FlexOS™ User's Reference Guide

DIGITAL RESEARCH®

FlexOS™
User's Reference Guide

FlexOS™

User's Reference Guide

COPYRIGHT

DISCLAIMER

DIGITAL RESEARCH MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. Further, Digital Research Inc. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Digital Research Inc. to notify any person of such revision or changes.

NOTICE TO USER

From time to time changes are made in the filenames and in the files included on the distribution disk. This manual should not be construed as a representation or warranty that such files or materials and facilities exist on the distribution disk or as part of the materials and programs distributed. Most distribution disks include a "README.DOC" file. This file explains variations from the manual which do constitute modification of the manual and the items included therewith. Be sure to read this file before using the software.

TRADEMARKS

First Edition: August 1987

Order Number: 1073-2064-001

Foreword

FlexOS™ is an operating system designed for microcomputer systems. FlexOS controls the flow of data between various application programs such as word processors or spreadsheets (software), and the individual components of your computer system such as video display terminals, disk drives, and printers (hardware).

FlexOS is both a multiuser and multitasking operating system. Multiuser means that more than one person at a time can use the system. Multitasking means that you can run several programs simultaneously, moving from one to another as you need to instead of starting and stopping each program. Multitasking also means that software like a print spooler can print files unattended while you continue to edit with a word processor or perform calculations with a spreadsheet. This helps you work faster and more efficiently.

What This Guide Contains

The FlexOS User's Reference Guide (hereinafter called the User's Reference Guide describes each FlexOS command, its options, and gives examples of its use.

What You Need To Know

This book assumes you are already familiar with personal computers, including the general terminology used to describe hardware and software. You should read and be familiar with the documentation that accompanies your computer before proceeding.

The FlexOS Documentation Set

The User's Reference Guide is one of several manuals in the FlexOS documentation set. Other user-oriented documentation includes:

- FlexOS User's Guide: This book describes the FlexOS user interface, including the command line, file specifications, directories and subdirectories, wildcards, logical names, input/output redirection, batch processing, and error handling.
- FlexOS Configuration Guide: This book describes system configuration and maintenance, including how to install new hardware devices and software, and how to format a hard disk.
- FlexOS Window Manager Guide: This book describes how to use the FlexOS Window Manager provided by Digital Research®.
- Running DOS Applications under FlexOS: This book describes how to install and run certified DOS application software under FlexOS.

The following books are written for application programmers and system developers. End users do not need them to use FlexOS.

- FlexOS Programmer's Guide: This book describes how to write applications and utilities to run under FlexOS.
- FlexOS System Guide: The book describes how to modify FlexOS to run on different computer systems. Information presented in this guide includes driver and supervisor interfaces, FlexOS's driver services, and how to construct a boot loader.
- FlexOS Supplements: Microprocessor-dependent supplements to the Programmer's Guide and the FlexOS System Guide that supply information about FlexOS for a specific microprocessor.
- FlexOS Programmer's Utilities Guides: The reference to FlexOS assembly language programming tools. There is a separate utilities guide for each microprocessor supported by FlexOS.

Terms and Conventions

This book uses the terms "diskette," "hard disk," and "disk" in the following manner:

diskette	The $5^{1/4}$-inch or $3^{1/2}$-inch magnetic disks that can be inserted into and removed from your microcomputer. Your system has at least one diskette drive.
hard disk	An internal or external fixed magnetic disk.
disk	The term used when information applies to both diskettes and hard disks.

User Input and FlexOS Output

In this book, user input is in **boldface** type and FlexOS system output, including the system prompt A>, is in regular type. The following example illustrates the convention:

```
A>DIR | FIND "ACCOUNTS PAY"

ACCOUNTS PAY         1429   8-18-86   1:42P
```

In this example, user input is all uppercase letters. However, you can enter commands in uppercase, lowercase, or a combination of the two.

Contents

1 FlexOS Commands

Tables

FlexOS Commands

1.1 Command Summary

Tables 1-1 through 1-6 summarize FlexOS commands according to logically related tasks.

Table 1-1. Working with Files

Task	Command
Compare two files	COMP
Copy a file	COPY
Erase a file	ERASE
Define search order of file extensions	ORDER
Send files to print spooler for background printing	PRINT
Analyze and repair a file	RECFILE
Rename a file	RENAME
Display text	TYPE

Table 1-2. Working with Directories

Task	Command
Change the current directory	CHDIR
Display files in a directory	DIR
Make a directory	MKDIR
Set up a search path	PATH
Analyze and repair a directory	RECDIR
Remove an empty directory	RMDIR
Display subdirectories	TREE

Table 1-3. Working with Disks

Task	Command
Back up a disk	BACKUP
Display disk statistics	CHKDSK
Compare two diskettes	DISKCOMP
Copy a diskette	DISKCOPY
Partition a hard disk	FDISK
Prepare a disk	FORMAT
Restore backed up files to disk	RESTORE
Verify information written to disk	VERIFY
Display the volume label	VOL

Table 1-4. Data Protection

Task	Command
Write a volume and enable or disable disk protection	DISKSET
Change the protection or attributes of a file	FSET
Log off the system	LOGOFF
Log onto the system	LOGON
Change a password	PASSWORD
Display or change file access privileges	SECURITY

Table 1-5. Modifying Your User Environment

Task	Command
Reroute drive requests	ASSIGN
Create a new shell	COMMAND
Configure a serial port	CONFIG
Change the standard input and output devices	CTTY
Create a logical name	DEFINE
Exit a shell you have created	EXIT
Change the system prompt	PROMPT

Table 1-6. Additional Commands

Task	Command
Start a command that runs in the background	BACK
Run batch files	BATCH
Enable or disable control breaks	BREAK
Cancel a running program	CANCEL
Clear the terminal screen	CLS
Set or display the date	DATE
Echo batch file commands as they are run	ECHO
Search files or command output for text strings	FIND
Enable conditional execution of batch commands	FOR
Transfer control to specific command in batch file	GOTO
Enable conditional execution of batch commands	IF
List the built-in FlexOS commands	LIST
Display a file or command output one screen at a time	MORE
Suspend batch processing	PAUSE
Display information on running programs	PROCESS
Insert remarks in batch files	REM
Allow more than 10 parameters on a batch file command line	SHIFT
Sort a file or command output	SORT
Transfer operating system files to another disk	SYS
Set or display the time	TIME
Display the version of FlexOS	VER

1.2 ASSIGN

Forms Give a drive a new assignment:

ASSIGN original_drive=new_drive

Reset a drive to its original assignment:

ASSIGN
ASSIGN original_drive=
ASSIGN original_drive=original_drive

Explanation

The ASSIGN command routes requests for one drive to another drive.

Changes to drive assignments only affect the program you are running in your current window. They do not affect other programs, commands you have already entered, or other users on the system.

ASSIGN also permits applications designed to run only on drive A or B to use other drives, such as hard disk drives C or D.

Colons after the drive letters are optional.

Examples

```
A>ASSIGN A=C
```

Routes requests for drive A to drive C.

```
A>ASSIGN
```

Resets all drives to their original assignments.

```
A>ASSIGN B=
```

Resets drive B to its original assignment. After the drive is reset, the command DIR B: displays the directory of drive B.

```
A>ASSIGN A=A
```

Explicitly resets drive A to itself.

```
A>ASSIGN A=C B=C
```

Combines assignments on a single command line. This example routes requests for both drive A and drive B to drive C. You must separate the assignments with blank spaces.

```
A>ASSIGN A=C B=
```

Routes requests for drive A to drive C and resets drive B to its original assignment.

1.3 BACK

Forms BACK command
BACK batchfile

Explanation

The BACK command starts a command or batch file and runs it in the background. The background command continues to run even after you have invoked LOGOFF.

BACK invokes the command or batch file and then displays its process ID. To stop the background command or batch file, use the process ID with the PROCESS or CANCEL command.

After invoking BACK, you can run other programs from the shell from which you invoked BACK. However, the background command cannot get access to the shell from which it was started.

You can redirect a background command's input and output as described in Chapter 2 of the User's Guide. Redirecting a background command's output to a file is one way of determining whether the command ran successfully.

Examples

```
A>BACK DIR > LOGFILE.DOC
```

Writes the output of DIR to the file LOGFILE.DOC and displays the following message:

```
Process id = 0007 started Mon 08-26-86 08:05:00.00
```

```
A>BACK BATFILE
```

Runs all commands in the batch file BATFILE.BAT in the background and displays the following message:

```
Process id = 0023 started Fri 10-04-86 17:12:16.04
```

1.4 BACKUP

Forms Back up the current directory:

```
BACKUP source_drive: backup_drive:
BACKUP source_drive: backup_drive: -option
```

Back up one or more filespecs:

```
BACKUP source_drive: filespec backup_drive:
BACKUP source_drive: filespec backup_drive: -option
```

Explanation

The BACKUP command makes archival copies of one or more files from a source disk onto a backup diskette. You can then restore the backed up files to the source drive with the RESTORE command. You might typically back up your files in preparation for formatting your hard disk.

Caution: Unless you specify the -A option (see Table 1-7), BACKUP erases the backup diskette before writing files to it from the source disk.

Table 1-7 summarizes the BACKUP options. You can use these options singly or in combination.

Table 1-7. BACKUP Options

Option	Description
-A	Adds files to the backup disk without erasing files already on the disk.
-D:mm-dd-yy	Only backs up files modified on or after the specified date. The date must be specified in month, day, and year form. -D can be specified with the -A option.
-M	Only backs up files modified since the last backup.
-S	Backs up the current directory of the source disk, as well as all the subdirectories attached to the current directory.

If you do not specify a directory path, files are backed up from the current directory. If you do not specify a filespec, all the files in the directory are backed up. Files are backed up to the root directory of the backup diskette.

As BACKUP fills each diskette, it prompts you to insert another, until the directory has been completely backed up.

BACKUP displays the name of each file as it backs it up.

Examples

```
A>BACKUP C: A:
```

Backs up the files from the current directory of the source disk in drive C onto the backup disk in drive A. FlexOS displays the following message:

```
Insert the backup diskette #01 in drive A:
Warning! All files on diskette in drive A: will be erased.
Press any key when you are ready ...
```

```
A>BACKUP D: A: -A
```

Uses the -A option to add files to the backup disk. Files already on the backup disk are not erased, and the warning message is not displayed.

```
A>BACKUP C: *.TXT A:
A>BACKUP C: *.TXT A: -A -M
```

Specifies a filespec with the BACKUP command. This enables you to back up selected files rather than the entire disk. The combination of the -A and -M options backs up only those .TXT files modified since the last backup and adds them to the files already on the backup disk.

