code | Meaning |
|---|---|
| **CARDR** | Serial Card Reader |
| **CRn** (0≤n≤1) | MPC Parallel Card Reader or Reader/Punch |
| **DISK pdisk** | Physical Disk Partition (**pdisk** is a partition (volume) number) |
| **GS0 - GS3** | Vector General graphics display terminal |
| **MG0 - MG3** | Megatek graphics display terminal |
| **MTn** (0≤n≤7) | Magnetic Tape Unit |
| **PRn** (0≤n≤3) | Line Printer |
| **PTR** | Paper Tape Reader |
| **PUNCH** | Paper Tape Punch |
| **PLOT** | Printer/Plotter |
| **SMLCnn** | Synchronous Communications Line (00≤nn≤07) |

If the device is currently assigned to another user, the system replies:

```
The device is in use.  (ASSIGN)
ER!
```

unless the optional argument –WAIT was supplied. In this case, the ASSIGN command is queued until the device is unassigned by another user, or until the other user presses the CONTROL-P or BREAK key, or logs out. The terminal from which the ASSIGN command is invoked is unavailable for use until the device assigned is again available for assignment, or until the CONTROL-P or BREAK key is pressed by the user at that terminal.

Examples:

```
ASSIGN CENPR -WAIT
```

assigns the serial line printer and queues the assignment if the printer is already assigned.

```
AS PTR
```

assigns the paper tape reader.

If the user does not ASSIGN a device and attempts to perform I/O to or from the device, the error message:

```
Device not assigned.
ER!
```

is printed at the terminal.

## Assigning Magnetic Tapes

For magnetic tapes, the ASSIGN command either indicates by number which physical tape drive the user wants, or provides a description of a tape drive that will meet the user's requirements. Additionally, ASSIGN allows special requests to be made of the system operator; for example, removing the WRITE-ring or mounting a particular tape. (These requests are useful primarily for Batch jobs.) The command format, complete with optional arguments is:

$$\text{ASSIGN} \left\{ \begin{array}{l} \textbf{MTpdn [-ALIAS MTldn]} \\ \textbf{MTX -ALIAS MTldn} \end{array} \right\} \textbf{[options]}$$

**options** are:

$$\begin{array}{l} \textbf{-TPID id} \\ \left\{ \begin{array}{l} \textbf{-7TRK} \\ \textbf{-9TRK} \end{array} \right\} \\ \textbf{-DENSITY bpi} \\ \left\{ \begin{array}{l} \textbf{-RINGON} \\ \textbf{-RINGOFF} \end{array} \right\} \\ \textbf{-MOUNT} \end{array}$$

The options and arguments are described below:

| Option | Description |
|---|---|
| MTpdn | Magnetic tape (MT) unit number from 0 to 7, inclusive. **pdn** is the physical device number assigned to each drive at system startup. |
| MTldn | The logical drive number, from 0 to 7, inclusive. **ldn** is a user-specified number assigned to a particular physical drive unit; mapped into **pdn** in subsequent magnetic tape operations. |

| | |
|---|---|
| **MTX** | Indicates "any available drive"; MUST be accompanied by – ALIAS MTldn option. The particular drive assigned depends on any other options that appear on the command line. |

**Note**

The MTX option *cannot* be used from the system terminal.

| | |
|---|---|
| **–WAIT** | Indicates user is willing to wait until requested drive is available. |
| **–TPID** | Indicates that a tape "id" is to follow. Used to request mounting of a particular tape; requires operator intervention. |
| **id** | A list of tape identifiers (arguments) describing a particular reel of tape, and/or type of tape drive (name, number, etc.). Identifiers must not contain the following delimiters: commas, spaces, .NL., and /'. They may not begin with a hyphen or dash (–), which is a reserved character indicating the next control argument on the ASSIGN statement line. |
| **{ –7TRK / –9TRK }** | Specifies seven- or nine-track tape drive. Nine-track is the default. These options are usually used in conjunction with the MTX argument. Requires operator intervention. |
| **–DENSITY bpi** | Specifies tape density in bytes per inch. bpi must be one of the following: |

**800**
**1600**
**3200**
**6250**

On Version 3 tape drives (those which can handle 6250 bpi), the –DENSITY option sets tape density automatically. On Version 0, 1, and 2 drives (which handle 800 bpi and 1600 bpi tapes), the –DENSITY option requires operator intervention so that density may be set manually. (This option replaces the previous –800BPI, –1600BPI, and –6250BPI options.)

| | |
|---|---|
| **–RINGON** | Requests the operator to place the write ring on the tape, so the tape may be both read and written. |
| **–RINGOFF** | Requests the operator to remove the write ring from the tape. This protects the tape against writing. |

**–MOUNT** Requests the operator to mount a new tape reel.

This option is used in the middle of a mag tape procedure (such as MAGSAV or MAGRST), when one reel of tape is exhausted and another is needed. It is usually accompanied by the –TPID option.

Unlike other uses of the ASSIGN command, use of the –MOUNT option requires that the drive have been assigned already by a previous ASSIGN command.

Examples:

```
OK, AS MT1
Device MT1 assigned.
OK,
```

Magnetic tape drive MT1 is assigned. (1 is the physical device number.)

```
OK, AS MT1 -ALIAS MT0
Device MT1 assigned.
OK,
```

Physical device MT1 will now be referred to as logical MT0. The physical-to-logical number correspondence can be listed with the STAT DEV command:

```
OK, AS MT1 -ALIAS MT0
Device MT1 assigned.
OK, STAT DEV

Device User name                          Usrnum Ldevice
MT1    CLAM                                67    MT0
```

```
OK, assign mtx -alias mt4
```

The operator is requested to assign any available tape drive as logical device 4. A message is displayed at the user's terminal, indicating which physical drive has been assigned. For example:

```
Device MT0 Assigned.
```

The operator has assigned magnetic tape drive MT0 in response to the above request.

```
as mtx -alias mt3 -tpid power -9trk -ringoff -density 6250
```

The operator is requested to mount the "POWER" tape (a 9-track tape) on any drive that can handle 6250 bpi. In addition, the user wants write-protection and is assigning an alias of MT3 (ldn) to whatever device the operator chooses.

This request, if processed, might be acknowledged with this display:

```
Device MT0 Assigned.
```

If a request cannot be handled by the operator for any reason, the following message appears at the terminal:

```
MagTape Assignment Request Aborted  (ASSIGN)
ER!
```

**Operator Intervention**

The System Administrator uses the SETMOD command to determine the operator's role in tape assignments. Three modes are possible:

- Each user can assign a tape drive from any terminal; operator intervention is necessary only for processing special requests. This is the default mode.
- Each user must send all assignment requests through the operator, who controls all access to tape drives. The operator then sends messages to the user terminal indicating the status of the assignment request.
- Tape drive assignment from any user terminal is strictly forbidden. This feature is used to restrict access to tape drives in security-conscious environments, or to warn users when the operator is not available to process requests. In this mode, any attempt by a user to assign a magnetic tape drive will result in the message:

```
No MagTape Assignment.Permitted.  (AS)
ER!
```

**Disks:** In assigning disks, **pdisk** is a physical disk number (as printed by STATUS DISKS). A user may only assign a disk that is not already assigned and that appears in the Assignable Disks Table. The operator prepares this table by using the DISKS command from the supervisor terminal. This restriction provides a degree of system integrity because it prevents users from assigning a disk without the supervisor terminal operator's knowledge, or from assigning disks or partitions the operator wishes to reserve for special use. For a disk to be assigned to a user, it must not be the paging disk, nor assigned to another user, nor a disk specified in a previous STARTUP command.

To assign a disk that has been started by STARTUP or ADDISK, it must first be shut down at the supervisor terminal by the SHUTDN command.

For complete disk-assignment details, refer to the **System Operator's Guide.**

```
AS DISK 460

AS DISK 54
```

Each of the above command lines assigns a disk partition. The number is the partition (volume) number.

The maximum number of disk partitions that may be assigned to all users at any one time is ten. If an attempt is made to assign too many disks, the message:

```
ASSIGN TABLE FULL
```

is printed.

**ALMC Lines:** A user may assign an AMLC line as follows:

ASSIGN AMLC **[protocol] line [config]**

where **line** specifies a line number. Using this form of the ASSIGN command, a user may assign the AMLC line number and may set a terminal protocol and line configuration word for the line specified by the line number. Refer to the description of the AMLC command in the **System Operator's Guide** for a complete description of possible parameters for the arguments **protocol** and **config.** Values for protocol and config are summarized below.

| Protocol | Meaning |
|---|---|
| **TTY** | Normal terminal |
| **TTYHS** | TTY with per-character interrupt |
| **TRAN** | Transparent (no character conversion) |
| **TRANHS** | TRAN with per-character interrupt |
| **TTYUPC** | TTY with lower-to-uppercase translation for output |
| **TTYHUP** | High-speed TTYUPC |

| Config | Meaning |
|---|---|
| 2033 | 110 Baud |
| 2213 | 300 Baud |
| 2313 | 1200 Baud (Default) |
| 2413 | Programmable clock (default = 9600) |

A user may only assign an AMLC line if it has been configured to be assigned, and if it is not assigned to another user. For information on configuring lines, see the **System Operator's Guide**.

**SMLC Lines:** A user may assign an SMLC line as follows:

**ASSIGN SMLCnn [-WAIT]**

where **nn** is the SMLC line number between 00 and 07.

## ▶ ATTACH pathname

The ATTACH command changes the user's current directory to **pathname.** Pathnames are specified as follows:

- To specify a top-level UFD, simply give the directory name. For example:

```
ATTACH BEECH
```

- To specify a directory subordinate to your current directory, begin the pathname with an asterisk (*). For example:

```
ATTACH *>BRANCH5>TREEHOUSE
```

- To specify a directory on a particular disk partition, or to attach to the MFD of a partition, begin the pathname with the partition name. For example:

```
ATTACH <FOREST>OAK>ACORNS
```

The logical device number (ldev) may be substituted for the partition name. For example:

```
ATTACH <3>OAK>ACORNS
```

**Note**

An obsolete form of the ATTACH command used ldisk and key parameters to keep the user's "home" and "current" directories separate. This version of the ATTACH command is documented in Appendix C, Old Commands.

### Protection

To attach to an ACL directory, you must have use (U) access to all directories in the pathname.

To attach to a password directory, you must supply the appropriate password following the directory name. For example:

```
ATTACH BEECH SECRET
```

If the password to be supplied contains lowercase letters, you must enclose the password in quotes. Otherwise, all lowercase letters are converted to uppercase. If the password occurs within a pathname, the entire pathname must be enclosed in quotes. For example:

```
ATTACH 'BAGGAGE>SUIT case>PASSPORT'
```

attaches you to PASSPORT, which is a subdirectory of the password directory SUIT, for which the password is case.

### Search Order

If you do not specify a disk name as part of the pathname, all the disks to which you have access are searched. Disks are searched in the following order:

1. All local disks are searched in order of their logical device (ldev) numbers.
2. If your local system is on a network, remote disks are then searched in order of their logical device numbers.

**Note**

With duplicate UFD names, the search rules may cause some unexpected attaches.

Such unexpected attaches are illustrated by the following example. Suppose two users, Bob and Joe, are on the same system. Bob has rights to the UFD <LOCAL> on that system. Joe, however, has not got rights to the UFD. When Bob gives the command:

```
ATTACH GOSSIP
```

he is attached to <LOCAL>GOSSIP, on his local system. If Joe gives the same command, <LOCAL> is not searched, since Joe has no rights to it. The search continues, and if the system is on a network, Joe might be attached to a UFD called <FAROUT>GOSSIP on a remote system.

**Recovering From Errors While Attaching:** If an error message is returned following an ATTACH command (for example, if a UFD is not found), the user remains attached to the previous working directory.

Several examples show all ordinary attachments:

```
OK, attach <cooper>sport
OK, create baseball
OK, attach sport>baseball
OK, create gehrig
OK, create ruth
OK, ld

<COOPER>SPORT>BASEBALL (ALL), Records= 1, Quota= 3 / 0

Directories= 2.

GEHRIG           RUTH
OK, attach *>ruth
OK, create pitching
OK, create hitting
OK, ld

<COOPER>SPORT>BASEBALL>RUTH (ALL), Records= 1, Quota= 3 / 0

Directories= 2.

HITTING          PITCHING
OK,
```

## ▶ ATM

The ATM command logs you into the Advanced Text Management Option Selection Menu of Prime's Office Automation System (OAS). For detailed information, see the **OAS Advanced Text Management Guide (PT65).**

▶ ATM_ADMIN

The ATM_ADMIN command allows the Office Automation System (OAS) Administrator to create and maintain the document data base and files relating to the Document Database Indexing function. For detailed information, see the **OAS System Administrator's Guide.**

▶ **AVAIL** $\begin{bmatrix} \textbf{packname} \\ \textbf{-LDEV n} \\ * \end{bmatrix}$ **[-NORM]**

The AVAIL command prints disk usage statistics at the terminal. These statistics give the number of disk records available for use in the specified logical disk. The units of measurement are physical disk records, 2048 bytes long. If you wish AVAIL to report "normalized" records, 880 bytes long, use the –NORM option. In either case, numbers are given in decimal.

If no argument is specified, AVAIL types the number of available records on the current logical disk and the percentage of space used up on that disk, provided that users are allowed sufficient access. If these conditions are not met, the message "INSUFFICIENT ACCESS RIGHTS" is returned when AVAIL is invoked with no argument. Thus, if the current disk was logical disk three, with the packname DOCUMN, AVAIL would produce the following message:

```
OK, avail
     Volume DOCUMN
      37035 total records
      21503 records available
      41.9

OK,
```

Users can also check availability by specifying either the packname or the logical disk number of a disk. The number is given as a decimal number with the –LDEV option (e.g., –LDEV 3). For example:

```
OK, avail -ldev 3
     Volume DOCUMN
      37035 total records
      21503 records available
      41.9


OK, avail MARKET
     Volume MARKET
      59256 total records
       7082 records available
      88.0
```

Users can check availability and status of all started disks by giving AVAIL with an asterisk ( ) as argument:

```
OK, avail *

VOLUME      TOTAL       FREE
  ID        RECS        RECS   FULL
--------------------------------------------------------
SHIP       140733       3924   97.2     4463
MARKET      59256       7080   88.1    32060
DOCUMN      37035      21503   41.9    71061

OK,
```

If the System Administrator has not set up the file SYSTEM>DISCS, the AVAIL command will print the error message "NO DISCS FILE IN UFD SYSTEM". Similarly, if the System Administrator restricts access to disk information, all versions of AVAIL will fail.

The AVAIL command does not work under PRIMOS II.

See *also*: **STATUS**

▶ **BASIC [pathname]**

The BASIC command loads the Prime BASIC language interpreter. Single-precision arithmetic is standard. For further information, refer to the **Interpretive BASIC Programmer's Guide.**

When BASIC is invoked with the pathname parameter, BASIC loads and runs the contents of **pathname** as a BASIC language program, then returns to PRIMOS command level. BASIC invoked without **pathname** or **filename** starts the BASIC interpreter, for editing or other user-specified BASIC operations.

See *also*: **DBASIC**, **BASICV**, and **NUMBER**

▶ **BASICV [pathname]**

BASICV invokes a virtual-memory BASIC subsystem (BASIC/VM). If a pathname is given, BASICV takes control and runs the contents of **pathname** as a BASIC-language program. If no pathname is given. BASIC/VM assumes control and prints a prompt character and waits for input. For example:

```
OK, BASICV
BASICV REV19.0
>NEW MUMBLE
>QUIT
OK,
```

For further information, refer to the **BASIC/VM Programmer's Guide.**

▶ **BASINP pathname**

The BASINP command invokes a program that loads, from paper tape, a BASIC source program written for a computer system other than a Prime computer. **pathname** is the name of the file into which the contents of the paper tape is to be read.

*See also:* **BASIC**

▶ $$\text{BATCH} \left\{ \begin{array}{l} \text{-DISPLAY} \\ \text{-STATUS} \end{array} \right\}$$

The BATCH –STATUS and –DISPLAY commands allow users to gain information on jobs being processed by the Batch subsystem, and to judge how heavily the Batch subsystem is being used.

BATCH –DISPLAY provides the fuller information. It tells how many Batch queues hold jobs waiting to execute, and how many jobs are waiting in each queue, in addition to the user name, job-id, user number, and queue for each executing job.

BATCH –STATUS prints a one-line description that includes the number of waiting and held jobs, the number of queues with waiting and held jobs, and the number of executing jobs. If there are both waiting and held jobs, in addition to executing jobs, the total number of active jobs is also printed. If there are no active jobs, the message "No batch jobs" is displayed.

▶ $$\textbf{BATCH} \left\{ \begin{array}{l} \text{-START} \\ \text{-STOP} \\ \text{-PAUSE} \\ \text{-CONTINUE} \end{array} \right\}$$

The BATCH command for the operator tells the Batch monitor to start, pause, continue, or stop processing user's jobs. The BATCH –START command can be typed only from the system terminal. The other three commands may be used only by the operator, from any terminal.

For detailed information on the BATCH command, see the **System Operator's Guide.**

▶ $$\text{BATGEN} \left\{ \begin{array}{l} \text{-STATUS} \\ \text{-DISPLAY} \end{array} \right\}$$

BATGEN –STATUS lists name and status (i.e., blocked or unblocked) for each Batch queue. An unblocked queue accepts jobs; a blocked queue does not.

BATGEN –DISPLAY displays queuename, status, and characteristics for each queue.

BATGEN accepts other arguments, but these are used only by the System Administrator to define, modify, or delete queues. For details, see the **System Operator's Guide** or the **System Administrator's Guide.**

**Note**

> If the BATDEF file is not read-enabled by the System Administrator, the BATGEN −STATUS and −DISPLAY commands will produce error messages. In this case, users needing information about queues should consult their supervisor, the operator, or the System Administrator.

*See also:* **BATCH, JOB**

### ▶ BINARY pathname

BINARY opens a file for writing on PRIMOS File Unit 3, usually as a binary output file for use by a compiler or assembler. The file is assigned the name **pathname.** If **pathname** is a simple filename, it will be opened in the current directory. This command has the same effect as OPEN pathname 3 2.

The F77, PL1G, FTN, COBOL, and RPGII compilers, and the PMA assembler, normally open a file named filename. BIN or B_filename as the binary output file (where **filename** is the source filename). A BINARY command is useful if the user wants to send the output of several compilations to a single file.

### ▶ CDML

CDML invokes the COBOL data manipulation language. CDML is an external command. For further information, refer to the **DBMS COBOL Reference Guide** or the **DBMS Data Manipulation Language Reference Guide.**

### ▶ CHANGE_PASSWORD [old-password]

CHANGE_PASSWORD allows you to change your login password This is a string of up to 16 characters, known only to you. The string may contain any ASCII characters, with the exception of PRIMOS reserved characters. (A complete list of PRIMOS reserved characters can be found in the **Prime User's Guide.**)

After you enter the old password, the system prompts for the new password. This is done twice for verification purposes. For security reasons, the new password is not echoed back on the terminal screen or paper. (The only exception occurs if your terminal is in half-duplex mode. In this case, the password is echoed on the terminal.)

If you did not have a login password, so that your old password is null, then you can create a password by typing CHANGE_PASSWORD followed by a carriage return, and then entering the new password in response to the system prompts.

You receive an error message if you type the old password incorrectly or if, during the confirmation dialogue, the second new password differs from the first. Your password is changed only if you enter the old password correctly and if both new passwords match. In the event of an error, the old password (if any) remains valid.

In the following example, your current password is SECRET and you wish to change it to a new password. (The new password is not echoed back on your terminal screen or paper.)

```
OK, CHANGE_PASSWORD SECRET
New password? User types new password.

Reenter new password for confirmation: User types password again.

OK,
```

**Note**

You should contact your System Administrator before you attempt to create a null password. This command will not allow you to create a null password if your installation does not allow them. Similarly, you should contact the System Administrator if you have forgotten your password. If this happens, the System Administrator is the only person who can create a new password for you.

▶ **CHAP** $\left\{ \begin{array}{c} \textbf{-nn} \\ \text{[ALL]} \end{array} \right\}$ **[priority [timeslice]]** *Operator command*

CHAP is an internal command that changes a specified user's hardware priority and the amount of time that user has for processing (timeslice). A user's default priority is 1, in a range of 0 (lowest) to 3 (highest). CHAP is an operator command.

**Note**

The timeslice or priority of user number 1 cannot be modified.

▶ **CLOSE** $\left\{ \begin{array}{l} \textbf{pathname} \\ \text{-UNIT } \textbf{unit-1 [...unit-n]} \\ \text{-ALL} \end{array} \right\}$

You use the CLOSE command to close files in one of three ways:

- By name. To do this, you use the command in the form CLOSE pathname. If the file specified by **pathname** cannot be found, an error message is printed and CLOSE returns you to PRIMOS command level. Closing an already-closed file unit is not an error.

- By file unit number. To do this, you use the command in the form CLOSE –UNIT unitnumber. **unitnumber** must be expressed in decimal, and may have a value in the range from 1 to 8.

- By specifying –ALL. CLOSE –ALL closes all files and units from '1 to '176. This form of the command does not close file unit '177. (In a command file or CPL program, specify each item to be closed; do not use CLOSE –ALL or the command file itself will be closed.)

An older form of the CLOSE command allows the following syntax:

$$\text{CLOSE} \left\{ \begin{array}{l} \textbf{pathname} \\ \textbf{funit-1 [...funit-n]} \\ \text{ALL} \end{array} \right\}$$

With this form of the command, each funit represents an octal number. The octal number must be in the range from '1 to '177.

The CLOSE –ALL command also makes sure that buffers are retrieved properly and resets the state of the file system. CLOSE –ALL is sometimes useful when the user is uncertain as to the state of the files in the current directory.

If a program is stopped with a BREAK, then the CLOSE –ALL command should be given. If CLOSE –ALL is not given, a difficulty such as:

```
File in use.  OMEGA  (OPENR)
File open on delete.   OMEGA
ER!
```

may arise with the next command.

Once CLOSE –ALL has been given, the stopped program cannot be continued (started).

When issued from the supervisor terminal with a pathname, the CLOSE command closes access to the specified file for all users.

*See also:* **OPEN**

## ▶ CLUP

CLUP is the DBMS Cleanup Processor command. See the **DBMS Administrator's Guide.**

▶ **CMPF file-a file-b [file-c...file-e]** [ **–BRIEF** | **–MINL number** | **–REPORT pathname** ]

The CMPF command allows the user to compare up to five files containing ASCII text lines of any length. The files may be specified by pathnames. One file, **file-a,** is treated as the original file. The CMPF command produces output that shows text lines that were (1) added; (2) changed from a previous line or lines; or (3) deleted from the original file, and not present in the new files.

The CMPF command, along with the MRGF command, helps ease the problems of parallel software development.

| | |
|---|---|
| **file-a** | The treename of the original file (i.e., the file that is to be treated as the common ancestor of files **file-b** through **file-e**). |
| **file-b, file-c, file-d, file-e** | Optional additional files to be compared with **file-a.** |
| **–BRIEF** | Suppresses the printing of differing lines of text within the special files at the terminal. Instead, only the file identification letters and line numbers are printed. |
| **–MINL number** | Sets the minimum **number** of lines that must match, after a difference in the files being compared, in order to resynchronize all file comparison. The default value is: –MINL 3. |
| **–REPORT pathname** | Produces a file, **pathname,** containing the differences, instead of printing (or displaying) them at the terminal. |

**CMPF Operation:** When a difference is found between **file-a** and any other file specified, CMPF attempts to get the files being compared back into synchronization. The process is accomplished only when a certain minimum number of lines (default = 3) match in all files that were specified in the CMPF command line. After resynchronization is complete, lines that differ between **file-a** and any of the other specified files are printed at the terminal (or written into the report file if the –REPORT option was specified). Then, the comparison between files continues.

When the differences between files are printed (or displayed, or written), each line from **file-a** is identified by the letter A and the line number of that line. Lines of **file-b, file-c, file-d,** and **file-e** are similarly printed, using the letters B through E, respectively.

Consider the following two files:

```
FILEA               FILEB
The                 The
quick               swift
brown               red
fox                 fox
jumps               jumps
over                over
the                 the
lazy                dog
dog
```

A CMPF comparison of these two files works as follows:

```
OK, CMPF FILEA FILEB
[CMPF 19.0]

A2          quick
A3          brown
CHANGED TO
B2          swift
B3          red


A8          lazy
DELETED BEFORE
B8          dog

COMPARISON FINISHED.
2 DISCREPANCIES FOUND.

OK,
```

*See also:* **FILVER, MRGF**

### ▶ CNAME oldname newname

CNAME changes the name of a file (or directory) from **oldname** to **newname. oldname** may be specified by **pathname; newname** may not. When a pathname has been used for **oldname, newname** replaces only the final element of the pathname. The user is cautioned not to change names of special UFDs such as CMDNC0. CNAME requires delete (D) and add (A) rights to the directory.

Examples:

```
OK, CNAME OSS3 OLD.S3
OK,
```

changes the name of a file from OSS3 to OLD.S3.

```
OK, CNAME <USER>SPARE2 JHNDOE
OK,
```

changes the name of the UFD SPARE2 in the volume USER to JHNDOE.

▶ COBOL **pathname [options]**

The COBOL command loads the Prime COBOL compiler and compiles the object program from an ASCII source file, **pathname.**

For complete details, default values and examples, see the **COBOL Reference Guide.**

*See also:* **NCOBOL**

▶ COMINPUT **[pathname] [funit]** $\left\{ \begin{array}{l} \textbf{-CONTINUE} \\ \textbf{-END} \\ \textbf{-PAUSE} \\ \textbf{-START} \\ \textbf{-TTY} \end{array} \right\}$

The COMINPUT command causes PRIMOS to read its input stream from a specified file rather than from the user's terminal. Commands are processed as if they were entered at the terminal.

| | |
|---|---|
| **pathname** | The file from which input is to be read. If it is only a filename, not a pathname, it will be found in the current directory. |
| **funit** | The PRIMOS file unit number on which the input file is to be opened. If omitted, File Unit 6 is assumed. |
| **-TTY, -END, -PAUSE -CONTINUE, and -START** | Specify control flow. Their effect is discussed in the following paragraphs. |

**Note**

The COMINPUT command must be specified with at least one parameter. If CO is specified with a null parameter, the message NOT FOUND is printed at the terminal.

Command input files are especially useful for repetitive processes such as compiling and loading a series of programs, building libraries, running production jobs, etc. Command files are normally constructed with the text editor, ED.

PRIMOS reads commands from the command input file, **pathname,** by opening the specified file unit and reading, then processing the series of commands, one line at a time.

When the command line:

**CO –TTY or CO –END**

is encountered, PRIMOS switches the input stream and takes subsequent commands at the terminal. If the user has specified a file unit (funit) other than the default Unit 6 when COMINPUT was invoked, then it is necessary to specify a file unit using the form:

**COMINPUT –TTY funit**

The form:

**COMINPUT –PAUSE**

returns to command level, leaving the current command input file unit open. Thus, a user can invoke other commands or use the form:

**COMINPUT pathname funit**

to start another command file on another unit, before issuing a command of the form:

**COMINPUT –CONTINUE funit**

to continue the original command file.

The command line:

**COMINPUT –START [funit]**

may be used to restart processing of a command file following either a BREAK (or CONTROL-P) or a warm start of the PRIMOS system.

The –START option must not be used if PRIMOS command level has been entered from a command file process by means other than a QUIT; for example, by an error message an ER!. In these instances, to continue processing of a command file, the command:

**COMINPUT –CONTINUE [funit]**

should be issued.

It is advisable to use COMINPUT –START when the quit is from inside a processing program, and to use COMINPUT –CONTINUE if an error in a command line in the command file causes an automatic COMINPUT –PAUSE

Any error message that occurs during the processing of a command file causes the command input stream to be switched to the terminal. However, the command input file remains open, which allows the user to reinvoke the command that caused the error message and, subsequently, resume the command file at the command following the one that caused the error.

**Caution**

> The use of the command CLOSE -ALL within the command file closes the command input file unit (funit) and causes the message:
>
> ```
> Unit not open.  Cominput  (Input from terminal)
> ```
>
> to be displayed (or printed) at the terminal.

**Comments:** Comments may be inserted in a command input file. The comment format is:

**/*string**

The characters /* begin the comment, which ends at the end of the line. A comment may also be appended to a command line. For example:

```
SLIST BENCH.MAP  /*PRINT MAP FILE
```

Comments may be inserted at PRIMOS command level and at subcommand level for certain utilities such as the SEG LOAD subprocessor.

**Chaining Command Files:** The -CONTINUE option allows command files to be chained. The last command file in the chain must be terminated with a CO -TTY (or CO -END) to ensure that files opened in the process are closed and that the chain is terminated properly. Furthermore, the user must be careful to keep track of what file units are being opened and closed as the chain progresses, especially those file units that are opened for command input files.

*See also:* **CPL**

## ▶ COMOUTPUT [pathname] [options]

The COMOUTPUT command causes PRIMOS to direct its output stream to a specified command output file, **pathname,** as well as the user's terminal. If the file is specified by a filename, rather than an entire pathname, it will be placed in the current directory.

COMOUTPUT is not available under PRIMOS II.

The options are described below. Logical combinations of options are permissible. (Refer to the examples in this section.)

| | |
|---|---|
| **–CONTINUE** | Continues command output to a file. With the –CONTINUE option, subsequent terminal output is appended to the file specified by **pathname.** |
| **–END** | Stops command output to a file and closes the command output file unit. |
| **–NTTY** | Turns off the terminal output (i.e., does not print or display responses to command lines, including the prompt OK). Once –NTTY has been specified, terminal output is not turned on until either –TTY is specified in a subsequent COMOUTPUT command, an error occurs, or the BREAK key is hit. |
| **–PAUSE** | Stops command output to **pathname.** However, the command output file, **pathname**, remains open. |
| **–TTY** | Turns on the terminal output. |

Command output files are useful when the user desires to keep a record of his terminal transactions in the system. PRIMOS writes all command input and output responses on the file specified by **pathname.** The file is opened on File Unit 177 (or the highest file unit provided by the System Administrator).

Examples:

```
OK, COMO OUTPUT
```

This command line arranges for subsequent terminal output to be written into the file named OUTPUT. Commands and echoed responses continue to be displayed at the terminal. The file named OUTPUT is overwritten if it already exists.

```
OK, COMO -NTTY
```

Terminal output continues to the file named OUTPUT, but no terminal output is printed or displayed at the terminal.

```
OK, COMO -END
```

The file named OUTPUT is closed. Furthermore, since –TTY was not specified following a –NTTY, terminal output will not be printed or displayed until the command line:

```
OK, COMO -TTY
```

is issued.

```
OK, COMO OUTPUT -C -P
```

The file named OUTPUT is opened and positioned to end of file, but no terminal output is sent to the file.

## ▶ CONCAT [pathname] [options]

CONCAT is used to combine a number of input files into an output file suitable for spooling. Various options allow the user to specify input and output file units, title and banner generation, etc. CONCAT accepts commands either from a terminal or from a command file.

### Using CONCAT

CONCAT is invoked from PRIMOS command level. Various options can be specified on the command line, as explained below. After processing the command line, CONCAT enters one of two modes. If –COMMAND is specified on the command line, command mode is entered and CONCAT accepts commands changing the operating environment (banners, titles, page numbering, disposition of files, etc.). Commands should be given one per line. If –COMMAND is not specified on the command line, insert mode is entered, and CONCAT accepts a list of input files, one file per line.

When CONCAT is in insert mode, it prints a colon prompt (:). To leave insert mode, the user follows the prompt with an empty line or carriage return. CONCAT responds by going into command mode and printing a right angle prompt (>). The user leaves command mode by typing QUIT.

### Command Line Options

The following options may be given on the command line only. **outfile** (if specified) must be the first option; –BANNER (if specified) must be the last. Other options may be given in any order.

To specify output file:

| | |
|---|---|
| **outfile** | A filename or pathname specifying the output file. If **outfile** is omitted, the file open on the output unit is used instead. (The default output unit is Unit 2, but this can be changed with the –OUNIT option.) If no file is open, and **outfile** is not specified, CONCAT returns an error message: |

```
Output file not open  (CONCAT)
ER!
```

To specify file units:

| | |
|---|---|
| **–IUNIT n** | Specifies the unit on which the input files will be opened. The default is 1. |
| **–OUNIT n** | Specifies the unit on which the output file will be opened. The default unit is 2. If **outfile** is omitted from the command line, the file open on unit **n** will be used for output. |

To specify output file verification:

| | |
|---|---|
| **–VERIFY** | If **outfile** already exists, causes an OK TO MODIFY OLD query followed by an OVERWRITE OR APPEND query. This is the default mode. |
| **–OVERWRITE** | Causes **outfile** to be overwritten if it already exists. |
| **–APPEND** | Causes output to be appended to **outfile** if it already exists. |

To specify output file disposition:

| | |
|---|---|
| **–CLOSE** | Truncates and closes **outfile** on exit. This is the default. |
| **–OPEN** | Leaves **outfile** open on exit. No truncation of **outfile** is done. |
| **–TRUNCATE** | On exit, truncates **outfile** but leaves it open. |

To specify mode:

| | |
|---|---|
| **–INSERT** | Goes directly into insert mode (prompt is ":") and accepts a list of files to be inserted into the output file. If neither –INSERT nor –COMMAND is specified on the command line, –INSERT is assumed. |

## Options/Subcommands

The following instructions may be given as options on the command line (preceded by a hyphen), or as subcommands when CONCAT is in command mode. (For example, –HEADER is a command line option, while HEADER is a subcommand.)

**Note**

When CONCAT is in command mode, it ignores blank lines and text preceded by /*.

To specify input file disposition:

| | |
|---|---|
| **DELETE** | Deletes input file after copying it to the output file. This option has no abbreviation. |
| **NDELETE** | Does not delete input file after copying it. This is the default. |

To specify formatting:

| | |
|---|---|
| **HEADER** | Specifies title generation and banner page suppression. This is the default. The first line of each input file is read. If it is a title (begins with octal 1 in the left byte), then the line only appears as the title for the file. Otherwise the line appears both as the title line and as the first line of the file. |
| **BANNER [banner-line]** | Specifies title and banner page generation. It must be the last option on the line. A banner page is inserted between input files. It contains two lines of up to 14 large characters. **banner-line** specifies the first of these lines. It is read as raw text so that spaces are accepted. The second line is the last component of the input file treename. Titles are generated as in –HEADER mode. The banner page has the same title as the routine following it. If the optional banner-line is omitted, then that line of the banner will be left blank. |
| **NHEADER** | Suppresses both title and banner page generation. The input files are copied to the output file without modification. |
| **EJECT** | Generates a page eject between input files. Suppresses title and banner page generation. |
| **RESETP** | Resets spooler page numbering between input files. |
| **NRESETP** | Does not reset spooler page numbering between input files. This is the default. |

**Subcommands**

CONCAT recognizes three further subcommands:

| | |
|---|---|
| **TITLE [new-title]** | Use **new-title** as the header for the next input file. It is read as raw text so that spaces are accepted. If **new-title** is omitted, the filename is used. |

| | |
|---|---|
| **INSERT [file-name-list]** | If **file-name-list** is omitted, CONCAT will go into insert mode and accept the names of the files concatenated one per line. To exit from insert mode, a blank or null line may be entered. If **file-name-list** is specified, the files in the list are concatenated into the output file without entering insert mode. If an error is made in the line, the rest of the line after the error is ignored. Up to 40 files may be specified on one line, separated by spaces or commas. Treenames with imbedded spaces (i.e., passwords) must be enclosed in quotes. |
| **QUIT** | Exit from CONCAT. This is the only clean way to exit from CONCAT. |

▶ **CONFIG –DATA filename** *Operator command*
or
**CONFIG ntusr pagedev comdev [maxpag]
[altdev] [namlc] [nphan] [nrusr] [smlc]**

The CONFIG command defines system parameters. CONFIG is specified upon cold start of PRIMOS. It is an operator command, and it is issued at the supervisor terminal. For further information, refer to the **System Administrator's Guide.**

▶ **COPY pathname [new-pathname] [options...]**

The COPY command allows you to copy objects (files, directories, segment directories, and access categories) either from one directory to another or within a directory. When you copy an object, it is not removed from its original directory or altered in any way, unless you specify the –DELETE option, explained below.

**Note**

COPY does not allow the MFD, BOOT, or DSKRAT files of an MFD to be overwritten or copied. To copy a boot file to an MFD, you must first restore the new boot to memory and then save it with the name "BOOT". This restriction does not apply when these files exist somewhere other than in an MFD.

**pathname** identifies the location and name of the object you wish to copy (source object). **new-pathname** identifies the destination and name of the copied object (target object). If you do not specify a new-pathname, the object is copied into your current directory under its original name. You must have add (A) access to the directory specified in the new-pathname. You must also have delete (D) access to this directory if the object specified in the new-pathname already exists. In all cases, you must have read (R) access to the source directory in order to copy any objects.

Command line processor options, iteration, wildcards, and name generation are particularly useful with COPY. For example, the command:

```
COPY (@.LIST @.BIN) ARCHIV>(=.OLDLIST =.OLDBIN)
```

copies all files with suffixes .LIST or .BIN in the current directory to the directory ARCHIV, replacing the suffix .LIST with .OLDLIST, and the suffix .BIN with .OLDBIN, and preserving the rest of each filename. For instance, a file called TRASH.BIN in the current directory is copied to the directory ARCHIV, and renamed TRASH.OLDBIN. Chapter 4 of this guide provides brief explanations of iteration, wildcards, and name generation, which are discussed in more detail in the **Prime User's Guide.**

### Copying With Password Directories

The requirements for access are different under password directories. In all cases, you must have owner access on the target object. To delete an object, delete (D) access is required for the object. If you specify –PROTECT, then the password protecting each source object is also copied. Protection attributes include protection keys (for files, directories, and segment directories), and passwords (for directories only). If you do not specify –PROTECT, the object is copied with the system default rights. The system default rights are RWD NIL. This means that the owner has all rights and nonowners are denied all rights.

To copy the passwords of a directory, owner rights in the source object are required. PRIMOS asks your permission to copy a directory, unless you have specified the –NO_QUERY option, in which case the directory is copied without your permission. If you do not have owner rights, the copied directory acquires the system default passwords. For owner passwords the default is blank. For nonowners the default provides no password (that is, the password is null).