1.5 BATCH

Form BATCH filespec

Explanation

BATCH directs FlexOS to perform a nested operation. Program control flows from the batch file containing the BATCH filespec command to the named filespec.

filespec is the name of a batch file. If the batch file is not in the current directory, you must specify a directory path with filespec.

When the commands in the second filespec are finished, control returns to the first batch file and any additional commands are then run.

Note: Invoking a batchfile in the background causes a shell to be invoked, and the shell, in turn, runs the batchfile. Thus, the Process ID returned is that of the shell. Therefore, batchfiles cannot be stopped with the CANCEL command, but the shell can be stopped.

For examples of nested batch files, see the User's Guide.

Note: The BATCH command is only valid for use with batch files.

1.6 BREAK

Forms
BREAK
BREAK ON
BREAK OFF

Explanation

The BREAK command has been included with FlexOS for compatibility with other DOS operating systems.

In some operating systems the BREAK command turns the Ctrl-C function on and off. Ctrl-C stops running programs. In FlexOS, Ctrl-C always stops a running program whether BREAK is set to ON or OFF.

If you use any of the command forms shown above, either from the command line or within a batch file, FlexOS ignores the command and continues processing.

1.7 CANCEL

Forms
CANCEL process_ID
CANCEL process_name
CANCEL process_name window_number
CANCEL -F family_ID

Explanation

The CANCEL command stops a running program and removes it from the system. A program is defined as any user application or FlexOS command. You can only cancel your own programs; however, a system manager can cancel any program running on the system.

When you run a program, it is identified by a process name and three ID numbers: a process ID, a window number, and a family ID. To determine these values for a particular program, use the PROCESS command described later in this section.

Under certain circumstances the CANCEL command does not return an error message when it fails to cancel a process. To determine if the process was in fact canceled, you must issue a PROCESS VIEW command after the CANCEL command. See the description of the PROCESS command later in this section.

Note: You can also stop a running program by using Ctrl-C. However, you can only do this if the program is running in the current window. For more information on using windows, see the FlexOS Window Manager Guide.

Table 1-8 summarizes the options you can use with CANCEL.

Table 1-8. CANCEL Options

Option	Description
process_ID	Decimal number up to 10 digits long. FlexOS automatically assigns each running program a Process ID number.
process_name	Name up to eight characters long.
window_number	Decimal number up to three digits long.
-F family_ID	Decimal number up to five digits long.

Examples

```
A>CANCEL 6
```

Identifies the program to be canceled by its Process ID. This example stops a running program whose process ID is 6.

```
A>CANCEL ERASE
```

Cancels the specified program, regardless of the window in which it is running. If more than one program has the specified name, CANCEL terminates only the program most recently started. This example cancels the ERASE command.

```
A>CANCEL WPROC 3
```

Cancels the specified program in the specified window. Use this form if two or more programs have the same process name. In this example, the program WPROC is running in both Window 3 and Window 5. This command line cancels WPROC in Window 3.

```
A>CANCEL -F 3
```

Uses the -F family ID option to cancel all the programs in a process family. Process families include all the commands in a batch file or commands chained together using I/O redirection. This form cancels a process family in any window. This example cancels process family 3.

1.8 CHDIR

Forms CHDIR
CHDIR directory_path

Abbreviation CD

Explanation

CHDIR lets you change the current directory. The current directory is the directory FlexOS looks in first when you enter a filename without specifying which directory it is in.

Changing the current directory implies that you have another directory on your disk to make current. If you try to change your current directory to a directory that does not exist, you receive an error message. For information on how to create a directory, see the MKDIR command in this section.

If you do not specify a drive with the CHDIR command, the current drive is assumed.

Examples

```
A>CHDIR

A:
```

Displays the current directory path of the current drive. In this example, because the current directory is the root directory, CHDIR displays only a drive letter.

```
A>CHDIR B:/INFO
```

Changes the current directory of drive B to the subdirectory INFO. If you change your default drive to drive B, your current directory will be INFO. If you copy a file from drive A to drive B, it will be copied to INFO.

```
A>CHDIR BASEBALL/NATIONAL
```

Changes the current directory of drive A to the path root/BASEBALL/NATIONAL. (The subdirectories BASEBALL and NATIONAL must already exist or you receive an error message.) If you are changing the directory path all the way back to the root directory, you must include a slash or back slash for the root directory.

You can confirm your change by doing the following:

```
A>CD

A:BASEBALL/NATIONAL

A>CHDIR GIANTS
```

Attaches additional directory levels to your current path, provided these directories already exist and are logically related to the current directory. This example adds the subdirectory GIANTS to the current directory of drive A.

Again, CHDIR confirms your change:

```
A>CHDIR

A:BASEBALL/NATIONAL/GIANTS
```

1.9 CHKDSK

Forms
CHKDSK
CHKDSK -F
CHKDSK -V
CHKDSK filespec
CHKDSK filespec -F -V

Explanation

CHKDSK (check disk) displays statistics about disks, directories, files, and memory. You should run CHKDSK periodically for each disk to ensure the integrity of the file structures.

To prevent accidental changes to your disk, all Yes or No (Y/N) prompts from CHKDSK require you to press the Enter key after entering your response.

If you specify a filespec, CHKDSK reports the number of non-contiguous blocks. Non-contiguous blocks in a file can degrade system performance.

Table 1-9 decsribes the CHKDSK options.

Table 1-9. CHKDSK Options

Option	Description
-F	Causes CHKDSK to correct the errors it finds in the directory or File Allocation Table and write the corrections to the disk. Requires a Yes response to its prompt before it actually makes the corrections.
-V	Causes all pathnames on the specified or default drive to be displayed.

Examples

A>CHKDSK

Displays the following status report on the disk in drive A:

```
  Volume in A: is DISKETTE1          Created MAY 10, 1986 11:25

        1228800 bytes total disk space.
            512 bytes in boot area.
           7168 bytes in two FATs.
           7168 bytes (224 entries) in root directory.
        1213952 bytes total file space.
         266752 bytes in 2 hidden files.
         160256 bytes in 12 user files.
            512 bytes in 1 bad clusters.
         786432 bytes available on disk.
MEMORY:
         913776 bytes total memory.
         148970 bytes system memory.
         206806 bytes allocated memory.
         558000 bytes free memory.
```

The CHKDSK memory status report provides the following information:

total memory	Total memory available after FLEXOS.SYS has been loaded into memory.
system memory	Memory allocated by the system.
allocated memory	Memory allocated by a process.
free memory	Memory currently not in use: the Transient Program Area (TPA).

The sum of system memory, allocated memory, and free memory equals the total memory.

Note: CHKDSK reports two files that have the "hidden" attribute set. Some application programs create hidden files. Hidden files are not displayed in directory listings.

A>CHKDSK -F

Analyzes the directories and File Allocation Table on your disk and corrects any errors it finds.

If CHKDSK finds any lost allocation units (clusters) on the disk, it asks if you want to recover the lost data. If you answer yes, each chain of lost allocation units is recovered into its own file. The name of each file is FILEnnnn.CHK, where nnnn is a sequential number starting with 0000. The files are stored in the root directory of the current or specified drive. You can look at these files to see if they contain any useful information. If they do not, you can erase them.

A>CHKDSK -V

Displays a list of all files, including hidden ones, and their pathnames in addition to the memory status report.

A>CHKDSK C:*.TXT

Analyzes all TXT files in the current directory of drive C. In this case, CHKDSK's report includes the number of non-contiguous areas occupied by the specified file(s). By entering a wildcard in the filespec, many files can be analyzed at once.

Badly fragmented files (files with many non-contiguous blocks) can slow down system performance when they are accessed; the disk drive head has to move more often to find parts of the file. To determine how fragmented your files are, use CHKDSK periodically with a filespec of *.*. You can correct fragmented files by copying them to another disk.

A>CHKDSK *.* -F -V

Displays the number of non-contiguous areas occupied by the specified filespec and corrects any errors in the directory or File Allocation Table. CHKDSK only looks for the filespec in the current directory. In this example, the system responds by displaying all file pathnames with non-contiguous block information, as well as the memory status report.

```
Directory A:/
        A:/FLEXOSLDR.IMG
        A:/FLEXOS.SYS
        A:/RAMDSK.DRV
        A:/ATHD.DRV
Contains 1 non-contiguous blocks.
        A:/CONFIG.SAV
        A:/CONFIG.BAK
        A:/PRINTER.DRV
        A:/LOGON.286
Contains 2 non-contiguous blocks.
        A:/COMMAND.286
Contains 5 non-contiguous blocks.
        A:/WMEX.286
Contains 1 non-contiguous blocks.
        A:/AUTOEXEC.BAT
        A:/USER.TAB
        A:/CONFIG.BAT
Contains 1 non-contiguous blocks.
        A:/CONFIG2.BAT
Scanning FAT for lost chains.
```

```
  DISK:
    Volume NONE   Created January 1, 1980   00:00:00

     1228800 bytes total disk space.
         512 bytes in boot area.
        7168 bytes in two FATs.
        7168 bytes (224 entries) in root directory.
     1213952 bytes total file space.
      266752 bytes in 2 hidden files.
      160256 bytes in 12 user files.
         512 bytes in 1 bad clusters.
      786432 bytes available on disk.

MEMORY:
      913776 bytes total memory.
      148970 bytes system memory.
      206806 bytes allocated memory.
      558000 bytes free memory.
```

1.10 CLS

Form CLS

Explanation

CLS clears the screen. Output returns to the screen when you run your next command, application, or batch file.

1.11 COMMAND

Forms Invoke a copy of the primary shell:

```
COMMAND
COMMAND <command>
COMMAND -option <command>
```

Invoke a different shell:

```
COMMAND filespec -option
```

Explanation

The **shell** is a program that determines your user environment--what system prompt FlexOS displays, how the system interprets the command line, what command abbreviations it recognizes, and so on. The shell invoked when you start FlexOS is called the **primary** shell.

COMMAND lets you change from the primary shell to a secondary shell that can be a copy of the primary shell or different from the primary shell. The secondary shell might display a different prompt or use a set of specially defined commands. Note, however, that a secondary shell must be able to recognize the commands you are using. Some FlexOS commands are not compatible with other shells.

To return to the primary shell from a secondary shell, use the EXIT command. If you have used the -P option with COMMAND (see Table 1-10), EXIT has no effect.

Table 1-10 summarizes the options you can use with COMMAND.

Table 1-10. COMMAND Options

Option	Description
-P	Makes the secondary shell permanent in memory. You cannot EXIT from this shell. To leave this shell, you must LOGOFF or turn off your computer.
-C <command>	Runs a command on the secondary shell, then causes the secondary shell to terminate automatically when the command is finished. Substitute the command for <command> when you use this option.

Examples

```
A>COMMAND
```

Invokes a secondary shell that is a copy of your primary shell. The secondary shell has the same system prompt and recognizes the same commands as the primary shell.

```
A>COMMAND DIR
```

Accepts a command to run in the copy of the primary shell.

```
A>COMMAND -C DIR
```

Invokes a copy of the primary shell, runs the DIR command in the new shell, then automatically terminates the new shell.

You can precede -C with -P, but this has the same effect as using -C by itself. However, you cannot have -P following -C, because the -C option causes COMMAND to treat everything after the -C as input to another shell, not as input to COMMAND.

```
A>COMMAND SHELL2.EXE

%EXIT
```

Invokes the shell identified by SHELL2.EXE. The file containing your shell must be an executable file, which includes files with the extension 286, 386, 68K, COM, or EXE. In this example, the secondary shell contained in SHELL2.EXE uses a percent sign as the system prompt. Because the secondary shell is not permanent in memory, EXIT returns you to the primary shell.

You can also use one or both of the COMMAND options following the filespec designating the secondary shell.

1.12 COMP

Forms
COMP
COMP filespec_1
COMP filespec_1 filespec_2

Explanation

COMP compares the contents of two files of equal size. For example, you can use COMP after copying a file to make sure the copy is identical to the original.

The COMP command lets you compare files as follows:

- on the same or different drives
- in the same or different directories
- with the same path name or filename (if the files are on different drives)

You can use wildcards in the filespecs to compare two groups of files in which each pair of files is of equal size. COMP compares all the files matching filespec_1 with the corresponding files in filespec_2.

COMP displays the path names and filenames as each file is compared. An error message is displayed if:

- COMP cannot find a file matching filespec_1.
- filespec_1 and filespec_2 are different sizes.
- A specified directory path is invalid.

If COMP finds mismatching information in the files, it displays the offset of the mismatching bytes in hexadecimal and decimal, as follows:

```
Compare error at offset 17H (decimal 23)
File 1 = 49H (decimal 73)
File 2 = 4FH (decimal 79)
```

After 10 mismatches, COMP concludes that further comparison is unnecessary and displays:

```
10 Mismatches - ending compare
```

If the two files are identical, COMP displays:

```
Files compare ok
```

COMP continues comparing pairs of files that match the two filespecs until no more files can be found that match filespec_1. COMP then displays:

```
Compare more files (Y/N)? _
```

To compare two more files, type Y. You do not have to press Enter. You are immediately prompted for a new filespec_1 and filespec_2.

If you are done comparing files, type N to return to the system prompt. You do not have to press Enter.

Note: To compare diskettes instead of files, use the DISKCOMP command.

Examples

```
A>COMP

Enter first filespec :
GUESTS.LST

Enter second filespec or drive name :
B:

A:GUESTS  .LST  and  B:GUESTS  .LST
Files compare ok

Compare more files (Y/N)? N
```

Prompts you for filespec_1 and for filespec_2 or a drive name. This example compares GUESTS.LST on drive A with GUESTS.LST on drive B.

```
A>COMP INIT.BAT
```

Specifies filespec_1 only. COMP prompts you for filespec_2 or a drive name and then compares the files and displays the appropriate messages.

```
A>COMP ANSI.SYS B:
```

Specifies both filespecs. If you only specify a drive name for filespec_2, COMP assumes that filespec_2 has the same filename and file extension as filespec_1 and looks for it in the default directory. This example compares ANSI.SYS on drive A with ANSI.SYS on drive B.

```
A>COMP A:ORIGINAL.TXT B:ARCHIVE.TXT
```

Compares ORIGINAL.TXT on drive A with ARCHIVE.TXT on drive B.

```
A>COMP A:/DIR1 A:/DIR2
```

Compares all the files in one directory with all the files in another directory. The files are listed as they are compared:

```
A:/DIR1/FILE1.DAT  and  A:/DIR2/FILE1.DAT
Files compare ok

A:/DIR1/FILE2.DAT  and  A:/DIR2/FILE2.DAT
Files compare ok

Compare more files (Y/N)?
```

1.13 CONFIG

Form CONFIG [devname [baudrate [wordlength [parity [stopbits]]]]]

Explanation

The CONFIG command lets you specify the parameters of a serial port on your command line. FlexOS can support as many serial ports as its loaded drivers can handle.

If your system does not have a serial port, you receive an error message when you try to use CONFIG.

Table 1-11 defines the CONFIG parameters and their acceptable values.

Table 1-11. CONFIG Options

Parameter	Definition
devname	Any known serial device name or serial port driver.
baudrate	The transmission rate in bits per second. Acceptable values are 50, 75, 100, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600 and 19200.
wordlength	The number of data bits for a transmitted character. Acceptable values are 5, 6, 7, or 8.
parity	The type of parity, if any, to be used. Acceptable values are O(DD), E(VEN), or N(ONE).
stopbits	The number of stopbits added to the end of a transmitted character. Acceptable values are 0, 1, 1.5 or 2.

Note: When you specify a baudrate value, you need only specify the first two digits. You can specify 11 for a baudrate of 110, 15 for a baudrate of 150, and so on. When specifying a parity value, you need only specify the first character of the value: O for ODD, E for EVEN, and N for NONE.

Examples

```
A>CONFIG P0:
```

Displays the values assigned to the specified device as follows:

```
Device:
Baudrate:  9600
Wordlength:  7
Parity:  EVEN
# of Stopbits:  2
```

Note: In this example you specify only the device name with CONFIG.

```
A>CONFIG P0: 9600 8
```

Changes the wordlength to 8 data bits. Notice that you must specify all parameters preceding the one you want to change. You do not have to change these values, but you must include them on the command line.

```
A>CONFIG P0: 9600 8 EVEN 1.5
```

Changes the number of stopbits to 1.5.

1.14 COPY

Forms Copy a single file:

COPY source
COPY source destination

Combine files:

COPY source_1+source_2+...source_last destination

Add one or more files to the end of another file:

COPY source_1+source_2+...source_last

Explanation

The COPY command lets you:

- Copy one or more files to another disk.
- Copy files to the same disk, provided you rename the copy or specify a different directory.
- Combine files while copying.
- Add one or more files to the end of another file.
- Transfer data between any of the system devices.

Source is the filespec you want to copy; **destination** is the filespec of your copy. If you do not specify a drive with the source or destination, the current drive is assumed. If the destination is a directory path without a filename, files are copied into that directory without their names being changed. To copy a file to the same directory, you must change its name.

When you copy a file, the source file's date and time are copied to the destination file's directory information.

You can use drive names, device names, and directory paths with the source and destination files. You can use wildcards with the source files, but not in the destination. Notice that you can only use the asterisk (*) and the question mark (?) wildcards with COPY. If you use wildcards, the names of the files are listed as they are copied. The rules for wildcards are described in Section 1.4 of the User's Guide.

Files are usually copied in text (ASCII) mode. To copy or combine binary files, use the -B option. You can combine ASCII and binary files by using the -A and -B options on the same command line.

When you copy a file marked Read-Only, the copy does not have the Read-Only attribute.

Table 1-12 summarizes the options you can use with COPY.

Table 1-12. COPY Options

Option	Description
-A	Copies an ASCII (text) file. -A applies to the file preceding it and all remaining files until you specify another -A or -B. When files are being combined, -A is the default for the source and destination files.
-B	Copies a binary file. -B applies to the file preceding it and to all remaining files until you specify another -A or -B. When files are being copied, -B is the default for the source and destination.
-S	Copies hidden and system files in addition to the specified file or files. You only need to specify -S once and it can appear anywhere on the command line.
-V	Verifies that sectors are properly written on the target disk. Although -V slows down performance by forcing FlexOS to verify your file twice, it is good practice to verify an important copy with -V. You need only specify -V once; it can appear anywhere on the command line.

Examples

```
A>COPY B:MYFILE
```

Copies MYFILE.TXT from drive B to the current drive (drive A). The file is not renamed.

If you use this form of COPY on a single-drive system, COPY prompts you first to insert the diskette containing your source file, and then the diskette that is to contain the copy. Depending on the size of your computer's memory, you might be prompted to insert your diskettes more than once.

```
A>COPY B:SCORES.DAT -A
```

Copies SCORES.DAT as an ASCII file from drive B to the current drive, drive A. You might do this if you had previously combined several binary files into SCORES.DAT using the -B option. This would leave the end-of-file character in each source file intact in SCORES.DAT. When you copy with the -A option, COPY reads only to the first end-of-file character it encounters, and thus only copies the first source file contained in SCORES.DAT to drive A.

```
A>COPY A:CURRENT.TXT B:ARCHIVE.TXT
```

Copies CURRENT.TXT to drive B and renames it ARCHIVE.TXT. Because the file was renamed, the destination could have been the same drive as the source.

```
A>COPY *.* B: -V
```

Copies all the files on drive A to drive B. COPY lists the files as it copies them. The files are not renamed. The -V option verifies the copies.

Copying files in this manner is useful if the source files are fragmented, which means that parts of the file are written on non-contiguous (physically separate) areas of the disk. When a file is copied, these fragments are consolidated and stored as one block in the new file (assuming there is contiguous space available on the destination disk). To determine if your files are fragmented, use the CHKDSK command described in this section.

```
A>COPY B:/EXPENSES.DAT B:/BUDGET
```

Copies EXPENSES.DAT from the root directory of drive B to the subdirectory root/BUDGET, also on drive B. The copy is not renamed.

CAUTION: This command assumes the directory BUDGET exists on drive B. If the directory does not exist, EXPENSES.DAT is copied into a file named BUDGET in the root directory of drive B.

1.14.1 Combining Files

In combining files, COPY adds a copy of one or more source files to the end of source_1, and then copies the combined files to a destination file. If the destination filespec already exists, the new source_1 overwrites it. If the destination filespec does not exist, COPY creates it. The contents of the source files themselves are unaffected by this operation.