### Command Line Options

There are several options you can specify to modify the copy procedure. They may appear in any order on the command line. The options are:

| Option | Meaning |
|---|---|
| **–COPY_ALL** | Preserves all attributes of a copied object. It is important to note that the protection of the file system object is also copied. In other words, COPY will place a specific ACL on the new-pathname (target object), if it can, so that the protection of the target object is the same as the source object. A copied directory will have all entries protected in the same way as in the source directory. This includes putting entries into access categories. The |

| | |
|---|---|
| | target directory, however, will be protected by a specific ACL. |
| | If you do not specify this option, attributes are set to their system defaults unless you specified one of the other attribute-copying options (–DTM, –PROTECT, –QUOTA, or –RWLOCK). Also, if you do not specify –COPY_ALL, no access categories in a subdirectory are copied to the target. |
| | If you are working with an ACL directory, you must have protect (P) access on the directory to use this option. |
| **–DAM** | Converts all copied SAM files to DAM files. If you do not specify this option, the original file type is preserved. |

**Note**

The –DAM option is not applicable to directory tree subentries.

| | |
|---|---|
| **–DELETE** | Deletes the source object after copying it. The default is no deletion. You must have delete (D) access on the source directory to use this option. |
| **–DTM** | Preserves the date/time modified stamp of all copied objects. When you copy a directory, this option preserves the date/time modified stamp of each subentry in the directory. If you do not specify this option, PRIMOS resets the date/time to the current date/time (the system default). If you are working with an ACL directory, you must have protect (P) access on the directory to use this option. |
| **–FORCE** | Force-deletes delete-protected objects. If you specify the –DELETE option, this would include both the source object and an existing target object. For example, the command: |

```
COPY FRED.LIST FRED.OLDLIST -DELETE -FORCE
```

copies the file FRED.LIST to FRED.OLDLIST, and deletes both FRED.LIST and the previous version of FRED.OLDLIST, if a previous version existed. If you do not specify –FORCE, PRIMOS asks you before force-deleting an object unless you have specified the –NO_QUERY option. In this case, the protected objects are not deleted

–FORCE is very useful when you wish to overwrite a directory tree that may contain delete-protected objects.

|  |  |
|---|---|
| **–INCREMENTAL** | Copies only those objects whose dump bit is off (=0). This may be, for example, files that have not been dumped to tape. If you do not specify this option, objects are copied regardless of their dump bit settings. This option is analogous to the function of the INCREMENTAL command in the MAGSAV subsystem.<br><br>**Note**<br><br>If a directory is the object of your COPY command, all entries within that directory are copied regardless of their dump bit settings. |
| **–LEVELS [dec]** | Copies only down to the level indicated by **dec** when you are copying a directory tree. **dec** is a decimal integer from 0 to 999. If you omit this option, the entire tree is copied. If you omit the decimal integer, the default value is 0. In this case, only the top level of the tree is copied – the directory itself and none of its subentries. |
| **–NO_QUERY** | Instructs PRIMOS to resolve any unexpected or potentially dangerous situations during a COPY procedure. The default is –QUERY. |
| **–PROTECT** | Preserves the protection attributes – original protection keys, passwords, and access rights – of all copied objects. If you do not specify this option, PRIMOS uses the default ACLs in the target directory. Also, if –PROTECT is not specified, any access categories in a subdirectory will not be copied to the target. If you are working with an ACL directory, you must have protect (P) access on the directory to use this option. When you copy a directory and specify –PROTECT, all protection within the directory (passwords, ACLs, or access categories) is preserved in the copied directory. |
| **–QUERY** | Asks you to resolve any unexpected or potentially dangerous situations during a COPY procedure. This is the default mode. |
| **–QUOTA** | Preserves the maximum quota information associated with a copied directory and any of its subdirectories. If you do not specify –QUOTA, there is no limit for the maximum quota information (no restriction on the maximum directory size). If you are working with an ACL directory, you must have protect (P) access on the directory to use this option. |
| **–REPLACE** | Copies only the objects in the source and target directories. If you do not specify –REPLACE, only |

| | |
|---|---|
| | objects you select from the source directory are copied. |
| **–REPORT** | Reports the results of each successful copy operation. |
| **–RWLOCK** | Preserves the concurrency lock setting of the object you are copying. If you do not specify –RWLOCK, the read/write locks are set to the system default. If you are working with an ACL directory, you must have protect (P) access on the directory to use this option. |

**Note**

You may only alter the locks on DAM and SAM files and segment directories. All other file system types that you copy will have the system default.

| | |
|---|---|
| **–SAM** | Converts all copied DAM files to SAM files. If you do not specify this option, the original file type is preserved. |

**Note**

This option is not applicable to directory tree subentries

The following example illustrates the use of the COPY command:

```
COPY ORCHARD FOREST -LEVELS 3 -PROTECT -DTM
```

In this example, you copy the top 3 levels of the directory tree ORCHARD to the directory FOREST. Access categories or the password protecting ORCHARD are copied, as is the date/time modified information on each source object

▶ **COPY_DISK** $\begin{bmatrix}\textbf{-NOVERIFY}\\ \textbf{-DO\_VERIFY}\end{bmatrix}$ **[–NO_BADS] [–LOWEND]** *Operator command*

This operator command copies one physical disk, of any format, to another physical disk. COPY_DISK is a command for system operators, and is discussed in full detail in the **System Administrator's Guide.**

## ▶ CPL filename

The CPL command invokes the CPL interpreter and executes a CPL program.

**filename** is the name of the file you want run. If **filename** ends in '.CPL', that file is run. Otherwise, CPL looks first for **filename.CPL**, and runs it, if found. If **filename.CPL** cannot be found, CPL looks for **filename**, running it, if found.

For more information regarding CPL, see the **CPL User's Guide.**

## ▶ CPMPC pathname [–CRn] [–PRINT]

CPMPC is a card-reader or punch utility command that causes a file specified by **pathname** to be punched onto a card deck. CPMPC does not punch an end-of-file ($E) card at the end of the output deck of punched cards. (For further information on $E cards, refer to the discussion of CRMPC command.)

Parameters to the CPMPC command may be specified in any order. The value of the –CRn option is CR0 or CR1, depending on whether the first (CR0) or second (CR1) card reader/punch is specified. Before invoking the CPMPC command, the card reader/punch must be assigned by an ASSIGN CR0 or ASSIGN CR1.

The –PRINT option causes punched data to be interpreted (printed) on the card if the punch has that capability.

## ▶ CREATE pathname –PASSWORD

The CREATE command creates a new directory as specified by **pathname.** (If a simple directory name is given, the new directory is created subordinate to the current directory.)

If the new directory is created subordinate to an ACL directory, the new directory becomes an ACL directory. It is protected by default protection from its parent directory.

If you wish to create a password directory subordinate to an ACL directory, use the –PASSWORD (or –PW) option.

You must have add (A) rights to create a new directory within an ACL directory.

If the new directory is created subordinate to a password directory, it is automatically created as a password directory, whether you give the –PW option or not. (ACL directories cannot be subordinate to password directories.) The directory is created with a blank owner password, and a null nonowner password (any password will match it). The protection keys are set to RWD NIL, allowing read, write, and delete rights to the owner, no rights to the nonowner.

## ▶ CREATK

CREATK invokes a program to build a template for multiple-keyed index files. It is a part of the MIDAS subsystem. When invoked, CREATK sets up a dialog that asks the user about the kinds of keyed index files that are to be created. For further information, refer to the **MIDAS User's Guide.**

## ▶ CRMPC pathname [–CRn] [–PRINT]

CRMPC reads cards from the parallel interface card reader connected to the MPC controller and loads card image ASCII data into the file specified by **pathname.** Reading continues until one of the following occurs:

- A card is read that has $E in columns 1 and 2 (the recommended way to stop).
- There are no more cards on the reader.
- The STOP button in the card reader is pressed.
- BREAK is pressed on the terminal, stopping CRMPC.

The parameters may be specified in any order. The option – CRn specifies the number of the card reader/punch. **n** may be 0 or 1.

If the –PRINT option is specified, the contents of each card are printed on the card (if the card reader has that capability).

## ▶ CSUBS

CSUBS, an external command, invokes the COBOL DBMS subschema. See the **DBMS COBOL Reference Guide** or the **DBMS Data Description Language Reference Guide** for details.

## ▶ DATE [option]

DATE prints (or displays) the current date and time at the user's terminal. DATE is useful to date command output files.

**option** is one of the following:

| Option | Sample Output |
|---|---|
| –FULL | 82-05-06.11:05:08.Thu |
| –USA | 05/06/82 |
| –UFULL | 05/06/82.11:05:08.Thu |
| –DAY | 6 |
| –MONTH | May |
| –YEAR | 1982 |
| –TIME | 11:05:08 |
| –AMPM | 11:05 AM |
| –DOW | Thursday |
| –CAL | May 6, 1982 |
| –TAG | 820506 |
| –FTAG | 820506.110508 |
| –VIS | 06 May 82 11:05:08 Thursday |
| –VFULL | 06 May 82 |

If DATE is invoked with no option it prints:

```
OK, DATE
06 May 82 11:05:40 Thursday
OK,
```

## ▶ DBACP

Invokes data base administrator command process. Refer to the **DBMS Administrator's Guide.**

## ▶ DBASIC [pathname]

DBASIC invokes the Prime version of interpretive BASIC that has double-precision arithmetic capabilities. The operation of the DBASIC command is the same as the operation of the BASIC command.

## ▶ DBG

Invokes the source level debugger (a separately priced product). For a full discussion of debugging with DBG, see the **Source Level Debugger Guide.**

## ▶ DBUTL

The DBUTL command invokes a data base dump utility that allows users to monitor the contents of a data base schema and shared user table. DBUTL accepts only uppercase commands. For details on its use, consult the **DBMS Schema Reference Guide** and its updates.

## ▶ DEFINE_GVAR $\left\{ \begin{array}{l} \textbf{pathname} \text{ [-CREATE]} \\ \text{-OFF} \end{array} \right\}$

The DEFINE_GVAR command manipulates global variable files.

The form "DEFINE_GVAR pathname", with or without the –CREATE option, activates an existing global variable file **pathname** is the file to be activated.

The –CREATE option must be used to create a new global variable file.

The command DEFINE_GVAR –OFF turns off an active global variable file. Global variable files are also turned off when the user logs out, or when a new global variable file is activated by a DEFINE_GVAR command. (You may create more than one global variable file, but may only have one global variable file active at any time.)

Global variable files are deleted using the standard PRIMOS DELETE command. They should always be turned off before being deleted.

The DEFINE_GVAR command may be used at command level or inside a CPL program. You must create a global variable file before you define any global variables. You must activate the global variable file at the beginning of any work session during which you want to use the global variables it contains. For example:

```
DEFINE_GVAR MY_VARS -CREATE
```

creates an empty global variable file named MY_VARS

To use the file again in a later session, use the command:

```
DEFINE_GVAR MY_VARS
```

Whenever the file is active, you may add to, delete, list, and make use of any variables it contains.

**Note**

If you have a password on the directory containing your global variable file, you must use a full pathname with the password when using the DEFINE_GVAR command. For example:

```
DEFINE_GVAR <FOREST>BEECH SECRET>MY_VARS
```

*See also:* **SET_VAR**, **LIST_VAR**, and **DELETE_VAR**

## ▶ DELAY [min] [max] [margin]

The DELAY command defines a time function that delays the printing of a character after a carriage return (CR) has been output to a terminal. **min** defines the number of character-times (time it takes the system to type a character on a line) to delay when CR is output at the left margin. **max** defines the number of character-times to delay when CR is output at the right margin. **margin** defines the number of characters required to move to the right margin. If a CR is typed at some point within a line, the time delay is proportional to the number of characters typed. If **margin** is not specified, 72 is assumed; if **max** is not specified, 12 is assumed. If the command, DELAY, is given with no parameters, the default values 6, 12, and 72 are assumed; these values are adequate for most 30 cps terminals.

The DELAY command can be used prior to logging in.

Example:

```
DELAY 0 10 100
```

The DELAY command may be issued from the supervisor terminal if it is designated to be user 1. (Refer to the USRASR command.)

Delay is particularly useful for a terminal with a nonstandard line speed. In this case the command DELAY 10 should be sufficient to allow the terminal to function in the Prime computer configuration.

## ▶ DELETE pathname [options...]

The DELETE command allows you to delete objects (files, directories, segment directories, and access categories). For example, the command:

```
DELETE MARK>ONE
```

deletes the file MARK>ONE.

**Note**

> This command will not delete the MFD, BOOT, or DSKRAT files in an MFD. You may use this command to delete these files only if they exist somewhere other than in an MFD.

**pathname** identifies the name and location of the object you wish to delete (target object). You must have delete (D) access on the directory specified in the pathname.

To delete objects from password directories, you must have delete (D) access on the target object. If the file does not have deleted access, then you must have owner access on the directory.

You may specify options in any order on the command line following the pathname. The options are:

| Option | Meaning |
|---|---|
| –FORCE | Force-deletes all delete-protected objects that you select. This option is especially useful if you wish to delete an entire directory that may contain delete-protected objects. If you do not specify –FORCE, PRIMOS asks you before force-deleting an object, unless you have specified –NO_QUERY. If you do specify –NO_QUERY, no delete-protected objects will be deleted. |
| –NO_QUERY | Instructs PRIMOS to resolve any unexpected situations during a DELETE procedure. |
| –QUERY | Asks you to resolve any unexpected situations during a DELETE procedure. This is the default mode. PRIMOS always queries you for directory and access category deletion, unless you specify –NO_QUERY. |
| –REPORT | Reports the results of each successful deletion. |

## How DELETE Handles Wildcards

Some command processor options are particularly useful with the DELETE command. When your pathname includes wildcards, you can specify the –NO_VERIFY (–NVFY) command processor option with the DELETE command. If you specify –NO_VERIFY, PRIMOS queries you about the deletion of subdirectories and access categories only. If you specify both the –NO_QUERY and –NO_VERIFY options, PRIMOS does not query you about any deletions. For complete information on all command processor options, see the **Prime User's Guide.** Wildcards and other command line features are also discussed in Chapter 4 of this guide.

▶ **DELETE_VAR variable-names**

The DELETE_VAR command deletes one or more global variables from an active global variable file. For example:

```
DEFINE_GVAR MY_VARS
DELETE_VAR.UFD
```

deletes the variable .UFD from the file MY_VARS. The command:

```
DELETE_VAR .A .B .C
```

deletes the three variables .A .B .C from the active file.

*See also:* **SET_VAR, LIST_VAR**

▶ **DELSEG** $\left\{ \begin{array}{l} \textbf{segno-1 [–TO segno-2]} \\ \textbf{ALL} \end{array} \right\}$

DELSEG is an internal command that frees (deletes) segments.

The parameter **segno** is the segment number of the segment to be freed, **segno** must be '2000 or above for user 1 ('4001 and above for all other users) and not equal to '4000. To delete all the segments in a range of numbers, you use the –TO option, in the following format:

**DELSEG segno-1 –TO**

Specifying ALL deletes all segments belonging to the user at that terminal. A "BAD PARAMETER" message is the response to an illegal segment number. Deleting an already nonexistent segment has no effect.

Example:

```
OK, DELSEG 4003
```

▶ **DISKS [NOT] pdisk-0 [pdisk-1]...[pdisk-7]** *Operator command*

The DISKS command adds or removes the specified physical disk(s) to/from the Assignable Disks Table.

▶ **DMSTK [options]**

The DMSTK command produces a call/return trace of the user's command loop stack and static mode stack (if any). Its **options**, which may be given in any order, specify how the dump is to be done. Addresses are always printed in octal.

| Option | Meaning |
|---|---|
| **-BRIEF** | Specifies a short format dump, omitting condition frames and fault frames. If this option is not given, the dump will be done in full format. (Since "full format" is the default, there is no full format option.) |
| **-ALL** | Specifies that dumping is to begin with the frame from which DMSTK was called. If -ALL is not specified, dumping begins with the most recent condition frame (if there is one) or with the frame from which DMSTK was called. |
| **-FROM n** | Specifies that dumping is to begin from frame **n**. (The frame from which DMSTK is called is frame 1.) If -FROM is not given, then the -ALL option determines the starting point for the dump. |
| **-FRAMES n** | Specifies that only **n** frames of the stack are to be dumped. (**n** must be a positive decimal integer.) The default is to dump the entire stack. |
| **-ON_UNITS** | Specifies that a list of on-units established by each activation (i.e., frame) that is dumped is to be produced. |

For information on the format of the stack dump, see Appendix B.

▶ **DPTCFG** *Operator command*

The DPTCFG command constructs the configuration file needed to set up a DPTX system. For details, see the **Distributed Processing Terminal Executive Guide.**

▶ **DPTX** *Operator command*

The System Administrator uses the DPTX command to configure and enable the DPTX (Distributed Processing Terminal Executive) system, which allows the use of IBM 3271/

3277 terminals as Prime terminals and/or the use of 3271/3277 terminals or OWL *1200* terminals attached to Prime as IBM host terminals. For full details, see the **Distributed Processing Terminal Executive Guide.**

## ▶ DROPDTR

The DROPDTR command forces the dropping of the DTR (Data Terminal Ready) signal associated with an AMLC line. As such, it is used only in the following situation:

1. A user has been talking to a Prime computer over a dial-up (AMLC) line, using a "port selector" or modem.
2. The user has logged out.
3. The user now wants to disconnect from the current line and reconnect to a new line. (For example, the user may wish to login to a different node on a network.) To force the disconnect, the logged-out user gives the DROPDTR command.

(Normally, DTR is dropped following a "grace period" of up to 10 minutes. The length of the grace period is set by the System Administrator, using the AMLTIM or DTRDRP configuration directives.)

## ▶ ED [pathname]

The ED command loads and starts EDITOR, the most commonly used version of the Prime text editor. If **pathname** is specified, that file is loaded into EDITOR's text buffer in memory, and EDITOR is started in EDIT mode. Otherwise, the editor is started in INPUT mode with an empty text buffer. Files and units are automatically opened and closed. For details of ED operation, refer to the **New User's Guide to EDITOR and RUNOFF.**

**Restarting EDITOR:** If the user accidentally returns control to PRIMOS (for example, by a QUIT), the user can restart ED without losing any of the text buffer by issuing the command: START 1000 (continue from break) or START 1001 (resume in EDIT mode).

## ▶ EDB inputfile [outputfile]

The EDB command loads and starts EDB, the binary editor, which prints ENTER and waits for command input. EDB is used primarily for building and maintaining libraries of subroutines. The input and output files may be on disk or paper tape. If the pathname, **outputfile,** already exists, it is overwritten by the output file. If paper tape is used for either **inputfile** or **outputfile**, use the name –PTR. For example: EDB –PTR NEWLIB. For details, see the **Subroutines Reference Guide.**

**Note**

As of Rev. 19, EDB no longer executes properly under PRIMOS II.

### ▶ EDIT_ACCESS target acl [-NO_QUERY]

You use the EDIT_ACCESS command to modify existing access control lists (ACLs). (See the SET_ACCESS command for a description of ACLs.)

The target in the command line may be either an ACL-protected file or an access category. The recommended file suffix for access categories is .ACAT, but you need not include the suffix in the target name, unless you have another file with the same name in your current directory. If you specify the filename without a suffix, EDIT_ACCESS will search first for the filename as you gave it, and then for the filename with .ACAT suffixed to it.

If the object is an access category, its contents will be changed as specified. If the object is a file or directory protected by a specific ACL, the ACL is changed as specified. If the file is otherwise protected, you will be asked whether you want to create a specific ACL. A YES or OK creates a specific ACL containing the edited ACL.

If the object does not exist, you will be asked if you want to create an access category.

To add IDs to an ACL, or to change access for existing ACLs, specify the ID and access as usual. To delete an ID, specify it with no access (as in "FRED:").

If you specify -NO_QUERY, any actions are carried out without question.

```
EDIT_ACCESS MYFILE FRED:ALL JOHN:LUR PHIL:  /* Add FRED,
                                            /* change JOHN, delete PHIL.
```

For additional information on ACLs and access categories see the SET_ACCESS command or the **Prime User's Guide.**

### ▶ EDIT_PROFILE [pathname] [-PROJECT project-id]

The EDIT_PROFILE command invokes the PRIMOS profile editor. System and Project Administrators use the profile editor interactively to add, change, and delete information about user profiles, and the projects to which users belong.

Each user of a system has a **profile.** The profile associates each user-id with certain **attributes,** such as:

- The user's Initial Attach Point (called the **origin directory**)
- The projects to which the user belongs
- The Access Groups to which the user belongs

A **project** is an administrative grouping, such as a class, project team, or department. Note that a project is *not* the same as an access group. See Table 2-1 for further information on profiles.

Profiles and projects are discussed more fully in the **System Administrator's Guide.** For explanation of access groups, see the SET_ACCESS command, or the **Prime User's Guide.**

## How EDIT_PROFILE Operates

EDIT_PROFILE operates in three distinct modes:

- System Initialization mode
- System Administrator mode
- Project Administrator mode

Unless you are a System Administrator, you can only use EDIT_PROFILE in Project Administrator mode. Only Project Administrator commands are discussed here; the other modes are explained in the **System Administrator's Guide.**

### The Project Administrator's Job

Every project has one Project Administrator. This person is named by the System Administrator, who appoints him or her to administer that particular project. As Project Administrator, you may manage several projects. For each of your projects, you can use EDIT_PROFILE to perform the following functions:

- For existing project members, you can change their attributes as needed. You can also check their current rights by listing them.
- For the project as a whole, you can change the project profile, add new members to the list of project users, delete users from that list, and maintain the files which keep track of this information for you.
- If you manage more than one project, EDIT_PROFILE has some subcommands and options to make life easy for you. See the following discussion on administering several projects.
- You may also be responsible for the access groups for your project team. (See the SET_ACCESS command.)

Table 2-1
User Profiles

| Item | Required? | Modifiable | Notes |
|---|---|---|---|
| **User id** | X | **SA** | May be used by one or more people. |
| **Login password** | | SA or user | Usually unique. Users can change their passwords with the CHANGE_PASSWORD command. On some systems, blank or null passwords are allowed by SA. |
| **Initial Attach Point*** | X | SA or PA | Attaches user to **origin directory** at LOGIN. |
| **Project Affiliation** | X | SA or PA | SA must assign one project to each user. This may be a default project. PA may add or remove members from his/her project. |
| **List of Access Groups** | | SA or PA | SA creates **system-wide** and/or **project-specific** access groups. SA assigns users to system-wide groups, active whenever user logs in. PA may assign users to project-specific groups, active when a user logs in as a member of a particular project. |

*Initial Attach Point depends on the project, if any, that the user specifies when he or she logs in.

## Project Administrator Mode of EDIT_PROFILE

A Project Administrator can only change the attributes of members of the particular project or projects that he or she administers.

To use EDIT_PROFILE in Project Administrator mode, you must specify the –PROJECT option with the project id of your project. You therefore give the command in the following form:

**EDIT_PROFILE [pathname] –PROJECT project-id**

You supply a pathname only when your project is not on your local system. If your project is on a system other than the one to which you logged in, then you give the name of the disk partition (MFD) in which your project is kept.

For example, suppose HARRY is Project Administrator for a project called HARKNESS in a partition called HAMPER, which is on a system called SYS.H. If HARRY is logged into another system, he gives the EDIT_PROFILE command as follows:

```
EDIT_PROFILE <HAMPER> -PROJECT HARKNESS
```

## Project Administrator Commands

As Project Administrator, you can use the following EDIT_PROFILE subcommands:

| Command | Meaning |
|---|---|
| ADD_USER | Adds a new member to your project. |
| CHANGE_PROJECT | Changes the profile of your project. |
| CHANGE_USER | Changes the attributes of an existing individual project member. |
| DELETE_USER | Removes a user from the list of project members. |
| LIST_PROJECT | Lists the attributes of your project, and of one or more project members. |
| LIST_USER | Lists the attributes of an individual project member. |
| REBUILD | Rebuilds project lists and project files. |

If you manage more than one project, you may also use the ATTACH_PROJECT and DETACH_PROJECT commands, explained in the discussion of administering several projects.

### The ADD_USER Command

The Project Administrator adds users to projects and sets up user profiles with the ADD_USER command. The command format is:

**ADD_USER [user-id] [–PROFILE] [–LIKE reference]**
**[–NO_QUERY] [–PROJECT project-id]**

If you specify only a user id, that user is added to your project as a new member with the default attributes described in the project profile. To establish a new profile, use the -PROFILE option. The system then queries you to provide the profile you want.

Normally, the user id argument and the -PROFILE option will be the most useful. Project Administrators may also use the following options:

| Option | Meaning |
|---|---|
| **-LIKE reference** | Specifies existing user attributes the new user will assume. The reference identifies the user who has the existing attributes. |
| **-NO_QUERY** | Suppresses the "Check" and "Change" questions posed by the system after the new user is added. |
| **-PROJECT project-id** | Specifies the project to which the user is added. You only use this option if you administer several projects. |

### The CHANGE_PROJECT Command

You use this command to change the project profile. The format is:

**CHANGE_PROJECT [project-id] [-PROFILE] [-LIST]**

You need not specify the project id unless you administer several projects.

The -LIST option displays the latest version of project attributes. You must specify the -PROFILE option, which specifies that you want to change or list the project profile.

### The CHANGE_USER Command

You use this command to change the profile of an individual member of your project. Note that your System Administrator may restrict the attributes that you can change. For example, a Project Administrator may only assign access groups for project members from the list of groups assigned to that project by the System Administrator. The format of the command is:

**CHANGE_USER [user-id] [-PROJECT project-id] [-LIST]**

The user id identifies the project member whose attributes you wish to change. If you do not supply a user id, the system prompts you to enter one.

The -PROJECT option is useful only if you administer several projects.

The -LIST option prints the user's attributes after you have changed them.

### The DELETE_USER Command

You use this command to delete a user from your project. The command format is:

**DELETE_USER [user-id] [-PROJECT project-id]**

The -PROJECT option is useful only if you administer several projects.

### The LIST_PROJECT Command

This command lists the attributes of the specified project, as well as those of either one or all users in the project. The list always includes the project limits imposed by your System Administrator. The command format is:

```
LIST_PROJECT [project-id] [-PROFILE] [-ALL         ]
                                     [-USER user-id]
             [-OUTPUT pathname] [-APPEND] [-TTY]
```

You need not specify a project id unless you administer several projects. The options are as follows:

| **Option** | **Meaning** |
|---|---|
| **-ALL** | Lists profiles for all users in the project. |
| **-OUTPUT pathname** | Directs the output from the command into the file specified by **pathname**. This option is very useful if the output that would be listed from the -ALL option is extremely long. If you do not supply a pathname, the command will fail. |
| **-APPEND** | Adds command output at the end of the file you specified with -OUTPUT. Otherwise, if you have specified an existing file and do not use -APPEND, the contents of that file will be overwritten. |
| **-TTY** | Displays output of the command at the terminal. If you use the -OUTPUT option and also want to see the output at your terminal, use the -TTY option as well. Unless you specify -TTY when you use -OUTPUT, your terminal will *not* display the output file. |
| **-PROFILE** | Displays the project profile. |
| **-USER user-id** | Lists the profile of the project member identified by the user id. You can only specify one user id; if you want to list only user attributes, use the LIST_USER command. |

### The LIST_USER Command

You use this command to display the attributes of an individual member of your project. The format of the command is:

**LIST_USER [user-id] [–PROJECT project-id] [–ALL]**

The user id identifies the member of your project whose attributes are to be listed. If you do not specify it, the system will prompt for one.

You need not use the –PROJECT option unless you administer several projects.

You use the –ALL option to display the user's attributes in each project to which that user belongs. However, it will only display those projects which you administer.

### The REBUILD Command

You use this command to rebuild your project to hold more members. You use the command in the following form:

**REBUILD [–PROJECT project-id] [–SIZE entry-count]**

You do not use the –PROJECT option unless you administer several projects.

You use the –SIZE option to specify the total number of members you want in your project. This total should include the number of new members you expect to add to the project. If you do not use the –SIZE option, EDIT_PROFILE determines the new project size, based on the current total of project members.

Note that project members cannot log into your project while EDIT_PROFILE rebuilds it.

## Administering Several Projects

The System Administrator may appoint one person to manage several projects. If you are Project Administrator for more than one project, you will find the following features helpful.

### Current Project

When you give the EDIT_PROFILE command in Project Administrator mode, you specify a project id. The project you specify is then known as your **current project**.

Every command you give subsequently will be performed on the current project, unless you use a –PROJECT option or specify an optional project id with an EDIT_PROFILE subcommand.

There are two EDIT_PROFILE subcommands you can use specifically to set or change your current project. These are the ATTACH_PROJECT and the DETACH_PROJECT commands.

### The ATTACH_PROJECT Command

This command sets your current project for all subsequent operations. The format of this command is:

**ATTACH_PROJECT [project-id]**

### The DETACH_PROJECT Command

You use this command to clear the setting of a current project set by a previous ATTACH_PROJECT or other EDIT_PROFILE command. The format is:

**DETACH_PROJECT [project-id]**

You do not have to specify the project id. If you choose to do so, you must specify the correct id of your current project.

Once you have given this command, you have no current project. If you give subsequent commands within EDIT_PROFILE, you must therefore specify in the command line what project the command applies to.

## ▶ ELIGTS tenths *Operator command*

The ELIGTS command allows modification of the user-eligibility time slice. For details, see the **System Operator's Guide.**

## ▶ EMACS

This command invokes the EMACS character-oriented screen editor, a separately priced product. For detailed information, see the **EMACS Reference Guide** and the **EMACS Extension Writing Guide.**

## ▶ EVENT_LOG [-NET] $\begin{bmatrix} \text{-ON} \\ \text{-OFF} \end{bmatrix}$ *Operator command*

EVENT_LOG turns system and network logging on and off.

If you specify the -NET option on the command line, network logging is affected. Otherwise, system logging is affected. The other two options (-ON and -OFF) turn the type of logging either on or off. If both are omitted on the command line, the default is -ON.

Turning on system logging opens a file in the UFD LOGREC*. The name of this file is LOG.mm/dd/yy, where mm, dd, and yy comprise the date the command is issued. This file may be specified as an input file to LOGPRT. The System Administrator must create the UFD LOGREC* before system logging is turned on. For network logging, the UFD used is PRIMENET*, and the filenames are in a similar format.

For detailed information on EVENT_LOG, see the **System Operator's Guide.**

## ▶ F77 pathname [options]

The F77 command loads the Prime FORTRAN 77 compiler and compiles the object program from the ASCII file specified by **pathname.** If no conflicting options are specified, F77 will use the following default options:

```
-64V -OPTIMIZE -INTEL -UPCASE -NOBIG -LOGL -L NO -B YES
```

The F77 compiler can be used to compile programs written in FORTRAN IV. The programs may be compiled in V-mode or I-mode. They must be loaded with SEG; F77 does not support R-mode compilation.

For further information on using the F77 compiler, see the **FORTRAN 77 Reference Guide.**

*See also:* **FTN**

## ▶ FAP

FAP invokes the FORMS Administrative Processor. For further information, see the **FORMS Programmer's Guide.**

## ▶ FDL

FDL invokes the FORMS Definition Language. For further information, see the **FORMS Programmer's Guide.**

## ▶ FDML

FDML invokes the FORTRAN DML preprocessor. See the **DBMS FORTRAN Reference Guide** or the **DBMS Data Manipulation Language Reference Guide.**

## ▶ FED

This command invokes the FORMS Editor. For further information, see the **FED User's Guide.**

## ▶ FILMEM [ALL]

Under PRIMOS, FILMEM with no argument fills memory location '100 to the top of 32K with zeros. If running under PRIMOS II, FILMEM clears '100 to the top of 64K bytes, except for those locations occupied by PRIMOS II.

FILMEM ALL clears all of the user space (up to 128K bytes) (segment '4000).

## ▶ FILVER [pathname-1] [pathname-2]

FILVER invokes a file comparison and verification process for comparing runfiles.

When **pathname-1** and **pathname-2** are used, FILVER causes them to be compared for equivalence. If any differences exist, a message is printed indicating failure to verify. If the files are exactly the same, a message is printed that confirms successful verification. FILVER is an external command.

The differences are reported in a form useful for comparing runfiles. It is suggested that the user also have listings of both files compared to make use of the different information printed by FILVER. Four numbers are displayed for each difference:

**DIFF wwwwww xxxxxx yyyyyy zzzzzz**

where **wwwwww xxxxxx** describes the position of the file: **wwwwww** is a sector number (in octal); **xxxxxx** is the offset within the sector (in octal). The user must take into account the nine-word header in runfiles and any offset from a sector boundary in the starting location of the runfile. The parameter **yyyyyy** is the value of the differing word in File 1, **zzzzzz** is the value of the word in File 2; both of these parameters are in octal.

Example:

```
OK, FILVER FILEA FILEB

DIFF 000000 000000 105000 140704     DIFF 000000 000010 120255 142723
DIFF 000000 000001 124303 142337     DIFF 000000 000011 160743 152240
DIFF 000000 000002 124640 151305     DIFF 000000 000012 105000 152317
DIFF 000000 000003 127301 146717     DIFF 000000 000013 124303 153705
DIFF 000000 000004 110407 152305     DIFF 000000 000014 124640 146240
DIFF 000000 000005 160742 157711     DIFF 000000 000015 127314 126717
DIFF 000000 000006 161362 142240     DIFF 000000 000016 110407 147240
DIFF 000000 000007 162766 143725     DIFF 000000 000017 160742 151731
DIFF 000000 000020 161362 151656
DIFF 000000 000021 162766 143640
DIFF 000000 000022 120255 126720
DIFF 000000 000023 166351 151317
DIFF 000000 000024 171764 145305
DIFF 000000 000025 105000 141724
DIFF 000000 000026 110404 120307
CONTINUE= YES
DIFF 000000 000027 151720 151301
DIFF 000000 000030 110407 147304
DIFF 000000 000031 171760 105000
FILE LENGTHS DIFFER AS FOLLOWS:
FILE 1: 108 WORDS
FILE 2: 26 WORDS

 FILES NOT EQUAL
ER!
```

*See also:* **CMPF**

## ▶ FIXRAT [options] *Operator command*

FIXRAT is a maintenance program for disks running under Rev. 18 or earlier PRIMOS. When FIXRAT is run under PRIMOS, the disk on which it is working must be an assigned disk; it cannot have been started by a STARTUP or ADDISK command. For a complete discussion of FIXRAT, see the **System Operator's Guide.**

**Note**

FIXRAT does not run on disks that are running under Rev. 19 or later Primos.

*See also:* **FIX_DISK**

## ▶ FIX_DISK *Operator command*

The operator uses the FIX_DISK utility to maintain the integrity of a disk pack. FIX_DISK facilitates the handling of increasingly complex physical disk formats brought about by ACLs, quotas, and improved badspot handling.

FIX_DISK works only under PRIMOS. FIXRAT is required for PRIMOS II. Do not, however, use FIXRAT on Rev. 19 disks. For detailed information, see the **System Operator's Guide.**

## ▶ FSUBS

FSUBS invokes the FORTRAN DBMS subschema. See the **DBMS FORTRAN Reference Guide** or the **DBMS Data Description Language Reference Guide.**

## ▶ FTGEN

*Operator command*

FTGEN is the File Transfer Service (FTS) command for the System Administrator. It allows the Administrator (logged in as "SYSTEM") to configure the FTS system at a particular site, to initialize, and to validate the FTS data base. It also enables the Administrator to display and modify the configuration. For detailed information on FTGEN, see the **System Administrator's Guide.**

*See also:* **FTS**

## ▶ FTN pathname [options]

The FTN command loads the Prime FORTRAN compiler and compiles the object program from an ASCII file, **pathname.** The **FORTRAN Reference Guide** gives a detailed discussion.

*See also:* **F77**

## ▶ FTOP

*Operator command*

FTOP is the File Transfer Service (FTS) operator command. It provides special FTS operational privileges to the operator logged in as "SYSTEM". The operator can use the FTR command under FTS to monitor and control user file transfer requests. With FTOP, the operator can initialize, monitor, and control the FTS servers. For detailed information on FTOP, see the **System Operator's Guide.**

*See also:* **FTS, FTR**

## ▶ FTR

The File Transfer Request utility (FTR) allows you to request and manage file transfers interactively between Prime computers connected by PRIMENET. After you have made a request, FTR enables you to display, modify, suspend, release, abort, or cancel it. FTR is part of the File Transfer Service (FTS), which is a separately priced product.

You request a file transfer using FTR. FTR then submits it to FTS, which queues it for transfer. Since FTS queues all requests on the local computer, you can make requests even when a communications link or remote computer is down.

### Source and Destination Sites

File transfers take place between sites. A **site** is a single computer, identified by a unique site name; Prime sites normally use their PRIMENET nodenames as site names. Files are transferred from a source site to a destination site. One of these must be your local site; the other is usually a remote site. (FTR cannot be used to transfer files between two remote sites.)

The following discussion assumes that you are using FTR between Prime machines that have been configured using the FTGEN command. To transfer files to or from sites which are not configured, see the **PRIMENET Guide.**

### Request Names

Each request has two names associated with it; an external name and a unique internal name. You can use either name to identify the request. The **external name** is either the name of the file to be transferred, or a specific name that you assign to the request when you submit it. You usually refer to the request by its external name. FTS assigns the **internal name**, always a number, which you may use to distinguish between two requests which have the same external name.

FTR offers two distinct functions. You use one to submit a transfer request; you use the other to manage requests. Each function has its own FTR command life format.

## Submitting a Transfer Request

To request the transfer of a file you use the FTR command in the following format:

**FTR source-file [destination-file] [options]**

The **source-file** is a pathname that specifies the name of the file to be transferred. If you specify a filename rather than a full pathname, then the file must be in the directory to which you are currently attached in order for the request to be honored.

The **destination-file**, also a pathname, specifies what name the file will be given at the destination site after it has been transferred. You do not specify the destination file if you are using the –DEVICE option.

The **request submittal options** allow you to specify the source and destination sites, and how FTS will handle the request. Some of the most useful options are as follows:

| **Option** | **Meaning** |
|---|---|
| **–DSTN_SITE sitename** | Specifies the name of the destination site. This option is required when you are transferring a file *to* your local site. The default is your local site. |
| **–DSTN_USER username** | Specifies the owner of the file at the destination site. |
| **–DEVICE devicename** | Allows you to transfer a file to a line printer at a remote site, by specifying –DEVICE LP. |
| **–HOLD** | Holds the request until a user or operator releases it. (See the –RELEASE request management option.) Requests are not held unless you specify this option. |
| **–LOG pathname** | Specifies the pathname of the request log file. This file shows a log of events such as the submission or termination of your request. You can check the log file to see whether your request was successful. |
| **–NAME external-name** | Specifies the external name of the request. The default external name is the name of the source file. |
| **–SRC_SITE sitename** | Specifies the source site. This option is required when you are transferring a file *from* a remote site. The default is your local site. |
| **–SRC_USER username** | Specifies the owner of the file at the source site. |

For a list of all the request submittal options and a full description of their functions, see the **PRIMENET Guide**.

### Sending a File

To send a file called NEWS from your local site to a Prime computer called AB.B you could use FTR as follows:

```
FTR SALT>NEWS PRESS>LATEST -DS AB.B -LOG PEPPER>DEADLINE
```

In this case, you send the file NEWS to PRESS>LATEST on system AB.B. Events such as the submission of your request are logged in the file PEPPER>DEADLINE.

### Fetching a File

To fetch a file from system AB.C you could use FTR as follows:

```
FTR LEAD>LAST>COMMENTS SALT>COMP -SS AB.C -LOG PEPPER>DEADLINE
```

In this case, you transfer the file LEAD>LAST>COMMENTS from system AB.C into SALT>COMP on your local system. Again, your log file is PEPPER>DEADLINE.

### Managing Transfer Requests

To manage requests after they have been submitted, you use the FTR command in the following format:

**FTR option**

**requestname** identifies the file transfer request. You may use either the internal or external name as the requestname.

The **request management options** specify the type of action desired. The available options, for which there are no abbreviations, are:

| Option | Meaning |
|---|---|
| **–ABORT requestname** | Aborts a file transfer request. |
| **–CANCEL requestname** | Cancels a file transfer request. |
| **–DISPLAY requestname** | Prints the contents of one or more request(s). |
| **–HOLD requestname** | Delays a file transfer until a user or operator releases the request using the –RELEASE option. |
| **–MODIFY requestname [options]** | Modifies the characteristics of the specified request. The options you can use with –MODIFY are the request submittal options. |
| **–RELEASE requestname** | Releases a held request. |
| **–STATUS [requestname]** | Displays the status of one or more request(s). |

### Checking the Status of a Request

To check on the progress of requests that you have submitted, you use the –STATUS request management option, as follows:

```
FTR -STATUS
```

This produces a one-line report on the status of each of your waiting transfer requests, as shown in the following example:

```
OK, ftr -status
[FTR rev 1.0]
82-06-06.10.43:15 steven news (36949288) Status -transferring
```

For complete information on FTR, see the **PRIMENET Guide.**

## FTS

The Prime communications File Transfer Service, FTS, is a separately priced product which allows you to transfer files between computers connected by PRIMENET. FTS is not a command. The commands used with FTS are as follows:

- The FTR command provides the user interface to FTS.
- The FTOP command provides operator control of FTS.
- The FTGEN command allows the System Administrator to define the FTS environment.

*See also:* **FTR**, **FTOP**, and **FTGEN**

For full information on FTS, see the **PRIMENET Guide.**

## ▶ FUTIL

FUTIL invokes a file utility command that provides subsystem commands for the user to copy, delete, and list both files and directories. FUTIL also has an ATTACH command that allows attaching to subdirectories by giving a directory treename for either the MFD or home UFD to the specified subdirectory, FUTIL allows operations with files within UFDs and files within segment directories. FUTIL may be run from a command file.

At Rev. 19, the new COPY, DELETE, LD, and SIZE commands, with their ability to use command line enhancements such as iteration, treewalking, and name generation, provide easier methods for tasks previously done by FUTIL. However, work done under PRIMOS II still requires the use of FUTIL, as the new commands operate under PRIMOS only.

See Appendix C for a detailed discussion of subsystem commands available under FUTIL.

## ▶ HDXSTAT

The HDXSTAT command allows you to display the current status of all lines and sites of a half-duplex (HDX) network configuration. When a network connection is defined as half duplex, communication between two Prime systems lasts for the duration of the phone call only. For complete details, see the **PRIMENET Guide.**

The HDXSTAT command shows each defined HDX site, each defined HDX line, the association (where one exists) between individual sites and lines, the state of the telephone connection, and the status of the link. The link status will be one of the following:

| | |
|---|---|
| **assigned** | The line has been reserved for use by HDX PRIMENET, but is not yet set to receive or initiate calls. |
| **awaiting call** | A line has been set to receive a call from any remote site but no remote site has yet called in. |
| **not assigned** | The line is not in use by HDX PRIMENET. |
| **offline** | No connection to the site exists. |
| **running** | The network connection to the remote site is up. |
| **trying to establish** | A line has been set to initiate calls to a remote site, but contact with the site has not been made. |

## ▶ HELP $\begin{bmatrix}\textbf{command-name}\\ \textbf{topic-name}\end{bmatrix}$

The PRIMOS HELP command allows you to access online information about commands, categories or groups of commands, and other PRIMOS topics. If you type HELP with a command or topic name, the system prints the information on your terminal. If you type HELP without a command or topic name, the system prints a list of commands and topics for which information is available. For example, if you want information on the PRIMOS ASSIGN command, type:

```
OK, HELP ASSIGN
```

The system prints the command name, a one-line description of the command, the command abbreviation (if any), the command line format, and text summarizing the command function, options, etc. At the end of the listing are any references to further information, along with the date the HELP information was created or updated. Similarly, if you want information on Prime technical publications, type:

```
OK, HELP DOCUMENTS
```

The sytem prints the list of current manuals available from Prime and how to order them.