Example

```
A>COPY ABC.DAT+XYZ.DAT B:OLD.DAT
```

Adds a copy of XYZ.DAT to the end of ABC.DAT on the current drive. The result is copied to drive B. The destination file is named OLD.DAT.

COPY lists the source files as it combines them and then displays the message "1 File combined." to indicate that one destination file has been created.

You can also combine files from different drives, and use the -A and -B options with different files on the same command line.

1.14.2 Appending Files

In appending files, COPY adds copies of one or more source files to the end of source_1. This allows you to create one large file from several smaller, related files.

COPY adds the source files to the end of source_1 in the order you specify them. However, if COPY cannot find source_1, it adds the remaining source files to source_2. If it cannot find source_2, it adds the remaining source files to source_3, and so on. If COPY cannot find any source files, it displays an error message.

This process does change the file to which the other source files are appended.

Examples

```
A>COPY A.DAT+B:B.DAT
```

Adds a copy of B.DAT from drive B to the end of A.DAT. The message "1 file combined." means COPY found at least one source file to add to A.DAT.

```
A>COPY BASIC.DAT+*.DAT
```

The next example adds all the DAT files, except BASIC.DAT itself, to the end of BASIC.DAT. COPY displays the message "1 file combined."

```
A>COPY *.PRT+*.FUL B:
```

Creates multiple destination files. In this example, the PRT files (DAY.PRT, SWING.PRT, and GRAVEYRD.PRT) are lists of part-time employees for different work shifts, and the FUL files are lists of full-time employees. The wildcards make COPY add each FUL file to the end of the corresponding PRT file, producing PRT files on drive B that are lists of all employees on each shift. The files can then be renamed to something like DAY.EMP, etc.

As it combines, COPY displays the following message:

```
Combining -

DAY.PRT
SWING.PRT
GRAVEYRD.PRT

3 files combined.
```

1.14.3 Using Device Names with COPY

You can also use device names with COPY. FlexOS device names are included in the list of logical names described in Section 1.5 of the User's Guide.

When you copy a file to or from a device, the file is copied in ASCII (-A) mode.

Example

```
A>COPY CON SCREEN.TXT
I'm using the COPY command
to copy text from my keyboard
to a file. [Ctrl-Z]

1 file copied.
```

Copies what you enter at the keyboard into a file. The name of the source file is CON, which is another name for your console (keyboard and screen).

After each line of text you type, press Enter. When you have typed your last line of text, press Ctrl-Z to insert an end-of-file character in the file, and then press Enter. COPY puts the text in the destination file and returns you to the system prompt. In this example, the destination file is SCREEN.TXT.

1.15 CTTY

Forms CTTY
CTTY device_name

Explanation

On most computers, the console (the keyboard and the screen) is the standard input and output device. Unless you specify otherwise, input is entered at the keyboard and output goes to the screen. The CTTY command lets you define any character-oriented device that can handle both input and output as the standard input and output device. Examples of these devices include terminals and teletypes.

Entering the command:

```
CTTY CON0:
```

causes FlexOS to use the Console Resource Manager to interact with the port linked to con0:. Entering the command:

```
CTTY SER:
```

causes FlexOS to use the Miscellaneous Resource Manager to interact with the port linked to SER:.

CTTY also lets you reset the standard input and output device to its default value. If you want to change the standard input device to one device and the standard output device to another, you must use the DEFINE command.

Examples

```
A>CTTY
```

Resets the standard input and output device to its default value.

```
A>CTTY AUX
```

Defines AUX as the standard input and output device.

```
A>CTTY CON
```

Defines the console as the standard input and output device.

1.16 DATE

Forms Set the date:

```
DATE
DATE mm-dd-yy
```

Display the date:

```
DATE -D
```

Explanation

The DATE command displays, sets, or changes the date known to the system. When you create or modify a file, FlexOS records the current date in the file's directory entry.

Note: If you are on a multi-user system, only the system manager can set the date. If you are on a single-user system, you can use any form of the DATE command.

All dates must be in the form mm-dd-yy or mm/dd/yy, where:

- mm (month) is a number 1-12.
- dd (day) is a number 1-31.
- yy (year) is a number 80-99 (the 19 is assumed) or 1980-2099.

DATE does not accept invalid dates such as 2-30-86 (February 30, 1986) or 9/31/86 (September 31, 1986).

If you are a privileged user and you enter a valid date with the DATE command, FlexOS accepts it and displays the system prompt. If you enter an invalid date or separator, DATE displays an error message.

Examples

```
A>DATE

Current date is Tue  10-14-1986
Enter new date : 10-15-1986
```

Displays the current date known to the system and prompts you for a new one. You can enter any valid date as the new date. This example changes the date to October 15, 1986. To leave the date unchanged, press Enter.

```
A>DATE 5-14-1986
```

Changes the date from the command line.

```
A>DATE -D

Current date is Wed  5-14-1986
```

Displays the current date but does not prompt you for a new one. This form is useful if you want to use the DATE command in a batch file; the date is displayed and then the next batch command (if there is one) is immediately executed.

1.17 DEFINE

Forms Create a logical name:

```
DEFINE logical_name=value
DEFINE -S logical_name=value
```

List the logical names that have been created:

```
DEFINE -N
DEFINE -S -N
DEFINE -N logical_name
DEFINE -S -N logical_name
```

Delete a logical name:

```
DEFINE -D logical_name
DEFINE -D -S logical_name
```

Explanation

The DEFINE command lets you substitute a logical name for a more complex device name, directory name, filename, or other logical information.

In addition, DEFINE lets you:

- List your logical names and the names they replace.
- List any drive assignments you have changed with the ASSIGN command.
- Delete any of the logical names you created.
- Define the error message help level. (The help level is described in detail below.)

The help level you specify determines the amount of information FlexOS includes with its error messages. The higher the help level, the more detailed the error message. The help level can have a value between 1 and 4. Your computer manufacturer sets a default help level which you can override with DEFINE.

Table 1-13 summarizes the type of information you get with each help level.

Table 1-13. Error Message Help Levels

Help	Description Level
1	Displays the FlexOS function, the error source module, and the return code.
2	Identifies the command and type of error in one sentence. An example of a level 2 error message is "COPY: Write error."
3	Expands on the level 2 message and includes more specific information. An example of a level 3 message is "COPY: An error occurred writing report.txt on a:".
4	Expands on the level 3 message and often suggests a possible solution to the error. An example of a level 4 message is "COPY: An error occurred writing report.txt on a: The disk a: is full. You can erase unnecessary files to free up space."

You can create a logical name at the process or system level. A process level logical name can only be used by the particular program it was created for; it does not have any meaning to programs you may be running on other screens, commands you have already started, or programs being run by other users.

A system level logical name can be used by all the programs and users on the system. You can create a system level logical name only if you are on a single-user system, or if you are the system manager on a multi-user system.

You can only specify one logical name on a command line.

For more information on logical names, see Section 1.5 of the User's Guide.

Examples

```
A>DEFINE WP=A:/WORK/WORDPROC.EXE
```

Defines a process level logical name. In this example, the command line defines the filespec A:/WORK/WORDPROC.EXE as WP. This logical name can only be used by the program running in the current window. WP does not have any meaning to programs running in other windows.

```
A>DEFINE HELPLVL=2
```

Sets the error message help level to 2. If an error occurs, you receive a brief error message that does not include a possible solution.

```
A>DEFINE -S DATA=C:/CLIENTS/DATABASE.EXE
```

Uses the -S option to define DATA as a system level logical name that can be used by all users and programs on the system to access the file DATABASE.EXE.

A>DEFINE -N

Displays all process level logical names, their equivalent names, and any drive assignment names created with the ASSIGN command. The display takes the following form:

```
Process level logical names:

WP = A:/WORK/WORDPROC.EXE
INT = A:/PROGRAMS/BASIC/INTEREST.BAS
AVE = A:/PROGRAMS/BASIC/AVERAGE.BAS
```

A>DEFINE -S -N

Displays all system level logical names and their equivalent names, as follows:

```
System level logical names:

DATA = C:/CLIENTS/DATABASE.EXE
A: = FLPY0:
B: = FLPY1:
C: = HD1:
```

A>DEFINE -N WP

WP = A:/WORK/WORDPROC.EXE

Displays the equivalent name of the specified process level logical name, in this case WP. This form also displays any drive assignments you have created with the ASSIGN command.

A>DEFINE -S -N DATA

DATA = C:/CLIENTS/DATABASE.DAT

Displays the specified system level logical name and its equivalent name.

A>DEFINE -D WP

Deletes the process level logical name WP.

A>DEFINE -D -S DATA

Deletes the specified system level logical name.

1.18 DIR

Forms List all the files in a directory:

DIR
DIR -option

List specific files in a directory:

DIR filespec
DIR filespec -option

Explanation

The DIR command lists files in the current or specified directory. Depending on the form and options you use, you can list some or all of the files in a directory, and any hidden or system files.

Regardless of the form you use, DIR displays the name of each file, volume identification, and the amount of free space left on the disk. In each form of the DIR command entries that refer to subdirectories are identified with <DIR> in the file size column.

If you do not specify the -W (wide display) option, the listing includes the size of each file in decimal bytes and the date and time you last modified it.

If you do not specify the -H (hidden files) option, DIR does not list hidden and system files, even if they are present.

You can use wildcards with DIR.

Unless you specify a path with DIR, the command lists the files in the current directory.

If either filename or file extension is omitted, an * is assumed.

You can also use the DIR options in any combination. Table 1-14 summarizes these options.

Table 1-14. DIR Options

Option	Description
-H	Displays all hidden and system files in addition to the usual listing.
-P	Pauses after each screenful of directory listings. Press any key to display the next screenful of data.
-W	Produces a wide display of the directory listing.

Examples

A>**DIR**

Lists the current directory of the current drive. This form is equivalent to entering a DIR *.* command. The display takes the following form:

```
 Volume in drive A: is DISKETTE1
 Directory of A:

FILE1     A       10368   9-04-1986   1:19p
FILE3     A        1613   6-27-1986  12:14p
BUDGET    DAT        31   2-15-1986   8:52a
LEVEL2        <DIR>       5-09-1986  12:10p
FILE1             2288   7-02-1986   3:25p

          5 Files     141312 bytes free
```

A>**DIR -W**

The -W option produces a wide display of the current directory. Each line can contain five directory names or filenames. Use -W only if you have an 80-column display.

```
 Volume in drive A: is DISKETTE1
 Directory of A:

FILE1    TXT    FILE3    DOC    BUDGET   DAT    LEVEL2
EXAMPLE2 FIL    EXAMPLE3 FIL    GOFORIT  EXE    LEVEL3

        8 Files       141312 bytes free
```

A>**DIR B:**

Lists the files in the current directory of drive B.

A>**DIR BUDGET.DAT**

Lists the file, if any, in the current directory with the filename and file extension BUDGET.DAT.

A>**DIR B:*.DAT**

Uses a wildcard to list all the DAT files in the current directory of drive B.

A>**DIR FILE1**

Lists all entries in the current directory of the current drive with the filename FILE1, regardless of their file extension. Notice that the filename is entered without a file extension. This form is the same as **DIR FILE1.*** and lists filenames like FILE1.TXT, FILE1.BAS, and FILE1.

A>**DIR FILE1.**

Lists the directory entry FILE1 with no file extension, if present. Notice that in this form the filename is followed by a period and the file extension does not default to *.

A>DIR LEVEL2

Lists the files in the subdirectory LEVEL2 on drive A, as follows:

```
  Volume in drive A: is DISKETTE1
  Directory of A:/LEVEL2/

.            <DIR>        5-09-1986  11:45a
..           <DIR>        5-05-1986   9:00a
MYPROG   COM      2463    7-30-1986   8:55a
HOMEWORK TXT         1    9-18-1986   2:20p

           3 Files      141312 bytes free
```

Note: All subdirectory listing include a single period and a double period. The single period represents the subdirectory being listed (LEVEL2). The double period represents the directory one level above LEVEL2, called the "parent" directory. In this example, the parent directory is also the root directory.

A>DIR ..

Lists the files in the parent directory of your current directory.

1.19 DISKCOMP

Forms Compare two diskettes on a single-drive system:

DISKCOMP

Compare the diskette in the specified drive with the one in the current drive:

DISKCOMP drive_1

Compare the diskettes in two specified drives:

DISKCOMP drive_1 drive_2

Explanation

DISKCOMP compares the contents of two entire diskettes.

This command is used only for comparing diskettes, not individual files. Use the COMP command to compare individual files. If you specify a hard disk with DISKCOMP, you receive an error message.

DISKCOMP determines the number of sides and sectors per track to be compared based on the diskette in drive_1.

If you use DISKCOMP on a single-drive system, you are prompted to insert the diskettes at the appropriate time. DISKCOMP waits for you to press any key before it continues.

If an error occurs while DISKCOMP is comparing the diskettes, it issues a message indicating the track and side on which the error occurred. Then DISKCOMP continues to compare the rest of the diskette.

When DISKCOMP is finished and if it has found no errors, it prompts:

```
Diskettes compare ok

Compare more diskettes (Y/N)?_
```

If you type Y, DISKCOMP prompts you to insert another pair of diskettes. It compares these diskettes on the drives you originally specified.

If you do not want to compare any more diskettes, type N.

Examples

```
A>DISKCOMP
```

Performs a single-drive comparison on the current drive. DISKCOMP displays the following prompt:

```
Insert first diskette in drive A:
Press any key when you are ready
```

It reads the first disk and then prompts you for the second disk:

```
Insert second diskette in drive A:
```

```
Press any key when you are ready
```

Depending on the amount of available memory, DISKCOMP might prompt you to insert your diskettes more than once.

```
A>DISKCOMP B:
```

Performs a two-drive comparison, with drive B as the first drive and the current drive (drive A) as the second drive. DISKCOMP displays the following prompt:

```
Insert first diskette in drive B:

Insert second diskette in drive A:
Press any key when you are ready
```

Note: Specifying the current drive in the command line causes DISKCOMP to perform a single-drive comparison.

```
A>DISKCOMP A: B:
```

Compares the diskette in drive A with the diskette in drive B. DISKCOMP displays the following prompt:

```
Insert first diskette in drive A:

Insert second diskette in drive B:
Press any key when you are ready
```

Note: If you specify the same drive twice, DISKCOMP performs a single-drive comparison.

1.20 DISKCOPY

Forms
DISKCOPY
DISKCOPY source
DISKCOPY source destination

Explanation

DISKCOPY copies the entire contents of one diskette to another. In the forms above, the **source** drive contains the diskette you want to copy; the **destination** drive contains the diskette you are copying to.

Caution: DISKCOPY writes over any information already on the destination diskette.

DISKCOPY only copies diskettes. If you specify a hard disk, DISKCOPY displays an error message. If you wish to copy individual files, use the COPY command.

DISKCOPY prompts you to insert the diskettes at the appropriate times and waits for you to press a key before continuing.

After copying, DISKCOPY prompts:

```
Copy another (Y/N)?_
```

If you press Y, DISKCOPY prompts you to insert another pair of diskettes. The copy is performed on the drives you originally specified.

Press N to end the command.

Examples

```
A>DISKCOPY
```

Performs a single-drive copy on the current drive. FlexOS first displays this prompt:

```
Insert source diskette in drive A:

Strike any key when ready
```

After reading from the source diskette, FlexOS displays this prompt:

```
Insert target diskette in drive A:

Strike any key when ready
```

Depending on the size of your computer's memory, FlexOS may prompt you to insert your diskettes more than once.

```
A>DISKCOPY B:
```

Copies the diskette in drive B to the current drive, drive A. When you specify only the source drive, DISKCOPY uses the current drive as the destination drive. If you specify the current drive as the source drive, a single-drive copy takes place. DISKCOPY displays the following prompt:

```
Insert source diskette in drive B:

Insert target diskette in drive A:
```

A>DISKCOPY A: B:

Copies the diskette in drive A to the diskette in drive B. If you specify the same drive for the source and destination, DISKCOPY performs a single-drive copy. DISKCOPY displays the following prompt:

```
Insert source diskette in drive A:

Insert target diskette in drive B:
```

1.21 DISKSET

Forms Protect and/or label the current disk:

```
DISKSET
DISKSET -P=ON|OFF
DISKSET -L=volume_label
DISKSET -M=ON|OFF
DISKSET -P=ON|OFF -L=volume_label -M=ON|OFF
```

Protect and/or label a specified disk:

```
DISKSET drive:
DISKSET drive: -P=ON|OFF
DISKSET drive: -L=volume_label
DISKSET drive: -M=ON|OFF
DISKSET drive: -P=ON|OFF -L=volume_label -M=ON|OFF
```

Explanation

The DISKSET command lets you enable or disable the protection on a disk, and it also lets you label a disk. Disk protection cannot be enabled on a disk that does not have a label.

When protection is enabled, the options you specified with the FSET command determine who may access a particular file on your disk and how they may work with it. In addition, your User ID and Group ID are written on the disk label. Your User ID and Group ID determine which users have owner, group, and world privileges when working with your files.

Caution: When protection is disabled, all users can read, write, delete, or execute the files on your disk, regardless of the restrictions you specified with FSET.

You can use DISKSET on either a single- or multi-user system. On a multi-user system, the system manager or anyone whose User and Group IDs match your own can modify the protection of your disk.

The -P value is recorded on the disk and does not need to be set again unless you want to change it. To change the protection, you must be the only person using that disk when you invoke DISKSET. If anyone else is working with files on the disk, DISKSET refuses to operate.

The label is also recorded on the disk. The label can be changed only by the person who set it initially or the system manager.

Examples

```
A>DISKSET
```

Shows the volume label of the disk and whether protection is on or off.

```
A>DISKSET -P=ON
```

Enables protection on the current disk, which must have a volume label. When you enable protection, access to the files on your disk is based on the options you specified with the FSET command.

When you disable protection (-P=OFF), all users on your system can read, write, delete, or execute any file on your disk.

When you enable mixed mode media (-M=ON), both upper and lowercase characters can be used in filenames. When you disable mixed mode media (-M=OFF), all filenames are created in upper case only.

```
A>DISKSET -L=DISKETTE1
```

Writes a volume label of 11 characters or fewer on the disk. The volume label can contain only valid filename characters and cannot contain any blank spaces. When you create or change your volume label, your User ID and Group ID are written on the disk label. Only you or a privileged user can create or change your volume label.

```
A>DISKSET B: -P=OFF -L=EVERYBODY -M=ON
```

Disables the protection on the diskette in drive B, gives it the volume label EVERYBODY, and turns on mixed mode filenames.

1.22 ECHO

Forms
ECHO
ECHO ON
ECHO OFF
ECHO message

Explanation

A batch file is a collection of command lines. The ECHO command determines whether these commands lines are displayed as the batch file runs. Any output and messages produced by a command are always displayed, whether ECHO is on or off.

Examples

```
A>ECHO

ECHO is ON
```

Displays whether ECHO is on or off.

The command ECHO ON displays the command line of each command as it runs.

When you use ECHO OFF, the command lines in the batch file are not displayed as the file runs. In addition, any remarks in the file made with the REM command are not displayed. Instead, the cursor sits suspended below the command line and no system prompt appears. FlexOS waits for you to enter another command.

The last form consists of the ECHO command followed by a message. This message is always displayed when you run your batch file, whether ECHO is on or off. The message is a string of ASCII characters up to 127 characters long. The following batch file, saved as ECHOTEST.BAT, illustrates this form:

```
ECHO OFF
TIME -D
ECHO Although ECHO is OFF, this message
ECHO is displayed.
ECHO ON
REM  ECHO is ON.  Now remarks are displayed
REM  as well.
```

When you run ECHOTEST, the following output is displayed:

```
A>ECHOTEST

A>ECHO OFF

Current time is 09:41:22.15

Although ECHO is OFF, this message
is displayed.

REM  ECHO is ON.  Now remarks are displayed
REM  as well.
```

1.23 ERASE

Forms ERASE filespec
ERASE filespec -Q

Abbreviation DEL

Explanation

ERASE deletes the specified filespec. You can also use the alternate form DEL. ERASE and DEL function the same way.

If you enter ERASE by itself, you receive an error message. If you do not specify a drive with the ERASE command, ERASE deletes the file from the current drive. If you do not specify a path, the ERASE deletes the file from the current directory.

If you use the filespec *.* to erase all the files in a directory or on a disk, the ERASE command issues the following message to make sure you want to erase them:

```
Are you sure (Y/N)? _
```

If you want to erase all the files on the disk, type Y and press Enter; otherwise, type N and press Enter.

You cannot erase the system file and the two special directory entries (. and ..) in each subdirectory.