## ▶ HPSD

HPSD loads a version of PSD (Prime Symbolic Debugger) that is stored in the upper portions of memory. See the **Assembly Language Programmer's Guide** for details.

## ▶ INPUT pathname

INPUT opens a source file on File Unit 1 for reading. The file is assigned the name **pathname**. If **pathname** is only a simple filename, it is in the current UFD. This command has the same effect as OPEN pathname 1 1. (For PMA and FTN, the source filename is usually provided with the command that starts assembly or compilation.) INPUT is an internal command.

## ▶ JOB $\left\{\begin{array}{l}\textbf{filename}\\ \textbf{job-id}\end{array}\right\}$ [options]

The JOB command controls individual jobs run in the Batch subsystem. It has three major groups of options that allow users to submit, monitor, and control their jobs, and a fourth group that provides operator control for waiting or executing jobs.

### Submitting a Job

The format and options for submitting a job are:

$$\textbf{JOB filename} \left[\begin{array}{l}\textbf{-ACCOUNTING information}\\ \textbf{-ARGS cpl-arguments}\\ \textbf{-CPL}\\ \textbf{-CPTIME seconds}\\ \textbf{-ETIME minutes}\\ \textbf{-FUNIT number}\\ \textbf{-HOME pathname}\\ \textbf{-PRIORITY value}\\ \textbf{-QUEUE queuename}\\ \textbf{-RESTART} \left\{\begin{array}{l}\textbf{YES}\\ \textbf{NO}\end{array}\right\}\end{array}\right]$$

If the simplest form, **JOB filename,** is used, the Batch monitor looks for a file in the user's working directory. It looks first for **filename.CPL,** then for **filename.** (Batch assumes that all files whose names end in .CPL are CPL files, and that all other files are command files.)

If it finds either **filename.CPL** or **filename**, the monitor gives the job the user's login name, places it in an appropriate queue, assigns it a **job-id, #snnnn** (where **s** is the number of the queue, and **nnnn** is the job's number within the queue), and fills in the remaining options with the chosen queue's default values.

**Note**

If the file being submitted as a Batch job resides in a passworded directory, or if it is to be run from a passworded directory, the –HOME option must be used, with the requisite password provided within it. Otherwise, the Batch monitor will not be able to perform the necessary ATTACH commands, and the job will fail.

The options themselves have the following values and meanings:

**Note**

All numbers must be given as decimal integers.

| Option | Description |
|---|---|
| **–ACCOUNTING information** | Allows the user to specify accounting information for his job. **information** must be 80 characters or less in length. It may not be an explicit register setting or be preceded by an unquoted minus sign. If it contains spaces, commas, or comment designators (/*), it should be enclosed in apostrophes. The information will be included in job displays, but will otherwise be ignored by Batch. |
| **–ARGS cpl-arguments** | Used to pass CPL arguments to the job being processed. –ARGS must be the last option issued on a command line, as everything that follows the –ARGS option on the command line (except comments) is assumed to be the CPL arguments being passed. JOB doesn't read the CPL arguments; it just passes them to the CPL file when execution of the file begins. |
| **–CPL** | Runs submitted file as a CPL file, no matter what the file's name is. |
| **–CPTIME** { **seconds** / NONE } | Specifies the maximum amount of CPU time (in **seconds**) to be allotted to the job. **NONE** requests that no time limit be placed on the job. If the job exceeds the time limit, it will be aborted. If the CPTIME specification conflicts with the CPTIME limit of the requested queue, an error message will ask the user to specify a new limit. If it conflicts with all queues (and no queuename was specified) the error message "no queue available for job" will be printed. |

| | |
|---|---|
| **–ETIME** {**minutes** / NONE} | Specifies (in **minutes**) the elapsed time to be allowed before the job is aborted. Details are the same as those for –CPTIME. |
| **–FUNIT number** | This option is used for command input files only. It *cannot* be used for CPL programs. When used, it specifies the file unit to be used for command input. Permissible values are from 1 to 126, unless the System Administrator has set a lower limit. (Smallest possible range is 1 to 16.) Default depends on queue parameters; it is usually 6. (CPL jobs have file units allocated dynamically.) |
| **–HOME pathname** | Specifies the UFD in which a job is to run. Using this option has the same effect as providing an ATTACH command as the first line of the command file. The pathname for a –HOME option, however, may not be a null specification or a relative pathname (i.e., it may not begin with *>), and may not exceed 80 characters in length. |
| **–PRIORITY value** | Determines the job's priority within its queue. Possible **values** are from 0 to 9, with 9 being the highest (most favored) priority. Default is queue-dependent. |
| **–QUEUE queuename** | Allows the user to specify the name of the queue in which his job should be placed. (To learn the names and characteristics of queues, use the BATGEN –DISPLAY command.) If a conflict results between –QUEUE and some other option (e.g., if a user specifies –CPTIME NONE –QUEUE ALPHA, when queue ALPHA allows a maximum of 120 seconds CPU time), an error message will request the user to resubmit the job with different options. |
| **–RESTART** {YES / NO} | Determines whether a job can or cannot be restarted following a –RESTART command or a system shutdown. The default is always –RESTART YES. |

**Note**

Any or all of the above options may be specified within the command input file itself by using a $$ JOB command as the first non-comment line of the file. The command format is:

**$$ JOB user-name [options]**

where **user-name** is either * or the name the user logged in with, and **options** represents the eight JOB options listed above. If **user-name** is *, any user may submit the file to Batch; otherwise, only people logged in as **user-name** can submit it.

Example:

```
    $$ JOB JONES -HOME JONES>REPORTS -CPTIME NONE -ETIME NONE
```

Any job may specify options via the JOB command, the $$ JOB command, or both. If one option is specified twice, the JOB option overrules the $$ JOB option. For example, if $$ JOB called for –CPTIME 20 and JOB called for –CPTIME 10, the job would be allowed only 10 seconds of CPU time before being aborted.

## Monitoring the Job

While a job is in a Batch queue, its progress can be monitored by the JOB command form:

$$\text{JOB} \begin{bmatrix} \textbf{job-id} \\ \textbf{jobname} \end{bmatrix} \begin{Bmatrix} \text{–STATUS} \\ \text{–DISPLAY} \end{Bmatrix}$$

Within this format, the –STATUS and –DISPLAY options govern the amount of information to be shown, while the jobname and job-id options allow the user to specify the jobs on which he wants information, as follows:

| Option | Description |
|---|---|
| **job-id** | A five-digit number preceded by a #, assigned to a job by the monitor when the job is placed in a queue. Use **job-id** to request information on one job only. |
| **jobname** | The name of the file being run. If the job was submitted as a pathname (e.g., JOB FELLOWSHIP>HOBBITS>FRODO), its **jobname** is the final element of the pathname (e.g., FRODO). Use this format to request information on multiple submissions of a file. |

**Note**

Omitting **jobname** and **job-id** requests information on all the user's jobs.

| | |
|---|---|
| –STATUS | Prints out the job's **jobname** and **job-id**, the name of the queue in which it is placed, and its execution status: whether it is held, waiting, or running. |
| –DISPLAY | Provides status information and values for all JOB and $$ JOB command options, both those specified by the user and those assumed from queue-defined defaults. |

### Job Control

The following forms of the JOB command may be used to control the job's execution or characteristics once it is in the queue.

```
                 -  ORT
JOB job-id       -  AN  E [options]
                      CEL
                  E   ART
```

**Note**

**jobname** may be used in place of **job-id** only if the user has only one active job of that name.

| **Option** | **Description** |
|---|---|
| **–CANCEL** | Prevents the execution of jobs waiting to run. Canceling a running job will not halt its execution. However, it will make it impossible to restart that particular job, unless –CHANGE –RESTART YES is performed. |
| **–ABORT** | Terminates the execution of a running job and cancels a held or waiting job. |
| **–CHANGE** | Allows the user to change the –ACCT, –ARGS, –CPTIME, –ETIME, –FUNIT, –HOME, or –RESTART options on a waiting job. –CPL, –QUEUE, and –PRIORITY cannot be changed. |

If the job is already running, changes can be made by invoking first the JOB –CHANGE command and then the JOB –RESTART command.

Example:

```
JOB TEST4 -CHANGE -CPTIME 120 -ACCT MARKETING -RESTART YES
JOB TEST4 -RESTART
```

The job will abort, then restart under the newly specified options.

### Operator Commands

The operator's form of the job command is:

```
                 -ABORT
                 -CANCEL
JOB job-id       -HOLD
                 -RELEASE
                 -RESTART
```

With the –HOLD option, the operator can hold a waiting job in the queue, keeping it active but preventing it from running until he releases it. Users with jobs that need unavailable peripheral equipment, or jobs that should run after a particular time may ask the operator to hold the jobs until the equipment is available or the time is right.

The operator's –ABORT, –CANCEL, and –RESTART options are identical to the user options with those names.

## ▶ KBUILD

KBUILD builds a keyed-index file of fixed-length records. It is part of the MIDAS subsystem. When invoked, KBUILD asks the user a series of questions about the file to be built. For further information, refer to the **MIDAS Reference Guide.**

## ▶ KIDDEL

KIDDEL invokes a program to delete all or part of the records in a keyed index file. It is part of the MIDAS subsystem. For further information, refer to the **MIDAS User's Guide.**

## ▶ LABEL

```
                                ⎧ -VOLSER ⎫
LABEL MTn [-TYPE type] [        ⎨ -VOLID  ⎬ volume-id]
                                ⎩ -VOLUME ⎭
[-OWNER owner] [-ACCESS access] [-INIT]
```

LABEL initializes magnetic tapes just as MAKE initializes disks. LABEL writes either IBM (9-track EBCDIC or 7-track BCD) or ANSI (9-track ASCII) level 1 volume labels followed by dummy HDR1 and EOF1 labels. LABEL can also be used to read existing VOL1 and HDR1 labels. ANSI labels are written in accordance with the American National Standards Institute standard ANSI X3.27-1978. IBM labels are written in accordance with IBM's specifications (IBM) manual GC28-6680-5). Any nonstandard labels such as 7-track ASCII or user-defined labels *cannot* be read or written.

### Using LABEL

To read existing labels type the command:

**LABEL MTn [–TYPE type]**

To write labels type the command:

**LABEL MTn [–TYPE type] –VOLID volume-id [–OWNER owner]**
**[–ACCESS access] [–INIT]**

| | |
|---|---|
| **MTn** | The tape drive where the tape to be labelled is located. **n** is a number between 0 and 7. This keyword is required and must be the first on the command line. |

| | |
|---|---|
| **type** | May have one of three values: |
| | **A** 9-track ASCII (ANSI) (This is the default.) |
| | **B** 7-track BCD (IBM) |
| | **E** 9-track EBCDIC (IBM) |
| **volume-id** | A 1-6 character string which uniquely identifies this tape reel. If less than 6 characters are specified, they are blank-padded on the right. The keywords −VOLUME or −VOL may be substituted for the keyword −VOLID. |
| **owner** | 1-14 characters long for ANSI labels, 1-10 characters long for IBM labels. If less than 14 (or 10) characters are specified, they are blank-padded on the right. If this keyword is omitted, the default is the user's login name. The keyword −OWN may be substituted for the keyword −OWNER. |
| **access** | A single character defining access to this tape. ACCESS is not used by Prime software but is included for completeness. If it is omitted it is left blank on ANSI labels. ACCESS is ignored for IBM labels. |
| **−INIT** | Necessary keyword for previously unformatted tapes. |

Improper use of the LABEL command causes an error message to be printed. These errors are the result of bad syntax in the LABEL command itself or a system magnetic tape I/O error. Most of the messages are self-explanatory. For details consult the **Magnetic Tape User's Guide.**

If LABEL successfully writes a label, the message "TAPE LABEL WAS WRITTEN SUCCESSFULLY" is displayed. On read operations, LABEL prints out the volume and owner ids, creation date, access (ANSI tapes only), and other information.

### HELP Facility

The command LABEL −HELP causes LABEL to print out a description of the command similar to that found in this document.

For a complete description of tape labels and their use, refer to the **Magnetic Tape User's Guide.**

## ▶ LATE

LATE requests the time at which the next command is to be accepted. The LATE command responds as follows:

```
Time of day (HHMM) to execute next command:
```

The user then types in the time of day. The time of the next command is expressed as a number of the form **HHMM**. **HH** is the hour (00 through 23), and **MM** is the minute (00 through 59). LATE responds to this input with the message:

```
Next command will be executed at HH:MM.
```

If the specified time is earlier than the current time, execution of the command is deferred until the following day. For example:

```
OK, late
[LATE rev 18.0]

Time of day (HHMM) to execute next command: 0230

Next command will be executed at 02:30 tomorrow.

OK, late
[LATE rev 18.0]

Time of day (HHMM) to execute next command: 2245

Next command will be executed at 22:45.
```

If the colon is omitted from the time specification, LATE assumes the last two digits designate minutes. For example, 30 is interpreted as 00 hours and 30 minutes. If no time specification is made, the system defaults to 00 (midnight).

If the colon is included, the first two digits designate hours and the last two minutes.

No other commands can be executed before the specified time except a CONTROL-P to abort from LATE. LATE is useful if a user wishes to defer execution of a process, such as a command file, until a time when it is expected that system load will be light, such as during the second shift.

▶ **LD [pathname] [wild1...wild15] [options]**

The LD command allows you to list the contents of a directory and, optionally, the various directory entry attributes. You may select entries by the standard command processor options and also by the particular type of access control protecting the entry.

**pathname** identifies the directory to be listed It may also contain the first wildcard name. For example, the pathname:

```
TEST>MYDIR>@.SPEC
```

would list entries in the directory TEST>MYDIR whose names match the wildcard @.SPEC. If you do not specify a pathname, all entries in your current directory are listed.

You may also specify additional wildcard names on the command line. An entry is listed if it matches either the entry that is part of the pathname and/or one of the wildcard names. Wildcards, and command processor options, such as –FILE, –DIR, and –ACAT that are useful when using wildcards with the LD command, are discussed in Chapter 4 of this guide and in the **Prime User's Guide.**

## Command Line Options

You may specify one or more options on the command line in any order. Options are listed as follows: formatting options, sorting options, detail options, protection options. Available options are:

| **Formatting Option** | **Meaning** |
|---|---|
| –SINGLE_COLUMN | This option applies only if the default output format is used (names only displayed). It prints the names of the entries one per line, as opposed to four or fewer per line in the default format. |
| –NO_HEADER | Suppresses the header line, which shows pathname, current user's access rights in an ACL directory, number of records used by the directory and its files, and quota information. If you do not specify this option, output includes a header line with this information. If you specify –NO_HEADER with –SORT_NAME or –SORT_DTM, this option also suppresses the type counts. This option is most useful if you combine it with the –SINGLE_COLUMN option. |

| **Sorting Option** | **Meaning** |
|---|---|
| –NO_SORT | Does not sort listed entries. If this option is not specified, entries are listed and sorted alphabetically by name within the entry type. Types are always sorted in the order: file, segment directory, directory, and then access category. |
| –REVERSE | Reverses the sort order from its default. This never affects the sort order of entry types. |
| –SORT_DTM | Sorts entries by descending date/time modified within their type. You cannot specify this option if you also specify –SORT_NAME on the command line. |

| | |
|---|---|
| **–SORT_NAME** | Sorts entries alphabetically by name only, not within their type. You cannot specify this option if you also specify –SORT_DTM on the command line. |

| Detail Option | Meaning |
|---|---|
| **–DETAIL** | Lists all attributes for each entry you select. (The default output format lists only the names of the entries, four across.) The attributes are displayed from left to right in the following order: |

1. Access rights available to you; corresponding protection keys are displayed for password-protected entries.
2. The entry size in physical disk records.
3. The entry quota in physical disk records (for directories only).
4. The entry type (file, segment directory, directory, or access category).
5. The entry concurrency lock setting, which specifies how many people can be accessing an entry at any given time. The setting will appear as a blank (      ) for the system default, **excl** if the entry allows any number of readers or one writer at a given time, **updt** for any number of readers and one writer at a given time, or **none** for any number of readers and writers at a given time.
6. The incremental dump switch. **dmp** appears if the entry has already been dumped.
7. The delete-protection switch. **pr** appears if the entry is protected from deletion.
8. The date/time modified information.
9. The entry name.
10. The type of protection on the entry. This can be the name of an access category that protects the entry. Or, if the entry is protected by a specific ACL, (Specific) appears. A blank (      ) appears if the entry is protected by default access.

The following example illustrates output from the LD –DETAIL command.

If you wish to list only a subset of the attributes provided by –DETAIL, you may specify one or more of the following options:

```
OK, ld -detail

<MARKET>SALLY (ALL), Records= 50, Quota= 54 / 0

Files= 11.

ALL           7 sam       dmp   16 Apr 82 10:15:44  $EMEMO
ALL           7 sam       dmp   16 Apr 82 13:17:48  $EMEMO2
ALL           1 sam       dmp   26 Mar 82 13:58:00  SALLY.ABBREV
ALL           1 sam       dmp   02 Feb 82 10:19:48  C*LOGIN
ALL           6 sam       dmp   16 Apr 82 10:15:08  EMEMO
ALL           6 sam       dmp   16 Apr 82 13:17:20  EMEMO2
ALL           1 dam       dmp   22 Apr 82 11:56:56  LAC.COMO
ALL           1 sam       dmp   15 Apr 82 11:59:04  LOGIN.CPL
ALL           1 dam       dmp   26 Mar 82 11:40:52  ROAN.COMO
ALL           5 dam       dmp   09 Apr 82 11:16:48  STAR
ALL           7 dam       dmp   05 Apr 82 16:36:32  STATUS.COMO

Directories= 3.

ALL           1 dir     0 dmp   23 Apr 82 09:48:56  BARE       (Specific)
ALL           1 dir     0 dmp   23 Apr 82 09:47:12  FOOD       (Specific)
ALL           1 dir     0 dmp   23 Apr 82 09:50:08  ZOO        hunk.acat

Access categories= 1.

ALL            acat             23 Apr 82 09:50:00  HUNK.ACAT
OK,
```

| Option | Meaning |
|---|---|
| **–DTM** | Prints only the date/time modified information for each entry. |
| **–PROTECT** | Prints only the protection information for each entry. Protection information includes the access rights, the delete-protect switch, and the type of protection. |
| **–SIZE** | Prints size information for each entry. A size of –1 is shown for all entries for which you do not have read (R) or list (L) access. |

The LD command line by default lists all entries that match the wildcards you specify. The following three options allow you to list only those entries that are protected by a specific type of access control.

| Protection Option | Meaning |
|---|---|
| **–CATEGORY_PROTECTED [name]** | Lists only the entries that are protected by the access category identified by |

| | |
|---|---|
| | name. If you do not supply the access category name, then all entries protected by any access categories are listed. |
| **-DEFAULT_PROTECTED** | Lists only the entries protected by default access. |
| **-SPECIFIC_PROTECTED** | Lists only the entries protected by specific ACLs. |

## ▶ LISTF

LISTF prints the current UFD pathname (if the pathname is less than 80 characters long), the logical device upon which the UFD resides, and all filenames in the UFD.

At Rev. 19.0, LISTF has been made obsolete by the more powerful and flexible LD command. See the entry for LD for details.

For further information on LISTF, see Appendix C, Old Commands.

## ▶ LISTING pathname

LISTING opens a file for writing on File Unit 2, usually as a listing output file for a compiler or assembler. All subsequent compilation and assembly listings go to this file until it is closed. **pathname** specifies the directory and filename of the listing file. If **pathname** is only a simple filename, it is in the current UFD. LISTING is an internal command, and has the same effect as OPEN pathname 2 2.

## ▶ LIST_ACCESS [object]

You use the LIST_ACCESS command to examine the access rights for a file system object. If you do not specify a particular object, the command lists the access rights for your current directory.

If the object you specify is an access category, the system lists the contents of that category, the Access Control List (ACL). Although the file suffix for access categories is .ACAT, you need not include the suffix when you specify an access category, unless there is a file with the unsuffixed name in your current directory.

If the object you specify is not an access category, the command lists the ACL that protects the object you specify. The priority ACL, if any, is always included in the display.

For example, if you want to list the contents of TOP.ACAT, you can give the command:

```
LIST_ACCESS TOP
```

This will list the contents of TOP.ACAT, unless you have a file called TOP in your current directory, in which case it will list the contents of the ACL protecting TOP.

The following example illustrates the use of LIST_ACCESS without a specified object:

```
OK, LIST_ACCESS

ACL protecting   "<Current directory>"
      ALEX:        ALL
      .LEAD:       ALL
      $REST:       LUR
```

In this example, the user with id ALEX and members of the access group .LEAD have all rights to the current directory. All other users have list, use, and read rights. For more information on ACLs and access categories, see the SET_ACCESS command or the **Prime User's Guide.**

## ▶ LIST_GROUP

This command lists the ACL groups to which you belong. The groups may determine your access rights to certain files. Each user may be a member of up to 32 groups. If you are not a member of any group, PRIMOS gives you the message "No groups." Otherwise, it lists group names, as follows:

```
OK,  LIST_GROUP
Groups are:   .HELP   .ADMINISTRATORS   .LEADERS
```

See the **Prime User's Guide** for further information on ACLs and groups.

## ▶ LIST_PRIORITY_ACCESS disk-name

This command lists the priority access on any given disk partition. You normally use the LIST_ACCESS command to examine all access rights, including priority ACLs, that apply to a particular file system object. The LIST_PRIORITY_ACCESS command has been provided since a priority ACL can prevent you from accessing directories and from using the LIST_ACCESS command.

## ▶ LIST_QUOTA [pathname] [-BRIEF]

This command lists current disk quota and storage information for the directory specified by **pathname**. If you do not specify a pathname, LIST_QUOTA gives the information for your current directory. To use LIST_QUOTA, you must have list (L) access to the target directory and use (U) access to all higher directories.

**Quotas** are limits placed on the size of a directory. The limits are sized in disk records. A directory on which a quota has been set cannot obtain records that cause it to exceed its quota. This restriction is enforced on the entire subtree. If various levels of a subtree have multiple quotas, then the most restrictive quota is enforced.

The following example shows quota information for the subdirectory STATS in the UFD TEST:

```
OK, LIST_QUOTA TEST>STATS

Maximum records allowed on "TEST>STATS" = 500
Total records used = 425.
Records used in this directory = 28.
```

The output shows that the maximum quota for the directory STATS is 500 records. The directory tree is currently using 425 records, while the entries in STATS itself use 28 records.

**Note**

In a quota directory "Total records used" must always be less than "Maximum records allowed".

If no quota had been set on the directory STATS, PRIMOS would give you the following response to the LIST_QUOTA command:

```
LIST_QUOTA TEST>STATS

"TEST>STATS" is not a quota directory.
Total records used = 425.
Records used in this directory = 28.
```

Although the directory has no quota, PRIMOS prints information on the records used by both the subtree and by the single directory.

The –BRIEF option allows you to request a brief form of output, which supplies summary information on one line, as illustrated in the following example:

```
OK, LIST_QUOTA ABRAM>SARAH -BRIEF
Max:        0, Used:      101, Records:        71, ABRAM>SARAH
```

In this example, there is no quota on the directory. This is indicated by the figure 0 for maximum records allowed. The number 101 shows the total number of records used by the directory and its entire subtree, and the number 71 shows the number of records used by the directory. The last entry on the line shows the name of the directory for which information is listed. The directory name is only displayed if you specify it when you give the command.

For more information on quotas, see the **Prime User's Guide.**

## ▶ LIST_REMOTE_ID [-ON nodename]

The LIST_REMOTE_ID command displays any remote id's you have set up with ADD_REMOTE_ID commands. The output will show only the user and project ids, with nodenames. It will never show passwords. For example, suppose you have set up three remote ids using the ADD_REMOTE_ID command. If you then give the LIST_REMOTE_ID command, you see the following display:

```
OK, LIST_REMOTE_ID
System  User id                           Project id
SYSG    GUEST                             TOWEL
SYSK    KEN                               CHURCH
SYSM    MARK.K
OK,
```

If you specify the –ON option with a nodename, only the id associated with the specified system is listed. If you do not specify the option on the command line, all remote ids (a maximum of 16) are listed with the system where they are used.

*See also:* **ADD_REMOTE_ID**

## ▶ LIST_VAR [variable-name [...variable-name]]

The LIST_VAR command without arguments lists all global variables contained in an active global variable file.

If the LIST_VAR command is given variable names as arguments, it lists only those variables. The arguments may be expressed as wildcard names. In this case, those variables whose names match the wildcards will be listed. For example:

```
OK, list_var
 .AWAY                         BEECH>BRANCH2>TWIG4
 .HOME                         BEECH>BRANCH5>TWIG3
OK, list_var .away
 .AWAY                         BEECH>BRANCH2>TWIG4
```

## ▶ LOAD

This command loads and starts LOAD, Prime's Linking Loader for PRIMOS. For a complete discussion of the loaders and an example of the use of LOAD, refer to the **LOAD and SEG Reference Guide**.

LOAD loads programs for R-mode code generated by PMA, FORTRAN, or RPGII. To load segmented code, use the command SEG.

## ▶ LOGIN [user-id [password] [–ON nodename]] [–PROJECT project-id]

You use the LOGIN command to start work on the system. Every time you want to begin an operating session at a **terminal**, you must give this command. After you log in, PRIMOS connects you to your origin directory. (Your System Administrator decides which directory this is )

Your **user-id** identifies you to your local system. (It may also identify you to other systems which recognize ids defined on your system.) You can only specify an existing user id. This id need not be the same as the name of your origin directory. If you do not have a user id, see your System Administrator.

The **id** is a name up to 32 characters long, which must begin with an uppercase letter. The characters that follow may be any combination of uppercase letters, numbers, and the special characters period (.), underscore (_), and dollar sign ($). PRIMOS converts lowercase letters to uppercase.

A password may be required for you to log in on your particular system. If your system allows, you may include the password on the same line as the LOGIN command. Otherwise, you will be prompted to enter it. In this case, your terminal does not echo the password.

You are the only person who knows your password, if you have one. A **password** is a string of up to 16 characters, which may contain any ASCII characters except PRIMOS reserved characters. (See the **Prime User's Guide** for a list of these characters.)

The options for the LOGIN command are as follows:

| **Option** | **Meaning** |
|---|---|
| **–ON system-name** | Specifies a remote system to which you wish to log in. The **system name** is usually the PRIMENET nodename of that system. |
| **–PROJECT project-id** | Specifies a project id, which may be required for you to log in on your system. This is determined by your System Administrator. The project ids must conform to the same naming rules as user ids. |

You cannot log in without giving a user id. However, you can give the LOGIN command without specifying any other information on the same line. If you do this, PRIMOS will prompt you to enter your user id, and then your password and project id, if these are required on your system. (LOGIN will not prompt for your project id if your System

Administrator has defined a default project for you.) A user with id CLAM might log in as follows:

```
OK, LOGIN
User-id? CLAM
Password? FISH
Project-id? FARM

CLAM (user 27)  logged in Wednesday, 31 Mar 82 10:47:04
Welcome to PRIMOS version 19.0
Last login Tuesday, 30 Mar 82 11:49:32
```

You may be able to specify all this information on one line, without waiting for system prompts. If you are using a login password or the –ON option on the command line, you must type in your user id as well. The user CAROLINA might log in on one line, as follows:

```
LOGIN CAROLINA REDWING -PROJECT SAIL

CAROLINA  (user 16)  logged in Sunday, 06 Jun 82 09:06:16
Welcome to PRIMOS version 19.0
Last login Thursday, 03 Jun 82 08:05:32
```

In this example, CAROLINA gives her password REDWING. She uses the project id SAIL.

If your system is linked by PRIMENET to a remote system, you may be able to log in on that system. To do this, you need to know:

1. The name of an existing id on that system.

2. The PRIMENET nodename of the system.

3. Any passwords that may be required.

The following example illustrates a remote login by user HARRY onto a system with nodename HQA.

```
LOGIN HARRY -ON HQA
PRIMENET 19.0 HQA
Password? TUDOR
Welcome to PRIMOS version 19.0
Last login Monday, 29 Mar 82 10:50:48

Enter validation code: ARAGON
OK,
```

In this example, HARRY is prompted for a login password. He must also give a validation code before he can work on the remote system. These requirements vary depending on the needs of your particular site.

If you leave out the –ON option and give a user id that is valid on your local system, you are logged into the local system. The same thing happens if you use the –ON option with the nodename of your local system, specifying a valid user id. If you specify the wrong nodename, or the system you try to access does not have a nodename, the PRIMENET connection will be broken.

### Login Errors

If your attempt at login is not successful, PRIMOS often gives you a message explaining why you could not log in. If your attempts repeatedly fail, make a note of the message and see your System Administrator.

## ▶ LOGOUT [–uu]

LOGOUT is the last command the user issues when giving up access to the system.

During LOGOUT, all user files are closed, all devices assigned to the user's terminal are released, the UFD is detached, and a logout message is printed at the user's terminal and at the supervisor terminal. All segments that were used by the user are returned to the supervisor when the user logs out.

If a user number is specified (–**uu**), that user (which may be a phantom or a process on another terminal) will be logged out, provided that user –**uu** has the same login name as the user giving the command. (At the system terminal, the user specified by –**uu** will be logged out, *regardless* of login name.)

Example:

```
OK, LO
```

A typical response at the user terminal and also at the supervisor terminal might be as follows:

```
CLAM (user 73) logged out Wednesday, 12 May 82 11:47:52.
Time used: 00h 01m connect, 00m 02s CPU, 00m 01s I/O.
```

Every logout message includes information on 'Time used', which shows the user's connect time in hours and minutes, CPU time in minutes and seconds, and I/O time in minutes and seconds. For more details about time accounting, see the TIME command.

A user may be logged out because there has been no activity at that user's terminal for longer than the maximum inactive time allowed. If this happens, a message including the normal logout message is printed at both user and supervisor terminals. The normal message is preceded by the message:

*****From PRIMOS: maximum inactive time limit exceeded.**

If the user is logged out by a command issued at the supervisor terminal, the message "FORCED LOGOUT" is printed at both the user terminal and the supervisor terminal, followed by the normal logout message.

▶ **LOGPRT [destination] [option(s)]** *Operator command*

LOGPRT writes the contents of either LOGREC (the system event log) or NETREC (the network event log) into a disk file or displays them at a terminal. For a detailed discussion of event logging, see the **System Operator's Guide**.

▶ **LON** $\begin{bmatrix} \textbf{-ON} \\ \textbf{-OFF} \end{bmatrix}$

You use the LON command to control phantom logout notification. When you initiate a phantom process, it is logged in under your user id. When the phantom is finished, PRIMOS automatically logs it out, and notifies you of the logout, with a message at your terminal, such as:

```
Phantom 99: Normal logout at 10:33
Time used: 00h 02m connect, 00m 07s CPU, 00m 10s I/O
```

If you prefer not to receive logout notification messages while you are doing something else at the terminal, you can use the LON command to defer the messages. To do this, you use the command in the following form:

```
LON -OFF
```

After you have given the LON –OFF command, PRIMOS stores notification messages instead of displaying them at your terminal.

When you want to see logout notification messages at your terminal, you use the –ON option, as follows:

```
LON -ON
```

After you have given the LON –ON command, PRIMOS displays any logout notification messages that have been deferred by a LON –OFF command. For example, suppose you gave the LON −OFF command and then initiated three phantoms. PRIMOS tells you the phantoms have been assigned user numbers, as follows:

```
PHANTOM is user 110
PHANTOM is user 115
PHANTOM is user 107
```

Later on you may give a LON −ON command, which will display the logout notification messages from these phantoms, as in the following example:

```
OK, lon -on

Phantom 110: Normal logout at 11:40
Time used: 00h 00m connect, 00m 00s CPU, 00m 00s I/O.

Phantom 107: Abnormal logout at 11:40
Time used: 00h 00m connect, 00m 03s CPU, 00m 01s I/O.

Phantom 115: Normal logout at 11:40
Time used: 00h 00m connect, 00m 04s CPU, 00m 01s I/O.
```

The default option is −ON. Logout notification is sent to your terminal unless you specifically prevent it by using the −OFF option.

Phantoms and phantom logout are discussed in more detail in the **Prime User's Guide**.

▶ **LOOK [−uu] [segno] [access] [mapseg]** *Operator command*

LOOK is an operator command that provides access to any segment in the system. This command is issued at the supervisor terminal, and it must be preceded by an OPRPRI 1 command. LOOK is discussed in the **System Administrator's Guide**.

▶ **MAGNET [−SILENT] [−USER]**
**[−OPERATOR]**

You use MAGNET to read, write, and copy magnetic tapes in various interchange formats. Using MAGNET, you can read files from disk or tape, write files to disk or tape, and create output spool files. The **Magnetic Tape User's Guide** explains how to use MAGNET, and also discusses tape concepts, including formats and labelling. The following are some of the features of MAGNET:

- Object declaration and modification
- Character translation and seven-track binary packing
- User-definable character sets
- Tape positioning and copying
- Logical data transfer between devices

- Support of unlabelled tapes
- Support of ANSI and IBM labelled tapes
- Support of fixed- and variable-length records

When you give the MAGNET command, you may use the following options:

| Option | Meaning |
|---|---|
| **–SILENT** | Prints only severity 2 and 3 errors. |
| **–USER** | Displays mount or dismount messages at your terminal. |
| **–OPERATOR** | Displays mount or dismount messages at the supervisor terminal. |

Some features of the interactive, pre-Rev. 19 MAGNET are still supported. The **Magnetic Tape User's Guide** provides full details of these features.

**Note**

MAGNET is not an archiving or system backup facility. For archiving, use MAGSAV or MAGRST. For system backup, use either the MAGSAV and MAGRST or PHYSAV and PHYRST commands.

For detailed information on MAGNET, MAGSAV, MAGRST, PHYSAV, and PHYRST, see the **Magnetic Tape User's Guide**.

## ▶ MAGRST [–7TRK] [–TTY]

MAGRST restores information from a magnetic tape created by MAGSAV into the PRIMOS system. When invoked, MAGRST responds with a series of questions. These questions and appropriate user replies are discussed after the paragraphs that describe general information and the –7TRK and –TTY options.

**General Information:** MAGSAV and MAGRST are utility programs that move files on any disk including the storage module, to a 7- or 9-track magnetic tape and vice versa. The files may be SAM, DAM, segment directories, UFDs, or an entire disk. Whenever a directory is specified, the directory and all components (the subtree) are transferred.

**Logical Tapes:** A logical tape consists of a header record, a file mark, file records, and two file marks. A logical tape may span multiple physical tapes, or a single physical tape may contain multiple logical tapes. The header record contains the tape name, date, and revision number.

**Pathnames:** A disk file appears on tape as a record containing a pathname, followed by as many data records as are required for the file. The pathname contains the unique path to the file specified by the user relative to the current logical volume and/or directory. When an entire disk is saved, all pathnames begin in the MFD. For example, a file might have a pathname of MFD>UFD>JUNK or MFD>UFD>SUBUFD>JUNK. MAGRST's $A (ATTACH) command accepts *only* filenames; it does not accept pathnames Therefore, it is simplest to attach to the UFD into which you wish to restore the tape before invoking MAGRST.

**Note**

An attempt to restore a file called MFD while attached to an MFD results in the message:

```
FILETYPE MISMATCH, FILE OMITTED: MFD
```

The file is not restored, therefore.

**Use of MAGRST:** If the option –7TRK is specified, it indicates to MAGRST to use the 7-track tape format. The default value is 9-track. All restore operations take place in the home UFD. MAGRST asks for the tape unit and logical tape number. If the –TTY option is specified, MAGRST takes the unit number from the terminal, but takes all its other information from its current input stream. (This might be a command file or a CPL file.) The tape name, the date, and the revision are printed on the user terminal. Then MAGRST asks:

```
Ready to Restore:
```

The responses are as follows:

| Response | Meaning |
|---|---|
| YES | Restore the entire tape. |
| NO | Don't restore; request a different tape unit and logical tape. |
| PARTIAL | Permits a restore of part of the tape. |
| NW [filename] [level] | Creates an index of the magnetic tape but does not write the tape to disk. **level** indicates the level of the index (e.g., 3). If **level** is not specified, the default level is 100. **filename** is the name of a file that will contain the index. If **filename** is not specified, the index is printed (or displayed) at the terminal. |

| | |
|---|---|
| $I **[filename] [level]** | Causes an index to be printed when a YES (entire) or PARTIAL restore is done. If **level** is not specified, the default level is 2. |
| $A **directory [passwd] [ldisk] key** | Attach to a directory. **ldisk** must be specified if it is not 0. (The STATUS DISK command tells what the ldisk numbers are.) To attach a subdirectory, first attach the top directory, then use a key of 2 to attach downward. For example: `$A UFD` `$A SUBUFD 6 2` Attaching with a pathname is not supported. The syntax of $A is described in greater detail in the **Magnetic Tape User's Guide**. |

MAGRST provides the ability to enter multiple pathnames for a partial restore. For example, pathnames might be input in response to the query TREE NAME as follows:

```
TREE NAME:MFD>LIB>FTNLIB
TREE NAME:MFD>LIB>JUNK
TREE NAME:MFD>CMDNC0>PRINT
TREE NAME:
```

After each file is restored, the message:

```
FILE COMPLETE
```

is printed at the terminal, if the $I command was specified. For a partial restore, files that have bad records are omitted. The pathnames of these files are printed along with an error message. The message:

```
*** RESTORE COMPLETE ***
```

is printed when the end of logical tape is reached.

MAGRST checks for conflicting file types when a file is going to be overwritten. Conflicts generate an error message, and the file is skipped.

It is not necessary for MAGRST to read through all logical tapes when restoring sequential logical tapes. After MAGRST has exited to PRIMOS, the magnetic tape is not rewound. Instead, it is positioned at the location before the beginning of the next logical tape in sequence. In the case of sequential logical tapes, the user must run MAGRST again and specify 0 for logical tape number where LOGICAL TAPE NO: is requested. Then, the next logical tape is restored without rewinding and reading through the preceding logical tapes.

**Index:** MAGRST allows a user to index a tape and direct the listing of the index to a disk file rather than the user terminal. To use this feature, follow the NW or the I command with a filename, and then with the number of index levels.

Example of index with MAGRST:

```
        Ready to Restore: nw tape#1  5
```

In this case, MAGRST writes an index to level 5 into the file specified by TAPE#1.

**Physical End of Tape:** When physical END OF TAPE is encountered in either MAGSAV or MAGRST, a message is logged on the user terminal and a new tape unit is requested. The new unit may be the same as the old unit.

**Errors:** Tape read or write errors are retried and tested for various conditions until considered unrecoverable. The first record on a tape is not retried. Both recovered and unrecovered errors are logged. If recoverable tape I/O errors occur, a total of the errors is printed at the end of the logical or physical tape (e.g., 5 RECOVERED MT I/O ERRORS). For further discussion of error messages, see the **Magnetic Tape User's Guide**.

**Assigning Tapes:** When running MAGRST under PRIMOS, the magnetic tape drive must be assigned. Refer to the description of the ASSIGN command for further details. Users must avoid restoring files into UFDs that are in use by other users because this action could either confuse the other users or could cause either MAGRST or the user program to abort with the message:

```
        FILE IN USE
```

In this case, the abort is caused by two different programs attempting to gain access to the same file at the same time.