Caution: Be careful when you use ERASE with the wildcard characters *, ?, and ^ in the filename and file extension because you can inadvertantly erase entire directories. For example, the command:

```
A>ERASE ^*.TXT
```

on a directory that contains no files with the TXT extension has the same effect as the command

```
A>ERASE *.*
```

except that you do not receive the "Are you sure?" warning message.

For more information on wildcards, see Section 1.4 of the User's Guide.

Examples

```
A>ERASE MYFILE.1
```

Erases MYFILE.1 from the current directory of drive A.

```
A>ERASE B:*.*

Are you sure (Y/N)? Y
```

Erases all files in the current directory of drive B.

```
A>DEL B:/ACCOUNTS

Are you sure (Y/N)? Y
```

Erases all files in the subdirectory /ACCOUNTS on drive B.

```
A>ERASE *.DAT -Q
```

Queries you before erasing each DAT file in the current directory of drive A. Press Enter after you type each response. The -Q option helps you avoid erasing important files. The display looks like this:

```
FLPY0:LIST.DAT, delete (Y/N)? N
FLPY0:MARKET.DAT, delete (Y/N)? Y
FLPY0:REPORT.DAT, delete (Y/N)? Y
```

1.24 EXIT

Form EXIT

Explanation

The EXIT command returns you from a secondary shell to the primary shell. The secondary shell is invoked with COMMAND. Notice that if you used the -P option with COMMAND, EXIT will not return you to the primary shell; the secondary stays in effect until you use LOGOFF or turn off your computer.

Example

```
%>EXIT
```

Returns you to the primary shell. The secondary shell was not permanent in memory.

1.25 FDISK

Form FDISK

Explanation

The FDISK command prepares a hard disk.

Since preparing a hard disk with FDISK is an essential step in setting up the computer system, this task is described in the Configuration Guide.

1.26 FIND

Forms Search one or more files:

```
FIND "string" filespec
FIND -option "string" filespec
FIND "string" filespec_1 filespec_2...filespec_last
FIND -option "string" filespec_1 filespec_2... filespec_last
```

Search command output:

```
COMMAND | FIND "string"
COMMAND | FIND -option "string"
```

Explanation

FIND searches files or command output for a specified character string, and sends lines that include the string to the standard output device or to a file.

The string you want to search for must be enclosed in quotation marks. If the string contains a quotation mark, use double quotation marks as string delimiters.

You can specify more than one filespec on the command line. Separate filespecs with blank spaces. FIND searches for the string in each file, in the order the files appear on the command line.

Table 1-15 summarizes the options you can use with FIND.

Table 1-15. FIND Options

Option	Description
-C	Displays the matched string once, followed by total number of matches in the specified file.
-N	Displays the line number of each match, followed by the line in which the string occurs.
-V	Displays each line that does not include the string.

Examples

```
A>FIND "Twinkle, Twinkle," STAR
```

Searches the file STAR for the string "Twinkle, Twinkle," and displays each line that contains the string. The resulting display looks like this:

```
---------- STAR
Twinkle, Twinkle, little star
Twinkle, Twinkle, little star
```

```
A>FIND -C "Nevermore" /POETS/POE/RAVEN.TXT
```

Uses the -C option to count the number of times the string "Nevermore" appears in the file A:/POETS/POE/RAVEN.TXT, with this result:

```
---------- POETS/POE/RAVEN.TXT: 8
```

```
A>FIND -N "Twinkle, Twinkle," STAR
```

Displays the line number of each match:

```
---------- STAR
[1]Twinkle, Twinkle, Little star
[5]Twinkle, Twinkle, Little star
```

```
A>FIND -V "BRONTE, EMILY" AUTHORS.TXT
```

Lists the lines in AUTHORS.TXT that do not contain the string "BRONTE, EMILY".

```
A>DIR | FIND "ACCOUNTS PAY"
```

Searches the output of the DIR command for the string "ACCOUNTS PAY" and displays the following:

```
ACCOUNTS PAY      1429   1-18-1985   1:42p
```

1.27 FOR

Form FOR %c IN (item_1 ... item_n) DO command %c

Explanation

The FOR command lets you repeat another command a number of times for different files, commands, or applications. FOR can be used on the command line or in a batch file, but it is usually used in a batch file. Only one FOR command is allowed on a command line or on one line in a batch file.

The words FOR, IN, and DO are required. The **c** in the "%c" component can be any single character. You can specify two percent signs with the %c component if you wish.

item_1 ... item_n can be the names of files (including other batch files), commands, applications, or text. Separate items with one or more blank spaces. There is no limit to the length of your item list; however, the entire FOR command line must be 128 characters or fewer. You can use wildcards with the FOR command, but not path names.

The **command** at the end of the FOR command can be the name of a FlexOS command, an application, or a batch file. This command is run against each of the items in your list in left-to-right order.

Examples

```
FOR %v IN (FILE1 FILE2 FILE3) DO TYPE %v
```

Runs the TYPE command against FILE1, FILE2, and FILE3, in that order. This is the same as entering TYPE FILE1, TYPE FILE2, and TYPE FILE3 on successive command lines or on successive lines in your batch file.

Items in a FOR command line must be filenames.

1.28 FORMAT

Forms Format the specified drive:

FORMAT drive:
FORMAT drive: -option

Format the current drive:

FORMAT
FORMAT -option

Explanation

FORMAT prepares a disk in a recording format acceptable to FlexOS. It analyzes the entire disk for any defective tracks, and prepares the disk to accept FlexOS files.

Caution: Formatting destroys any existing data on the disk. Because of this, you should always specify the disk you want to format. Forms 3 and 4 leave more room for error because FORMAT uses a default value for the current drive. If the default value is different from what you expect, you could mistakenly format the wrong disk.

You must format all new disks before they can be used by FlexOS. You may format your disk with the default values supplied by your manufacturer or, if you are formatting a diskette, you can also use the -P option to select a standard format from the Format Menu.

Table 1-16 summarizes the options you can use with FORMAT. Unless otherwise noted, you can use these options in any combination on the same command line.

Table 1-16. FORMAT Options

Option	Description
-B	Copies the boot loader, but not the system files to the newly formatted disk. If you do not use -B, the disk can only be used to store data files. To make your disk a boot disk, use -B.
-P	Displays the Format Menu and prompts you for a diskette format. -P can only be used to format diskettes. -P overrides -1 and -8 if they appear on the same command line.
-S	Copies the operating system file, the boot image, and all hidden, system and Read-Only files to the newly formatted disk. If you specify -S, it is unnecessary to also specify -B.

Table 1-16. (Continued)

Option	Description
-V	Prompts you for a volume label and whether you want to enable protection on the disk. A volume label must be 11 characters or fewer. All characters acceptable in filenames are acceptable in volume labels. The disk must have a volume label for you to enable disk and file protection. Volumes labels are also used to identify your disks.
-1	Selects a single-sided format from the internal library of standard formats. -1 can only be used with diskettes.
-8	Selects an 8-sectored format from the internal library of standard formats. -8 can only be used with diskettes.

1.28.1 Formatting a Diskette

When you use FORMAT to prepare a diskette, FORMAT prompts you to insert a diskette in the current or specified drive, warns you that any data already on the diskette will be lost, and asks you if you want to continue the FORMAT operation.

If you decide not to continue, type N next to the prompt and press Enter. FORMAT returns you to the system prompt.

If you decide to continue, type Y and press Enter. FORMAT then displays the message "Formatting:" while it formats the disk. The colon blinks while the disk is being formatted.

When FORMAT has finished, it displays the following:

```
Format complete.
  XXXXXX bytes of total disk space
   XXXXX bytes used by system
  XXXXXX bytes available on disk

Format another (Y/N)?
```

The total disk space and space available on disk vary according to the options you use to format your disk and the type of disk you have. Enter Y to format another diskette. Enter N to end the FORMAT program.

1.28.2 Formatting a Hard Disk

Formatting a hard disk is a three-step procedure:

1. If the hard disk has never been formatted, use Form 1 to format the entire hard disk (this is called a **physical** format) with the driver installed as unpartitioned. If the hard disk has been formatted, skip to step 2.

2. Create a partition for FlexOS using the FDISK command.
3. Format the partition (this is called a **logical** format) with the driver installed as partitioned. Use Form 2 of the FORMAT command with the -S and -V options.

When you format a hard disk, the responses you receive from FORMAT are slightly different from those you receive when you format a diskette:

- You are not prompted to insert a diskette. However, you are warned that any data already on the disk will be lost, and you are asked if you want to continue the FORMAT operation.
- When you physically format a hard disk, FORMAT prompts you to enter defective track information provided by the hard disk manufacturer.
- When FORMAT is done, you do not receive the prompt "Format another (Y/N)?".

Examples

```
A>FORMAT A:
```

Formats the disk in drive A with the default values supplied by your manufacturer. When you format a hard disk for the first time (physical format), use this form of the command.

```
A>FORMAT B: -P
```

Uses the -P option to display a list of formats, as follows:

```
Select a format [1 - 6], or else press 'ESC' to cancel

1) 8 sector, 40 track, double-sided (ff)
2) 8 sector, 40 track, single-sided (fe)
3) 9 sector, 40 track, double-sided (fd)
4) 9 sector, 40 track, single-sided (fc)
5) 15 sector, 80 track, double-sided (f9)
6) Manufacturer's diskette format

Select a format [1 - 6], else 'ESC' to exit:
```

The letters in parentheses after each format are media codes, which are used by FlexOS and certain programs to identify the disk format you are using.

```
A>FORMAT B: -S
```

Uses the -S option to format the disk in drive B and copies the operating system file, boot image, and all hidden, system, and Read-Only files to the newly formatted disk. When formatting is completed, FlexOS displays the following message:

```
System transferred
```

```
A>FORMAT -SC:FLEXOS.SYS
```

Uses the -S option with a filespec to format the diskette in drive A and copy the operating system file FLEXOS.SYS from drive C. The boot image and all hidden, system, and Read-Only files are also copied. Do not put a blank space between the -S and the filespec.

When formatting is complete, FlexOS displays the "System transferred" message.

```
A>FORMAT B: -V
```

Formats the diskette in drive B and then prompts you for a volume label:

```
Enter VOLUME label (11 chars. max. or press Enter for none)?
```

A volume label longer than 11 characters results in an error message, and you are prompted to enter another one. If you enter a volume label, FlexOS asks if you want to enable file security:

```
Enable file security (Y/N)?
```

```
A>FORMAT B: -1
```

Formats your diskette for single-sided use. FORMAT does not change the default values for the number of sectors and tracks if it has a single-sided format with those values in its library.

```
A>FORMAT B: -1
```

Formats your diskette at 8 sectors per track. FORMAT does not change the default values for the number of sectors and sides if it has a 8-sector format with those values in its library.