The following example shows a MAGRST of the tape made by MAGSAV in the example in the following section:

```
OK, magrst
[MAGRST Rev. 19.0]
You are not attached to an MFD.
Tape unit (9 Trk): 0
Enter logical tape number: 2
Name: stevex
Date(MM DD YY): 06-12-82
Rev no:      0
Reel no:      1
Ready to Restore: nw 1
*** Starting Index ***
FILE1 (dam)
FILE5 (dam)
FILE7 (sam)
FILE17 (sam)
*** End Logical tape ***
*** Index Complete ***
OK,
```

▶ **MAGSAV [options]**

MAGSAV writes information from a disk configured to PRIMOS to a 7- or 9-track magnetic tape. When invoked, MAGSAV responds with a series of questions. These questions and appropriate user replies are discussed after the paragraphs that describe the options that may be specified with the MAGSAV commands. Optionally, the user may specify: 1024-word or variable-length tape records, use of 7-track magnetic tape, keeping track of date-time stamped files and directories, and incremental dumping of a disk to tape. The value of options are:

| Option | Meaning |
|---|---|
| **–LONG** | Use a 1024-word record size. |
| **–VAR** | Allows variable-length records, up to 2048 words; overrides –LONG option. This record length is the default, and improves the speed of MAGSAV operation. If this option is selected, MAGSAV prints the record size after the REV message in the MAGSAV dialog. |
| **–P300** | Use 512-word records. |
| **–7TRK** | Use 7-track magnetic tape format. The default is 9-track magnetic tape. |
| **–UPDT** | Indicates update; i.e., the DUMPED switch in the UFD entry will be set for files and directories that are saved from disk onto tape. The default is not to set the DUMPED switch. |

| | |
|---|---|
| **–INC** | Indicates incremental dump; that is, only files and directories with the DUMPED switch set to 0 will be saved. The default action is to save all files and directories. |
| **–SUFD** | Specifies that MAGSAV is always to save directories, whether or not they have been modified. –SUFD is useful only with the –INC option. |
| **–TTY** | MAGSAV takes tape unit number from terminal, all other information from current input stream. |
| **–NO_ACL** | Specifies that MAGSAV is not to save any ACLs or ACL references. Tapes saved with the –NO_ACL option can be restored by Rev. 18 MAGRST onto a Rev. 18 system. If this option is not used, ACL information is saved as explained below. |

## How MAGSAV Handles ACLs

A file or directory using ACL protection can be protected in one of the following ways:

- It can be protected by a specific ACL.
- It can be protected by an access category.
- It can use the default protection of the directory in which it resides.

**Specific ACLs:** Specific ACLs are always saved (unless the user gives the –NO_ACL option). Specific ACLs are saved immediately *after* the objects they protect.

**Access Category:** If a UFD is saved, all access categories within the UFD are saved. They are written onto the tape before any other files or directories. Each object protected by an access category is then followed on the tape by the information that it is so protected.

If individual files are being saved by name, then any desired access categories must also be saved by name. They are not saved automatically, as specific ACLs are.

**Default Protection:** No ACL information is saved for file system objects that use default protection.

## MAGSAV Dialog

MAGSAV requests information in the following order:

| | |
|---|---|
| **TAPE UNIT:** | The proper response is the physical unit number of the tape (0-7). |
| **ENTER LOGICAL TAPE NUMBER:** | The response is 1 for the first logical tape, 2 for the second, etc. MAGSAV rewinds the tape, then positions itself correctly. A response of 0 implies the tape is already positioned correctly and MAGSAV takes no action. |

| | |
|---|---|
| **TAPE NAME:** | Any six-character name. |
| **DATE:** | The response format is **mmbddbyy** where **b** represents a space and **mm** = month, **dd** = day, and **yy** = year. The date is checked for validity and rejected if it is not valid. For example, 07 35 03 would be rejected. If not specified, the default is current date. (Carriage return causes system date to be used.) |
| **REV. NO:** | An arbitrary number (usually 0). |
| **NAME OR COMMAND:** | NAME asks the user what to save. The response is either a treename or one of the alternate action commands. These alternate action commands are: |

| | |
|---|---|
| $A **dirname [passwd] [disk] [key]** | Attach to the UFD specified by **dirname** on the disk **ldisk**. (Use **passwd** if required.) To specify a sub-UFD, attach first to its UFD, and then attach to the sub-UFD, giving the ldisk number and the key "2". $A does not accept pathnames. |
| **$Q** | Terminate a logical tape, and return to PRIMOS. |
| **$R** | Terminate a logical tape, rewind, return to PRIMOS. |
| **$I [filename] [level]** | Cause an index to be printed. If **filename** is specified, the index is written into the file specified by **filename**. If **level** is specified, then index is to that level. For example: $I 3 prints an index of the MFD, and UFDs and any filenames.<br>Defaults: if no **level** is specified, then two levels are printed.<br>If **filename** is not specified, then the index is printed on the terminal. |
| * | Save current directory. |
| **$UPDT** $\left\{ \begin{matrix} \text{ON} \\ \text{OFF} \end{matrix} \right\}$ | **ON** indicates to set the DUMPED switch of each file magsaved. The default is **OFF**. |
| **$INC** $\left\{ \begin{matrix} \textbf{ON} \\ \textbf{OFF} \end{matrix} \right\}$ | If **ON** is specified, MAGSAV only those files and directories having a set DUMPED switch. Default is **OFF**. |

| | |
|---|---|
| **MFD** | Save entire volume. |

Example:

```
OK, as mt0
Device MT0 assigned.
OK, magsav
[MAGSAV Rev. 19.0]
Tape unit (9 Trk): 0
Enter logical tape number: 2
Tape name: s
Date (MM DD YY):
Rev no:
Name or Command: file1
Name or Command: file5
Name or Command: file7
Name or Command: file17
Name or Command: $R
OK,
```

The MAGSAV rewind command, $R, causes the tape to be rewound and exits to command level.

To save an entire disk, the user must attach to the MFD and respond to the query NAME OR COMMAND: with the name MFD. To save a UFD, the user must attach ($A) to the MFD and give the name of the UFD that is to be saved. To save a file in the UFD the user must attach to the UFD (e.g., $A dirname) and give the name of the file. MAGSAV also saves a disk that contains nested segment directories.

MAGSAV can handle up to 13 levels of directories and subdirectories. Disks or UFDs containing more than 13 levels must be saved in a series of steps.

Except under PRIMOS II, the magnetic tape must first be assigned using the ASSIGN command. Files or directories that are in simultaneous use by other users must not be accessed by MAGSAV, as MAGSAV will ignore such files and so will not write them onto tape.

**Error Recovery:** A separate error message for each recoverable error is not printed. If recoverable errors occur on the magnetic tape when writing tape, the total number of such errors is printed when the end of tape is reached.

▶ **MAKE [-OLD] [-LOWEND] [-AUTO]** *Operator command*

MAKE formats disks (and partitions of multisurface disks) for use by PRIMOS. MAKE creates a PRIMOS disk that has the following:

- Disk pack or partition name (packname) as specified by user
- MFD (Master File Directory)
- BOOT (Bootstrap, in Record 0)
- BADSPT (only if badspots are present on disk)
- DOS (which is empty)
- CMDNC0 (which is empty)

The −OLD option creates a disk in pre-Rev. 19 format, for use on machines running Rev. 18 (or earlier) PRIMOS.

See the **System Operator's Guide** for details.

▶ **MAXSCH n** *Operator command*

The MAXSCH command is used to set the variable MAXSCH in the system data base named SUPCOM, which controls the circumstances in which the backstop process (part of the scheduler) adds processes to the ready list. The default value of MAXSCH is 3.

▶ **MAXUSR n** *Operator command*

MAXUSR is a PRIMOS III, or PRIMOS operator command that controls and limits the number of users logged in to the number specified by **n**.

▶ **MCLUP** *Operator command*

MCLUP invokes the MIDAS Cleanup Utility. For details, see the **MIDAS User's Guide.**

## MEDUSA

MEDUSA™ is Prime's Computer Aided Design (CAD) system. It is a separately priced product. MEDUSA is not a command. To invoke MEDUSA, you execute a command file to initialize your workstation for use with the MEDUSA software. For information on MEDUSA, see the **MEDUSA 2-D User's Guide.**

▶ **MESSAGE** $\left[\begin{matrix}\textbf{username}\\ \textbf{-usernumber}\end{matrix}\right]$ **[-NOW] [-ON nodename]**
**text of message**

The MESSAGE command is used to send or receive messages. Either users or the operator may send messages. Messages may be sent:

- From any user terminal to any user terminal.
- From any user terminal to the supervisor terminal.
- From the supervisor terminal to all users.
- From the supervisor terminal to a specified user.
- From the supervisor terminal to another supervisor terminal on a different node on the network.

## Sending Messages

**username** is the user id with which the user logged in. **usernumber** is the number of a specific terminal.

To find out what the user numbers are for the various terminals, issue the STATUS USER command. A list of users, their user numbers, line numbers, and physical device numbers will be printed.

If you send a message to **username**, all users logged into that name receive the message.

If you send a message to **usernumber**, only the specific terminal with that number receives the message.

If you omit this argument, the message is sent to the operator on your system.

**text of message** is a single line to be sent. Sending a message produces two lines of information on the receiver's terminal. The top line contains information about the sender: the second contains the text of the message. The format is:

```
***uu hh:mm
   text of message
```

where **uu** is the user name and user number and **hh:mm** is the time of day in hours and minutes. For example:

```
        ***BEECH (55) 11:16
```

If the –NOW option is specified, the message is printed immediately on the receiver's terminal.

If the –NOW option is not specified, the first message sent to the receiver is stored in a buffer and printed when the receiver returns to PRIMOS command level.

### Sending Messages to Remote Sites

If your local system is attached to a network, you may be able to send messages to a user at a remote site, using the –ON option. To do this, you need to know the PRIMENET nodename of the remote system to which you are sending the message, as well as a user id or user number for the person you want to receive the message.

For example, to send a message to a user with id QUEEN on a system with PRIMENET nodename SYS.Q, you might give the MESSAGE command in the following form:

```
        MESSAGE QUEEN -NOW -ON SYS.Q
```

### Setting Receive States

Users may set the receive state of their terminal with the MESSAGE command. One of three different states may be selected to control the flow of messages:

| **Option** | **Meaning** |
|---|---|
| MESSAGE –ACCEPT | Enables reception of all messages. |
| MESSAGE –DEFER | Inhibits immediate messages. |
| MESSAGE –REJECT | Inhibits all messages. |

Setting a receive state to defer or reject messages is useful when you do not want messages to interrupt a terminal session. For example, this can be critical in situations where you are printing the contents of a file. Deferring or rejecting messages in this instance would prevent the message from being printed along with your file's contents.

Sending a message while in MESSAGE –REJECT mode or sending an immediate message while in MESSAGE –DEFER mode is not permitted because the receiver will not be able to respond.

## Querying Receive States

You may determine what a user's terminal receive state has been set to with the –STATUS option of the MESSAGE command. Issuing the command MESSAGE –STATUS lists users' login names, terminal numbers, and receive states.

| Option | Meaning |
|---|---|
| **MESSAGE –STATUS** | Lists the receive state of all users. |
| **MESSAGE –STATUS username** | Lists the receive state of all users with the name **username.** |
| **MESSAGE –STATUS usernumber** | Lists the receive state of the terminal with the number **usernumber.** |
| **MESSAGE –STATUS ME** | Lists the receive state of your own terminal. |

## Error Messages

The following are possible error messages:

**Improper command usage or arguments. (MESSAGE)**

This message may mean:

1. The MESSAGE command contains a typing error.
2. The MESSAGE command was addressed to a user number which does not represent a logged in user.
3. The MESSAGE command contains an illegal or unknown option.

***Unknown addressee.**

The MESSAGE command specified the user id of a user who is not logged in.

***Usernumber not receiving now. (SMSG$)**

This message may mean one of two things:

1. If you are trying to send an immediate message (M –NOW), it means that the recipient's receive state is either DEFER or REJECT.
2. If you are sending a message without the –NOW option, this warning means that the recipient's receive state is REJECT.

*****Unknown node.**

You have specified a system name unknown to your network.

*****User usernumber busy, please wait.**

The user already has a deferred message waiting. Only one deferred message is allowed.

*****Requires –ACCEPT enabled.**

Sender must issue MESSAGE –ACCEPT before sending message.

*****Requires –ACCEPT or –DEFER enable.**

Sender must issue MESSAGE –ACCEPT or MESSAGE –DEFER before sending message.

## ▶ MPACK

MPACK is a MIDAS utility for packing and restructuring MIDAS files. It recovers space occupied by data records and index entries marked for deletion. It also unlocks locked records and restructures the data subfile. For details, see the **MIDAS User's Guide.**

## ▶ MRGF file-a file-b [file-c...file-e] –OUTF ofiie

```
[ -BRIEF           ]
[ -MINL number     ]
[ -FORCE           ]
[ -REPORT pathname ]
```

The MRGF command allows a user to merge ASCII files. Two to five files may be merged. The files may be specified by **pathname.** One file, **file-a**, is treated as the original file, and it is assumed that changes have been made in this file to produce the other files, **file-b** through **file-e**.

Unchanged lines of text and unconflicting changes between files are copied automatically into the output file, **ofile.** When corresponding lines of text in the specified files differ, the user is asked by the MRGF program to resolve the conflicts. The user's response to these queries determines the manner in which the conflicts arising out of the differences are resolved.

**Purpose of MRGF:** The MRGF command, along with the CMPF command, helps ease the problems of parallel software development. The MRGF command allows automated merging of program changes, and obviates the need for tedious editing of programs when two (or more) sets of changes made to a program are to be combined. It is anticipated, however, that the resulting merged output will be checked carefully before use.

MRGF is especially useful for combining changes to a program that have been made in parallel by several programmers. It can also be useful for distributing software changes to one or more sites, or one or more persons.

| Parameters | Meaning |
|---|---|
| file-a | The pathname of the original file. **file-a** is treated as the original file by MRGF (i.e., the file that is the common ancestor of **file-b** through **file-e**). |
| **file-b, file-c file-d, file-e** | Pathnames of files that trace their ancestry to **file-a**. |
| **ofile** | The pathname of the merged output file generated by MRGF. It must appear immediately after –OUTF. |
| **–BRIEF** | An optional control argument that suppresses the printing (or display) at the terminal of differing lines of text within the specified files. (Only the file identification letters and line numbers are printed.) |
| **–MINL number** | Sets the minimum number of lines that must match, after a difference in the files being merged, in order to resynchronize all file merging. The default value is –MINL 3. |
| **–FORCE** | Causes **file-b** to be the preferred file if conflicts exist between several files. When –FORCE is used, the user is never asked by MRGF to resolve a conflict. (See the following paragraphs for a discussion of resolving conflicts.) |
| **–OUTF** | Immediately precedes **ofile**, the output file. –OUTF is mandatory. |
| **–REPORT pathname** | Produces a file, named **pathname**, that contains the differences (in lines of text) between files encountered while merging. Resolvable differences are not displayed or printed at the terminal. User-resolvable differences (conflicts) are written into the report file, **pathname**, and they are also displayed or printed at the terminal. |

**MRGF Usage: file-a** is treated as an original file (i.e., as a file that is the common ancestor of **file-b** through **file-e**). **file-a** is compared line by line with each of the other files. Lines that match in all files are copied into **ofile**. When differences are found between specified files, MRGF attempts to get all files back in synchronization. Rematching is completed only when a certain minimum number of lines match in all files. This minimum number may be set with the –MINL control argument.

After resynchronization is complete, selection of lines to be output must take place. If only one file differed from **file-a**, the changes in that file are copied into **ofile**. If all files differed identically from the original, those changes are also copied. If conflicting changes are found in several files (or if only one file is being merged with the original), the user can select manually the lines that are to be copied into **ofile.** If the –FORCE control argument is used, the user is not queried; and the changes in **file-b** are considered the preferred changes to be inserted into **ofile.**

If the –FORCE control argument is not used, the differing lines from each of the files are reported. Each line from **file-a** is identified by preceding it with the letter A and the line number of that line. Lines of **file-b** through **file-e** are similarly identified, using the letters B through E, respectively. The –BRIEF control argument causes only the identification letter and the line numbers of the differing lines to be printed. After an unresolvable discrepancy is reported, EDIT mode is entered to allow the user to select lines to be placed in **ofile** (refer to the following paragraphs). After selection (either automatic or manual) is completed, the line-by-line comparison continues.

If the –REPORT control argument is used, the resultant report file contains all discrepancies between files (i.e., both the resolvable and the unresolvable differences). Unresolvable differences are always displayed on the user's terminal as well. Resolvable differences, however, are never displayed on the user's terminal. The action taken by MRGF (or the user) is placed in the report file following each discrepancy.

**Manual Selection of Changes:** After each unresolvable difference is displayed, EDIT mode is entered. The user must select which lines are to be inserted into **ofile** by issuing the following commands:

| | |
|---|---|
| **A** | Insert all of the differing lines in **file-a.** |
| **B** | Insert all of the differing lines in **file-b.** |
| **C** | Insert all of the differing lines in **file-c.** |
| **D** | Insert all of the differing lines in **file-d.** |
| **E** | Insert all of the differing lines in **file-e.** |
| **An** | Insert line **n** of **file-a.** |
| **Bn** | Insert line **n** of **file-b** (similarly for **file-c** through **file-e**). |
| **Am,n** | Insert lines **m** through **n** of **file-a** (similarly for **file-b** through **file-e**). |
| **PA** | Print all of the differing lines in **file-a** (similarly for **file-b** through **file-e**). |
| **PAn** | Print line **n** of **file-a** (similarly for **file-b** through **file-e**). |
| **PAm,n** | Print lines **m** through **n** of **file-a** (similarly for **file-b** through **file-e**). |
| **OOPS** | Undo all previous editing for this discrepancy. |
| **GO** | Terminate editing and proceed with merge. |
| **QUIT** | Terminate editing, close all files, and exit from MRGF. |

In addition to the above commands, new text can be inserted at any point in a difference resolution by entering a blank line. INPUT mode is entered, and lines typed are copied into **ofile.** A blank line will terminate input. No text editing can be performed on lines that are copied or entered in INPUT mode No tab character expansion is performed on lines entered in INPUT mode.

**Line Length:** The MRGF command operates on compressed lines of any length. It assumes that files of common ancestry contain lines compressed in identical fashion. It is, however, possible for a mismatch to occur between two lines that appear identical, but were compressed differently.

For example, consider the following three files:

```
The                The                The
quick              quick              quick
brown              red                brown
fox                fox                fox
jumps              jumps              jumps
over               over               over
the                the                the
lazy               sleeping           snoring
dog                dog                dog
```

A MRGF of these files would produce the following:

```
OK, MRGF FILEA FILEB FILEC -OUTF FILEX
[MRGF 19.0]

A8        lazy
CHANGED TO
B8        sleeping
BUT ALSO CHANGED TO
C8        snoring
EDIT.
B
GO

MERGE FINISHED.
1 MANUAL CHANGE.
1 AUTOMATIC CHANGE AS FOLLOWS:
       1 FROM FILE B

OK,
```

In the above example, rust-colored lines were typed by the user. The merged output file from the above MRGF would appear as follows:

```
The
quick
red
fox
jumps
over
the
sleeping
dog
```

If the –FORCE control argument had been used in the example given, the same merged output would have been produced. However, the change from FILEB would have been inserted automatically, and the user would not have been queried.

## ▶ NCOBOL

NCOBOL is a COBOL compiler that generates R-mode code and uses the nonshared COBOL library, NCOBLB. Its options and defaults are the same as those of the V-mode COBOL compiler, COBOL.

## ▶ NET

*Operator command*

The operator uses the NET command to activate, assign, and deactivate half-duplex (HDX) network connections. For details, see the **PRIMENET Guide.**

## ▶ NETCFG

*Operator command*

The System Administrator uses the NETCFG command to establish the configuration of a PRIMENET network. For details, see the **PRIMENET Guide.**

## ▶ NETLINK

You may connect to any system on the Public Data Network by using the NETLINK command. This means that systems other than Prime systems and software other than PRIMENET software may be accessed across geographic boundaries. Other sites or other networks as well as jobs within these other sites and networks may be accessed.

Several NETLINK commands exist to aid users in using other systems and networks. There are basic commands and advanced commands. Basic commands allow you to enter and exit the remote systems. Advanced commands allow you to:

- Transfer files across networks.

- Set data transmission characteristics.
- Print the status of your connection.
- Connect to and use up to four different remote systems at the same time.
- Specify the various fields of the connect packet when data transmission characteristics of a foreign system differ from that of Prime's.

Only NETLINK's basic usage will be presented here. For a list of all commands and error messages see the **PRIMENET Guide.**

**NETLINK Usage**

The basic steps to using NETLINK are as follows:

1. Enter NETLINK Command mode by issuing the NETLINK command. When Command mode is entered, the @ prompt appears.
2. Connect to the remote system by issuing the NC address or C address command. **NC** means no reverse charge. This is required for many international calls. **address** is either the host address assigned by the Public Data Network or a PRIMENET system name. For example, 'NODE 1' and '308 67' are both valid addresses.

   International calls must be specified as follows: 'national network identifier: host address.' For example:

   ```
   '2080:12300011'
   ```

   When a connection has been established, the message "address Connected" appears.
3. Log into the system as you would normally, entering any validation codes as required.
4. Once you finish a terminal session, log out as you would normally. The message "address Disconnected" appears. When a connection to a remote host has been terminated by logging out, Command mode is reentered and the @ appears You may now connect to another site or return to PRIMOS.
5. To return to PRIMOS enter the QUIT command

### NETLINK Example

Below is an example of a basic terminal session.

```
OK, netlink
[NETLINK  Rev. 19.0]

@ c hqa

HQA Connected
PRIMENET 19.0 HQA
login HAPPY

PRIMOS Version 19.0
HAPPY (48) LOGGED IN AT  12:54 051282

Enter validation code:  HQCODE
 .
 .
 .
 .            /* continue with normal terminal session
 .
 .
 .
OK, LO
HAPPY (user 48) logged out Wednesday, 12 May 82 11:47:52.
Time used: 00h 03m connect, 00m 12s CPU, 00m 02s I/O.

WAIT...


HQA Disconnected

@ quit

OK,
```

## ▶ NETPRT

The functionality of the NETPRT command has been subsumed to the LOGPRT command. Use the LOGPRT command to read or print the network event log, NETREC. For details, see the **System Operator's Guide.**

## ▶ NSED

NSED is the nonshared version of the text editor. It is identical in function to ED, the shared EDITOR. PRIMOS II, which does not support the shared EDITOR, can use NSED.

## ▶ NUMBER

NUMBER numbers or renumbers statements in a BASIC program. NUMBER asks the user:

```
INTREENAME, OUTTREENAME, START, INCR,
```

The user gives four responses on one line:

| | |
|---|---|
| **intreename** | The pathname of the input file (i.e., the file that contains the BASIC program with the statements to be renumbered). |
| **outtreename** | The name of the output file. If **outtreename** is omitted, output is to the input file |
| **start** | The starting statement number, from 1 to 9999. If **start** is omitted, the value 1 is assumed. |
| **incr** | The statement number increment, from 1 to 9999. If **incr** is omitted, the value 1 is assumed. |

If **incr** is specified, **start** must be specified also.

The maximum line length that NUMBER can handle is 75 characters, plus 5 for the line number. Lines longer than 75 characters are truncated. Only those BASIC programs with their commands in UPPERCASE characters are renumbered correctly.

Examples:

Assume an input file contains the following statements:

```
11 PRINT 'AB'
12 INPUT A
30 PRINT 'H'
35 INPUT H
40 B=H/A
50 PRINT ' ',B
55 IF b<>0 THEN 11
99 END
```

Then, the following sequence of commands:

```
OK, NUMBER
INTREENAME, OUTTREENAME, START, INCR,
FOO FOOBAR 10 5
OK,
```

produces the following output:

```
10 PRINT 'AB'
15 INPUT A
20 PRINT 'H'
25 INPUT H
30 B=H/A
35 PRINT ' .',B
40 IF b<>0 THEN 10
45 END
```

The input file may be only partially numbered. In such a file, statements are numbered in the order of their occurrence.

**Error Messages:** The following messages may be printed, if an error occurs:

| **Message** | **Remarks** |
|---|---|
| BAD PARAMETERS | If either **start** or **incr** are specified with more than four digits, NUMBER requests a new parameter line. |
| BAD SYNTAX | If the value of either **start** or **incr** is invalid, NUMBER requests a new parameter line. |
| XXXXXX NOT FOUND | The specified input file does not exist. Control returns to PRIMOS. |
| XXXX DUP LINE NUMBER | **XXXX** occurs as a line number more than once. Control returns to PRIMOS. |
| INPUT FILE NULL | The specified input file is empty. Control returns to PRIMOS. |
| MEMORY OVERFLOW | There is not enough memory to contain a map of line numbers. Control returns to PRIMOS. |
| LINE NUMBER OVERFLOW | A new line number greater than 9999. Control returns to PRIMOS. |

## ▶ OAS

The OAS command logs you into the Master Function Selection of Prime's Office Automation System (OAS). It allows you access to all Word Processing and Management Communications and Support options.

For detailed information, see the following documents:

- **OAS Word Processing Guide (PT25)**
- **OAS Word Processing Guide (PT45)**

- **OAS Word Processing Guide (PT65)**
- **OAS Management Communications and Support Guide (PT25)**
- **OAS Management Communications and Support Guide (PT45)**
- **OAS Management Communications and Support Guide (PT65)**

## ▶ OA_ADMIN

The OA_ADMIN command logs the System Administrator into Prime's Office Automation System (OAS) Administrator Option Selection. It allows the Administrator to create and maintain Calendar, Country, Network Directory, and other master OAS files. For detailed information, see the **OAS System Administrator's Guide.**

## ▶ OA_TERM

The OA_TERM command downline loads the PT65 Terminal (Administrative Workstation) for Prime's Office Automation System.

For detailed information, see the **OAS Word Processing Guide (PT65)** and the **OAS Management Communications and Support Guide (PT65).**

## ▶ OPEN [pathname] funit key

OPEN opens the file unit specified by **funit** (between 1 and 177 octal), associates it with the specified **pathname**, and assigns a status according to **key.**

The key parameters consist of octal values for the type of file and the action to be taken when the file is opened. The format of **key** is as follows:

New file (file type) key; octal values are:

| | |
|---|---|
| **0000** | File will be sequential file (SAM). |
| **2000** | File will be direct access file (DAM). |
| **4000** | File will be a SAM segment directory. |
| **6000** | File will be a DAM segment directory. |
| **10000** | File will be a UFD (avoid this; use CREATE command instead). |

Action key; octal values are:

| | |
|---|---|
| 1 | Open for reading. |
| 2 | Open for writing. |
| 3 | Open for reading and writing. |
| 4 | Close. |

| | |
|---|---|
| 5 | Delete (avoid this; use DELETE command instead). |
| 6 | Test for existence. |
| 7 | Rewind. |
| 10 | Truncate at current position (no pathname). |

Reference key; values are:

| | |
|---|---|
| 0 | File is entry in current UFD. |
| 100 | File is entry in segment directory open on **funit.** |
| 1000 | Change open mode of **funit.** |

The octal values for file type, action, and reference are logically ORed to form key. For example, the key for opening a DAM file for writing, in the current UFD, is formed by ORing together the values:

| | |
|---|---|
| 2000 | Direct access file (DAM) |
| 0000 | In current UFD |
| 0002 | Open for writing |
| 2002 | The resulting key |

*See also:* **CLOSE**

▶ **OPRPRI** $\left\{ \begin{matrix} 1 \\ 2 \end{matrix} \right\}$ *Operator command*

OPRPRI (operator privilege) allows certain commands to be issued at the supervisor terminal. Under PRIMOS, OPRPRI 1 must precede the SHARE or LOOK commands. The command line OPRPRI 0, resets the protection against these commands.

▶ **ORIGIN**

The ORIGIN command changes your home and current attach point to your Initial Attach Point (that is, the directory to which you are attached when you first log in). This directory is called your **origin directory.** There are no command line arguments or options.

For more information on ORIGIN, see the **Prime User's Guide.**

▶ **OWLDSC [-FAST] [-NOLOCK] [-REPORT]**

The OWLDSC command invokes the OWL interface program, which allows an OWL-1200 terminal to emulate an IBM 3277 Model 2 display station on systems where DPTX/DSC is running. For details, see the **Distributed Processing Terminal Executive Guide.**

## ▶ PASCAL pathname [options]

The PASCAL command loads the PRIME PASCAL compiler and compiles the object program from an ASCII source file named **pathname.** For complete details about the programming language PASCAL and compiler **options**, see the **PASCAL Reference Guide.**

## ▶ PASSWD [owner-password] [nonowner-password]

**Note**

The PASSWD command applies only to directory passwords. It is part of the old password protection system. If you use the PASSWD command on a directory protected by access categories, the password will only be in effect after you give the REVERT_PASSWORD command.

To change or create a login password, use the CHANGE_PASSWORD command, documented earlier in this chapter.

The PASSWD command replaces any existing passwords in the current directory with two new passwords. The first is **owner-password;** the second is **nonowner-password.** The nonowner-password is optional. If it is not specified, the nonowner-password becomes blank. The PASSWD command must be given by the owner while attached to the directory. A nonowner cannot give this command. Passwords may be of any length; but they are matched by the first six characters only.

Passwords may be specified in lower- or uppercase. To specify a lowercase password, type the password in lowercase and enclose the password in single quotes. To specify an uppercase password, type the password in uppercase characters; do not enclose the password in single quotes.

Under PRIMOS II, only the owner password may be given.

Example:

```
OK, A GOUDY OLDPW
OK, PASSWD US THEM
OK,
```

**Caution**

The MAGRST command cannot restore directories with passwords unknown to the user of MAGRST

▶ **PHANTOM command-file** [**funit** | **CPL-arguments**]

A user may initiate a phantom user to perform a job. A phantom user is similar to any other PRIMOS user, but has no terminal associated with it; all controlling input is read from a command-file instead of a user terminal. **command-file** may be a CPL program or a command input file. It may be specified as a pathname or as a file in the user's current directory. The file specifies the sequence of commands and/or user program invocations and necessary input data specifications to complete a given job.

### Running Command Input Files as Phantoms

To run a command input file as a phantom, you must be sure that LOGOUT is the last command in the file, so that the phantom logs out properly. COMINPUT −TTY or COMINPUT −END as the last line may cause an abnormal termination of the phantom. If LOGOUT is not the last line, the phantom will report an abnormal ending when it finishes processing the file.

You can control the file unit on which the command input file is opened, by specifying an octal value for **funit** in the command line.

### Running CPL Programs as Phantoms

In a CPL program run as a phantom, the &RETURN directive at the end of the program is interpreted as a LOGOUT command. You cannot specify the file unit on which a CPL phantom will run.

When running CPL phantoms, you can specify **CPL-arguments** in the command line. These arguments are passed to the program, where they are used by the &ARGS CPL directive.

For further information on CPL programs, see the **CPL User's Guide.**

The phantom user feature is useful for running programs that are not interactive, and therefore do not require the services of a terminal. However, terminal output should be directed to a COMOUTPUT file.

When PRIMOS is started up, a fixed number of phantom users is specified. The line printer spooler, or the card reader spooler, is usually run as a phantom user, thereby releasing a terminal for interactive work.

**Startup of PHANTOM:** The PHANTOM command checks if there is a process available for a phantom user to be logged in. If no free processes are available, the message:

```
No phantoms are available.  FILENAME
```

is printed at the user terminal. If a process is available, the phantom user is logged into the login directory of the user who invoked the phantom. Then, the phantom feature of the operating system attaches to the user's current directory. If the command-file specified in the phantom command is a command input file, it is opened on File Unit 6 (or on the specified **funit**). If the command-file is a CPL file, it is opened on some available unit. PRIMOS takes all further commands from the file specified by **command-file** in accordance with COMINPUT or CPL operation. An example of phantom startup is:

```
OK, ph test.phant
PHANTOM is user 106
OK,
```

It is suggested that the COMOUTPUT command be invoked within the phantom command-file to keep a historical record of the phantom's processing. A process running as a phantom user *must not* perform any terminal input. An attempt to read input from a terminal causes the command file to abort and causes the phantom user to be logged out. If this condition occurs, the logout message at the system terminal is preceded by the line:

```
User 106:  Phantom requested terminal input.
```

An error that causes the command file to abort also causes the phantom user to be logged out.

Any terminal ouput that is generated by the phantom user program or directed to the user terminal by system commands, such as LD, is ignored, unless the COMOUTPUT command is invoked in the command file.

A user may monitor the status of any phantom user by the STATUS command. For each phantom user that is logged into the same login UFD, STATUS prints the UFD name followed by the user number of that phantom in decimal. When the user phantom job is complete, it is logged out, and will no longer appear in the output printed by the STATUS command. If a user wishes to stop a phantom user that was started at his terminal, the command:

**LOGOUT -uu**

must be issued, where **uu** is the number of the phantom user as reported by the PHANTOM command when it was invoked.

A user may log out, return later, and log in to the same UFD. The STATUS and LOGOUT commands may be used as before to control the phantom.

Any phantom or user may be logged out by use of the LOGOUT command at the supervisor terminal.

The PHANTOM command may be issued from a CPL program or a command file. Command files running in phantoms may also include PHANTOM commands (i.e., phantom command files may be chained in a manner similar to COMINPUT command files).

When a phantom logs out, it attempts to send a notification of its logout to the user who started it. A sample message, following an orderly logout, might be:

```
Phantom 106: Normal logout at 10:33
Time used: 00h 02m connect, 00m 07s CPU, 00m 10s I/O
```

The phantom can send this message to the user's terminal only if the user is still logged in on the terminal from which the phantom was started. In other cases, the messages can be recorded by a user program using the subroutines LON$R and LON$CN. (For further information on these subroutines, see the **Subroutines Reference Guide.**)

*See also:* **JOB, LON**

## ▶ PHYRST *Operator command*

The PHYRST command copies partitions to disk as saved by PHYSAV on magnetic tape.

For a complete description of this command, see the **System Operator's Guide.**

## ▶ PHYSAV [-LOWEND] *Operator command*

The PHYSAV command copies the contents of one or more assigned disk partitions to magnetic tape.

For a complete description of this command, see the **System Operator's Guide.**

## ▶ PL1G **pathname [options]**

PL1G is Prime's compiler for subset G of PL/I. It accepts either a filename or a pathname. However, neither the pathname nor any filename created from it may contain more than 32 characters.

At Rev. 17.2 and later PRIMOS, PL1G generates V-mode and I-mode code. Thus, SEG must be used to load the compiled program. Moreover, a new library (PL1GLB) must be loaded before loading the regular library. The commands are:

```
$LI PL1GLB
$LI
```

As distributed, PL1G uses the following default options:

```
        -B YES -L NO -64V -OPTIMIZE -UPCASE
```

The defaults may be changed by using the program TOOLS>PL1GDF. For a listing of all options and their uses, see the **PL/I Subset G Reference Guide.**

## ▶ PM

The PM (Post Mortem) command is used to print or display the contents of the RVEC vector (described in Appendix A). PRIMOS first prints labels for the items in RVEC, then prints the values on the line in the same order.

For the Prime 50 Series, the PM command also displays the procedure base register (PB), the stack base register (SB), the link base register (LB), and the temporary base register (XB). These 32-bit registers are displayed at the user terminal on a text line separate from the other registers. Each of the Prime 350-class registers is displayed as two 16-bit octal numbers separated by a ring number and a slash (/) character.

Example:

```
        OK, PM
        SA,EA,P,A,B,X,K=
        100 6271 16525 0 0 0 14000

        PB,SB,LB,XB:
        2000(3)/16525  4000(3)/112462  4000(0)/3714  0(0)/0
```

The above example of PM under PRIMOS shows a PB of 4000(3)/1106, which indicates: ring 3, segment '4000 octal. The word number portion of PB indicates the same number as the P parameter of PM. This number, which is the same as the P parameter, specifies the location within the segment to execute the next instruction upon possible receipt of a START command.

**Note**

PM will not give an accurate picture of the machine state of a program if the program has been halted by an on-unit that does not allow the static mode overseer to update the PM data. This would occur if a user on-unit returned to command level by calling COMLV$. However, the DMSTK command will always produce an accurate display of the program's state.

▶ **PMA pathname [options]**

PMA loads the Prime Macro Assembler and starts assembly of a source file specified by **pathname.**

| Option | Meaning or Remarks |
|---|---|
| **–I NPUT filename-1** | Specifies the name of the input file. The default is the pathname following the command PMA. |
| **–L ISTING filename-2** | The name of the listing file. The default is filename-1.LIST or L_filename-1. Use –LISTING NO for no listing. |
| **–B INARY filename-3** | The name of object file. The default is filename-1.BIN or B_filename-1. Use –BINARY NO for no object file. |
| **–EXPLIST** | Generates full assembly listing. (See the **Assembly Language Programmer's Guide** for further details.) |
| **–ERRLIST** | Generates errors-only listing. |
| **–RESET** | Resets all A-, B-, and X-Register settings. |
| **–XREFL** | Generates complete concordance. |
| **–XREF S** | Generates partial concordance; only symbols actually referenced are listed. |

For a complete discussion of the assembler, including register settings, see the **Assembly Language Programmer's Guide.**

▶ **POWER**

Invokes the PRIME POWER data management facility. For information, see the **PRIME/POWER Guide.**

▶ **PR ERR**

PRERR prints the message stored in ERRVEC and the first six locations of ERRVEC in octal. The PRERR command is useful in debugging a program. On encountering an error condition, PRIMOS sets up an internal vector called ERRVEC with several pieces of information. One of these pieces is an error message. Refer to the **Subroutines Reference Guide** for a description of ERRVEC.

Using the system subroutine ERRSET (see the **Subroutines Reference Guide**), a user may set the content of the error message and have the message printed or not printed, depending on the alternate return being zero or nonzero, in a user subroutine. If the user routine was the last routine to set ERRVEC, PRERR prints the user-stored message.

Example:

```
OK, PRERR
17 1 0 74 134640 120240
OK,
```

**Note**

The PRERR command will not display useful information following such conditions as access violation faults or illegal segment number faults. In these cases, the needed information will be printed as part of the system diagnostic for the error condition. It can also be obtained by using the DMSTK command.

## ▶ PRIMOS *Operator command*

This command is used solely by the operator or System Administrator to bring up the PRIMOS operating system. For detailed information, see the **System Administrator's Guide** or the **System Operator's Guide.**

## ▶ PRMPC pathname

PRMPC causes the file specified by **pathname** to be printed on an MPC parallel interface printer configured to PRIMOS. The printer (PR0) must be assigned before the PRMPC command is invoked.

## ▶ PROP

The PROP command allows the operator to control the spooler phantoms. It also allows users to discover the names and environments of these phantoms: that is, it allows them to find out how their system's printers and/or plotters behave.

Users can give two forms of the PROP command:

**PROP -STATUS**

which lists the spooler phantoms and tells which ones are running; and

**PROP printer-name -DISPLAY**

which provides information on a phantom's environment. For example:

```
OK, PROP HQAPRO -DISPLAY
[PROP rev 19.0.0]

DEVICE: PRO
PAPER:
DEST:
    HQ1
    HQ2
    HQ3
    PICKUP
MESSAGE:
This is a sample environment.

COMOUT: ON
UPCASE: OFF
PRINT:  ON
PLOT:   OFF
EVFU:   OFF
TYPE:       0
LENGTH:    38
LARGE:     50
LIMIT:    300
UPPER:     63
LOWER:      0
HEADER:     2
WIDTH:    108
LINES:    off
OK,
```

The parameters shown in the screen display are:

| Parameter | Definition |
|---|---|
| DEVICE: | Name of printer phantom. (Can also be used in a SPOOL –AT option.) |
| PAPER: | Type of paper mounted on printer. (Blank signifies default type.) |
| DEST: | Locations where printouts may be delivered by the operator. The synonyms listed are the acceptable synonyms for the –AT option of the SPOOL command. This parameter is optional; it appears only if the System Administrator has established a list of synonyms. |
| MESSAGE: | This message is printed on each header page. |
| COMOUT: | If on, the phantom is keeping a COMOUTPUT file of its activities. |

| | |
|---|---|
| UPCASE: | If on, printer prints all characters as uppercase. If off, prints both upper- and lowercase |
| PRINT: | If on, phantom controls a printer. |
| PLOT: | If on, phantom controls a plotter. |
| EVFU: | When engaged, defines the paper length of special forms on the spooler. |
| TYPE: | Shows printer type (0 indicates a 300-lpm printer/plotter; 1 indicates a band printer). |
| LENGTH: | Specifies the number of lines printed per page. |
| LARGE: | Files of less than this number of records receive priority in the spool queue. |
| LIMIT: | No files containing more than this number of records will be printed by this phantom. |
| UPPER: | Indicates the highest logical disk on which the phantom may search for spool queues. |
| LOWER: | Indicates the logical disk at which the search for spool queues will begin. |
| HEADER: | Specifies the number of header pages to be printed. |
| WIDTH: | Specifies the number of columns per page. |
| LINES: | Shows the number of physical lines per page. |

For a discussion of PROP commands available to the operator, see the **System Operator's Guide**.

## ▶ PROTEC pathname key1 key2

At Rev. 19, the PROTEC command has been replaced by the PROTECT command documented below.

For information on PROTEC, see Appendix C. Old Commands.

## ▶ PROTECT pathname [owner-code] [non-owner-code] [-REPORT]

The PROTECT command sets protection rights for password-protected files, directories, and segment directories.

**Note**

This command is not the same as the pre-Rev. 19 PROTEC command explained in Appendix C. The basic difference is that PROTECT uses mnemonic protection codes as opposed to the numeric protection keys used with PROTEC.

Although you may use PROTECT with objects protected by some type of access control, the protection rights are ignored when the objects are accessed. However, if you do this and then convert from an ACL directory to a password directory (with the REVERT_PASSWORD command), the protection rights set by PROTECT will come into effect.

On the PROTECT command line, the pathname identifies the object to be protected. You must have owner access to this object if it is a password directory, or protect (P) access if it is an ACL directory. If you specify a wildcard with no file type selection options, the default selects files, directories, and segment directories.