1.29 FSET

Forms View the attributes of a file:

```
FSET filespec
```

Change the attributes of a file:

```
FSET filespec -H=ON|OFF
FSET filespec -A=ON|OFF
FSET filespec -S=ON|OFF
FSET filespec -R=ON|OFF
```

Set the protection on a file:

```
FSET filespec -O=[R][W][E][D]
FSET filespec -G=[R][W][E][D]
FSET filespec -W=[R][W][E][D]
```

Explanation

The FSET command lets you:

- View the attributes of a file.
- Change the attributes of a file. You can change certain characteristics of your file without changing the information in it.
- Set the protection on a file. You can decide who has access to your file and how people may work with it. This protection will not go into effect until you use the DISKSET command to turn on disk protection, or use the FORMAT command to put a volume label on your disk.

You can use wildcards with FSET to change the attributes of or set the protection on several files at once.

1.29.1 Changing the Attributes of a File

Your FlexOS files can have one or more of the following attributes:

-H Hidden file. If you specify -H=ON in your FSET command, that particular file becomes a hidden file. It is not shown in your directory unless you use the DIR command with the -H option.

If you specify -H=OFF, the file appears in your directory. If you do not specify a value for -H, -H=OFF is the default. However, if your file is also a system file (-S=ON), it is not shown in your directory even if -H is off.

-A Archive file. If this attribute is off, it indicates the file has been archived since it was created or last modified. -A is automatically turned on when you modify your file. Programs that back up your files can turn off archiving so they can back up files that have changed since the last backup.

-S System file. If you specify -S=ON, your file becomes a system file. A system file may be accessed by any user on the system without the user needing a copy of the file in his or her directory. Applications that are useful to many people on the system make good system files. A system file is not shown in directory listings unless you use the DIR command with the -H option.

If you specify -S=OFF, your file is not a system file and it appears in the directory listing. If you do not specify a value for -S, -S=OFF is the default. Note however, that if your file is also a hidden file (-H=ON), it is not shown in your directory listing even if -S is off.

-R Read-Only file. If you specify -R=ON, your file becomes a Read-Only file. You can only read the file, even if your User or Group ID give you other privileges. -R=OFF turns off the Read-Only attribute. If you do not specify a value for -R, -R=OFF is the default value.

1.29.2 Setting Protection on Files

FSET lets you decide who may have access to each of your files and how people may work with them. You can set different levels of protection for different files.

There are three classes of user on the FlexOS system. They are:

-O = owner
-G = group
-W = world

The class of user you are depends on your User and Group ID numbers. On a multi-user system, the system manager assigns you a User and Group ID when he sets up your account. If you are on a single-user system, you assign yourself a User and Group ID. For more information on User and Group IDs, see the Configuration Guide.

You are the owner of every file you create. FlexOS keeps a record of this ownership information and uses it to determine who may have access to your file.

When you try to access a file, FlexOS compares your User and Group IDs with the file's ownership information. If your User and Group IDs match the owner information, you are given the owner privileges of the file. If only your Group ID matches, you are given group privileges. If neither your User or Group IDs match the owner information, you are given world privileges.

You can assign each class of user one or more of the following privileges:

- R = Read the specified file.
- W = Write to the specified file. To modify the contents of a file, you must have the write privilege.
- E = Run the specified file (if the file is a program).
- D = Delete part or all of the specified file. To change the security of a file, you must either have owner access or have the delete privilege.

Examples

```
A>FSET PERSONAL.TXT
```

Displays the file attributes of the file PERSONAL.TXT:

```
A:PERSONAL.TXT          A=ON,O=RWED
```

```
A>FSET BUDGET86.DAT -H=ON
```

Uses the -H option to set the hidden attribute ON for the file BUDGET86.DAT. Files with the hidden attribute on are not shown in a directory listing unless you use the -H option of the DIR command.

```
A>FSET BUDGET86.DAT -A=ON
```

Uses the -A option to set the archive attribute ON for the file BUDGET86.DAT.

```
A>FSET BUDGET86.DAT -S=ON
```

Uses the -S option to set the system attribute ON for the file BUDGET86.DAT. Files with the system attribute are not shown in a directory listing unless you use the -H option of the DIR command.

```
A>FSET BUDGET86.DAT -R=ON
```

Uses the -R option to set the Read-Only attribute ON for the file BUDGET.DAT. The Read-Only attribute overrides the access rights that are user/group based. A process cannot delete or write to a Read-Only file even if it has write and delete privileges for the file.

```
A>FSET DOCUMENT.PVT -O=R
```

Sets the security level for the owner category of your account. In this example, -O is set to read, which means that the owner can read from this file, but cannot write to it, execute it, or delete it. Set -O to any combination of the four settings, R, W, E, or D (Read, Write, Execute, or Delete).

```
A>FSET GAMEFILE -G=RWED
```

Sets the security level for the group category of your account. In this example, -G is set to read, write, execute, and delete. This means any other user in your group category has maximum security privileges to the file called GAMEFILE. Set -G to any combination of the four settings, R, W, E, or D.

```
A>FSET FORMULA.BAT -W=RW
```

Sets the security level for the world category of your account. In this example, -W is set to read and write. This means any user on the system can read from or write to this file, but cannot execute or delete it. Set -W to any combination of the four settings, R, W, E, or D.

1.30 GOTO

Form GOTO label . . .

:label

Explanation

The GOTO command transfers control to the line which has the label matching the one used on the GOTO line. GOTO lets you override the normal top-to-bottom running of commands. You can only use GOTO in a batch file.

Labels in a GOTO command appear in at least two places: after the word GOTO and again on a separate line anywhere else in your batch file. The label name next to GOTO is separated from GOTO by a blank space.

Either label name can be in uppercase or lowercase, or a combination of the two cases.

FlexOS treats the first eight characters in a label name as significant. FlexOS ignores any subsequent characters.

The second label name (the point to which control is passed) must meet three requirements:

- It must be on a line by itself. Any information after the label is treated as a comment.
- It must begin with a colon (:).
- The colon must be in column 1 of the new line.

If the label referenced by GOTO does not meet these conditions, the batch file ends with the message "Label not found."

If you use a label without a corresponding GOTO, then FlexOS treats the labeled line as a comment line.

Example

In the following batch file, GOTO2.BAT, the control passes from GOTO THREE to :THREE. FlexOS ignores the statements between the GOTO statement and the label.

```
REM  The name of this file is GOTO2.BAT.
REM  GOTO2.BAT exits normally but skips
REM  label :TWO.
REM
:ONE
REM  I'm in label ONE.
DIR A:
GOTO THREE
:TWO
REM  I'm in label TWO.
DIR B:
REM
A:CHKDSK
CLS
:THREE
REM  I'm in label THREE.
VER
```

1.31 IF

Forms
IF EXIST filespec command
IF NOT EXIST filespec command
IF string1==string2 command
IF NOT string1==string2 command
IF ERRORLEVEL decimal_number command
IF NOT ERRORLEVEL decimal_number command

Explanation

The IF command lets you test a condition to see if it is true or false. If the condition is true, FlexOS runs the command specified at the end of the IF statement. If the condition is false, FlexOS does not run the command.

The command on the IF/IF NOT line must be a FlexOS command or batch command.

A practical application of this command is to find out if a file is present or absent on disk before you take a certain action.

Examples

The form IF EXIST tests for the presence of a file in the current directory of the current or specified drive. In this case, the filespec is limited to a drive name, a filename, and a file extension; directory paths are prohibited. You can use wildcards in the filespec.

Here is how an IF EXIST statement might be used in a batch file:

```
IF EXIST %1.bak GOTO exit
REM  Else filename.BAK does not exist.
REM  We need to copy %1 to drive B
REM  as filename.BAK.
REM
DIR B:*.BAK
REM
REM  Observe the lack of filename.BAK
REM  on drive B.
REM
PAUSE
COPY %1 B:%1.BAK
DIR B:*.BAK
REM
REM  Confirm filename.BAK has been copied
REM  to drive B.
REM
PAUSE
GOTO BYPASS
:EXIT
ECHO filename.BAK already exists
DIR %1.BAK
REM  See?  I told you so.
:BYPASS
```

The form IF NOT EXIST inverts the logic of IF EXIST. Here is how the same backup task 1 might look using IF NOT EXIST:

```
IF NOT EXIST %1.BAK GOTO BAKUP
REM
:SHOWME
REM
REM filename.BAK exists; prove it
REM
DIR A:*.BAK
DIR B:*.BAK
PAUSE
CLS
GOTO EXIT
REM
:BAKUP
REM Copy %1 to drive B as
REM filename.BAK
REM
COPY %1 B:%1.BAK
GOTO SHOWME
REM
:EXIT
ECHO filename.BAK already exists
```

The preceding examples show that using IF or IF NOT is often a matter of style.

Forms 3 and 4 make sure two strings are identical before running the command. The strings can be any alphanumeric characters. If the two strings are identical, even to the matter of uppercase and lowercase letters, FlexOS runs the command at the end of the line.

For example, if a batch file called DISRAELI.BAT contains the following lines:

```
ECHO
IF %1==Gladstone ECHO %1 is a self-made man.
VOL
```

and if your command line looks like this:

```
DISRAELI Gladstone
```

FlexOS substitutes "Gladstone" for %1 in both places and tests to see if the parameter "Gladstone" matches the string on the right side of the double equal signs. If the test string in the batch file does match the string on the command line, FlexOS runs the ECHO command.

```
A>DISRAELI Gladstone

A>ECHO
ECHO is ON

A>ECHO Gladstone is a self-made man.
Gladstone is a self-made man.

A>VOL
Volume in drive A: is BATCHDISK

A>_
```

If the two strings are different, FlexOS does not run the ECHO command. FlexOS runs the VOL command whether the strings match or not.

Forms 5 and 6 test exit codes in a program that you create and run against a batch file. If the exit code decimal number in your program is equal to or greater than the decimal number in your batch file, FlexOS runs the specified command. Otherwise, that command is not run and control in your batch file passes to the line following IF ERRORLEVEL.