After the pathname, there are owner and nonowner codes that you may specify to designate various types of protection. These codes may have the following values:

| Code | Description |
|---|---|
| NIL | No access of any kind allowed. This is the default. |
| R | Read access only allowed. |
| W | Write access only allowed. |
| D | Delete access only allowed. |
| RW | Read and write access only allowed. |
| RD | Read and delete access only allowed. |
| WD | Write and delete access only allowed. |
| RWD | All rights (read, write, and delete) allowed. |

If you do not specify either the owner or nonowner code on the command line, the system default code is assumed (NIL). In other words, if you do not specify codes on the command line, neither owner nor nonowner will have any access to the object (protection defaults to NIL NIL). If protection has been set to NIL NIL, the owner must modify the protection rights before the files can be accessed.

The -REPORT option reports the results of each successful setting of protection rights using the PROTECT command. The following example illustrates the use of the PROTECT command with the -REPORT option:

```
OK, PROTECT SAIL>ARTICLE1 RWD R -REPORT
"SAIL>ARTICLE1" protected
OK,
```

## ▶ PRSER pathname

PRSER causes the file specified by **pathname** to be printed on the serial interface printer configured to PRIMOS. The printer (CENPR) must be assigned before the PRSER command is invoked.

▶ **PRTDSC station-1 station-2...**

The PRTDSC command invokes the printer emulation program on systems where DPTX/DSC is running. Printer output is spooled with form type equal to the first six characters of the station name. For details, see the **Distributed Processing Terminal Executive Guide.**

▶ **PRVER pathname**

PRVER prints a file specified by **pathname** on a printer/plotter configured to PRIMOS. The plotter must be assigned before the PRVER command can be issued.

Example:

```
OK, ASSIGN PLOT
OK, PRVER SALES.GRAPH
```

assigns the printer/plotter and prints at the printer/plotter.

▶ **PSD**

PSD loads and starts Prime Symbolic Debugger, an interactive debugging program that assumes control and waits for a command string. For details, see the **Assembly Language Programmer's Guide.**

To return to PRIMOS, type Q and carriage return at the user terminal.

▶ **PSD20**

PSD20 is a version of PSD for 16K PRIMOS II It is identical to PSD but occupies locations '17760 to '26552 It is described in the **Assembly Language Programmer's Guide.**

▶ **PTELE**

The PTELE command allows you to access the Office Automation System (OAS) Telephone Inquiry function. For detailed information, see the following documents:

- **OAS Management Communications and Support Guide (PT25)**
- **OAS Management Communications and Support Guide (PT45)**
- **OAS Management Communications and Support Guide (PT65)**

▶ **PT45DSC**

The PT45DSC command invokes the PT45 interface program, which allows a PT45 Terminal to emulate an IBM 3277 Model 2 display station on systems whree DPTX/DSC is running. For details see the Distributed Processing Terminal Executive PTU (PTU 2600-071).

▶ **RDY [option 1] [option 2]**

The RDY command allows users to choose the prompt messages they want displayed at their terminals and in their COMOUTPUT files.

The default message (the brief form) consists of the message OK, or ER! The long form includes clock time, the amount of CPU time used since the last prompt, and the amount of I/O time used since the last prompt.

If the user is at a command level above 1, the level number is also printed out. (This happens if the user has interrupted or terminated programs by typing CONTROL-P, or if error-handling mechanisms such as the condition mechanism have interrupted a program.) If the level is marked as static mode, a plus sign (+) follows the number. (This means that the last command executed was an external command.) Prompt messages may also be suppressed, so that they do not print at all.

The RDY command is given with one or more **options** to change the settings governing the printing of messages. It may also be given without **options** to produce a single long-form prompt.

| **Option** | **Function** |
|---|---|
| **-LONG** | Switches to the long format of prompt message. |
| **-BRIEF** | Switches to the short format. This is the default at login. |
| **-OFF** | Suppresses prompt messages. |
| **-ON** | Reenables the printing of prompt messages. Unless the -LONG or -BRIEF option is given with the -ON option, messages will appear in the format last specified. |
| **-READY_LONG text** | Changes the text portion of the long ready message to **text.** Default at login time is "OK,". |
| **-READY_BRIEF text** | Changes the text portion of the brief ready message to **text.** Default at login time is "OK,". |
| **-ERROR_LONG text** | Changes the text portion of the long error message to **text.** Default at login time is "ER". |
| **-ERROR_BRIEF text** | Changes the text portion of the brief error message to **text.** Default at login time is "ER!" |

The new prompt message, **text**, may be up to 20 characters in length. If it contains special characters or embedded blanks, it must be enclosed in single quotes. For example:

```
OK, RDY -RB Satisfactory. -EB Pfui!
Satisfactory. XYZZY
Not found. XYZZY (std$cp)
Pfui!
```

### ▶ REMOTE *Operator command*

The operator uses the REMOTE command from the supervisor terminal to permit or deny remote (network) access to local file system partitions (disk volumes) under FAM I. REMOTE has no effect on FAM II accesses. See the **System Operator's Guide.**

### ▶ REMOVE_PRIORITY_ACCESS disk-name *Operator command*

The System Administrator and the operator use this command to remove a priority ACL from the disk partition specified by **disk-name.**

For detailed information on priority ACLs and the REMOVE_PRIORITY_ACCESS command, see the **System Administrator's Guide** or the **System Operator's Guide.**

### ▶ REN

This internal command is used to reenter a subsystem following a QUIT or an error condition. It takes no arguments.

For REN to succeed, the subsystem being reentered must have defined an on-unit for the condition REENTER$ that can GO TO the appropriate point within the subsystem. At Rev. 19, only a few subsystems have defined such on-units If no on-unit exists, the REN command fails, and the user is returned to PRIMOS command level.

### ▶ REPLY *Operator command*

The operator uses the REPLY command to send messages to users in response to magnetic tape assignments requests. For details, see the **System Operator's Guide.**

### ▶ RESTOR pathname

The RESTOR command restores a runfile pathname from disk to memory, using the RVEC parameters saved with the file.

**Note**

Do not use RESTOR to restore a 64V segmented mode runfile, use the RESTOR subcommand of SEG.

Example:

```
OK, restor *bench9
OK, pm
SA,EA,P,A,B,X,K=
74 27250 6212 2 0 2 26100

PB,SB,IB,XB:
4000(3)/6212  0(0)/0  4002(0)/177400  4001(3)/1241
OK,
```

▶ **RESUME pathname [arguments...]**

RESUME runs (executes) the program specified by **pathname.** The program may either be a CPL program (in which case the name of the program must end in .CPL) or a runfile.

**File Suffix Search Order**

RESUME searches for files with the .SAVE suffix before searching for files with the .CPL suffix. If RESUME fails to find a file with the pathname specified suffix by.SAVE or.CPL, it assumes that the file specified is a saved file and executes it as a runfile.

**Note**

RESUME can also execute files with names ending with the .RUN suffix. This suffix represents a runfile format reserved for internal use by Prime. User files cannot use the .RUN suffix.

**arguments** are defined by the program being executed. If the file is a runfile (not a CPL program) the form of the arguments is:

**[p] [a] [b] [x] [keys] [program arguments...]**

where **p**, **a**, **b**, **x**, and **keys** may be used to set new values for the RVEC. (See Appendix A for details.) The program's runfile is loaded from disk to memory, using the saved values of SA and EA. RVEC is loaded from the saved RVEC parameters or from any new values specified in the command string. The processor registers and keys are then set from RVEC and the program is started at location **p**. Nonnumeric arguments are passed to the resumed program.

**Note**

Do not use RESUME to resume a 64V segmented mode program; use SEG instead. For further details, see SEG in this chapter.

If the file is a CPL program, **arguments** will not be interpreted as RVEC settings. Instead, they will be passed to the program as CPL arguments when the program executes. (For more information on CPL programs, see the **CPL User's Guide.**)

## ▶ REVERT_PASSWORD

This command converts your current directory from an ACL directory to a password directory. Before you perform any conversions, keep in mind the following:

- You must have protect (P) access to the directory before you convert it.
- When you convert a directory, its original password and protection keys are restored.
- You may not have ACL-protected subdirectories or access categories under a password directory. Therefore, if you give the REVERT_PASSWORD command for a directory that contains ACL-protected subdirectories or any access categories, the command will fail.

## ▶ RJ1004 RJ200UT RJ7020 RJX80 RJGRTS RJHASP

At Rev. 19, these commands have been replaced by the RJQ command. For details, see the **Remote Job Entry Phase II Guide.**

## RJE

Remote Job Entry (RJE) Phase II products are separately priced Prime software which enables multiuser Prime systems to emulate other vendors' RJE terminals over half-duplex, point-to-point, synchronous, and dial-up or dedicated communications lines. RJE is not a command. The commands used with RJE are as follows:

- The RJQ command provides the user interface to RJE.
- The RJOP command provides operator control of RJE.

Terminals that may be emulated through RJE are:

- IBM 2780 and 3780
- HASP
- CDC 200UT
- Honeywell GRTS
- Univac 1004
- ICL 7020
- XBM (CO3)

For detailed information on Prime's Remote Job Entry emulators, see the **Remote Job Entry Phase II Guide.**

*See also:* **RJQ, RJOP**

## ▶ RJOP *Operator command*

RJOP is the Remote Job Entry (RJE) operator command. The RJE local site operator uses RJOP to control RJE communication lines from a Prime computer to a remote site. Before invoking the command, however, the operator must create a site definition file which contains information about the particular type of RJE terminal being emulated, the line type used, and other emulator attributes.

RJOP allows the operator to monitor and control the transmission and reception of files, and to send messages to a remote machine. Other commands provided through RJOP allow the manipulation and control of access to RJE file transmission queues. An operator may also use the RJQ command to queue files for transmission, list RJE queue entries, cancel unwanted entries from the queue, and restart aborted file transmissions. For detailed information on RJOP, see the **Remote Job Entry Phase II Guide.**

*See also:* **RJE, RJQ**

## ▶ RJQ

RJQ is the Remote Job Entry (RJE) user command. It allows you to queue a file for transmission to a remote site. You can also use it to list the RJE file transmission queue entries, cancel entries from the queue, and restart aborted file transmissions. For detailed information, see the **Remote Job Entry Phase II Guide.**

*See also:* **RJE, RJOP**

## ▶ RLS [option]

The RLS command is used to discard unwanted stack history. (The **stack history** is a record of the calls and returns created by user commands. It is automatically saved by PRIMOS.) **option** determines how much of the stack is to be released, as follows:

| | |
|---|---|
| **–ALL** | Releases the entire stack down to listener level 1. |
| **–TO n** | Releases stack levels down to level **n. n** must be a positive decimal integer, and must be less than the current level number. |
| **–LEVELS n** | Releases **n** levels. The new stack level will be current level minus **n.** Since this must be a positive decimal integer, **n** must be a positive decimal integer such that the current level is –n≥1. Default is –LEVELS 1. |

If no options are given, one of two things happens. If the user's most recent command was an internal command, then the current level of the stack is released. If the user's most recent command was an external command, then the history of that command is released, but the level number of the stack is not changed.

If a large number of interrupts occurs, the stack grows large and unwieldy. If this happens, PRIMOS warns you that you are "Now at command level nn. To release use RLS. (listen_)". If the stack continues to grow and overflows its segment, PRIMOS reinitializes it automatically, sending the message "User environment reinitialized. (FATAL$)". Since most users don't need the stack history preserved, it is usually better to re-initialize the stack yourself when it begins to grow large than to wait for PRIMOS to do it for you.

## ▶ RPG

RPG invokes the Prime RPGII compiler. Refer to the **RPG II Programmer's Guide.**

## ▶ RSTERM [–READ] [–WRITE]

The RSTERM command empties the user terminal's read (input) and/or write (output) buffer. That is, it may be used to throw away typed-ahead input or output pending terminal characters.

Specifying –READ empties the input buffer. Specifying –WRITE empties the output buffer. Specifying neither option empties both buffers.

The command is useful in CPL programs for re-initializing the terminal state when a condition handler for the QUIT$ condition is specified.

## ▶ RUNOFF [pathname]

RUNOFF, Prime's text formatter, accepts commands to control margins, indentation, line spacing, column width, page numbering, running heads, and many other features of an ASCII source file. The commands may be entered from the terminal at runtime or be edited into the source file. RUNOFF produces a formatted output to the terminal, or to a designated disk file. For detailed information, refer to the **New User's Guide to EDITOR and RUNOFF.**

## ▶ RWLOCK pathname lock [–REPORT]

The RWLOCK command allows you to set the read/write concurrency lock on a file or segment directory. In other words, it allows you to determine how many readers and/or writers may access the file or segment directory at one time.

The pathname identifies the object to be protected. You must have owner access to the object if it is a password directory, or protect (P) access if it is an ACL directory. You may use a wildcard in the pathname. (Wildcards are discussed in Chapter 4.)

The lock specified on the command line identifies the read/write lock. You may specify only one of the following on the command line:

| Lock | Setting |
|---|---|
| SYS | Sets protection to the value of the system read/write lock. This is the default. The system lock is determined by the System Administrator. |
| EXCL | Sets for N readers OR one writer (exclusive OR). This means that an unlimited number of readers or one writer may access the object simultaneously. |
| UPDT | Sets for N readers AND one writer. This means that an unlimited number of readers and one writer may access the object simultaneously. |
| NONE | Sets for N readers AND N writers. This means that all users may have access to the object simultaneously. |

You may also specify an option, –REPORT, on the command line. This option acts as a verification tool, reporting each successful lock setting after it is completed.

## ▶ SAVE pathname [sa] [ea] [pc] [a] [b] [x] [keys]

The SAVE command saves the content of memory from **sa** (starting address) to **ea** (ending address) as a file named **pathname**. ("Memory" is actually segment '4000.) The other values are the RVEC parameters described in Appendix A. If any parameters are not specified, the existing values of RVEC are taken from the current register set and are stored with the program. The RVEC parameters are used to initialize the processor registers and keys when the program is restored or resumed.

Do not use SAVE to save 64V segmented mode runfiles. Use the SAVE subcommand of SEG instead. Refer to the **LOAD and SEG Reference Guide.**

**Note**

> All FORTRAN programs begin with ELM (Enter Load Mode). If macro assembler (PMA) users have ELM as the first instruction in the program, there is no need to set the keys after loading. The preferred way to save a memory image is to use the loader SAVE command.

▶ **SCHDEC [[–SCHEMA] schema-name [–LISTING] output-file]**

Invokes the DBMS Schema Decompiler. For information, see the **DBMS Schema Reference Guide.**

▶ **SCHED pathname**

Invokes the schema editor (SCHED), an interactive processor that allows a data base administrator to alter the definition of a data base. For information, see the **DBMS Administrator's Guide.**

▶ **SCHEMA source-filename [–VOL volume-name]**

Invokes the DBMS schema DDL Compiler. For information, see the **DBMS Schema Reference Guide** or the **DBMS Data Description Language Reference Guide.**

▶ **SEG [pathname]**

SEG invokes a utility for loading, modifying, running, and sharing segmented (V-mode and I-mode) programs. SEG **pathname** executes the V-mode runfile specified by **pathname.**

PRIMOS assigns segments to a user as they are accessed. These are not reassigned until either the LOGOUT or DELSEG command is issued.

The maximum number of segments available to a user program is between 32 and 256. It is a configuration dependent parameter, and is available from your System Administrator. User segments start at segment 2048 ('4000 octal). Segments in other ranges are for

PRIMOS itself, for shared code, and reserved for expansion. Information on installation of shared code is in the **System Administrator's Guide.**

A complete discussion of SEG is given in the **LOAD and SEG Reference Guide.**

▶ **SETIME -mmddyy -hhmm** *Operator command*

SETIME is an operator command that sets the system date and time of day.

▶ **SETMOD** $\left\{ \begin{array}{l} \textbf{-USER} \\ \textbf{-OPERATOR} \\ \textbf{-NOASSIGN} \end{array} \right\}$ *Operator command*

The SETMOD command determines whether users can assign their own tape drives or not. IF SETMOD is set to USER (the default), users can assign tape drives from their terminals by the ASSIGN command. If SETMOD is set to OPERATOR, the use of the ASSIGN command to request a magnetic tape assignment prints the request at the supervisor terminal. The operator responds to the request and notifies the user (via a message at the user's terminal) of the result. If SETMOD is set to NOASSIGN, no magnetic tape assignments are possible. In this mode, use of the ASSIGN command produces a message at the user's terminal saying that tape drives cannot be assigned.

▶ **SET_ACCESS target** $\left[ \begin{array}{l} \textbf{acl [-NO\_QUERY]} \\ \textbf{-LIKE reference} \\ \textbf{-CATEGORY category-name} \end{array} \right]$

You use the SET_ACCESS command to specify access rights for an object or a group of objects. An **object** is a file, directory, or segment directory. Access rights are defined in **Access Control Lists (ACLs)**, which are stored in **access categories.**

## Access Control Lists

An **ACL** is a list of users and access rights. The entries in the list together define who does or does not have particular rights to a file system object. Each entry in the list is an ordered pair, in the following format:

**identifier: rights**

**Note that the identifier and the rights must be separated by a colon (:), but not by any blanks. An ACL** may contain a maximum of 32 entries (pairs) but may not be more than a total of 160 characters long, including blanks.

**Identifiers** are of three types:

- An **individual identifier** identifies a single user. The user id serves as the identifier in this case.
- A **group identifier** identifies a set of users who are members of a particular access group. Your System or Project Administrator defines access groups for your system or project. The group name, which must always begin with a period (.), serves as the identifier in this case.
- The **special identifier** $REST identifies all users who do not have individual or group rights in a particular ACL. This identifier functions as a "catch-all".

The access right(s) identify the privilege(s) given to one or more users when using a particular file system object. ACLs provide access control by associating identifiers with lists of these access rights. Each right is specified by a mnemonic code. The available rights are:

| Code | Right | Applies to | Allows the User to: |
|---|---|---|---|
| **P** | Protect | Directories | Change accesses and attributes. |
| **D** | Delete | Directories | Delete directory entries. |
| **A** | Add | Directories | Add directory entries. |
| **L** | List | Directories | Read directory contents. |
| **U** | Use | Directories | Attach to directory. |
| **R** | Read | Files | Read file contents. |
| **W** | Write | Files | Change file contents. |

Two further special mnemonics apply to both directories and files:

| | | | |
|---|---|---|---|
| **ALL** | All | Directories and files | All of the above. |
| **NONE** | None | Directories and files | All access denied. |

### Access Categories

An **access category** is a named file system object that contains an ACL. You use it to protect a group of file system objects in a common manner. Basically, access categories provide an efficient way to group objects together for access control purposes.

Access categories stand alone: that is, they may be created before any objects are put into them and may exist after all objects have been removed from the category.

### Setting Access Control

There are four variations of the SET_ACCESS command line that you can use to specify access rights. In all cases, if the target is a password directory whose parent is an ACL directory, the target is converted to an ACL directory.

#### Specifying No Options

In the first case, you specify only a target with the command, as in:

**SET_ACCESS target**

The target must be a file, directory, or segment directory, and is set to use the default access for the parent directory. You cannot use this form of the command line if the target is an MFD. The following example illustrates this form of SET_ACCESS:

```
SET_ACCESS tramp
```

#### Specifying an ACL

In the second case, you specify an ACL in the command line. What happens depends on whether or not the target exists, what type of target it is, and how it is currently protected. The format is:

**SET_ACCESS target acl [-NO_QUERY]**

If the target is a file, the ACL for the file is set as specified. An ACL is automatically created if one does not currently exist for the file. If an ACL already exists, PRIMOS queries you before it replaces the contents of the ACL. You may suppress this query function with the -NO_QUERY command line option. If the file is protected by an access category, the contents of the category are not changed; instead, a new specific ACL is created to protect the file.

If the target is an access category, the previous contents of the category's ACL are replaced by the ACL you specify, with the same query function active.

If the target does not exist, a new access category is created with the ACL you specify. Whenever the object is an access category, the suffix .ACAT is added for you.

For example, suppose you want to set access on a file called SWILL. You want a user with id HOG to have read and write access to the file. You also want members of the access group .HERD to be able to read the file. You do not want anyone else to have access to it. You would give the command:

```
OK, SET_ACCESS SWILL HOG:rw .HERD:r $REST:none
```

### Using the –LIKE Option

In the third case, you specify the –LIKE option on the command line. The format is:

**SET_ACCESS target –LIKE reference**

Both the target and the reference must be existing file system objects. A specific ACL is created if one does not exist for the target. The ACL of the target is set identically to that of the reference. Again, if the target is protected by an access category, the target is removed from the category and the category is not changed.

For example, suppose you want the file HILL to be protected in the same way as the file SWILL in the previous example. You could give the command in the following form:

```
OK, SET_ACCESS HILL -LIKE SWILL
```

### Using the –CATEGORY Option

In the final case, you specify the –CATEGORY option on the command line. The format is:

**SET_ACCESS target –CATEGORY category-name**

**target** must be a file, directory, or segment directory, and **category-name** must specify an existing access category. The target you specify is added to the access category.

For example, suppose you have created an access category called TOP.ACAT, and you want to protect the file PAYSCALES with it. You would use the command in the following form:

```
OK, SET_ACCESS PAYSCALES -CATEGORY TOP
```

▶ **SET_DELETE pathname** $\left[\begin{array}{l}\textbf{-PROTECT}\\ \textbf{-NO\_PROTECT}\end{array}\right]$

You use the SET_DELETE command to prevent accidental deletion of a file, directory, or segment directory. SET_DELETE works only for objects in ACL-protected directories. You must have delete (D) access to the directory that contains the object in order to use this command. Command line parameters are:

| | |
|---|---|
| **pathname** | Identifies the object you wish to protect from deletion. |
| **-PROTECT** | Protects the object from accidental deletion by instructing PRIMOS to query you before it deletes the object. This is the default. |
| **-NO_PROTECT** | Allows PRIMOS to delete the object without querying you beforehand. |

In the following example, you delete-protect the file SPECS. You have specified -PROTECT on the command line, which instructs PRIMOS to protect the object. You then try to delete SPECS. When PRIMOS asks "ok to force delete?", you reply N (or NO) to the query. Next, you remove delete protection from the object SPECS by using the -NO_PROTECT option to SET_DELETE. When you next give the command DELETE SPECS, PRIMOS deletes the object without querying you before the deletion.

```
OK, SET_DELETE SPECS -PROTECT
OK, DELETE SPECS
"SPECS" protected, ok to force delete? N
OK, SET_DELETE SPECS -NO_PROTECT
OK, DELETE SPECS
OK,
```

## ▶ SET_PRIORITY_ACCESS disk-name acl *Operator command*

The System Administrator and the operator use this command to set a priority ACL on a given disk partition.

If a $REST identifier is given as part of the ACL in this command line, the access rights specified by that identifier override any other access control in effect on the partition.

For detailed information on priority ACLs and the SET_PRIORITY_ACCESS command, see the **System Administrator's Guide** or the **System Operator's Guide.**

## ▶ SET_QUOTA pathname [-MAX n]

The SET_QUOTA command sets the maximum quota on a directory or subdirectory to a specific number of records. (A **quota** is the maximum number of records that may be contained in a directory.) At most sites, only the System Administrator sets quotas on top-level UFDs. Users can set quotas on their own subdirectories as a check on their own storage use. To set a quota on a directory or subdirectory, you must have either protect (P) access (for Access Control systems) or owner access (for password systems) on the parent directory. If you attempt to use SET_QUOTA without having these rights, the quota will not be set. PRIMOS will return the following error message:

```
Insufficient access rights. directory-name (set_quota)
```

The **pathname** identifies the directory or a subdirectory on which you are setting a quota. If you want to set a quota on a subdirectory within your current directory, you only need to specify the final element of the pathname.

The command line option –MAX **n** specifies the maximum number of records to be allocated to the directory or subdirectory. **n** is a positive decimal integer between 0 and the maximum number of records on the disk partition. If **n** is 0, there is no quota and the UFD may use as many records as it can obtain.

If you attempt to set a quota on a directory or subdirectory which currently has none, when there are users currently accessing the directory or subdirectory, PRIMOS returns the message:

```
File in use.  directory-name (set_quota)
ER!
```

You can still successfully set the quota in this situation if you are the only user of the directory subtree. Use the ATTACH command to attach to another UFD or to a level that is higher than the level of the directory whose quota you wish to set. If you are not the only active user of the directory subtree, you must wait for all other users to attach somewhere else or log out before you can set the quota. In the following example, you are in the subdirectory PAGE that currently has no quota. You wish to set a quota of 400. PRIMOS returns the message "File in use." You then attach to a higher level in the subtree, CHAPTER, so that you can set the quota on PAGE:

```
OK, SET_QUOTA BOOK>CHAPTER>PAGE -MAX 400
File in use.  PAGE  (set_quota)
ER!  ATTACH BOOK>CHAPTER
OK,  SET_QUOTA BOOK>CHAPTER>PAGE -MAX 400
OK,
```

Once you have set a quota on a directory or subdirectory, you can modify it (raise, lower, or remove it). Simply give a new SET_QUOTA command with a different value specified with the –MAX option. Removing a quota is the same as setting the quota to 0.

▶ **SET_VAR name [:=] value**

The SET_VAR command defines a variable and places it and its value in the global variable file. Global variables are the only variables you can use at command level.

**name** is any legal variable name, up to 32 characters long. Names of global variables must begin with a dot (.).

**value** can be:

- A character string. At command level, the string must be short enough so that the entire command line does not exceed 160 characters. In CPL programs, the string may be up to 1024 characters long. A string must be enclosed in single quotes if it contains blanks or special characters. The single quotes are included in the character count.
- A numeric string representing an integer between the values of (−2**31)+1 to 2**31+1.
- A character string consisting of the logical value TRUE or FALSE.

The assignment symbol := is optional. The command defines the variable **name** if it was undefined, and assigns it the value **value.**

For example:

```
OK, set_var .a alpha
OK,
```

defines the global variable .A and assigns it the value ALPHA.

You can use the SET_VAR command interactively, at command level, to define global variables. Or, you may use it inside a CPL program to define either global or local variables. For further discussion of global variables, see Chapter 4 of this guide.

*See also:* **DEFINE_GVAR, DELETE_VAR,** and **LIST_VAR**

▶ **SHARE [pathname] segno [access-rights]** *Operator command*

The PRIMOS SHARE command is used for incorporating files into shared segments. It is issued only from the supervisor terminal. The command OPRPRI 1 must be given for SHARE to work.

The process of incorporating shared code into PRIMOS involves the use of both SEG's SHARE command and the PRIMOS SHARE command. The SHARE command of SEG gathers a number of SEG runfiles into a single segment runfile with an address below '4001. Then those R-mode runfiles below segment '4000 must be incorporated into PRIMOS using the PRIMOS SHARE command. For complete details, refer to the **System Administrator's Guide.**

▶ **SHUTDN** $\left\{ \begin{array}{c} \textbf{[ALL]} \\ \textbf{[pdisk0] [pdisk1...pdiskn]} \end{array} \right\}$ *Operator command*

The SHUTDN command performs tasks necessary to shutting down the PRIMOS system in an orderly manner.

▶ **SIZE pathname [–NORM]**

You use the SIZE command to find out the size of files, directories, and access categories. For existing files, SIZE tells you how many records the file contains. For directories, segment directories, and access categories, SIZE tells you the number of entries in the object.

In order to use SIZE in Access Control systems, you must have read (R) access to any file or segment directory you wish to size, and list (L) access to any directory you wish to size. You must also have use (U) and list (L) access to the object's parent directory.

**Note**

SIZE will not run under DOS. To find the size of file under DOS, use FUTIL.

The pathname you specify on the command line identifies the object whose size you wish to know. This pathname may be a wildcard name.

The report returned by SIZE depends upon the type of object specified by the pathname, as follows:

| **Object** | **Report** |
|---|---|
| **file** | Prints the size of the file in 1024-word records. If you specify the –NORM option, the size is printed in 440-word records. The number of words in the file and the file type ("sam file" or "dam file") are also printed. |
| **UFD** | Prints the number of top-level entries in the UFD. This includes files, segment directories, sub-UFDs, and access categories. The directory type ("pwd UFD" or "acl UFD") appears after the entry count. Finally, the size in words is printed. |
| **segment directory** | Prints the number of top-level entries in the segment directory, showing the directory type ("sam SEGDIR" or "dam SEGDIR") after the entry count. The size of a segment directory is printed in terms of the number of entries that it can hold. You can get the size in words by multiplying this number by 2. |

| | |
|---|---|
| **access category** | Prints the number of entries in the category (specifying the type "access cat"). |

SIZE always prints the pathname you specify on the command line. This shows you which SIZE output corresponds to which object.

To size all objects in your current UFD, specify:

```
OK, SIZE @@
     1 record  in sam file   "ABBREVS" (304 words)
     1 record  in sam file   "LOGIN.CPL" (56 words)
     1 record  in sam file   "C*LOGIN" (17 words)
    47 entries in acl UFD    "TREE" (763 words)
```

To size all objects in or below your current UFD, use the PRIMOS command preprocessor option –WALK_FROM 1 as in:

```
OK, size *>@@>@@ -walk_from 1
     1 record  in sam file   "ABBREVS" (304 words)
     1 record  in sam file   "LOGIN.CPL" (56 words)
     1 record  in sam file   "C*LOGIN" (17 words)
    47 entries in acl UFD    "TREE" (763 words)
     1 record  in dam file   "*>TREE>NUM1" (47 words)
```

**Note**

The PRIMOS command processor does not treewalk segment directories. Consequently, you cannot size any files inside a segment directory with SIZE *>@@>@@. To size files in a segment directory, use FUTIL. However, you can specify SIZE *>segdir_name>@@ to scan a segment directory and SIZE will work properly.

The following example uses a wildcard to size the entire contents of a UFD, and illustrates the range of reports returned by SIZE:

```
OK, SIZE MYUFD>@@
     3 entries in access cat "MYUFD>PROTECT-EM.ACAT"
     1 entry   in access cat "MYUFD>AB.ACAT"
    11 entries in acl UFD    "MYUFD>JUNK.UFD" (467 words)
    26 entries in acl UFD    "MYUFD>MAIL" (1058 words)
     1 entry   in acl UFD    "MYUFD>PRETENDER" (23 words)
     1 record  in sam file   "MYUFD>BB_PROFILE" (26 words)
     2 records in sam file   "MYUFD>XSTAT.SAVE" (1034 words)
     4 records in dam file   "MYUFD>O_RUNIT" (3606 words)
     4 entries in sam SEGDIR "MYUFD>SWU.SEG" (65 total)
     1 entry   in dam SEGDIR "MYUFD>DATABASE" (5 total)
    33 entries in access cat "MYUFD>TEST.ACAT"
    24 entries in pwd UFD    "MYUFD>OLD_SOURCE" (5040 words)
     2 entries in access cat "MYUFD>MAIL_DIRECTORY.ACAT"
```

**Note**

When SIZE needs to print a number greater than 32767 before the word "entries" or "records", it allows 12 full character positions for the number instead of the normal 6 character positions. For example:

```
OK, SIZE MYUFD>ELEPHANT
     32768 records in dam file       "MYUFD>ELEPHANT"
```

▶ **SLIST [pathname]**

SLIST prints (or displays) the contents of a file at the user's terminal.

**pathname** specifies the name of a file. If it is a simple filename, it is in the current directory. If **pathname** is omitted, SLIST asks the user to specify a treename (synonym for pathname).

SLIST is often used to display source listings of short programs or data files. If TERM –XOFF has been enabled, the user can halt the SLIST display by typing CONTROL-S, then restart the display by typing CONTROL-Q or quit by typing CONTROL-P.

Examples:

```
OK, slist
Usage: SLIST <treename>

OK, slist testfile
FTN TEST -64V
COMINPUT LOADTEST.COMI 7
CLOSE 7
COMINPUT -TTY

OK,
```

▶ **SORT [–BRIEF] [–SPACE] [–MERGE] $\begin{bmatrix}\text{-TAG}\\ \text{-NONTAG}\end{bmatrix}$**

The SORT command sorts up to 20 files into a single output file, sorting (in ascending or descending order) on up to 64 keys. SORT is a stable sort: that is, it preserves the order of input for records with equal keys.

SORT's options are as follows:

| Option | Meaning |
|---|---|
| –BRIEF | SORT program messages are not printed at the user's terminal. |
| –SPACE | Any blank lines are deleted from the SORT output file. |

| | |
|---|---|
| **–MERGE** | A merge of presorted files is requested. |
| **–TAG** | A TAG sort (described below) is requested. |
| **–NONTAG** | A NONTAG sort (described below) is requested. |

A **TAG sort** is specified when large files are sorted. For unordered files, it is a faster sort than nontag. Internally, the TAG sort stores input records separate from the key data. After all keys have been sorted and merged, the corresponding records are then located and output.

A **NONTAG sort** may be specified for smaller or well ordered input files. Internally the NONTAG sort stores each input record with its sort key in the work file. This eliminates the search for records after merging, but requires more disk space.

If –TAG or –NONTAG are not specified, the system defaults to TAG.

When invoked, SORT prints a message requesting:

- The name of the file to be sorted
- The name of the output file to be created
- The number of keys for the sort (default is 1)

This information should be given on a single line. When it has been supplied, SORT asks for:

- The starting and ending columns of each key field (default is ASCII, "A")
- The data type of the key
- Information on whether the sort on that key is to be done in ascending or descending order. Ascending is the default and need not be specified; descending is indicated by an "R" for reverse.

Information for each key field should be given on a separate line.

If the –MERGE option was given, SORT now asks for:

- The number of additional files to be merged
- The name of the files

Give the number and each filename on a separate line.

The dialog for a sample sort is as follows

```
OK, sort
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME — OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
alpha  alpha 2
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE—SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN—SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE—SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
     "UI" - UNSIGNED INTEGER
  DEFAULT IS ASCII.
1 5
8 10 ts r

BEGINNING SORT

PASSES           2              ITEMS         5
[SORT-REV18.0]
OK,
```

The same sort with -BRIEF in force would be requested by

```
OK, sort -brief
num4 num5
1 5

BEGINNING SORT



 PASSES        2           ITEMS        16

[SORT-REV18.0]
OK,
```

Several sorted files could then be merged with a merge-sort:

```
OK, sort -merge -brief
num2 num3 2
1 5
8 10 ts r
2
beta
gamma

BEGINNING MERGE

PASSES      1             ITEMS       5

[SORT-REV18.0]
OK, slist num5
able   49
able   15
able   96-
baker  44
baker  43
baker  14-
charl  95
charl  52
charl  78-
delta  66
delta  44
delta  67-
easy   97
easy   49
easy   12-
OK,
```

**Caution**

Giving identical names for input and output files is not a safe practice. When this is done, the disk space used by the file becomes free (and hence vulnerable) during the sort. If the space is taken over by another user during this time, a "disk full" error — and loss of the file being sorted — may result.

## File and Record Types

SORT can process four types of files: ASCII files (also called compressed files), uncompressed files, binary files (also called variable-length files), and fixed-length files. (All files created by Prime's text editor, ED, are ASCII files.) The file types are defined by the records they contain, as follows:

| Type | Definition |
|---|---|
| **COMPRESSED SOURCE (ASCII)** | Blank compressed record delimited by a NEWLINE character (:212). Source lines cannot contain data that may be interpreted as a blank compression indicator (:221) or a NEWLINE. |
| **UNCOMPRESSED SOURCE** | Uncompressed record delimited by a NEWLINE character (:212). They cannot contain data that may be interpreted as a NEWLINE. |
| **VARIABLE LENGTH** | Record stored with length (in words) stored in first word. (The first word is not included in the word count.) |
| **FIXED LENGTH** | Record containing data only, no length information. The length must be specified using the –INLENGTH or –OUTLENGTH keywords. If a NEWLINE character is appended to each record so that the file can be edited, it must be included in the character count. |

SORT has two default file types: ASCII (uncompressed) and binary (variable length). The type used depends on the type of key selected. If a user specifies any integer or real key, SORT defaults to binary records. Otherwise, the SORT file type defaults to ASCII.

## Specifying File and Record Information

By using whatever keywords are needed from the list below, users can specify multiple input files for SORT or SORT –MERGE, indicate file types, or give information about record length. These keywords replace the standard "inputfile outputfile number-of-keys" lines of dialog. They are given a single line, in any order. All files must be of one type. Input and output files may be of different types.

| Keyword | Usage |
|---|---|
| **–INPUTFILE name** | Specifies a file to be sorted. **name** may be a pathname of up to 80 characters. Repeat this keyword for each input file. |
| **–OUTPUTFILE name** | Creates a file to hold the sorted output. Only one output file per sort is allowed. |
| **KEYS n** | **n** is the number of keys for the sort. |
| **–INTYPE** $\left\{\begin{array}{l}\text{COMPRESSED}\\ \text{UNCOMPRESSED}\\ \text{FIXED}\\ \text{VARIABLE}\end{array}\right\}$ | Specifies the type of file(s) to be sorted. All input files must be of the same type. If this keyword is not given, a default file type will be taken from the key type. |
| **–OUTTYPE type** | Specifies file **type** for the output file. Types are the same as those for input files. If this keyword is not given, the output file will have the same type as the input file(s). |
| **–INLENGTH n** | Gives the maximum length of the input records (in bytes). (Greatest possible **n** is 32760 bytes. This is also the default.) This keyword must be given for fixed-length records. |
| **–OUTLENGTH n** | Specifies maximum length for records in output file. Defaults to length of input record. If you specify a fixed-length record output file, you must also specify its record length. |

## Key Types

SORT recognizes 13 types of keys. ASCII files (compressed and uncompressed) can be sorted on seven of these key types: A, LS, TS, LE, TE, and AU. Variable- and fixed-length files can use any key type.

Table 2-2 explains each key type. Note that the default, A, sorts upper- and lowercase characters differently. This represents a change from the Rev. 16 default, which converted lowercase characters to uppercase before sorting them. The lower-and-uppercase-equal sort is now provided by the AU key.

## Specifying Key Information

Key information may also be specified via keywords, as follows:

| Keyword | Usage |
|---|---|
| **–START n** | **n** is first column of key. |
| **–END n** | **n** is last column of key. |
| **–DESCENDING** | Requests sort in descending order. |
| **–TYPE code** | Any of the codes from Table 2-2. |
| **–EBCDIC** | Requests that the EBCDIC collating sequence, rather than the ASCII sequence, be used for sorting (used only with A or AU key types). |

An example of a SORT dialog using keywords might be:

```
OK, sort -brief
-input chaos.1 --input chaos.2 -output order -keys 2
1 10
15 20 r

BEGINING SORT


 PASSES        2          ITEMS       16

[SORT-REV18.0]
OK,
```

**Table 2-2**
**Key Types**

| Code | Key Type | Definition |
|---|---|---|
| **A** | **ASCII (default)** | Character strings, stored one character per byte. Their length is limited only by the length of the record. |
| **I** | **Single-precision integer (short)** | Length is 2 bytes; range is −32767 to +32767 |
| **J** | **Double-precision integer (long)** | Length is 4 bytes; range is −2**31 to +2**31 −1. |
| **F** | **Single-precision real** | Length is 4 bytes; range is ±(10**−38 to 10**38). |
| **D** | **Double-precision real** | Length is 8 bytes; range is ±(10**−9902 to 10**9825). |

**Table 2-2 (continued)**

| Code | Key Type | Definition |
|---|---|---|
| U | **Numeric ASCII, unsigned** | Like plain ASCII. These are stored one digit per byte, and are limited only by the length of the record. |
| LS | **Numeric ASCII, leading separate sign** | Numbers preceded by "+" or "−" to indicate positive or negative value. (A blank space will be treated as a positive sign.) |
| TS | **Numeric ASCII, trailing separate sign** | Same as LS, except that the "+" or "−" follows the number. |
| LE | **Numeric ASCII, leading embedded sign** | One digit per byte. Alphabetic characters may represent digits, as shown in the insert table below. The first character represents both a digit and the sign of the field (e.g., L579 represents −3579). |

| Digit | Positive | Negative |
|---|---|---|
| 0 | 0,−,+,{ | },− |
| 1 | 1 A | J |
| 2 | 2 B | K |
| 3 | 3 C | L |
| 4 | 4 D | M |
| 5 | 5 E | N |
| 6 | 6 F | O |
| 7 | 7 G | P |
| 8 | 8 H | Q |
| 9 | 9 I | R |

| | | |
|---|---|---|
| TE | **Numeric ASCII, trailing embedded sign** | Same as LE, except that the last digit carries the sign (e.g. 357R represents −3579). |
| PD | **Packed decimal** | A four-bit nibble represents each digit; the number ends with a sign nibble. A negative sign is represented by hex D in the sign nibble; any other value in the sign nibble indicates a positive number. A packed field must have an odd number of digits plus the sign; since they are stored two nibbles (digit or sign) per byte, this comes out to a full number of bytes. Packed decimal keys may be up to 63 digits plus sign. |
| AU | **ASCII, upper- and lowercase sort** | Storage is identical to regular ASCII. Lowercase characters are sorted as uppercase, put into output file as lowercase. |
| UI | **Unsigned integer** | Length is 2 bytes; range is 0 to 65535. |

## ▶ SPOOL [pathname] [options]

The SPOOL subsystem allows any user to submit disk files to be printed or plotted on the system line printers and/or plotter. The subsystem consists of a user interface program (SPOOL), a set of phantom programs to print files queued on the various printing or plotting devices, an operator interface program (PROP), and a queue management facility.

The SPOOL program allows users to submit files to the queue mechanism, interrogate the queue status, and cancel their queued requests. When SPOOL is invoked by a user submitting a file, it responds with a message of the following format:

```
PRT002 spooled, records:        1, name: SHORTFILE
```

The length of the spool file determines its priority in the queue. Short files are given a higher priority than long ones to ensure efficient queue operation. (To find out what boundary divides short files from long ones on your printers, use the PROP –DISPLAY command.)

The first parameter to give when submitting a file is **pathname**; subsequent parameters are specified as **options** in the SPOOL command format. Using **options**, the user may request that the spooler provide certain internal services when printing the spool file – for example, inserting line numbers in the left margin, using FORTRAN output conventions (column 1 represents carriage control), etc.

The user may also use **options** to specify other information regarding the spool file: for example, that it be deferred (held in the queue without printing) until a certain time of day, that the file be printed or plotted on a special form type, etc.

| **Parameter** | **Meaning or Remarks** |
|---|---|
| **pathname** | If **pathname** is specified, the file named **pathname** is copied onto the SPOOL queue. If the –OPEN option is used, a file is created in the SPOOL queue. This file has null contents, and it remains open on File Unit 2 (unless otherwise specified by the –TUNIT option). When SPOOL is invoked in this manner, it responds as follows: |

```
OK, SPOOL -OPEN
[SPOOL rev 19.0.0]
PRT004 opened.
OK,
```

|  |  |
|---|---|
|  | File Unit 2 (or the specified TO file unit) remains open until closed by specific user command or action. When closed, any information placed in the previously open file (such as listings from compilations) is printed, and then deleted, by SPOOL. |
| **–AS alias** | **alias** (a name of 16 or fewer characters) replaces **pathname** on the file header and in SPOOL –LIST displays. |
| **–AT destination** | Denotes the printer (or printers) which may print the file. Useful for specifying location when the computer is connected to others via PRIMENET, or when output is delivered to mailstops. |
| **–CANCEL** | Removes the print or plot request(s) (specified by their PRT numbers) from the queue. For example: |

```
OK, spool -cancel prt004
[SPOOL rev 19.0.0]
PRT004 cancelled.
OK,
```

|  |  |
|---|---|
|  | If used, this option must be the final option on the command line. |
| **–COPIES n** | Specifies number of times file is to be printed. (**n** must not exceed 99.) When this option is used, the file's length is considered to be its actual length multiplied by the number of copies. |
| **–DEFER [time]** | Defers printing of the file. The file remains in the spool queue but is not printed until after the specified time. **time** may be entered in either 24-hour format (00:00 = midnight) or in 12-hour format, with AM or PM (12:00 AM = midnight). |
| **–FORM [type]** | Specifies a paper type. This file is not printed until the system operator indicates that the requested form is mounted on the printer. **type** must be a 1-6 character name and does not have to start with an alphabetic character. If –FORM is not given, the default paper type is used. |
| **–FTN** | Indicates FORTRAN output conventions are to be used when printing the file. Characters in column 1 of each line have special significance, as follows: |

|  |  |
|---|---|
| **1** | Eject to top of page before printing. |
| **0** | Skip 2 lines. |
| **space** | Skip 1 line. |

| | |
|---|---|
| + | Overprint last line (available on MPC parallel-interface printers only). |
| – | Triple space. |

**–FUNIT n** Directs the spooler to use the file unit specified by **n** for input. The default unit is unit 1. **n** must be an octal integer between 1 and '176. It must not conflict with any default or explicit –TUNIT specification.

The spooler will not generate an initial page eject after the banner page is output when this mode is specified.

**–LIST** Lists all SPOOL-queue entries or selected SPOOL-queue entries. Possible values and associated meanings for list option are as follows:

| List Option | Meaning |
|---|---|
| **OWN** | Lists files spooled under the current user's id. |
| **DEFER** | Lists deferred files. |
| **PLOT** | Lists files in plot queue. |
| **PRINT** | Lists files in print queue. |
| **FORM type** | Lists files queued with the form specified by type. For default form do not specify type. |
| **PRINT** | Lists files queued with the form specified by type. For default form do not specify type. |
| **ALL** | Lists all files in queue. This is the default for list option. |

If used, this option must be the final option on the command line.

**–LNUM** Generates line numbers. The spooler prefixes each line of output with a four-to-five digit line number enclosed in parentheses and followed by a space. This option may not be used with the FORTRAN print convention option, –FTN. It also overrides carriage control characters.

**–NOFMT** Used for files containing EVFU "skip-to-channel" commands. This option disables normal spooler format control (pagination and header generation). It is incompatible with the –FTN and –LNUM options.

**–NOHEAD** Prints file without header or trailer pages.

| | |
|---|---|
| **–OPEN** | Specifies open-only operation. Instead of copying the named file to the spool queue, SPOOL creates a new queue file and opens it on the unit specified by –TUNIT. |
| **–PLOT [nwords]** | Denotes a plot file; **nwords** is the decimal number of words to be read and output per raster scan. If **nwords** is omitted, the default is 128 (for a 200 raster/inch plotter). This spool file is invisible to the line-printer spooler. |
| **–TUNIT unit** | Defines TO-unit number This is the file unit that is opened on an open-only operation and is used for the output of a copy operation. The restrictions regarding **unit** conflict and argument specification described with –FUNIT apply also when using –TUNIT. |

**Multiple Options:** Most of the options shown above can be used jointly on the command line. Exceptions are:

- Only one of the formatting options, –FTN, –LNUM, and –NOFMT may be used in a single command line.
- Only one of the two command options –LIST and –CANCEL may be used in a single command line.

An example of a SPOOL command line might be:

```
SPOOL SHORTFILE -AS MEMO.1 -AT BLDG_3 -DEFER 2200
```

This example requests that the file SHORTFILE (in the current directory) be printed under the name MEMO.1, at the BLDG_3 printer. It also requests that printing be deferred until 10 PM (22:00).

*See also:* **PROP, CONCAT**

## ▶ START [pc] [a] [b] [x] [keys]

START initializes the processor's registers and keys from the command line (or from RVEC, for any values not specified in the command line) and starts execution at location **pc**. This form of the command starts or restarts a static mode program previously loaded into memory by a RESTOR or RESUME command. (Static mode programs include external PRIMOS commands and most user software. Recursive mode programs, discussed below, include internal PRIMOS commands and all on-units.)

START can also restart a static or recursive mode program that has returned control to PRIMOS (for example, because of a BREAK, an error, or a FORTRAN PAUSE or CALL EXIT statement). If START is typed without arguments, one of the following two things occurs:

1. If no static mode program has been invoked or restored at this listener level (as is the case afer a BREAK, for example), the interrupted program is continued from the point of interruption, whether it is static or recursive mode.

2. Otherwise, the static mode program defined by the current RVEC is invoked.

If arguments are given, their register settings are applied to the RVEC contents (defining a new static mode machine state), and that static mode program is invoked.

**Note**

If a static mode program is interrupted, and another static mode program is invoked subsequently, an attempt to START the first program will be refused with an appropriate error message. This is done because the machine state for the first static mode program has been overwritten in the RVEC, and hence the system does not have the information needed to restart the program correctly.

▶ **STARTUP [PROTECT] pdisk0 [pdisk1...pdisk7]** *Operator command*

STARTUP initializes the configuration of disk drives by relating physical disk drive numbers to PRIMOS logical disk unit numbers. Physical device numbers for disks are given in the **System Administrator's Guide.**

*See also:* **ADDISK**

```
                 ┌             ┐
                 │ ALL         │
                 │ DEVICE      │
                 │ DISKS       │
                 │ ME          │
▶  STATUS        │ NETWORK     │
                 │ PROJECT     │
                 │ SEMAPHORES  │
                 │ SYSTEM      │
                 │ UNITS       │
                 │ USERS       │
                 └             ┘
```

You use the STATUS command to check the current status of various aspects of your system. For example, you can find out who is using your system, what file units you have open, what parts of your network are running, and other information in considerable detail.

The options to the STATUS command are as follows:

| Option | Meaning |
|---|---|
| ALL | Displays all information accessible through STATUS. |
| DEVICE | Displays the physical and logical device numbers of any assigned magnetic tape drives. |
| DISKS | Shows you which disks are running on your system, including disks on your network if your system is on a network. |
| ME | Shows you what id you are using, and whether anyone else, including any phantoms, is using the same id. The ME option also shows you your user number, and the line and devices you are using. |
| NETWORK | Displays the status of other systems to which your system is attached by PRIMENET. |
| PROJECT | Shows you the project id being used by each user on your system. This option lists all the users in order of their user number, showing the user id of each one. |
| SEMAPHORES | Displays the value of user semaphores that have been set on your system. A **semaphore** is a flag used for synchronizing processes. It is used by cooperating user processes to control access to a single shared resource. |
| SYSTEM | Shows you what version of PRIMOS is running on your system. |
| UNITS | Shows you what file units you have open. |
| USERS | Lists the ids of all the current users of your system. |

If you give the STATUS command with no options, it displays information provided by the SYSTEM, UNITS, DISK, SEMAPHORE, NETWORK, and ME options, in that order.

The following example illustrates the use of STATUS with no options:

```
OK, STATUS

System is currently running PRIMOS rev. 19.0


User SALLY                           NJR

File   File      Open    File
Unit   Position  Mode    Type  RWlock  Treename
 127  000001710   W      DAM   NR-1W   <MARKET>SALLY>STATUS.COMO


Disk    Ldev  Pdev  System
SYS.R     0   4463
SPOOLR    1    460
MARKET    2  32060
PARK      3  71061
SYS.E     4         NJE
INTEG     5         NJE
ABGRP2    6         NJE
OCEAN     7         NJK
SYSNJK   10         NJK
BARGR3   11         NJK
BARTST   12         NJK
CDTEST   13         NJK
SYSNJM   14         NJM
ABGRP1   15         NJM
SOFTM    16         NJM
SYSNJD   17         NJD
TRANSA   20         NJD
TRANSB   21         NJD

Sem. Value  Users
---- ------ -----


Full duplex network

     Node       State
   NJR          ****
   MK.K          Up
   TELENET       Up

Ring network

     Node       State
   NJR          ****
```

```
        NJD       Up
        NJE       Up
        NJK       Up
        NJM       Up
        NJ.D1     Up
        NJ.D2     Up
        NJ.D3     Up
        NJ.D5     Up
        NJ.D6     Down
        QA.PCB    Up
        QA.PC2    Up
        QA.LSI    Up
        NJ.G3     Down
        NJ.CP3    Up

    Public data network

          Node
        MK.B
        MK.D
        MK.WR
        SED1
        SED3

    User                              No  Line Devices
    SALLY                             68   rem <MARKET> (from NJE    )

    OK,
```

Under PRIMOS II, STATUS lists the login UFD, the logical device upon which the UFD resides, the low boundary of PRIMOS plus buffers, the open file units, and the physical-to-logical device correspondence. STATUS also lists physical device numbers, as described in the **System Operator's Guide.**

## ▶ SVCSW { }

The SVCSW command controls the handling of SVC instructions in a virtual memory environment. For more information on SVC instructions, see the **Assembly Language Programmer's Guide.**

The normal mode (SVC 0) causes all SVC instructions to be trapped and processed by the system supervisor. If the SVC SWITCH is on (SVC 1), almost all SVC instructions cause a virtual trap, and SVC instructions are handled through the user's location '65. The class of SVC instructions always processed by the PRIMOS operating system, regardless of the SVCSW command, are those determined by FUNCTION code 5XX. Currently the SVC's are RREC, WREC (for reading and writing to disk), TIMDAT (for obtaining the time and date from PRIMOS), and RECYCL.

The SVC switch is initialized to 0 by the LOGIN command.

▶ **TCF –HOST hname –TERMINAL tname [–QUIT 'q string']**

This command invokes DPTX/TCF (Transparent Connect Facility) on a system where DPTX/TSF and DPTX/DSC are running. The parameters for TCF are as follows:

| | |
|---|---|
| **hname** | Is the PRIMOS name for the IBM host mainframe the user wishes to connect to. |
| **tname** | Is the PRIMOS name of the IBM 3270-type terminal from which the user wants to communicate (the current terminal may be specified by ' '). |
| **q string** | Is a string of eight characters or less that will serve as the signal that the user wishes to return to PRIMOS command level. The default string is 'TCF$QUIT'; it must be given in all caps, as upper- and lowercase characters are considered different characters when the QUIT string is being evaluated. |

**Note**

When the quit string is entered at the terminal, the program breaks the connection with the host, prints out 'TCF HALT', and returns to PRIMOS. To the host, it seems that the terminal was powered off. Some applications programs may not tolerate this; consult your System Administrator for guidance.

For a detailed description of TCF, see the **Distributed Processing Terminal Executive Guide.**

▶ **TERM** $\left\{ \begin{array}{l} \textbf{–ERASE char} \\ \textbf{–KILL char} \\ \textbf{–BREAK ON} \\ \textbf{–BREAK OFF} \\ \textbf{–FULL} \\ \textbf{–HALF LF} \\ \textbf{–HALF NOLF} \\ \textbf{–DISPLAY} \\ \textbf{–NOXOFF} \\ \textbf{–XOFF} \end{array} \right\}$

The TERM command allows the user to define his terminal characteristics. It controls duplex and half-duplex operation, designates KILL and ERASE characters, and enables or disables the BREAK (CONTROL-P) key.

The TERM command with a null argument prints or displays a list of the optional parameters at the terminal.

Multiple options may be specified on one line with the TERM command. For example:

```
term -halt -xoff
```

may be specified on one line. See below for definitions of –HALF and –XOFF.

| **Parameter** | **Meaning or Remarks** |
|---|---|
| **–ERASE char** | Sets the ERASE character to the character **char**, for the duration of the user's process. The default ERASE character, set upon LOGIN, is the double quote ("). |
| **–KILL char** | Sets the KILL character to the character **char**, for the duration of the user's process. The default KILL character is the question mark (?). |
| **–BREAK OFF** | Inhibits the CONTROL-P or BREAK character from interrupting a running process. The default condition, 'BREAK ON', is set by the LOGIN command, and allows CONTROL-P or BREAK to interrupt a process. |
| **–BREAK ON** | Enables the CONTROL-P or BREAK character to interrupt a running process. |
| **–FULL** | Sets PRIMOS to treat the user terminal as a full-duplex terminal. Under PRIMOS, normal operation is full duplex. This option is not reset to a default by the LOGIN command; therefore the user must reset it. With full-duplex operation, all characters are echoed except a LINEFEED input character, which is ignored. A carriage-return input character is passed into the system as a LINEFEED and echoed to the user as carriage return followed by LINEFEED. |
| **–HALF [LF]** | Sets PRIMOS to treat the user terminal as a half-duplex terminal. No characters are echoed except carriage return, which echoes a NEWLINE. Input is sent to the LINEFEED system in the same manner as with TERM –FULL. LOGIN does not reset the terminal to a default setting of half duplex or full duplex. Therefore, the user must reset to half or full duplex, as desired. |
| **–HALF NOLF** | Functions in the same manner as –HALF except that the carriage return does not echo as LINEFEED. |
| **–DISPLAY** | Prints (or displays) the current values for the ERASE and KILL characters at the user terminal. |
| **–XOFF** | Sets the terminal to full duplex (default value) and enables the X-OFF (CONTROL-S) and X-ON (CONTROL-Q) keys. |

| | |
|---|---|
| **–NOXOFF** | Sets the terminal to full duplex (default value) and disables the X-OFF (CONTROL-S) and X-ON (CONTROL-Q) keys. |
| **–XOFF –HALF** | Half duplex with X-OFF enabled. |

**Note**

The X-OFF feature is not available under PRIMOS II.

## ▶ TIME

The TIME command prints the current value stored in the PRIMOS time accounting registers. The three values printed are the same as the three values in the logout message:

| | | |
|---|---|---|
| **Connect time** | *(hours, minutes)* | Time since LOGIN. |
| **Compute time** | *(minutes, seconds)* | Time accumulated executing commands or using programs (does not include disk I/O time). |
| **Disk I/O time** | *(minutes, seconds)* | Time accumulated for disk input/output. |

The disk I/O time includes user-requested I/O to files and also paging I/O time generated on the user's behalf. All times include supervisor services, such as the time spent executing supervisor subroutines on the user's behalf. Some supervisor service associated with the PRIMOS scheduler is charged to the supervisor (at the supervisor's terminal) and not the user. When the system is idle, CPU time is charged to the supervisor. Computer time does not include I/O time for diskette or for disks connected to a type 4000 Controller.

Example:

```
OK, time
Time used: 00h 52m connect, 00m 27s CPU, 00m 26s I/O.
OK,
```

## ▶ TRAMLC

TRAMLC transmits or receives a file over an assigned AMLC line between two Prime computer systems using transparent protocol

TRAMLC's first prompt is:

```
FNAME
```

Reply by giving the filename or pathname of the file you wish to transmit or receive. TRAMLC then prompts:

```
T/R LINE# BLOCK
```

This prompt is explained as follows:

| | |
|---|---|
| **T/R** | Transmit or receive? Answer with T if you want to send a file, R if you want to receive one. |
| **LINE#** | Reply with the number of your AMLC line. |
| **BLOCK** | When TRAMLC sends a message, it divides it into 60-word blocks and transmits one block at a time. If you want to monitor the progress of your transmission, TRAMLC will send messages to your terminal, saying "Block n transmitted." The number of messages you receive depends on the response you give to this prompt. If your response is "0", TRAMLC sends no messages. If your response is a positive integer (**n**), TRAMLC sends a message at every **n** blocks. |

When the entire file has been transmitted or received, the message:

```
FILE COMPLETE
```

is printed at the user's terminal.

Either the transmitter or receiver program can be started first. Both devices may be running at the same baud rate.

### Note

TRAMLC is designed for conveniently transmitting/receiving small files between two Prime computers. For maximum performance reasons, TRAMLC should not be used as a communications link.

Error messages give the reason for the error that occurred and the block number of the failure.

## ▶ TYPE text

The TYPE command prints text at the user's terminal or into a command output file.

**text** contains the message or information to be displayed. **text** may also contain variable references and function calls. Specifying variable references and function calls is more useful within CPL programs than at command level.

## ▶ UNASSIGN device

The UNASSIGN command may be entered at the user terminal to which **device** is currently assigned or at the supervisor terminal. The effect of this command is the opposite of ASSIGN. The UNASSIGN command, entered at the system terminal, unconditionally deassigns the peripheral assigned to any user. Entered from a user terminal, UNASSIGN deassigns only the device that was previously assigned to the user. On selected devices, this command turns off the device and clears the associated I/O buffers.

**device** is a previously assigned device, as defined in the ASSIGN command.

From the supervisor terminal, this command can be used to release a device if the user who assigned it has forgotten to log out and has left his terminal.

Example:

```
UN PTR
```

unassigns the paper-tape reader.

If an operator unassigns a user-dedicated tape drive, no message will appear at that user's terminal. Should the user subsequently attempt to unassign the same device, an error message will be displayed.

Magnetic tapes can be unassigned by physical or logical device number:

```
UNASSIGN  MTpdn

UNASSIGN -ALIAS MTldn
```

The –ALIAS option can be specified only if a logical device number was previously assigned to this particular drive. UNASSIGN returns only the "OK," prompt to the terminal from which it is issued.

For a disk to be assigned to a user as an I/O device, it must not be assigned to another user nor be in the PRIMOS file system. It must also be specified by an entry in the Assignable Disks Table. (Refer to ASSIGN.) If the disk is assigned to PRIMOS, it must be released by the SHUTDN command at the supervisor terminal. A disk that has been assigned by a user cannot be entered as an argument in the ADDISK command. The supervisor terminal can UNASSIGN any device that may be assigned. All devices assigned by a user are released when the LOGOUT command is invoked by that user.

## ▶ UPCASE in-pathname out-pathname

UPCASE reformats files that contain lowercase alphabetic characters, making them suitable for output to a device with only uppercase alphabetic characters. UPCASE scans through an input file, replacing all occurrences of lowercase characters with their counterparts. **in-pathname** is the input file and **out-pathname** is the output file.

## ▶ USAGE [option-1...option-n] *Operator command*

This command invokes the USAGE subsystem, a system-metering tool. It allows the operator to monitor performance factors of the PRIMOS operating system. Although USAGE is commonly used as an operator command, it may also be invoked from other user terminals.

USAGE provides both manual and automatic sampling modes through various command line options. The options are:

| Option | Meaning |
|---|---|
| **–ALL** | Displays metering information about the system, disks, and each user, at each sample time. |
| **–BRIEF** | Produces a short form of output. It generates an overview of the processes and users using system resources. The default long form produces more detailed information. |
| **–DISK** | Displays only system and disk-metering information at each sample time. |
| **–FREQ n** | Selects automatic sampling every **n** seconds. **n** must be an integer between 1 and 32767, although it is recommended that you specify an integer greater than 30. If you do not specify this option, the default is manual sampling. |
| **–TIMES n** | Specifies the total number of samples to be taken, if automatic sampling is in effect. The command terminates after it prints **n** sets of data. **n** must be an integer between 1 and 32767. If you do not specify this option, sampling continues indefinitely. |
| **–USER** | Displays metering information about the system, and each user, at each sample time. This is the default mode of operation. |

If you want manual sampling, do not specify the –FREQ or –TIMES options. USAGE is invoked each time you give a START command (with no arguments), and prints the most recent differential values. After the sample is taken, USAGE pauses to command level and allows you to enter other commands.

**Note**

> It is not recommended that manual sampling times be less than 30 real seconds.

The following example illustrates a possible command line for manual sampling:

```
    OK, USAGE -BRIEF
    [USAGE 19.0]
    Type "START" to continue.
    OK, START
```

The next example illustrates automatic sampling. PRIMOS will monitor the system 10 times with an interval of 1800 seconds (30 minutes) between each sampling for a period of 5 hours (1800 seconds x 10 times / 3600 seconds-per-hour):

```
    OK, USAGE -FREQ 1800 -TIMES 10
```

**Caution**

> If a user logs in or logs out during a sampling interval, incorrect or possibly negative meter values may result. You must, therefore, interpret per-user metering data carefully. Processes may accumulate CPU time without actually being logged in. These processes will be displayed in the USAGE output.

For detailed information on USAGE, including user, system, and disk I/O metering display definitions, see the **System Administrator's Guide** or the **System Operator's Guide.**

## ▶ USERS

The USERS command prints the number of users currently logged into PRIMOS.

Example:

```
    OK, USERS
    Users = 38
    OK,
```

▶ **USRASR userno** *Operator command*

The USRASR command allows the supervisor terminal to act as a user terminal. The parameter **userno** is the user number to be associated with the supervisor terminal.

▶ **VISTA**

VISTA is the PRIMOS command that invokes DBMS/QUERY, the Data Base Management System query language and report writer. For detailed information, see the **DBMS/QUERY User's Guide** and the **DBMS/QUERY Reference Guide.**

▶ **VPSD, VPSD16**

VPSD is the V-mode version of PSD. It is described in the **Assembly Language Programmer's Guide.**

VPSD16 is a version of VPSD that is loaded at '160000 octal. Its commands are the same as those for VPSD.

▶ **VRPG**

This command invokes Prime's RPG II V-mode compiler. Note that this is not the same compiler that is invoked with the RPG R-mode command. For detailed information on VRPG, see the **RPG II V-mode Compiler Reference Guide.**

*See also:* **RPG**

▶ **VRTSSW [number]**

The VRTSSW command sets the virtual sense switches. There are 16 switches. **number** is an octal representation of them as a 16-bit word; the default is 0. It is stored and made available to the user when a program written in PMA executes an INA '1620 (read sense switches) instruction. The virtual sense switches are initialized to 0 by LOGIN.

Example:

```
V 40100
```

**Note**

The Skip-On-Sense-Switch series of instructions always refers to the actual sense switches, not the virtual sense switches.

## ▶ WP_ADMIN

The WP_ADMIN command allows the Office Automation System (OAS) Administrator to create and maintain the Global Keyword List relating to the Document Keyword Indexing function. For detailed information, see the **OAS System Administrator's Guide.**

## ▶ WS1004 WS200UT WS7020 WSX80 WSGRTS WSHASP

At Rev. 19, these commands have been replaced by the RJOP command. For details, see the **Remote Job Entry Phase II Guide.**

## ▶ $$

The $$ command is used in command files to flag information to be passed to the Batch monitor. Its use is explained in the writeup on JOB.

## ▶ /* [Any-text-desired]

A command line that begins with a slash-asterisk (/*) is interpreted as a comment line; no action is taken. Note that a comment cannot be followed by a semicolon and a second command, since all text following the /* is taken as part of the comment.

Example:

```
/* I wish I were out in my sailboat.
```

## ▶ * [Any-text-desired]

A line that begins with an asterisk (*) signifies a PRIMOS null command, that is, a command that performs no action. However, even though no action is taken, the line itself is evaluated by the command processor. Thus, it is possible for these lines to produce error messages that will halt a program even though the lines were not intended to execute. For example:

```
OK,    [
Function call contains too many left brackets. (EVAL_AF)
Error in variable or command function reference. (std$cp)
ER!
```

# 3 Dictionary of Command Functions

## Command Functions

Command functions are intended primarily for use with Prime's Command Procedure Language, CPL. However, they may be invoked at command level as well.

Functions and their arguments are always enclosed in square brackets. For example:

```
OK, TYPE [CALC 254 * 19]
```

When the command line is processed, the function call is evaluated. The value returned by the function call then replaces the function call itself in the command line. For example, the command line shown above would evaluate to "TYPE 4826". The user does not set this intermediate stage. He sees only:

```
OK, TYPE [CALC 254 * 19]
4826
OK,
```

### Types of Command Functions

Command functions fall into four categories:

- Arithmetic Functions, which perform arithmetic operations and conversions.
- File System Functions: some of these provide information about file system objects. Others open, read, or write files.
- String-handling Functions, which manipulate or provide information about character strings.
- Miscellaneous Functions, which perform miscellaneous useful tasks, such as retrieving the current date and time, or requesting terminal input.

This chapter explains these functions in detail. The functions are organized according to their type. Under each type of function, specific functions are listed alphabetically.

## Arithmetic Functions

### ▶ [CALC expression]

The CALC function evaluates arithmetic and logical expressions. It accepts expressions containing the logical operators & (and), | (or), and ^ (not); the arithmetic operators +, −, *, /, unary +, and unary −; and the relational operators =, <, >, <=, >=, and ^=. The precedence is:

```
Highest:  ^   unary +   unary -
          /   *
          +   -
          =   ^=  <  >  <=  >=
          &
Lowest:   |
```

**Note**

All operators that are to be evaluated by CALC *must* be delimited by blanks. This restriction resolves the ambiguity that can arise from the fact that "*", "<", and ">" are also valid pathname characters.

Logical and relational operators return the Boolean values TRUE and FALSE. For example:

```
[CALC 9 > 3]
```

returns:

```
TRUE
```

Relational operators accept either numeric or non-numeric operands. If a relational operator is given a non-numeric operand, a string comparison will be done. If both operands are either numeric or Boolean, an arithmetic comparison is done. Boolean true is interpreted as "1" and false as "0".

Arithmetic operators *must* have operands that convert to integers. (Strings that convert to integers must contain only digits, except possibly for a preceding sign and leading and trailing blanks: the resulting value must be in the range −2**31+1 ... 2**31 −1.)

Arithmetic operators return a character string representation of the numeric result. Arithmetic operators apply only to integer values; CPL has no floating point arithmetic.

All the arithmetic operators have the usual definition, except for / which returns only the truncated integer part of any non-integer result. The final result is converted to a string and that string is returned as the value of CALC. For example:

```
[CALC 18 / 5]
```

returns:

```
        3
```

▶ **[HEX number]**

The HEX function converts a hexadecimal **number** to its decimal equivalent.

**number** is the hexadecimal number or letter you want converted. For example:

```
        OK, TYPE [HEX A]
        10
        OK,
```

The number 10 is returned as the decimal equivalent of A. Negative numbers are not supported.

▶ **[MOD string1 string2]**

The MOD function divides one number by another and returns the remainder (modulus).

**string1** is the dividend. **string2** is the modulus. For example:

```
        OK, TYPE [MOD 10 123]
        10
        OK, TYPE [MOD 360 23]
        15
        OK,
```

▶ **[OCTAL number]**

The OCTAL function converts an octal number to its decimal equivalent.

**number** is the octal number you want converted. For example:

```
        OK, TYPE [OCTAL 10]
        8
        OK,
```

8 is returned as the decimal equivalent of octal 10. Negative octal numbers are not supported.

▶ **[TO_HEX number]**

The TO_HEX function converts a decimal number to its hexadecimal equivalent.

**number** is the decimal number you want converted. For example:

```
OK, TYPE [TO_HEX 15]
F
OK,
```

F is returned as the hexadecimal equivalent of decimal 15.

▶ **[TO_OCTAL number]**

The TO_OCTAL function converts a decimal number to its octal equivalent.

**number** is the decimal number you want converted. For example:

```
OK, TYPE [TO_OCTAL 8]
10
OK,
```

The number 10 is returned as the octal equivalent of the decimal number 8.

## File System Functions

▶ **[ATTRIB pathname** $\left\{\begin{array}{l}\textbf{-TYPE}\\ \textbf{-DTM}\\ \textbf{-LENGTH}\end{array}\right\}$ **[-BRIEF]]**

The ATTRIB function returns information about a specified file or directory.

**pathname** is the name of the file or directory. One of the options **(-TYPE, -DTM, -LENGTH)** must be specified.

The **-DTM** option returns the date/time modified information on the file in the format produced by [DATE -FULL]. The **-LENGTH** option returns the length of the file in words.

If **-TYPE** is specified, ATTRIB returns the file system type of **pathname:**

- **ACAT**, for an access category
- **DAM**, for a direct access file
- **SAM**, for a sequential access file (such as those created by ED)
- **SEGDAM**, for a segmented direct access file
- **SEGSAM**, for a segmented sequential access file
- **UFD**, for a directory (UFD)
- **UNKNOWN** if a file is not of a recognized type

If **–BRIEF** is specified, most error messages produced by the function will be suppressed. Only error messages indicating improper invocation of the function or its arguments will be printed.

▶ **[DIR pathname [–BRIEF]]**

The DIR function returns the directory part of **pathname:** that is, all of **pathname** except its final component. For example:

```
[DIR SMITH>X>Y]
```

returns:

```
SMITH>X
```

"" is returned if pathname contains no >'s.

If **–BRIEF** is specified, most error messages produced by the function will be suppressed. Only error messages indicating improper invocation of the function or its arguments will be printed.

▶ **[ENTRYNAME pathname]**

The ENTRYNAME function returns the entryname portion of **pathname:** that is, its final component. For example:

```
[ENTRYNAME SMITH>X>Y]
```

returns:

```
Y
```

**pathname** itself is returned if it contains no >'s.

▶ **[EXISTS pathname [type] [–BRIEF]]**

The EXISTS function is a Boolean function that determines:

- Whether or not a file system object exists.
- Whether **pathname** is a specified type (file, directory, segment directory).

**pathname** is the name or pathname of a file or directory.

**type** may be omitted or may be one of the following:

- **–ANY**
- **–ACCESS_CATEGORY**

- **–FILE**
- **–DIRECTORY**
- **–SEGMENT_DIRECTORY**

The value TRUE is returned if **pathname** is found and it matches the file type specified. The value FALSE is returned if **pathname** cannot be found or does not match the file type specified. If **type** is **–ANY** or is omitted, only the existence of **pathname** is checked. For example, assume a UFD contains three files: PAYROLL.COBOL, COMPILE_ALL.CPL and PHONE_LIST. Assume also that it contains two sub-UFDs, WORKFILES and MEMOS. If you were attached to this UFD, the function call:

```
OK, TYPE [EXISTS PHONE_LIST -FILE]
```

would return:

```
TRUE
```

because PHONE_LIST is a file in the current directory. The function call:

```
[EXISTS MEMOS -SEGDIR]
```

would return:

```
FALSE
```

because MEMOS is not a segment directory.

But

```
[EXISTS MEMOS]
```

or

```
[EXISTS MEMOS -ANY]
```

returns

```
TRUE
```

If **–BRIEF** is specified, most error messages produced by the function will be suppressed. Only error messages indicating improper invocation of the function or its arguments will be printed.

## ▶ [GVPATH]

The GVPATH function returns the pathname of your active global variable file. If no global variable file is defined or active, the function returns "**–OFF**".

## ▶ [OPEN_FILE pathname status-var -MODE m]

The OPEN_FILE function opens a file for reading or writing. It returns the unit number of the file unit on which it opened the file.

**pathname** is the name or pathname of a file or directory.

**status-var** is the name of a global or local variable that is automatically set to 0 if the operation is successful and nonzero otherwise. **status-var** *must* be specified and must be a global variable if OPEN_FILE is used at command level.

**-MODE m** indicates how a file is to be opened; reading only, writing only, or reading and writing. If **mode** is omitted, the file is opened for reading.

**m** is one of the following:

| | |
|---|---|
| • R or r | *to specify reading only* |
| • W or w | *to specify writing only* |
| • **WR** or **wr** | *to specify reading and writing* |

## ▶ [PATHNAME path [-BRIEF]]

The PATHNAME function returns the full pathname (complete with volume name) of **path.** If **path** is *, PATHNAME returns the full pathname of the home directory. The directory specified by the directory part of **path** must exist, and the user's process must be able to attach to it. This is the only way the full pathname can be obtained.

For example:

```
[PATHNAME *>X>Y]
```

returns:

```
<VOLXX>USER>FILES>X>Y
```

if the home directory is <volxx>user>files.

If **-BRIEF** is specified, most error messages produced by the function will be suppressed. Only error messages indicating improper invocation of the function or its arguments will be printed.

## ▶ [READ_FILE unit status-var [-BRIEF]]

The READ_FILE function reads a line from the ASCII file opened on **unit** and returns the line as its value. The value is quoted if it contains special characters. The true null string (a string of zero length, containing no characters) is returned if end of file is reached.

**unit** is the decimal number (integer only) identifying the file to be read. It may be a variable whose value was set by the OPEN_FILE function.

**status-var** is a variable that is automatically set to 0 if the operation is successful and nonzero otherwise. If the end of file is reached, **status-var** will be set to 1.

If –**BRIEF** is specified, most error messages produced by the function will be suppressed. Only error messages indicating improper invocation of the function or its arguments will be printed.

▶ **[WILD wild-name-1 [...wild-name-n] [options] [–BRIEF]]**

The WILD function produces a list of all names within a directory that match one or more wildcard names. It has two forms, discussed below. The first form returns all matching names at once, in a single list. Names within the list are separated by blanks. The second form, which uses the –SINGLE option, returns one matching name per invocation until the list of names is exhausted.

**wild-name-1** through **wild-name-n** are wildcard names that the WILD function will match. If **wild-name-1** is a pathname, all the wild-names are looked for in the directory that **wild-name-1** specifies. Otherwise, all names are searched for in the current directory. (**wild-name-2** through **wild-name-n** may *not* be pathnames.) For example:

```
TYPE [WILD MYUFD>@.COBOL @.PMA]
```

prints a list of all COBOL and PMA source files in MYUFD.

**options** represents one or more optional control arguments. These place limits on the objects matched by the specified wildcard names. **options** are as follows:

| **Option** | **Meaning** |
|---|---|
| **–AFTER date** | Select only objects created or modified after the date specified by **date.** (This information is stored as the file's DTM, "date and time modified." Its format is mo/da/yr.) |
| **–BEFORE date** | Select only objects created or last modified before the specified date. |
| **–DIRECTORY** | Select only directories. |
| **–FILE** | Select only files. |
| **–SEGMENT_DIRECTORY** | Select only segment directories. |
| **–ACCESS_CATEGORY** | Select only access categories. |

If **–BRIEF** is specified, most error messages produced by the function will be suppressed. Only error messages indicating improper invocation of the function or its arguments will be printed.

The second form of the WILD function returns names one at a time, rather than as a list. Its form is:

**[WILD wild-names [options]–SINGLE unit-var]**

See the **CPL User's Guide** for details on using this form of the WILD function.

▶ **[WRITE_FILE unit text]**

The WRITE_FILE function writes the contents of **text** into the ASCII file opened on **file-unit. text** must be quoted if it contains special characters. One level of quoting is stripped prior to writing.

**unit** is the decimal number (integer only) identifying the file to be written.

**text** is the information you want written It is written as a single line in the file; that is, the newline is automatically added.

## String-handling Functions

▶ **[AFTER string find-string]**

The AFTER function prints all text or characters that appear after the specified text or characters.

**string** is the text or characters to be searched and **find-string** is the text or characters to be located. For example, if you had the string ABCDE and wanted to list all characters appearing after D you would say:

```
OK, TYPE [AFTER ABCDE D]
E
OK,
```

The system returns the character E as that character appears after ABCD. **string** can also be a global variable.

▶ **[BEFORE string find-string]**

The BEFORE function prints all text or characters that appear before the specified text or characters.

For example, if you had the string ABCDE and wanted to list all characters appearing before the character C you would say

```
OK, TYPE [BEFORE ABCDE C]
AB
OK,
```

The system returns the characters AB as they appear before the character C

▶ **[INDEX string find-string ]**

The INDEX function locates and prints the starting position number of a substring in a string

**string** is the text to be searched and **find-string** is the text or characters to be located in **string.** For example, if you wanted the position number of the characters DE in the string ABCDE you would say

```
OK, TYPE [INDEX ABCDE DE]
4
OK,
```

The system returns 4 as the string DE occupies the fourth and fifth positions in the string ABCDE

▶ **[LENGTH string ]**

The LENGTH function prints the number of characters in a given character string

For example

```
OK, TYPE [LENGTH This is a test]
14
OK,
```

The text line **This is a test** contains 14 characters including embedded blanks

▶ **[NULL string ]**

The NULL function tests a string for the occurrence of any text or characters and returns 'TRUE' if no text or characters exist and 'FALSE' otherwise

**string** is the text or characters to be tested

▶ **[QUOTE string1 string2. .stringn ]**

The QUOTE function places an outer pair of quotes around the text specified in **string** and places single quotes around the text within **string** The function is used to keep the meaning of special symbols from being interpreted during calls to subsystems. For example:

```
[QUOTE xy'|'z]
```

returns:

```
'xy''|''z'

[QUOTE 'abc ''de'' fg' ]
```

returns:

```
'''abc ''''de'''' fg'' '
```

▶ **[SEARCH string1 string2 ]**

The SEARCH function returns a number representing the position (beginning with 1) of the first character in **string1** that appears in **string2**. For example:

```
[SEARCH abcZdef <>Z+ ]
```

returns:

```
4
```

as Z is the fourth character in the string abcZdef If no character in **string1** appears in **string2**, SEARCH returns 0.

▶ **[SUBST string1 string2 string3 ]**

The SUBST (substitute) function replaces text in one string with text from another.

**string1** is the string to be modified. **string2** is the string characters or text in **string1** to be changed and **string3** is the text or characters replacing **string1**. **string2** and **string3** are taken whole, not interpreted character by character as in TRANSLATE (see TRANSLATE). Replacement occurs at all places **string2** is found. For example:

```
OK, TYPE [SUBST aabbaabbaa bb QR]
aaQRaaQRaa
OK,
```

All 'bb's' in **string1** were replaced with QR.

▶ **[SUBSTR string start-position [num_chars] ]**

The SUBSTR (substring) function identifies and prints a substring of specified length beginning in one specified position.

For example, if you had the string ABCDE and wanted to locate C and D you would say:

```
OK, TYPE [SUBSTR ABCDE 3 2]
CD
OK,
```

SUBSTR prints out the two characters beginning in the third position. In this case C and D are output.

If **num-chars** is omitted, SUBSTR prints all characters from the starting position to the end of the string.

▶ **[TRANSLATE string1 string2 string3]**

The TRANSLATE function replaces characters or text in one string with characters or text from another.

**string1** is the text or characters to be modified; **string2** holds the characters to replace **string1** with; and **string3** holds the characters to be removed.

TRANSLATE looks in **string1** for each character in **string3.** If it finds the character, it replaces the character with the corresponding character from **string2.** That is, if char(i) in **string1** = char(j) of **string3**, then char(i) of **string1** is replaced by char(j) of **string2.**

For example:

```
OK, TYPE [TRANSLATE abc 345 cba]
543
OK, TYPE [TRANSLATE 'a mixxpelled xtring' s x]
a misspelled string
OK,
```

If **string2** and **string3** are omitted, all text in **string1** is converted to uppercase.

▶ **[TRIM string** $\left[\begin{array}{l}\textbf{-LEFT}\\ \textbf{-RIGHT}\\ \textbf{-BOTH}\end{array}\right]$ **[char]]**

The TRIM function removes a specified character from the left, the right, or both sides of a given string.

**string** is the string of characters to be modified, and **char** is the character to be removed. For example, if you wanted to remove all B's from the beginning and end of the string:

```
BBBBABCBBB
```

you would say:

```
OK, TYPE [TRIM BBBBABCBBB -BOTH B]
ABC
OK,
```

All occurrences of B to the left and right of ABC are removed.

If neither **–LEFT**, **–RIGHT**, nor **–BOTH** is specified, **–BOTH** is assumed. If **char** is omitted, leading and/or trailing blanks are removed.

### ▶ [UNQUOTE string]

The UNQUOTE function removes the outer pair of quotes around the text specified in **string** and changes all other remaining quotes within **string** to single quotes. For example:

```
[UNQUOTE '''xx''''yy''' ]
```

returns:

```
'xx''yy'
```

### ▶ [VERIFY string1 string2]

The VERIFY function returns an integer that represents the position (beginning with 1) of the first character in **string1** that does not appear in **string2**. For example:

```
[VERIFY 1298s8 0123456789 ]
```

returns:

```
5
```

as S, the fifth character of 1298S8, does not appear in 0123456789. If all characters of **string1** appear in **string2**, VERIFY returns 0.

## Miscellaneous Functions

### ▶ [ABBREV –EXPAND text]

The ABBREV function retrieves the value of **text** from the currently active abbreviation file, and returns the expanded value as its result. If **text** is not an abbreviation, **text** itself is returned.

For example, if ZOT is an abbreviation in your abbreviation file, having the value "CLOSE –ALL; RLS –ALL", then the command:

```
TYPE [ABBREV -EXPAND ZOT]
```

would return:

```
CLOSE -ALL; RLS -ALL
OK,
```

## ▶ [CND_INFO flag]

The CND_INFO function allows a CPL condition handler to examine the condition information of the most recent condition on the stack The function returns information requested by the argument specified in **flag** **flag** must be exactly one of the following

| | |
|---|---|
| **-NAME** | Returns the name of the condition If no condition frame is on the stack, $NONE$ is returned |
| **-CONTINUE_SWITCH** | Returns the Boolean value of the continue-to-signal switch If no condition frame is on the stack FALSE is returned |
| **-RETURN_PERMIT** | Returns the Boolean value of the returned permitted switch If no condition frame is on the stack FALSE is returned |

For a complete discussion see the **Subroutines Reference Guide.**

## ▶ [DATE [option]]

The DATE function returns the current date and/or time in a variety of formats If **option** is omitted the date only is returned 82 03 17 The other possibilities are

| **Option** | **Format** |
|---|---|
| -FULL | `82-05-06.11:05:08.Thu` |
| -USA | `05/06/82` |
| -UFULL | `05/06/82.11:05:08.Thu` |
| -DAY | `6` |
| -MONTH | `May` |
| -YEAR | `1982` |
| -TIME | `11:05:08` |
| -AMPM | `11:05 AM` |
| -DOW | `Thursday` |
| -CAL | `May 6, 1982` |
| -TAG | `820506` |
| -FTAG | `820506.110508` |
| -VFULL | `06 May 82 11:05:08 Thursday` |
| -VIS | `06 May 82` |

## ▶ [GET_VAR expr]

The GET_VAR function returns the value of the variable name defined by **expr**. The string $UNDEFINED$ is returned if the variable named by **expr** has not been defined, or if no global variable file is active.

The GET_VAR function is useful for testing whether a variable has been set. If it is used at command level, only global variables may be accessed. For example:

```
OK, TYPE [GET_VAR .value_is zzz]
ZZZ
OK, TYPE [GET_VAR .no_value_yet]
$UNDEFINED$
OK,
```

## ▶ [QUERY text [default] [-TTY]]

The QUERY function prints the contents of **text** on your terminal screen and follows it with a question mark.

**text** is whatever information you want printed. If **text** is null, printing is suppressed. **text** and **default** must be quoted if they contain special characters. One level of quoting is stripped before printing. After **text** has been printed, the system awaits an answer of yes, y, ok, no, n. Upper- or lowercase characters may be used. QUERY returns TRUE if an answer of yes, y, or ok was supplied or FALSE if your answer was no or n. If you enter a carriage return, **default** becomes the value of QUERY.

If the value of **default** is not supplied, FALSE is assumed.

The **-TTY** option forces the QUERY function to take input from the terminal. If this option is omitted, the function takes its response from the command input stream. That is, if the function (or the CPL program containing it) was invoked interactively, it goes to the terminal for its response. If the function (or CPL program containing it) was invoked from a command input file, or from a &DATA group within a CPL program, it goes to the command input file or CPL program for its input.

## ▶ [RESCAN string]

The RESCAN function removes one level of quotes from **string** and evaluates any function calls or variable references no longer appearing in quotes. For example:

```
[RESCAN '[BEFORE '' [do not eval this] xxx'' x ] ' ]
'[do not eval this]'
```

returns:

```
OK,
```

The function may be used to force evaluation of quoted variable references and function calls.

▶ **[RESPONSE text [default] [-TTY]]**

The RESPONSE function prints the contents of **text** on your terminal screen and follows it with a colon.

**text** is whatever information you want printed. If **text** is null, printing is suppressed.

If **text** and **default** contain special characters, they must be enclosed in quotes. One level of quotes is stripped from **text** prior to printing.

After **text** has been printed, RESPONSE waits for user input. The input is then returned as the value of the function. If a null response is entered, the value of **default** is returned. If **default** is omitted, the null string is assumed.

The **–TTY** option forces the RESPONSE function to take input from the terminal. If this option is omitted, the function takes its response from the command input stream, whether that be the terminal, a command input file, or a &DATA group inside a CPL program.

# 4
# Command Line Features

## Introduction

This chapter summarizes various PRIMOS features that you can use in the command line. These features provide considerable flexibility in the use of commands. For instance, you can copy or delete several files by giving one command, using the wildcard feature.

The **Prime User's Guide** provides detailed tutorial information on each feature discussed in this chapter, which is a summary for quick reference, showing how to:

- Use variables at command level
- Suppress normal command processor syntax, using the ~ (tilde) character.
- Enter several commands on one line, using the semicolon.
- Iterate (repeat) commands, using parentheses.
- Use wildcards and treewalking with the @ + and ^ characters.
- Generate naming patterns for file system objects, with the = and ^ characters.

These features help you tailor your use of commands to satisfy your particular needs. You can also use the ABBREV command, discussed in Chapter 2, to create your own abbreviations for commands and command lines you often use. In addition to these features, you may want to use functions in the command line, as explained in Chapter 3.

### How PRIMOS Executes These Features

When PRIMOS encounters a user-defined abbreviation, variable, function call, or iteration list, it substitutes the value of that object for the object itself. Values for abbreviations and global variables are retrieved from the appropriate user file. Values for function calls are obtained by evaluating the function call when the command line is processed. The user supplies the values for iterated commands in parentheses when giving the command.

When PRIMOS encounters a wildcard or treewalk pattern in a command, the command processor searches the specified directory or directories for all file system objects that match the specified pattern. The command processor then creates one command for each matching object that it finds.

The command processor handles these features in the order shown in Figure 4-1.

## Using Variables at Command Level

There are four commands that govern the use of variables.

- **DEFINE_GVAR** creates or activates a file that stores global variables.
- **SET_VAR** defines individual variables and places them (with their values) inside an active global variable file.
- **LIST_VAR** displays global variables and their values.
- **DELETE_VAR** removes one or more variables from an active global variable file.

Each of these commands is explained in detail in Chapter 2. An example of their use is as follows:

1. A user decides to create a global variable file and name it VARFILE. She creates it with the command:

```
OK, DEFINE_GVAR VARFILE -CREATE
```

2. She places two global variables in the file, using the SET_VAR command:

```
OK, SET_VAR .HOME BEECH>BRANCH5>TWIG3
OK, SET_VAR .AWAY BEECH>BRANCH2>TWIG4
```

3. She checks the variables with the LIST_VAR command:

```
OK, LIST_VAR
.AWAY                                    BEECH>BRANCH2>TWIG4
.HOME                                    BEECH>BRANCH5>TWIG3
```

4. She uses the variables in a command:

```
FTN %.HOME%>FOO.FTN -B %.AWAY%>FOO.BIN
```

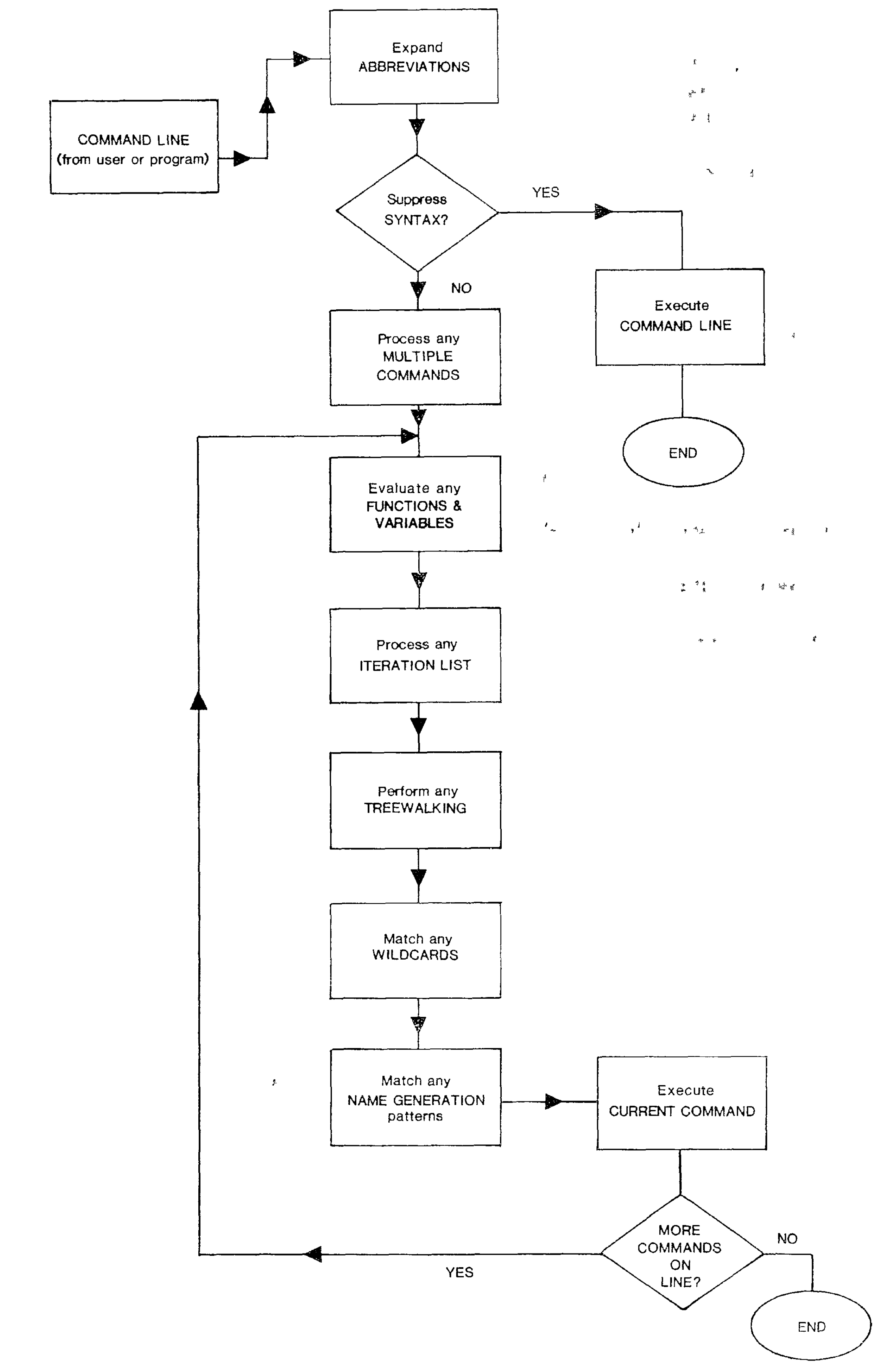


**Figure 4-1. Process Flow for Execution of a Command Line at Rev 19.0**

When the user logs out, her global variable file becomes inactive; but the variables remain within it. When the user logs in again, she can reactivate her file, use her variables, change their values, define new variables, and so on.

Global variables may be used:

- At command level
- Inside a CPL program
- Inside a user program

For further details, see the **CPL User's Guide.**

### Suppressing Normal Command Processor Syntax

You can use a tilde ( ˜ ) to force PRIMOS to ignore the following:

- Wild characters
- Name generation patterns
- Variables
- Command functions
- Semicolons

You do this by beginning the command line with a tilde. When PRIMOS encounters the tilde as the first character in the command line, it removes the tilde and interprets the rest of the line literally. This means that it does not recognize any special characters, so that it does not process features that use them.

## Multiple Commands

You can give several commands on one line if you separate each command with a semicolon. For example, the command:

```
ATTACH MYUFD; LD
```

attaches you to MYUFD and lists the contents of the directory.

### Errors in Multiple Commands

In a line containing several commands, one command may include an error. If this happens, PRIMOS still tries to execute the remaining commands on the line. For instance, should PRIMOS be unable to attach you to MYUFD in the above example, it will still list the contents of whatever directory you are currently attached to.

## Using Multiple Commands in Abbreviations

The ABBREV command interprets semicolons literally, as part of the abbreviation, rather than as a separator between ABBREV and another command. This means that you cannot give the ABBREV command followed by another command on the same line; if you do, the second command will become part of the preceding abbreviation.

Since you can include a semicolon in an abbreviation, you can create one abbreviation to execute several commands. For example:

```
ABBREV -AC CA CLOSE -ALL; RLS -ALL
```

defines one abbreviation, CA, which will execute first the CLOSE and then the RLS command.

## Iteration

You use **iteration** to specify a list of two or more objects as arguments for one command. The list must be enclosed in parentheses. Each item in the list must be separated by one or more blanks, or by commas. For example, the command:

```
DELETE (RAG TAG BOBTAIL)
```

deletes all three files specified in parentheses.

You can replace more than one argument in a command with **iteration lists.** Each list must be enclosed in parentheses, and the lists must be separated by one or more blanks, or by commas For example, the command:

```
COPY (EAR NOSE THROAT) (E N T)
```

copies files EAR, NOSE, and THROAT, renaming them E, N, and T.

Alternatively, you can use an iteration list as part of a single argument. For example, the command:

```
COPY BOOK>(CONT CHAP1 CHAP2) NEWBOOK>(FRONT SEC1 SEC2)
```

copies the three files named in the first iteration list into directory NEWBOOK, renaming them according to the second iteration list.

Further, you can use two iteration lists in one argument to create a cross-product iteration list. (You cannot use more than two iteration lists in a single argument.) For example, the command:

```
    DELETE (YOUR,MY,HIS).(MEMO,DRAFT)
```

deletes YOUR.MEMO, MY.MEMO, HIS.MEMO, YOUR.DRAFT, MY.DRAFT, and HIS.DRAFT.

## Wildcards

You use **wildcards** to specify groups of file system objects as arguments for commands. Each command that you give may include wildcards in only one argument. This may be a single wildcard name, or an iteration list containing many wildcard names. For example, the commands:

```
    LD ACCOUNTS>@.FTN
    DELETE @@(.BIN .LIST)
```

show how you can use wildcards alone or with iteration lists. The first command lists all objects in the directory ACCOUNTS that have names of the form filename.FTN. The second command deletes all the files in your current directory that are suffixed with .BIN or .LIST.

A **wildcard name** is a pathname containing wild characters. If the name is a compound pathname, the wild characters must be contained in the final element of the pathname. If the name is a simple filename, the wild characters are contained within the name. Wild characters are summarized in Table 4-1.

**Table 4-1**
**Wild Characters**

| Character | Function |
|---|---|
| @@ | Replaces any number of characters in any number of components within a file or directory name. |
| @ | Replaces any number of characters within one component of a filename or directory name. Stops matching at the period (.) that separates a name and its suffix. |
| + | Replaces a single character, except a period (.). |
| ^ | Negation character, matches all names that do not match the rest of the wildcard name. If you use the ^, it must be the first character in the wildcard name. |

## Wildcard Options

You can use **wildcard options** to:

- Select only file system objects of particular types.
- Select only file system objects last modified before or after a particular date and time.
- Enable or disable verification.

Table 4-2 lists the wildcard options.

**Table 4-2**
**Wildcard Options**

| Option | Selects |
|---|---|
| **–FILE** | SAM or DAM files |
| **–DIRECTORY** | Directories (UFDs or sub-UFDs) |
| **–SEGMENT_DIRECTORY** | SAM or DAM segment directories |
| **–ACCESS_CATEGORY** | Access categories |
| **–BEFORE date.time** | Objects last modified on or before date.time |
| **–AFTER date.time** | Objects last modified on or after date.time |
| **–VERIFY** | Causes PRIMOS to seek verification before executing a command on any object. |
| **–NO_VERIFY** | Suppresses verification requests. This is the default. |

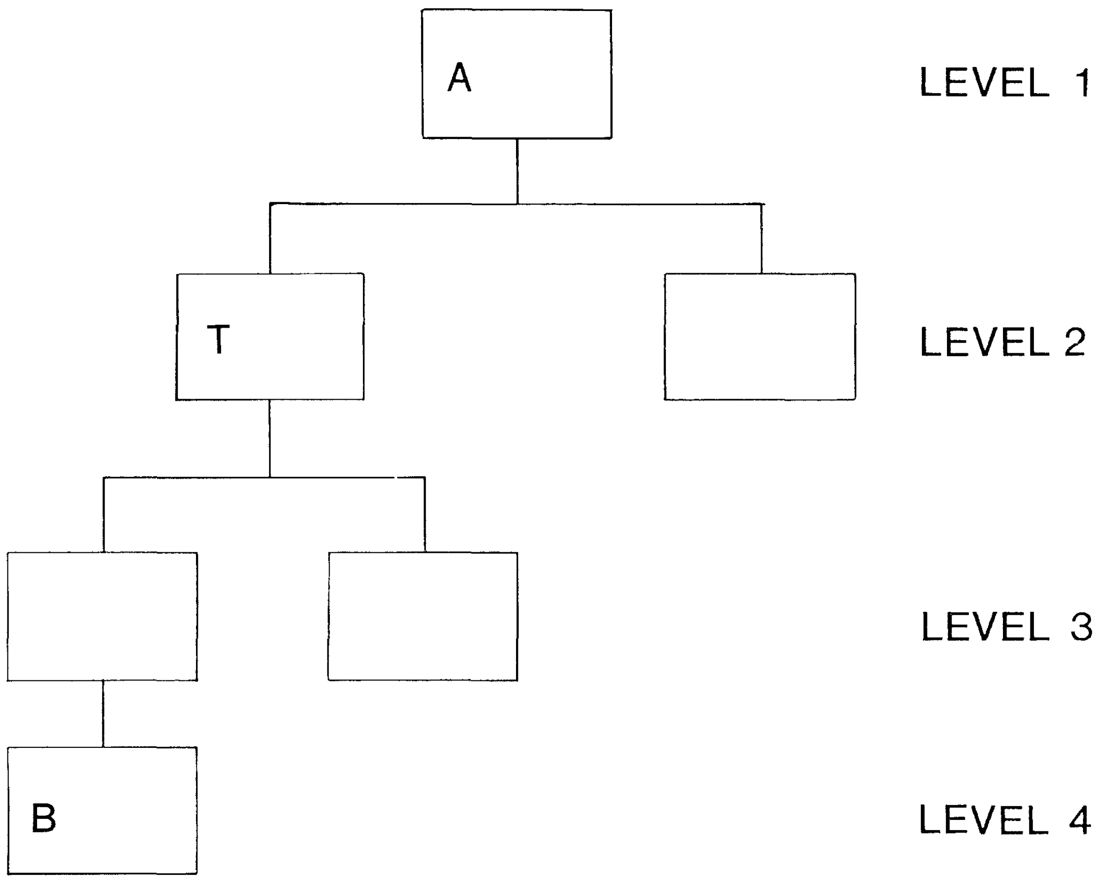
These options are explained in more detail in Chapter 18 of the **Prime User's Guide.**

## Treewalking

You use **treewalking** to make a command act on designated objects within a directory tree. (A **directory tree** includes the directory itself, its subdirectories, their subdirectories, and so on.)

You specify a **treewalking pattern** by using a wildcard name in an intermediate position within a pathname. The final position of the pathname may also contain a wildcard name. You cannot use wildcard characters in the first position of the pathname.

When you give a command containing a treewalk name, the command processor searches all directories subordinate to the specified starting directory for the file system objects whose names match the pattern in the treewalk name.



A — Attach point. If you are attached here, then....
T — Top-down treewalk starts here, either by default or with the option -WALK_FROM 2.
B — Bottom-up treewalk starts here.

**Figure 4-2. Sample Directory Tree**

## Treewalking Options

Command processor options you can use with treewalking pathnames are listed in Table 4-3.

Table 4-3
**Treewalking Options**

| Option | Meaning |
|---|---|
| **–WALK_FROM n** | Executes the command in directories at levels greater than or equal to **n**. The default is –WALK_FROM 2, which executes the command at the first directory under the starting directory. For execution in the starting directory, specify –WALK_FROM 1. |
| **–WALK_TO n** | Executes the command in directories at levels less than or equal to **n**. |
| **–BOTTOM_UP** | Executes the command in specified directories in "bottom-up" order, starting at the deepest level. By default, execution starts at the lowest level and proceeds to the higher levels. |

For example, the command:

```
LD ORCHARD>@@>@@ -WALK_FROM 1 -WALK_TO 3 -BOTTOM_UP
```

lists the contents of the directory ORCHARD, its subdirectories, and their subdirectories, starting at the level of ORCHARD's deepest subdirectory, and then proceeding upwards. Figure 4-2 shows where a bottom-up treewalk would start in a sample directory tree. The figure also illustrates how directory levels are numbered.

## Name Generation

You use **name generation** to save time when you want to specify several objectnames that are nearly identical, or to create a group of names that parallel a group of wildcard names.

Name generation requires the following:

- A source pathname, used to create new names. This is usually the first argument to the command, except for the RESUME and SEG commands. For these commands, the source pathname is the second argument.

- Generation patterns, contained in the objectname portions of one or more other pathnames. Generation patterns may consist of literal strings of characters, with some name generation symbols.

The number of components in the generated name is usually less than or equal to the number of components in the source pathname. The command processor composes the name from two sources.

- The literal strings you specify, each of which replaces a component in the source name.

- The components you specify with name generation symbols. These characters are listed in Table 4-4.

**Table 4-4**
**Name Generation Symbols**

| **Symbol** | **Meaning** |
|---|---|
| = | Copies a single component from the source name to the generated name. |
| == | Copies one or more components from the source name to the generated name. |
| ^= | Excludes a single component of the source name from the generated name. |
| ^== | Excludes one or more components of the source name from the generated name. |
| **literal-string** | Replaces a component of the source name with **literal-string.** |
| **+literal-string** | Adds the component given by **literal-string** to the generated name. |

**Note**

Only one double-equal sign, with or without the ^ symbol, may appear in any one name generation pattern.

For more detailed discussion of name generation,and examples showing how to combine various name generation symbols and literal strings, see the **Prime User's Guide**.

# A
# RVEC Parameters

## RVEC Parameters

The commands RESTOR, RESUME, SAVE, PM, and START process a group of optional parameters associated with the PRIMOS RVEC vector. These parameters are stored on disk for every runfile (executable program).

Initial values for the RVEC parameters are usually specified in the PRIMOS SAVE command, or by LOADer's or SEG's SAVE command, when the program was stored on disk.

Each parameter is a 16-bit processor word, represented by up to six octal digits.

| Parameter | Memory Location | Definition |
|---|---|---|
| **SA** | — | Starting Address (first memory word used by program) |
| **EA** | — | Ending Address (last memory word used by program) |
| **PC** | 7 | P Register (Program Counter) |
| **A** | 1 | A Register (Arithmetic) |
| **B** | 2 | B Register (Arithmetic) |
| **X** | 0 | Index Register |
| **Keys** | — | Status keys associated with INK, OTK instructions. |

The RVEC parameters are optional in the command string. Any item that is specified replaces the previous value in RVEC, which is saved with the program. Thus, for any parameters that are not specified, the value previously stored in RVEC is saved with the program.

**Slash convention:** An ordinal value followed by a slash and a value can be used to set a selected octal parameter without setting other octal parameters For example given the command format

**RESUME pathname [pc] [a] [b] [x] [keys]**

the command

```
R FILNAM 2/1000
```

sets the value of the RVEC parameter, B (i e skip two octal parameters and set the third to '1000)

**Supplying RVEC parameters:** RVEC parameters specified in RESUME or START commands replace the previous values in RVEC Also when a program returns to PRIMOS through the EXIT subroutine RVEC is loaded from the processor values in effect at the time of exit Only the SAVE command alters the values of RVEC stored on disk with the program

**RESTOR** returns a program from disk to memory and loads the SAVE parameters into RVEC in preparation for a START command

**RESUME** combines the functions of RESTOR and START

**PM** lists the current values of the RVEC parameters

External commands have RVEC parameters that can be modified at the time the command is started (e g , PMA filename 1/740) Providing RVEC parameters to a command that does not need them will cause unpredictable results

### Keys

The item **keys**, when specified among the RVEC parameters refers to the processor status keys handled by the INK and OTK instructions (Refer to the **System Architecture Reference Guide** ) These are represented by a single 16-bit word in one of the following formats (S mode and R mode programs use the first format V mode and I-mode programs use the second )

### Keys (SR)

Process status information is available in a word called the keys which can be read or set by the program Its format is as follows

| | | | | | |
|---|---|---|---|---|---|
| C | DBL | — | Mode | 0 | Bits 9 16 of location 6 |
| 1 | 2 | 3 | 4 6 | 7 8 | 9 — 16 |

| | |
|---|---|
| **C** (Bit 1) | Set by arithmetic error conditions |
| **DBL** (Bit 2) | 0 - Single Precision, 1 - Double Precision |

**MODE** (Bits 4-6) The current addressing mode as follows:

| | |
|---|---|
| 000 | 16S |
| 001 | 32S |
| 011 | 32R |
| 010 | 64R |
| 110 | 64V |
| 100 | 32I |

**C-bit (SR):** Bit 1 in the keys Set by arithmetic error conditions and shifts (Bit 1).

**Keys (VI)**

Process status information is available in a 16-bit register known as the keys. It may be referenced by the LPSW, TKA, and TAK instructions.

| C | 0 | L | MODE | F | X | LT | EQ | DEX | 0 — 0 | I | S |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4-6 | 7 | 8 | 9 | 10 | 11 | 12-14 | 15 | 16 |

**C** (Bit 1) C-bit

**L** (Bit 3) L-bit

**MODE** (Bits 4-6) Addressing Mode:

| | |
|---|---|
| 000 | 16S |
| 001 | 32S |
| 011 | 32R |
| 010 | 64R |
| 110 | 64V |
| 100 | 32I |

**F** (Bit 7) Floating point exception disable:

| | |
|---|---|
| 0 | take fault |
| 1 | set C-bit |

**X** (Bit 8) Integer exception enable:

| | |
|---|---|
| 0 | set C-bit |
| 1 | take fault |

**LT** (Bit 9)
**EQ** (Bit 10) Condition code bits:

| | |
|---|---|
| LT | set if result is negative |
| EQ | set if result is zero |

**DEX** (Bit 11) Decimal exception enable:

| | |
|---|---|
| 0 | set C-bit |
| 1 | take fault |

**I** (Bit 15) In dispatcher — set/cleared only by process exchange

**S** (Bit 16) Save done — set/cleared only by process exchange

**C-bit (VI):** Set by error conditions in arithmetic operations and by shifts.

**L-bit (VI):** Set by an arithmetic or shift operation except IRS, IRX, DRX. Equal to carry out of the most significant bit (Bit 1) of an arithmetic operation. It is valuable for simulating multiple-precision operations and for performing unsigned comparisons following a CAS or a SUB.

**Condition code bits (VI):** The two condition-code bits are designated "EQ" and "LT". EQ is set if, and only if, the result is zero. If overflow occurs, EQ reflects the state of the result after truncation rather than before. LT reflects the extended sign of the result (before truncation, if overflow), and is set if the result is negative.

# B
# DMSTK Format

The DMSTK command, explained in Chapter 2 of this guide, traces the sequence of calls and returns by which the user's process arrived at its current state. Machine states for internal commands, condition frames, and fault frames are preserved on the user's command stack. In addition, the most recent activation of a static mode program or command is preserved on the static mode stack. DMSTK can be used to display the stack dump on the terminal or into a COMOUTPUT file. As it is an internal command, it does not overwrite the static mode stack, and so does not preclude re-entry into the faulting program.

DMSTK lists each stack frame in the following general format (For an explanation of the registers and the rings involved, see the **System Architecture Reference Guide.**):

```
(nn) offset: Owner= procname (LB= ownerlb).
        Called from pcl_addr; returns to return_addr.
```

The information is as follows:

| Argument | Definition |
|---|---|
| **nn** | Frame index number of the stack frame |
| **offset** | The word number in the current stack segment where this activation's stack frame begins |
| **procname** | The name (if available) of the procedure that owns this stack frame |
| **ownerlb** | The value of the LB (linkage base) register belonging to the procedure that owns the stack frame |
| **pcl-addr** | Address of the PCL instruction that caused the procedure to be invoked |
| **return-addr** | The address to which the procedure will return |

If the frame is a fault frame, the following format is used:

```
(nn) offset: FAULT FRAME; fault type = fault type.
       Fault returns to ret_pb; LB= faulter_lb, keys= faulter_keys.
       Fault code= fcode; fault addr= faddr.
       Registers at time of fault:
            Save Mask=ssssss; XB= xb_value
            000001 000002 000003 000004 000005 000006
            000007 000010 000011 000012 000013 000014
            000015 000016 000017 000020 000021 000022
            000023 000024 000025 000026 000027 000030
```

| Argument | Definition |
|---|---|
| **fault-type** | Location in the fault table of the type of fault that occurred |
| **ret-pb** | Address to which the fault returns |
| **faulter-lb** | LB register belonging to the procedure in which the fault occurred |
| **faulter-keys** | CPU keys at the time of the fault |
| **register data** | If present, is a direct dump of the register save area (in the same format as that produced by the CPU RSAV instruction) |
| **fcode** | Fault code generated by this particular fault |
| **faddr** | Fault address generated by this particular fault |

If the activation is a condition frame, the following format is used:

```
(nn) offset: CONDITION FRAME for "condition_name"; returns to ret_pb.
       Condition raised at sigloc; LB= siglb; keys= sigkeys.
       [(Crawlout to outerpb; LB= outerlb; keys= outerkeys.)]
       [Registers at time of fault in inner ring:
            Save Mask= ssssss; XB= xb_value
            000001 000002 000003 000004 000005 000006
            000007 000010 000011 000012 000013 000014
            000015 000016 000017 000020 000021 000022
            000023 000024 000025 000026 000027 000030]
```

The latter two items are displayed only if the condition was signalled in an inner ring and subsequently a crawlout to the current ring occurred.

If, during the trace, the stack switches to a different segment, DMSTK will print, "STACK SEGMENT IS xxxx", giving the octal segment number of the new stack segment.

**Note**

A called-from or returns-to value such as 0(0)/0 or 0(0)/177776 usually means that the stack frame has an invalid return point and can never return. An example of such a frame is the first frame set up by SEG in a V-mode Static Mode program.

# C
# Old Commands

The commands in this appendix are either obsolete, or are rarely used in the detail given here.

ASRCWD is needed only by a user with a serial printer, a serial card reader, or a serial card punch.

The elaborate form of the ATTACH command became obsolete when pathnames were allowed in commands.

FUTIL has been made obsolete by the Rev. 19 COPY, DELETE, LD, and SIZE commands.

PROTEC has been made obsolete by the Rev 19 PROTECT command.

## ▶ ASRCWD number

ASRCWD sets a virtual control word which selects the input or output device for diverted terminal I/O. This command functions only on systems having the serial asynchronous interface used with serial printers, serial card readers, or serial card punches. After the command is given, input is taken from, and output sent to, the pair of devices selected by bits 11 through 16 of the octal value **number**.

| | | |
|---|---|---|
| **Input bits, xx:** | 00 | User terminal |
| | 01 | (Reserved) |
| | 10 | (Reserved) |
| | 11 | Serial card reader |
| **Output bits, yyyy:** | 0000 | User terminal |
| | 1000 | User terminal |
| | 0100 | Serial printer 1 |
| | 0010 | Serial printer 2 |
| | 0001 | Serial card punch |

For example, to choose the serial card reader for input and the user terminal for output, select the bits:

```
xx yyyy
11 0000
```

which may be regrouped as 110 000, to become octal 60. The command is:

```
        OK, ASRCWD 60
```

The virtual control word is normally set to the appropriate value by programs, and the ASRCWD command is needed only when a program exits abnormally, as by BREAK, leaving input or output diverted away from the terminal. ASRCWD 0 will correct that condition.

## ▶ ATTACH directory [password] [ldisk] [key]

The ATTACH command finds the disk file location of **directory**, checks **password** (if any), and places this information into two storage areas associated with the user. These two areas define the current directory and the home directory. They contain the name of the directory, its disk location, and a status flag which tells whether the user is an owner or a nonowner of the directory. As an option, the user may specify that the information be recorded to redefine only the current directory, leaving the home directory information alone.

It is only possible to ATTACH to file directories, not segment directories.

### Passwords

Any directory may have a pair of passwords to provide security. The owner password restricts owner-access to the files in the directory to those who know it. The nonowner password likewise restricts nonowner-access. See the PASSWD and PROTECT commands in Chapter 2 of this guide.

If a directory has both owner and nonowner passwords, a correct password — owner or nonowner — is required, and the user obtains OWNER or NONOWNER status appropriately. An incorrect or missing password results in the NO UFD ATTACHED status.

If a directory has only an owner password, then the correct password gives the user OWNER status. An incorrect password, or none, gives NONOWNER status.

If a directory has no password, OWNER status is given to an attaching user, whether he supplies a password or not.

### Logical Disk

The user may specify which logical disk is to be searched for the directory, as in the command:

**ATTACH directory password ldisk**

The logical disk, **ldisk**, is specified as an octal integer. Its values are:

| **Value** | **Meaning** |
|---|---|
| **n** | Search the MFD of logical disk **n.** The LDEV column of the STATUS DISKS printout shows the logical disk number for each disk. **n** is between 0 and the number of disks allowed. |
| **100000** | Search MFDs of all disks in order of increasing **ldisk** number. (Default) |
| **177777** | Search MFD of disk to which user is currently attached. |

The same actions may be accomplished when using pathnames:

**ATTACH < n > pathname**

**ATTACH ordinary-pathname**

**ATTACH < * > pathname**

correspond exactly to **ldisk** values of **n**, 100000, and 177777, respectively.

### Home Key Values

ATTACH ordinarily sets both the current and the home directories to the target directory. However, in the command:

**ATTACH directory password ldisk key**

the value of **key** may be chosen to allow or prevent the setting of home directory to be **directory.** The keys are:

| **Key** | **Meaning** |
|---|---|
| **177777** | Attach to a UFD in the MFD on **ldisk**; do not set as home. |
| **0** | Attach to a UFD in the MFD on **ldisk**; set as the home directory. (Default) |
| **1** | Attach to a subdirectory in the current directory; do not set as home. |
| **2** | Attach to a subdirectory in the current directory; set as the home directory. |

To specify a key without specifying **ldisk**, use the slash convention:

**ATTACH directory password 1/key**

or **ATTACH directory 1/key**

In this manner, a key may be used with a pathname:

**ATTACH pathname 1/key**

▶ **FUTIL [-NORM]**

FUTIL is a file utility whose subcommands permit the user to copy, delete, and list files and directories on a per-file or directory basis. FUTIL has an ATTACH subcommand that allows attaching to subdirectories by giving a pathname. FUTIL operates with files within User File Directories and segment directories. For complex or repetitive operations, FUTIL may be run from a command file (via the COMINPUT command) or a CPL program. The **-NORM** option causes LISTF to print sizes in 440-word records.

**Note**

FUTIL was designed for the file system as it existed before Rev. 19. It assumes that all protection is password protection and that files are labeled with prefixes rather than suffixes.

For best results with a Rev. 19 system which uses ACLs and quotas, use the Rev. 19 commands: COPY, DELETE LD, and SIZE.

## Invoking FUTIL From PRIMOS

To invoke FUTIL, input the command name FUTIL. When loaded, FUTIL prints the prompt character, >, and awaits a command string from the user terminal. To terminate long operations such as LISTF, type CONTROL-P and give the command FUTIL again.

The erase character " and the kill character ? may be used to modify the command string, as in other operating system commands.

## FUTIL Subcommands

FUTIL subcommands are described in the rest of this section in alphabetical order. Subcommand formats follow the same conventions as PRIMOS commands — abbreviations are shown in **rust**, optional arguments are in brackets, and an ellipsis (...) signifies that the previous element may be repeated.

Many FUTIL commands are significantly affected by the current value of the FROM and TO directories. For an explanation of FROM and TO directories, refer to the description of the FROM and TO commands in this appendix.

## ▶ ATTACH directory-pathname

ATTACH moves the home UFD to the directory defined by **pathname.** The pathname may contain, at most, 10 directories. The first directory in the pathname may be * (home UFD). All directories in the pathname must be UFDs or sub-UFDs. If segment directories are specified within the treename, a BAD STRUCTURE error is reported, and the home UFD is set to the last UFD that was specified in the treename before the error occurred. A subsequent ATTACH command will reset to the TO and FROM names relative to the home UFD (*), but will not affect absolute names.

## ▶ CLEAN prefix [level]

The CLEAN command is a conditional-delete command based upon a **prefix** match. If a filename begins with the characters specified as **prefix**, the file is deleted. If **level** is specified greater than 1, that many levels of sub-UFDs (including the home UFD) are scanned for **prefix** matches. In no case does CLEAN delete a UFD, sub-UFD, or a segment directory. A typical usage of CLEAN is:

```
CLEAN L_
CLEAN B_
```

to delete binary and listing files from a UFD.

## ▶ COPY filea [fileb] [,filec [filed]]...

COPY copies **filea** in the FROM directory to **fileb** in the TO directory and, optionally, **filec** in the FROM directory to **filed** in the TO directory, etc. Filename pairs must be separated by commas. If the second filename of a pair is omitted, the new file is given the same name as the old file. The files **filea**, **filec**, etc. must be SAM or DAM files and cannot be directories. Read access rights are required for **filea** and **filec**. If **fileb** exists prior to the copy, it must be a SAM or DAM file and the user must have read, write, and delete/truncate access rights to the target file (fileb in this case). If **fileb** exists, it is deleted; then **filea** is copied to **fileb**. The file type of **fileb** will be the same as **filea**.

Examples:

```
COPY FILEA
```

copies FILEA in the FROM directory to FILEA in the TO directory.

```
COPY FILEA  ,  FILEB  ,  FILEC
```

copies FILEA, FILEB, and FILEC in the FROM directory to FILEA, FILEB, and FILEC in the TO directory.

```
COPY FILEA FILEB
```

copies FILEA in FROM directory to FILEB in TO directory.

```
        COPY FILEA1 FILEA2, FILEB1 FILEB2, FILEC1 FILEC2
```

copies FILEA1, FILEB1, and FILEC1 in the FROM directory to FILEA2, FILEB2, and FILEC2 in the TO directory.

```
        COPY (0)
```

In this case, the FROM directory and TO directory must each be segment directories. COPY copies the file at position (0) of the FROM directory to position (0) of the TO directory. There are no access rights associated with these files, so PRIMOS checks instead the access rights of the directory. A user cannot set the FROM and TO directories if they are segment directories without access rights to them. No spaces are allowed in the name (0).

```
        COPY (0)   (1)
```

copies the file at position (0) of the FROM directory to position (1) of the TO directory, both of which are segment directories.

▶ **COPYDAM filea [fileb] [,filec filed]...**

The function is the same as COPY, but COPYDAM sets file type of **fileb** and **filed** to DAM, instead of copying the type of **filea** and **filec.**

▶ **COPYSAM filea [fileb] [,filec [filed]]...**

The function is the same as COPY but COPYSAM also sets file type of **fileb** and **filed** to SAM, instead of copying the type of **filea** and **filec.**

▶ **CREATE ufdname [owner-password [nonowner-password]]**

CREATE creates a UFD in the TO directory and assigns any **owner** and **nonowner passwords** specified. A UFD of the same name cannot already exist in the TO directory. If a password is not specified, it is set to six spaces (null). If a password is specified, longer than six characters, only the first six characters are used. The access rights of the new UFD are the default access rights set by PRIMOS. Refer to the CREATE command in Chapter 2.

▶ **DELETE filea [fileb]...**

The DELETE command deletes **filea** and optionally **fileb** from the FROM directory. **filea** and **fileb** cannot be directories. The user must have read, write, and delete access rights to each file specified. If **filea** and **fileb** are in a segment directory, read, write, and delete access rights are required for the FROM directory.

Examples:

```
DELETE FILEA

DELETE FILEA FILEB FILEC FILED

DELETE (1)  (2)  (185)  (186)
```

▶ **FORCE** $\left\{ \begin{array}{c} \text{ON} \\ \text{OFF} \end{array} \right\}$

As noted previously, LISTF, LISTSAVE, SCAN, UFDCPY, and TRECPY will not force read access rights on any files or subdirectories within the FROM directory. This restriction not only prevents the updating of this date-and-time stamp of files to be copied; it also permits these commands to work on write-protected disks. The price of this capability is that all files to be listed or copied must have read access. To override this restriction, the command **FORCE ON** must be specified. This command causes read access rights to be forced, but also causes LISTF, LISTSAVE, SCAN, UFDCPY, and TRECPY to fail on write-protected disks. The option remains in force until the command **FORCE OFF** is specified. UFDCPY never forces rights on the primary level of either the FROM or TO directory.

The user must have owner rights in the UFD to give the FORCE command.

▶ **FROM pathname**

where **pathname** is of the form:

> **<ldisk>**
> **<packname> directory [password-1] [ldisk] > directory [password-2]...**
> **< '>**

FROM defines the FROM directory in which files are to be searched for the commands COPY, COPYSAM, COPYDAM, DELETE, LISTF, LISTSA, SCAN, CLEAN, PROTECT, TREPRO, UFDPRO, TRECPY, TREDEL, UFDCPY, and UFDDEL. The FROM directory is defined from the **pathname** parameter. The pathname may contain up to 10 directories that can be segment directories as well as User File Directories. If segment directories are specified, the user must have read access rights to them. If any error is encountered, the FROM directory is set to the home UFD ( ). The first directory in the pathname may be ', which refers to the home UFD. The default FROM directory is the home UFD. Use of FROM never changes the home UFD. If the FROM name is a relative pathname (i.e., begins with ), any subsequent ATTACH commands that change the home UFD will reset the FROM name to .

Examples:

```
FROM  <0>BEECH
```

Set FROM directory to BEECH on logical disk 0. BEECH must be in the MFD on logical disk 0 and have a blank password:

```
FROM  BEECH  ABC
```

Search the MFD on all started disks for BEECH in logical disk order 0 to n. Set the FROM directory to the first directory encountered named BEECH. One of the passwords of BEECH must be ABC.

```
FROM  <TDISK>BEECH>SUB1>SUB2
```

Set the FROM directory to SUB2. SUB2 must be a directory in SUB1; SUB1 must be a directory in BEECH; and BEECH must be a directory in the MFD on a disk with packname TSDISK. The directories BEECH, SUB1, and SUB2 must also have a blank password.

## ▶ FROM *

Set the FROM directory to the home UFD. The home UFD is normally the last UFD the user has logged into, or attached to with either the ATTACH or FUTIL ATTACH command. If one were logged into BEECH, the above command sets the FROM directory effectively to BEECH. This command does not have to be given again if the user changes the home UFD Furthermore, this command does not have to be given at all unless the FROM directory has been made something other than the home UFD, since the home UFD is the default.

## ▶ LISTF [level] [FIRST] [LSTFIL] [PROTEC] [SIZE] [RWLOCK] [TYPE] [DATE] [PASSWDS]

Lists the FROM directory, the TO directory treename, and all files and directory trees in the FROM directory at the terminal. LISTF optionally follows each filename by its protection attributes: size in disk records (modulo 1024 words); file type; date/time modified; and, on directories, owner and nonowner passwords. The **–NORM** option causes LISTF to print sizes in 440-word records. The user must give the owner password in specifying the FROM directory. If the LSTFIL option is given, the list of files is sent to a file named LSTFIL in the home UFD instead of to the terminal. At a later time, a user may print that file on a line printer. **level** is a number specifying the lowest level in the FROM directory tree structure to be listed. The order of these options is not significant. The following list describes the output:

| **Level** | **Output** |
|---|---|
| **0** | The FROM directory name |
| **1** | The FROM directory and all files and directories within it (level 1 directories) |
| **2** | All output at level 1 and all files and directories in level 1 directories |

If the level is omitted, the default is 1.

The protection attribute of each file is printed as: Owner-Key Nonowner-Key. These keys have the following meanings:

| Octal Value | Key | Meaning |
|---|---|---|
| 0 | NIL | No access allowed |
| 1 | R | Read access only |
| 2 | W | Write access only |
| 3 | RW | Read and write access |
| 4 | D | Delete/truncate only |
| 5 | RD | Delete/truncate and read |
| 6 | WD | Delete/truncate and write |
| 7 | RWD | All access allowed |

Where the components of the key values have the following meaning:

| | |
|---|---|
| **R** | Read rights |
| **W** | Write rights |
| **D** | Delete and truncate rights |
| **NIL** | No rights |

Any combination of owner and nonowner rights is possible.

Examples:

```
O:RWD  N:NIL
```

means the owner has read, write, delete, and truncate rights; nonowners have no rights.

```
O:RWD  N:R
```

means the owner has all rights; nonowners have read rights.

The possible file types are:

| | | | |
|---|---|---|---|
| **SAM** | For SAM file | **SEGDAM** | For DAM segment directory |
| **DAM** | For DAM file | **UFD** | For User File Directory |
| **SEGSAM** | For SAM segment directory | | |

The DTM of a file or directory is printed as:

```
15:31:22 MON 08-NOV-1976
```

where 15:31:22 is 15 hours past local midnight (3 PM), 31 minutes, 22 seconds. The printed day of the week will be correct for all dates between 1 January 1972 and 31 December 2071. If the date is unreasonable (e.g., when SETIME -000000 –0000 is typed at the supervisor terminal) the DTM is not printed.

All dates are considered to be reasonable as long as the month is between 1 and 12. Note that the day of the week will be correct for dates such as 32 December and 0 April because they will be considered as 1 January and 31 March, respectively.

The passwords on subdirectories are printed as (owner, nonowner). Nonprinting characters are suppressed rather than replaced by blanks or printed on the user terminal although they will be sent to the output file (LSTFIL) (not to COMOUTPUT files, however). The default passwords on a UFD are printed as (          ,).

The FIRST option specifies that all files with names not beginning with * (the usual convention for runfiles) and B_ (the usual convention for PMA, FTN, and COBOL object files) are to have their first lines printed. If the file is not an ASCII file and the name does not begin with * or B_, the comment (NO FIRST LINE) will be printed. First lines are preceded by a colon and a space and are placed on the same line as the file and its options, if they fit.

LISTF traverses the file structure (as shown by the meandering line in Figure C-1) generating printed messages in sequence (as shown in the circles adjoining the meandering line).

If the optional parameter RWLOCK is specified, the FUTIL LISTF command also prints the value of per-file read-write lock settings for files. FUTIL does not print this information for UFD names, since it is not significant in that case. The value of the per-file read-write lock is printed as follows:

| **RWLOCK Value** | **Meaning** |
|---|---|
| **SYS** | Use the system read-write lock settings. |
| **W/NR** | Allows N readers or one writer to have the associated file open. |
| **1WNR** | Allows N readers and one writer to have the associated file open. |
| **NWNR** | Allows N readers and N writers to have the associated file open. |

When LISTF is used to produce a list of the sample file configuration shown in Figure C-1, the output level is set to 3 with the SIZE option.

▶ **LISTSAVE filename [level] [PROTEC] [SIZE] [TYPE] [DATE] [RWLOCK] [PASSWDS] [FIRST]**

The LISTSAVE command is identical in function to the LISTF command with the LSTFIL option specified, except the output listing file is named with the name specified by **filename** rather than LSTFIL, and the LSTFIL option is redundant.

▶ **PROTECT filename [owner-access [nonowner-access]]**

PROTECT will protect **filename** in the FROM directory with the owner and nonowner protection attributes specified. Refer to LISTF for a description of protection attributes. If the nonowner rights are omitted, they are set to 0. If the owner rights are omitted, they are set to 1 (read only). **filename** can be a file, a UFD, or a segment directory. If it is a UFD, the files and subdirectories within it will not be protected.

▶ **QUIT**

The QUIT command of FUTIL returns to PRIMOS command mode.

▶ **SCAN filename [level] [PROTEC] [SIZE] [TYPE] [DATE] [PASSWDS] [LSTFIL] [FIRST] [RWLOCK]**

The SCAN command is used to search the FROM directory tree for the occurrence of all files, sub-UFDs, and segment directories that are named with the name specified by **filename.**

If the level specified by the argument level is 1 (the default), only the filename followed by the information specified by the optional arguments is printed. If the level specified by **level** is greater than 1, the pathname (treename) to the file or directory, starting from the FROM directory, is printed. In addition, the information specified by any optional arguments may be printed after the treename. For example, with the sample tree structure shown in Figure C-1, the command:

```
SCAN FILEB 10 S F
```

will print the following information:

```
FROM-DIR = MFD
TO-DIR   = *

DIRECTORY PATH = MFD> UFD> SUFD12


       FILEB            1   (NO FIRST LINE)
```

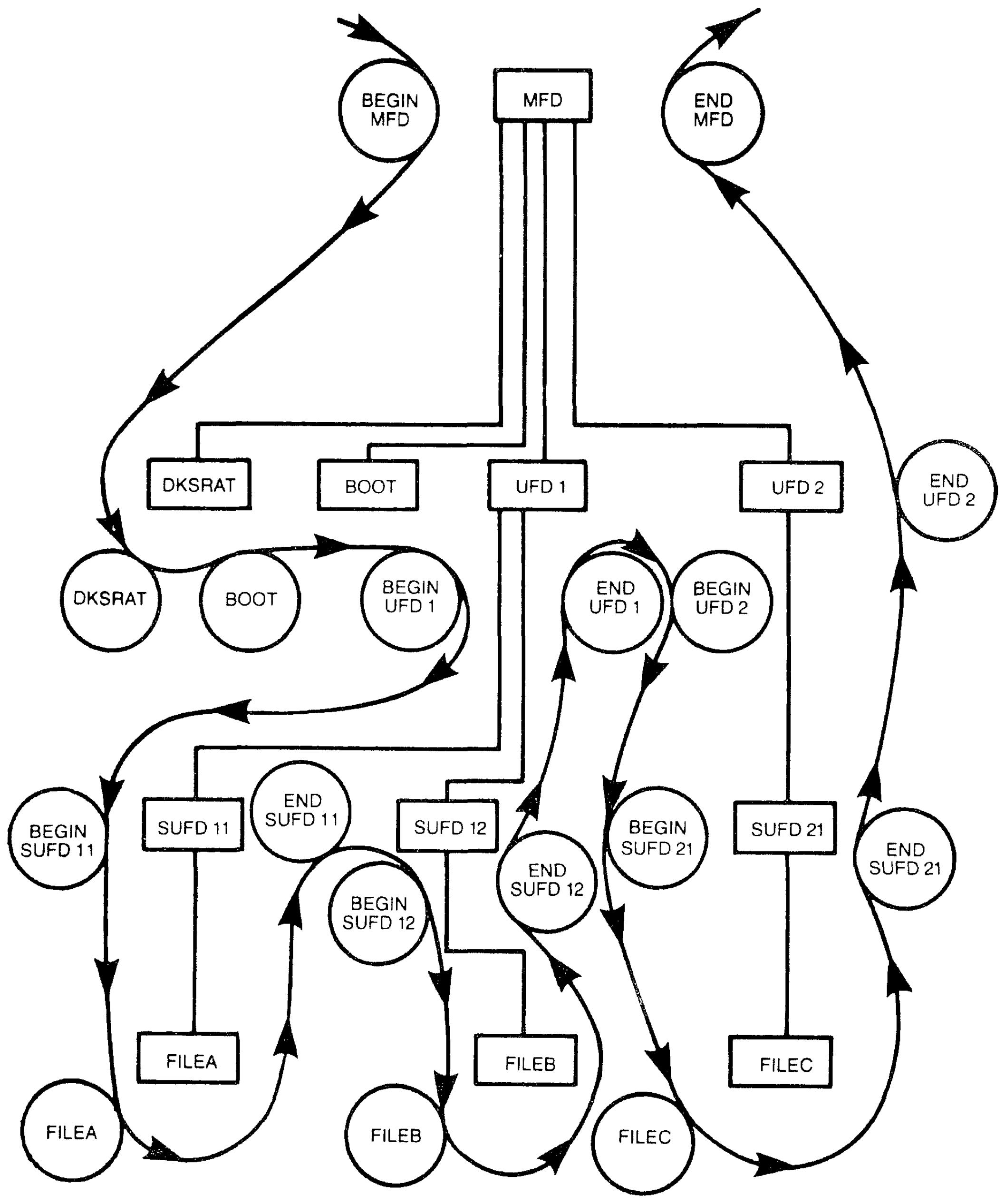


**Figure C-1. Listing File System Tree**

FILEB lacks a first line because it was empty. The name was indented three spaces because it is a sub-UFD that is a third level in a tree subordinate to the MFD.

## ▶ SRWLOC filename number

SRWLOC sets the per-file read-write lock for the file specified by **filename.** The parameter **number** is an octal number that is the read-write lock setting. The read-write lock is interpreted as modulo 4; 0 means use the system read-write lock, 1 means allow multiple readers or one writer, 2 means allow multiple readers and one writer, 3 means allow multiple readers and multiple writers.

## ▶ TO pathname

TO defines the TO directory in which files are searched for the commands CREATE, COPY, COPYSAM, COPYDAM, TRECPY, and UFDCPY. The TO directory is defined from the **pathname** parameter, which has a format similar to the directory pathname specified for the FROM command. The pathname may contain at most 10 directories that may be segment directories as well as UFDs. If segment directories are specified, the user must have read and write access to them. The first directory in the pathname may be the home UFD (*). The default TO directory is the home UFD. If any error is encountered, the TO directory is set to the home UFD (*).

**Note**

> The TO command never changes the home UFD. If the TO name is a relative pathname (i.e., begins with *), any subsequent ATTACH commands that change the home UFD will reset the TO name to *.

## ▶ TRECPY dira [dirb] [,dirc [dird]]

TRECPY is used to copy directory trees. A directory tree consists of all files and subdirectories that have their root in that directory.

The command TRECPY copies the directory tree specified by directory **dira** to directory **dirb**, and optionally **dirc** to **dird. dirb** and **dird** must not exist prior to the TRECPY command. If **dirb** is omitted, **dira** is taken as the name of the directory to be copied to. **dira** and **dirc** must be in the FROM directory; **dirb** and **dird** are created in the TO directory. Read access rights are required for **dira** and **dirc** and all files or subdirectories within them. The restriction on subdirectories can be overridden by use of the FORCE command (described in this appendix).

The directories **dirb** and **dird** are created with the same directory type and passwords as **dira** and **dirc**, and with default access rights. Also, the per-file read-write lock setting is copied by TRECPY. The names, access rights, and passwords of all files and subdirectories are also copied.

Example:

```
FROM MFD
TO   MFD
TRECPY   BEECH  LAUREL
```

copies the directory tree specified by BEECH in the MFD to a new directory, LAUREL, in the MFD.

### ▶ TREDEL dira [dirb]

The TREDEL command deletes the directory tree specified by directory **dira** and optionally deletes **dirb** from the FROM directory. **dira** and **dirb** must be directories. The user must have read, write, and delete rights to the **dira** and **dirb**; however, read, write, and delete rights are not required for files and subdirectories nested with the **dira** and **dirb.**

**Note**

The PRIMOS operating system DELETE command will not delete a directory if it is not empty.

### ▶ TREPRO directory [owner-access [nonowner-access]]

The TREPRO command is the same as PROTECT, except that directory is a UFD or segment directory in the FROM directory and it and all files under it (UFDs only) are protected with the specified access rights. The default access rights are <1 0>.

### ▶ TRESRW directory number

The TRESRW command sets the per-file read-write locks for all files in the sub-tree beginning with the directory (segment directory or UFD) specified by **directory**. The parameter **number** is the read-write lock settings, which are discussed in the description of the FUTIL command SRWLOC.

### ▶ UFDCPY

This command, UFDCPY, copies all files and directory trees from the FROM directory to the TO directory. The user must have owner rights in the FROM directory. Furthermore, all files and directories in the FROM directory must have read access rights. This restriction on subdirectories can be overridden with the FORCE command (described in this appendix).

Files already existing in the TO directory with names identical to those in the FROM directory are replaced. The user must have read, write, and delete access rights to files that are to be replaced.

Segment directories already existing in the TO directory with names identical to those in the FROM directory are not allowed and will not be copied. Files and directories created in the TO directory will have the same file type and access rights as the old files. If a file or UFD in the TO directory has the same name as a file or UFD in the FROM directory, the access rights must permit read, write, and truncate /delete. When the copy is finished, the new file will have the same protection attributes as the corresponding file in the FROM directory. The names, access rights, per-file read-write lock settings, and passwords of all files and subdirectories within directory trees being copied are also copied. Other existing files and directories in the TO directory are not affected. UFDCPY is effectively a merge of two directories including merging sub-UFDs. Both the FROM and the TO directory must be user-file directories.

Example:

```
FROM  BEECH
TO    LAUREL
UFDCPY
```

copies all files and directories from BEECH to LAUREL (BEECH and LAUREL are in the MFD). Note that, unlike the example for TRECPY, the user has not specified the MFD as the FROM directory; therefore, he does not need to know the MFD password. In the example, LAUREL exists prior to the UFDCPY. With the TRECPY example, LAUREL does not previously exist.

## ▶ UFDDEL

The UFDDEL command deletes all files and directory trees (specified by directories) within the FROM directory. The user must give the owner password in the FROM command and have read, write, and delete access to all files and directories within the FROM directory. These rights are not required for files and subdirectories nested within the directories in the FROM directory.

**Note**

Read and write access rights to a sub-UFD are sufficient to delete the contents of that directory, but not to delete the directory itself.

## ▶ UFDPRO [owner-access [nonowner-access [levels]]]

The UFDPRO command is used to protect all files and directories within the FROM directory with the specified rights, going down sub-UFD trees the specified number of **levels**. The default rights are <1 0> and the default level is 1. Thus, in the example structure of LISTF, SCAN, and CLEAN, the command: UFDPRO will protect the files DSKRAT and BOOT and the UFDs UFD1 and UFD2 with access rights <1 0> and will not change the rights of any of the subdirectory UFDs or files. The command: UFDPRO 1 0 10 will protect all files and directories within MFD. Both the owner and nonowner access rights must be specified in order to specify the number of levels.

▶ **UFDSRW number [levels]**

The UFDSRW command sets the per-file read-write locks for levels of files in the FROM directory. The parameter **number** is the read-write lock setting which is discussed in the description of the FUTIL command SRWLOC. (Default for **levels** is 1.)

## Restrictions

FUTIL cannot process UFD filenames that contain the characters () <> [] or , . Avoid using filenames containing these characters. (Segment directory filenames use parentheses around the numerical file specifier.) In using FUTIL under PRIMOS, certain operations may interfere with the work of other users. For example, a UFDCPY command to copy all files from a UFD currently used by another logged-in user may fail. If any file in that directory is open for writing by that user, UFDCPY will encounter the error "file in use", and will skip the file. If the user attempts to open one of his files for writing while UFDCPY is running, the user may encounter that error. The FUTIL LISTF and TRECPY commands cause the same interaction problems. Other FUTIL commands such as COPY and DELETE can also interfere with the other user, but the problem is not as serious because only one file is potentially involved in a conflict. To prevent the conflicts, users working together and involved in operations using each other's directory should coordinate their activities. If two users consistently use the same UFD at the same time, they should avoid the FUTIL LISTF command, and use the system LISTF command instead.

FUTIL operations when using the MFD should be done carefully. Never give the command TREDEL MFD, since the command will delete every file on the disk except the MFD, disk record availability table, BOOT, and BADSPT. A LISTF or UFDCPY of the MFD should be done only if one is sure no other user is using any files or directories on that disk. A UFDCPY of the MFD to the MFD of another disk has the effect of merging the contents of two disks onto one disk. A user should be sure there is enough room on the TO disk before attempting this operation or it will abort. Recall also that the names of segment directories on the two disks must not conflict. Files of the same name will be overwritten and UFDs of the same name will be merged. To avoid the name conflict, it may be desirable to UFDCPY the MFD of one disk into a user-file directory on another disk. Each directory originally on the FROM disk becomes a subdirectory in that UFD on the TO disk. A UFDCPY of an MFD does not copy the DSKRAT, MFD, BOOT or BADSPT to the TO directory. If a user wishes to copy BOOT to the TO directory, use the COPY command. Never copy the DSKRAT or the BADSPT file from one MFD to another.

The effect of a UFDCPY from the MFD of a disk in use to the MFD of a newly formatted disk is to reorganize the disk files so that all files are compacted, that is, have their records close to each other on the new disk. After such a compaction, the access time to existing files on the new disk is less than the access time on the old disk. Furthermore, new files tend to be compact since all free disk records are also compacted. The use of such compacted disks should improve the performance of all PRIMOS systems.

Users should not abort copying or deleting operations under PRIMOS II, but should allow them to run to completion. Aborting a copy or delete operation may cause a pointer mismatch or bad file structure or a directory with a partial entry. PRIMOS will not run correctly with a directory with a partial entry. FIXRAT should be run immediately if these conditions are encountered. (See the **System Administrator's Guide**.) Under PRIMOS, interruption of FUTIL with CONTROL-P will never result in a bad file structure.

## FUTIL Error Messages

The following are error messages generated by FUTIL. In many cases, FUTIL types error messages generated by PRIMOS and retains control, so users should be generally familiar with operating system error messages. The list given here includes those messages that may be encountered by FUTIL. Most messages are preceded by a filename identifying the file causing the error. Some of the error messages have the format:

```
reason for error
FILE =                      filename
```

In all cases except ALL FILE UNITS IN USE or DISK FULL on copies, FUTIL will continue with the operation, reporting all errors as it goes until the operation is complete.

**? CAN'T ABBREVIATE XXXXX COMMAND**

The command given is an abbreviation for a command considered too dangerous to abbreviate. Give the command in full if you wish to use it.

**? OPERATION ILLEGAL INSIDE SEGDIRS**

A PROTECT- or SWRLOC-class command was given while the FROM-dir was inside a segment directory.

**? UNKNOWN COMMAND – XXXXXX**

An unrecognizable command was given.

**ALL FILE UNITS IN USE. I NEED AT LEAST 6 (FUTIL)**

FUTIL no longer closes file units when it is invoked. Instead, it allocates its own units from any free units available. FUTIL needs between 6 and 14 free units

### ALREADY EXISTS or SEG DIR ALREADY EXISTS

An attempt has been made to TRECPY into or to CREATE a UFD or segment directory that already exists; or UFDCPY has attempted to copy a segment directory which already exists. If you intend to do the operation, the UFD or segment directory in the TO directory must first be deleted.

### file ALREADY EXISTS WITH WRONG FILE TYPE

Indicates an attempt was made to copy a file into an existing segment directory or to TRECPY a segment directory into an existing file.

### ALREADY OPEN

Indicates an attempt to UFDCPY a directory to itself, or an attempt to copy a file to itself, or an attempt to copy a directory to a subdirectory within itself.

### BAD NAME

A segment directory filename was given to a command which expected a UFD filename or vice versa. The type of filename must match the type of directory the file is contained in.

### BAD PASSWORD

An incorrect password has been given in a FROM, TO, or ATTACH COMMAND. PRIMOS will not allow FUTIL to maintain control in case of a bad password so the FUTIL command must be given to restart FUTIL after the user has attached to his directory.

Since FUTIL accepts lowercase input, it is conceivable that while FUTIL correctly interprets lowercase for commands, filenames, and options, some user may forget that all passwords are still interpreted literally. For example, a UFD with a password of ABC can only be attached with owner-rights if the password is entered literally as uppercase ABC.

### BAD STRUCTURE

Indicates any of various conditions in which the implied or explicitly specified structure is illegal. For example, an attempt to specify a UFD under a segment directory will cause this error.

### BAD SYNTAX

The command line processed by FUTIL is incorrect.

### CANNOT ATTACH

An attempt is made to UFDCPY a directory in which a sub-UFD has the same name as a file or segment directory on the TO side. The FROM side UFD is skipped.

### CANNOT ATTACH TO SEGDIR

The last directory in the directory pathname to an ATTACH command is a segment directory. It must be a UFD, as ATTACH sets the home-UFD to the last directory in the path.

### CANNOT COPY FILE TO DIRECTORY

On UFDCPY, indicates a file on the FROM side has the same name as a directory on the TO side.

### CANNOT DELETE MFD

User has given the UFDDEL command while attached to the MFD. This is not allowed.

### DISK ERROR

Same as **UNRECOVERED ERROR.**

### DISK FULL

The disk has become full before FUTIL has finished a copy operation. For operations involving many files, some files are not copied, creating only partially copied directories which may be of limited use. It is suggested that the user deletes such a structure immediately to prevent confusion as to what has been copied.

### END OF FILE

User has attempted to reference a nonexistent file beyond the end of a segment directory.

### IN USE

Indicates a FUTIL attempt to process a file in use by some other user. It may also indicate an attempt to copy a directory to a subdirectory within itself.

### INSUFFICIENT ACCESS RIGHTS

User has attempted an operation on a file which violates the file access rights assigned to that file. These rights may be changed by the PROTECT command, if the user has given the owner password on ATTACH.

### NO UFD ATTACHED

Self-explanatory. Often a result of misspelled/cased password.

### NOT A DIRECTORY

User has given a directory-treename which includes a regular file.

NOT FOUND

Self-explanatory. Often due to typographical errors.

NOT FOUND IN SEG-DIR

User has attempted to reference a file in a segment directory with an entry of 0, which indicates file does not exist or the user has attempted to reference a file past the end of the segment directory.

POINTER MISMATCH

Indicates a bad file structure. Running FIXRAT is in order.

STRUCTURE TOO DEEP

Directories may be nested to a depth of 100 levels. User has attempted to exceed this limit.

TOO MANY NAMES

A FROM, TO, or ATTACH treename was specified with more than 10 names.

UFD FULL

On a UFDCPY merge or a UFDCPY or TRECPY from a new partition to an old partition, the TO directory or a subdirectory has become full. FUTIL will report the error and then pop-up a level and continue as if the UFD had not become full.

UNRECOVERED ERROR

Indicates either an attempt to write to a write-protected disk, an actual disk error, or a FUTIL attempt to process a bad file structure. Running FIXRAT is in order if the disk was not write-protected.

WRONG FILE TYPE

An attempt was made to DELETE or COPY a directory, or to TREDEL, TRECPY, or TREPRO a file.

## ▶ PROTEC pathname key1 key2

Users (hereafter called owners) have the ability to open their files and directories to other users, giving controlled access rights to their files. This declaration of access rights can be made on a per-file basis. Access rights to a file are declared and specified through the PASSWD and PROTEC commands.

| | |
|---|---|
| **pathname** | The name of the file to be protected |
| **key1** | An integer that specifies the owner's access rights to **pathname** |
| **key2** | An integer that specifies the nonowner's access rights to **pathname** |

The values and meanings for **key1** and **key2** are:

| Value | Rights |
|---|---|
| **0** | No access of any kind allowed |
| **1** | Read only |
| **2** | Write only |
| **3** | Read and write |
| **4** | Delete and truncate |
| **5** | Delete, truncate, and read |
| **6** | Delete, truncate, and write |
| **7** | All access |

Example:

```
OK, PROTEC MYPROG  7 1
OK, PROTEC OLDIS  7 7
```

gives the owner all access rights of MYPROG, nonowners read-only access rights to MYPROG, and gives both owners and nonowners all access rights to the file OLDIS.

**Note**

The default protection keys associated with any newly created file or UFD are: 7 0. (Owner is given all rights and nonowner is given none.) However, if PROTEC is given without arguments, values are set to 0 0. (Neither owner nor nonowner has any rights.)

# D
# ASCII and EBCDIC Character Sets

**Table D-1 Prime ASCII Character Set**

| 8-Bit Octal Code | Char | | 8-Bit Octal Code | Char | 8-Bit Octal Code | Char | 8-Bit Octal Code | Char |
|---|---|---|---|---|---|---|---|---|
| 200 | NUL | | 240 | Sp | 300 | @ | 340 | ` |
| 201 | SOH | ^A | 241 | ! | 301 | A | 341 | a |
| 202 | STX | ^B | 242 | " | 302 | B | 342 | b |
| 203 | ETX | ^C | 243 | # | 303 | C | 343 | c |
| 204 | EOT | ^D | 244 | $ | 304 | D | 344 | d |
| 205 | ENQ | ^E | 245 | % | 305 | E | 345 | e |
| 206 | ACK | ^F | 246 | & | 306 | F | 346 | f |
| 207 | BEL | ^G | 247 | ' | 307 | G | 347 | g |
| 210 | BS | ^H | 250 | ( | 310 | H | 350 | h |
| 211 | HT | ^I | 251 | ) | 311 | I | 351 | i |
| 212 | LF | ^J | 252 | * | 312 | J | 352 | j |
| 213 | VT | ^K | 253 | + | 313 | K | 353 | k |
| 214 | FF | ^L | 254 | , | 314 | L | 354 | l |
| 215 | CR | ^M | 255 | - | 315 | M | 355 | m |
| 216 | SO | ^N | 256 | . | 316 | N | 356 | n |
| 217 | SI | ^O | 257 | / | 317 | O | 357 | o |
| 220 | DLE | ^P | 260 | 0 | 320 | P | 360 | p |
| 221 | DC1 | ^Q | 261 | 1 | 321 | Q | 361 | q |
| 222 | DC2 | ^R | 262 | 2 | 322 | R | 362 | r |
| 223 | DC3 | ^S | 263 | 3 | 323 | S | 363 | s |
| 224 | DC4 | ^T | 264 | 4 | 324 | T | 364 | t |
| 225 | NAK | ^U | 265 | 5 | 325 | U | 365 | u |
| 226 | SYN | ^V | 266 | 6 | 326 | V | 366 | v |
| 227 | ETB | ^W | 267 | 7 | 327 | W | 367 | w |
| 230 | CAN | ^X | 270 | 8 | 330 | X | 370 | x |
| 231 | EM | ^Y | 271 | 9 | 331 | Y | 371 | y |
| 232 | SUB | ^Z | 272 | | 332 | Z | 372 | z |
| 233 | ESC | | 273 | , | 333 | [ | 373 | { |
| 234 | FS | | 274 | < | 334 | \ | 374 | \| |
| 235 | GS | | 275 | = | 335 | ] | 375 | } |
| 236 | RS | | 276 | > | 336 | ^ | 376 | ~ |
| 237 | US | | 277 | ? | 337 | _ | 377 | DEL |

^ means CONTROL key is depressed

**Table D-2 EBCDIC Character Set**

(bank) = bank character

| Decimal | Octal | Hex. | Char. | Decimal | Octal | Hex. | Char. |
|---|---|---|---|---|---|---|---|
| 000 | 000 | 00 | NUL | 048 | 060 | 30 | |
| 001 | 001 | 01 | SOH | 049 | 061 | 31 | |
| 002 | 002 | 02 | STX | 050 | 062 | 32 | SYN |
| 003 | 003 | 03 | ETX | 051 | 063 | 33 | |
| 004 | 004 | 04 | PF | 052 | 064 | 34 | PN |
| 005 | 005 | 05 | HT | 053 | 065 | 35 | RS |
| 006 | 006 | 06 | LC | 054 | 066 | 36 | UC |
| 007 | 007 | 07 | DEL | 055 | 067 | 37 | EOT |
| 008 | 010 | 08 | | 056 | 070 | 38 | |
| 009 | 011 | 09 | | 057 | 071 | 39 | |
| 010 | 012 | 0A | SMM | 058 | 072 | 3A | |
| 011 | 013 | 0B | VT | 059 | 073 | 3B | CU3 |
| 012 | 014 | 0C | FF | 060 | 074 | 3C | DC4 |
| 013 | 015 | 0D | CR | 061 | 075 | 3D | NAK |
| 014 | 016 | 0E | SO | 062 | 076 | 3E | |
| 015 | 017 | 0F | SI | 063 | 077 | 3F | SUB |
| 016 | 020 | 10 | DLE | 064 | 100 | 40 | Sp |
| 017 | 021 | 11 | DC1 | 065 | 101 | 41 | |
| 018 | 022 | 12 | DC2 | 066 | 102 | 42 | |
| 019 | 023 | 13 | TM | 067 | 103 | 43 | |
| 020 | 024 | 14 | RES | 068 | 104 | 44 | |
| 021 | 025 | 15 | NL | 069 | 105 | 45 | |
| 022 | 026 | 16 | BS | 070 | 106 | 46 | |
| 023 | 027 | 17 | IL | 071 | 107 | 47 | |
| 024 | 030 | 18 | CAN | 072 | 110 | 48 | |
| 025 | 031 | 19 | EM | 073 | 111 | 49 | |
| 026 | 032 | 1A | CC | 074 | 112 | 4A | ¢ |
| 027 | 033 | 1B | CU1 | 075 | 113 | 4B | . |
| 028 | 034 | 1C | IFS | 076 | 114 | 4C | < |
| 029 | 035 | 1D | IGS | 077 | 115 | 4D | ( |
| 030 | 036 | 1E | IRS | 078 | 116 | 4E | + |
| 031 | 037 | 1F | IUS | 079 | 117 | 4F | \| |
| 032 | 040 | 20 | DS | 080 | 120 | 50 | & |
| 033 | 041 | 21 | SOS | 081 | 121 | 51 | |
| 034 | 042 | 22 | FS | 082 | 122 | 52 | |
| 035 | 043 | 23 | | 083 | 123 | 53 | |
| 036 | 044 | 24 | BYP | 084 | 124 | 54 | |
| 037 | 045 | 25 | LF | 085 | 125 | 55 | |
| 038 | 046 | 26 | ETB | 086 | 126 | 56 | |
| 039 | 047 | 27 | ESC | 087 | 127 | 57 | |
| 040 | 050 | 28 | | 088 | 130 | 58 | |
| 041 | 051 | 29 | | 089 | 131 | 59 | |
| 042 | 052 | 2A | SM | 090 | 132 | 5A | ! |
| 043 | 053 | 2B | CU2 | 091 | 133 | 5B | $ |
| 044 | 054 | 2C | | 092 | 134 | 5C | * |
| 045 | 055 | 2D | ENQ | 093 | 135 | 5D | ) |
| 046 | 056 | 2E | ACK | 094 | 136 | 5E | ; |
| 047 | 057 | 2F | BEL | 095 | 137 | 5F | — |

| Decimal | Octal | Hex. | Char. | Decimal | Octal | Hex. | Char. |
|---|---|---|---|---|---|---|---|
| 096 | 140 | 60 | - | 148 | 224 | 94 | m |
| 097 | 141 | 61 | / | 149 | 225 | 95 | n |
| 098 | 142 | 62 |  | 150 | 226 | 96 | o |
| 099 | 143 | 63 |  | 151 | 227 | 97 | p |
| 100 | 144 | 64 |  | 152 | 230 | 98 | q |
| 101 | 145 | 65 |  | 153 | 231 | 99 | r |
| 102 | 146 | 66 |  | 154 | 232 | 9A |  |
| 103 | 147 | 67 |  | 155 | 233 | 9B |  |
| 104 | 150 | 68 |  | 156 | 234 | 9C |  |
| 105 | 151 | 69 |  | 157 | 235 | 9D |  |
| 106 | 152 | 6A | ¦ | 158 | 236 | 9E |  |
| 107 | 153 | 6B | , | 159 | 237 | 9F |  |
| 108 | 154 | 6C | % | 160 | 240 | A0 |  |
| 109 | 155 | 6D | _ | 161 | 241 | A1 | ~ |
| 110 | 156 | 6E | > | 162 | 242 | A2 | s |
| 111 | 157 | 6F | ? | 163 | 243 | A3 | t |
| 112 | 160 | 70 |  | 164 | 244 | A4 | u |
| 113 | 161 | 71 |  | 165 | 245 | A5 | v |
| 114 | 162 | 72 |  | 166 | 246 | A6 | w |
| 115 | 163 | 73 |  | 167 | 247 | A7 | x |
| 116 | 164 | 74 |  | 168 | 250 | A8 | y |
| 117 | 165 | 75 |  | 169 | 251 | A9 | z |
| 118 | 166 | 76 |  | 170 | 252 | AA |  |
| 119 | 167 | 77 |  | 171 | 253 | AB |  |
| 120 | 170 | 78 |  | 172 | 254 | AC |  |
| 121 | 171 | 79 | ` | 173 | 255 | AD |  |
| 122 | 172 | 7A |  | 174 | 256 | AE |  |
| 123 | 173 | 7B | # | 175 | 257 | AF |  |
| 124 | 174 | 7C | @ | 176 | 260 | B0 |  |
| 125 | 175 | 7D | ' | 177 | 261 | B1 |  |
| 126 | 176 | 7E | = | 178 | 262 | B2 |  |
| 127 | 177 | 7F | " | 179 | 263 | B3 |  |
| 128 | 200 | 80 |  | 180 | 264 | B4 |  |
| 129 | 201 | 81 | a | 181 | 265 | B5 |  |
| 130 | 202 | 82 | b | 182 | 266 | B6 |  |
| 131 | 203 | 83 | c | 183 | 267 | B7 |  |
| 132 | 204 | 84 | d | 184 | 270 | B8 |  |
| 133 | 205 | 85 | e | 185 | 271 | B9 |  |
| 134 | 206 | 86 | f | 186 | 272 | BA |  |
| 135 | 207 | 87 | g | 187 | 273 | BB |  |
| 136 | 210 | 88 | h | 188 | 274 | BC |  |
| 137 | 211 | 89 | i | 189 | 275 | BD |  |
| 138 | 212 | 8A |  | 190 | 276 | BE |  |
| 139 | 213 | 8B |  | 191 | 277 | BF |  |
| 140 | 214 | 8C |  | 192 | 300 | C0 | { |
| 141 | 215 | 8D |  | 193 | 301 | C1 | A |
| 142 | 216 | 8E |  | 194 | 302 | C2 | B |
| 143 | 217 | 8F |  | 195 | 303 | C3 | C |
| 144 | 220 | 90 |  | 196 | 304 | C4 | D |
| 145 | 221 | 91 | j | 197 | 305 | C5 | E |
| 146 | 222 | 92 | k | 198 | 306 | C6 | F |
| 147 | 223 | 93 | l | 199 | 307 | C7 | G |

| Decimal | Octal | Hex. | Char. | Decimal | Octal | Hex. | Char. |
|---|---|---|---|---|---|---|---|
| 200 | 310 | C8 | H | 228 | 344 | E4 | U |
| 201 | 311 | C9 | I | 229 | 345 | E5 | V |
| 202 | 312 | CA |  | 230 | 346 | E6 | W |
| 203 | 313 | CB |  | 231 | 347 | E7 | X |
| 204 | 314 | CC | (bank) | 232 | 350 | E8 | Y |
| 205 | 315 | CD |  | 233 | 351 | E9 | Z |
| 206 | 316 | CE | (bank) | 234 | 352 | EA |  |
| 207 | 317 | CF |  | 235 | 353 | EB |  |
| 208 | 320 | D0 | } | 236 | 354 | EC | (bank) |
| 209 | 321 | D1 | J | 237 | 355 | ED |  |
| 210 | 322 | D2 | K | 238 | 356 | EE |  |
| 211 | 323 | D3 | L | 239 | 357 | EF |  |
| 212 | 324 | D4 | M | 240 | 360 | F0 | 0 |
| 213 | 325 | D5 | N | 241 | 361 | F1 | 1 |
| 214 | 326 | D6 | O | 242 | 362 | F2 | 2 |
| 215 | 327 | D7 | P | 243 | 363 | F3 | 3 |
| 216 | 330 | D8 | Q | 244 | 364 | F4 | 4 |
| 217 | 331 | D9 | R | 245 | 365 | F5 | 5 |
| 218 | 332 | DA |  | 246 | 366 | F6 | 6 |
| 219 | 333 | DB |  | 247 | 367 | F7 | 7 |
| 220 | 334 | DC |  | 248 | 370 | F8 | 8 |
| 221 | 335 | DD |  | 249 | 371 | F9 | 9 |
| 222 | 336 | DE |  | 250 | 392 | FA | \| |
| 223 | 337 | DF |  | 251 | 373 | FB |  |
| 224 | 340 | E0 | \ | 252 | 374 | FC |  |
| 225 | 341 | E1 |  | 253 | 375 | FD |  |
| 226 | 342 | E2 | S | 254 | 376 | FE |  |
| 227 | 343 | E3 | T | 255 | 377 | FF |  |

# Index

# Symbols