The ECHO command is often used with the IF and IF NOT statements to inform you of the exit conditions FlexOS encounters while running a program. Consider the following batch file, TESTPROG.BAT:

```
TESTHELP
REM
REM  TESTHELP is a program you are running
REM
:CHKERRS
        IF ERRORLEVEL 2 GOTO WRITERR
        IF ERRORLEVEL 1 GOTO READERR
        IF ERRORLEVEL 0 GOTO NORMAL
:WRITERR
        ECHO TESTHELP had a WRITE failure.
        GOTO REALEXIT
:READERR
        ECHO TESTHELP had a READ  failure.
        GOTO REALEXIT
:NORMAL
        ECHO TESTHELP finished normally.
        REM  It wasn't condition 2 or 1.
        GOTO REALEXIT
:REALEXIT
        REM  Th-Th-That's all, folks!
```

TESTPROG.BAT checks each IF ERRORLEVEL number value for a matching value in your running program. Of course, the messages you use with each ERRORLEVEL statement should reflect what is happening in the program.

1.32 LIST

Form LIST

Explanation

The LIST command displays a list of the FlexOS commands which are built-in to the system. Your computer manufacturer determines which commands are built-in.

1.33 LOGOFF

Forms LOGOFF
LOGOFF node::

Explanation

LOGOFF ends your current session, either on a local computer or over a network on a remote computer. Use form 1 to end the session on your computer and all remote computers you have logged on. Use form 2 to end the session on the the remote computer designated by "node::" only. See the FlexNet User's Guide for the description of the network and remote computer use.

After you see the logoff message, LOGOFF prompts you to begin another session.

Example

```
A>LOGOFF
Jones logged off 05-15-1986 08:30:00.00
```

In this example, user Jones ended his session at 8:30 A.M. on May 15, 1986.

Note: If you did not begin your session with a LOGON command, LOGOFF uses "noname" as a default name. In the example above, "noname" would have been displayed instead of "Jones". Setting up the LOGON/LOGOFF procedure is described in the Configuration Guide.

1.34 LOGON

Forms LOGON
LOGON node::

Explanation

LOGON tells FlexOS you want to start a session. Use form 1 to start a session on your computer. Use form 2 to start a session over the network with the remote computer designated by "node::". See the FlexNet User's Guide for the description of the network and remote computer use.

When you type LOGON, FlexOS prompts you for your username and password. When you type your password, the letters are not displayed on the screen. When you see the system prompt, you know you have logged on successfully and you can begin your work session. The PASSWORD command is discussed later in this section.

```
LOGON
Username:
Password:
```

The LOGON/LOGOFF procedure is an effective way to protect your data from unauthorized access; however, this procedure is optional. The system manager of your computer system decides whether or not users must use LOGON and LOGOFF. Setting up the LOGON/LOGOFF procedure is described in the Configuration Guide.

1.35 MKDIR

Forms MKDIR directory_path
MKDIR drive:directory_path

Abbreviation
MD

Explanation

MKDIR creates a subdirectory on the current or specified disk.

The number of subdirectories you can create is limited only by the available disk space. However, the maximum length of a directory path from the root directory to the level you want to access must be 128 characters or fewer.

Examples

The first example of the MKDIR command creates a new subdirectory called FOREST that is attached to the root directory.

```
A>MD /FOREST
```

To create a subdirectory beneath FOREST, use one of the following examples:

If your current directory path is root/FOREST, use this form:

```
A>MD TREES
```

The absence of a leading slash mark (/) tells FlexOS to create a subdirectory TREES and add it to the current directory.

The next form also creates TREES and adds it to the path root/FOREST, but you can use this command line regardless of your current directory path.

```
A>MD /FOREST/TREES
```

In this form of the command, the first / tells FlexOS to begin its directory search with the root directory.

The second form of the MKDIR command creates a directory on a drive other than the current drive. The following example creates a subdirectory called PHONE in the root directory on drive B:

```
A>MD B:/PHONE
```

1.36 MORE

Forms Display a file one screen at a time:

MORE < filespec

Display command output one screen at a time:

COMMAND | MORE

Explanation

MORE displays the contents of a file or the output from a command one screen at time. You can also direct MORE's output to a file.

MORE displays a screenful of data and then displays the message:

```
-- More --
```

If you press any character key, another screenful of data is written to the screen or file. This process continues until all input data is read.

Examples

The first form of MORE uses a file for input. The following example displays a screenful of data from the file TEST.HIS:

```
A>MORE < TEST.HIS

HISTORY 170 TEST ANSWERS
------------------------

1.  Louisiana Purchase
2.  Jefferson
3.  Monroe
.
.
.
20. Lincoln
-- More --
```

The second form of MORE uses the output from a command as input. The following example displays a screenful of output from the DIR command:

```
A>DIR | MORE

  Volume in drive A: is DISKETTE1
  Directory of  A:

SALES      JAN      2015   1-15-1985   1:19p
EXPENSES   JAN      1613   1-31-1985  12:14p
SALES      FEB      3256   2-15-1985   9:00a
EXPENSES   FEB      1300   2-21-1985   2:45p
SALES      MAR      8788   3-15-1985  10:21a
EXPENSES   MAR      1525   3-25-1985  12:14p
.
.
.
EXPENSES   OCT      2691  10-29-1985   8:30a
-- More --
```

1.37 ORDER

Forms
ORDER
ORDER EXT
ORDER EXT;EXT; ... EXT;

Explanation

The ORDER command defines the search order of specified file extensions. ORDER functions the same way as the PATH command except that it searches for file extensions rather than subdirectories.

ORDER is most often used to search for the file extensions of executable files when FlexOS is loaded. These files extensions include 286, 386, 68K, COM, and EXE.

If you specify a file extension that does not exist, FlexOS displays an error message when it tries to load the file.

If you specify more than one file extension with ORDER, you must separate file extensions with a semicolon, blank, or tab.

Examples

The ORDER command without a file extension displays the search order currently defined to FlexOS. In the following example, ORDER reports that, if you do not specify a file extension, FlexOS looks for files with the extension EXE.

```
A>ORDER
ORDER = EXE
```

The ORDER command with one or more file extensions tells FlexOS to look for files with one of those extensions if an extension is not supplied by the user. The following example tells FlexOS to look for files with the extension EXE or COM:

```
A>ORDER EXE;COM
```

1.38 PASSWORD

Forms PASSWORD
PASSWORD username

Explanation

The PASSWORD command lets you or the system manager change your LOGON password. Passwords are primarily used on multi-user systems to prevent unauthorized access to files. However, if you are on a single-user system you can still use a password to prevent others from accessing your work. Setting up the LOGON/LOGOFF procedure and working with passwords are explained in the Configuration Guide.

A password can be any string of ASCII characters. The first eight characters of the password are significant. When you attempt to log-on to your computer, FlexOS asks you to type your username and then your password. Although your username is displayed at the screen as you type it, your password, for security reasons, is not. If you correctly enter your username and password, FlexOS lets you access your files.

Examples

```
A>PASSWORD
```

Sets or changes your password. PASSWORD displays the following prompts:

```
Old password:
New password:
Confirmation:
```

Enter your old password, then your new password, and then enter your new password a second time to make sure you entered it correctly. None of your input is displayed on the screen.

If you decide not to change your password after you enter the PASSWORD command, press Ctrl-C to return to the system prompt.

```
A>PASSWORD CATHERINE
```

Specifies a username. To use this form of the command, you must be logged into a privileged account. The FlexOS system manager uses this form to set the initial passwords for each user. After the initial assignment, a user can change his or her password without consulting anyone.

This form displays the following prompts:

```
New password:
Confirmation:
```

By not prompting for the old password in this case, PASSWORD enables the system manager to assign a new password to a user who has forgotten the old one.

1.39 PATH

Forms
PATH
PATH directory_path
PATH directory_path;directory_path...
PATH drive:
PATH drive:directory_path
PATH drive:directory_path;drive:directory_path...

Explanation

PATH defines the search path of specified directories for commands or batch files that were not found by searching the current directory. PATH does not change your current directory.

If the command you entered is not found in any of the directories you specified, you receive an error message. If you specify a path that no longer exists, FlexOS ignores that path and goes on to the next one, if one exists.

Note: When you specify one or more search paths, you must type the words DEFAULT: and SYSTEM: in front of the first path.

Subdirectory names in a directory path must be preceded, separated, and followed by slashes (/), in the following manner:

/subdirectory/subdirectory/..../subdirectory/

If you specify more than one path with the PATH command, you must separate paths with a blank, semicolon, or tab.

Examples

```
C>PATH
PATH = C:/
```

Displays the search path currently defined to FlexOS.

```
C>PATH DEFAULT: SYSTEM:/COMMANDS/
```

Specifies a search path on the current drive. FlexOS searches these directories in the order they appeared in the PATH command. Note that, for FlexOS to search the current ("default") directory, you must explicitly name it in the PATH command. In the example, FlexOS first looks in the default directory on drive C, then in the directory /COMMANDS.

```
A>PATH DEFAULT: SYSTEM:/ACCOUNTS/;/BASIC/
```

Specifies more than one path on the current drive. In the example, FlexOS looks for a command in the current directory and in the subdirectories /ACCOUNTS and /BASIC.

1.40 PAUSE

Forms PAUSE
PAUSE comment

Explanation

PAUSE suspends batch processing so that you can evaluate the current situation and possibly take an appropriate action. For example, you might want to change diskettes between commands.

Insert PAUSE commands in your batch file where you want to end or continue batch processing. Each PAUSE command stops FlexOS and gives you time to decide whether to end processing. If you decide to end processing, press Ctrl-C. To continue processing, press any other key.

Examples

Form 1 of the PAUSE command suspends processing and displays the message, "Strike a key when ready...." If you press any key (except Ctrl-C), the batch file continues to run with the line following PAUSE. If you press Ctrl-C, batch processing ends. For example, suppose the file MAYBE1.BAT contains the following commands:

```
VER
DIR A:*.BAK
PAUSE
DIR A:*.TXT
PAUSE
COPY A:*.* B:
```

The two PAUSE commands let you examine your screen's contents before you proceed or stop.

Form 2 includes a comment with the PAUSE command. The comment is displayed when PAUSE suspends processing. The comment is a string of characters up to 121 characters long. For example, suppose MAYBE2.BAT contains the following commands:

```
VER
DIR A:*.BAK
PAUSE The next command erases *.BAK on A!
DEL A:*.BAK
DIR A:*.TXT
PAUSE The next command erases *.TXT on A!
DEL A:*.TXT
```

When you run MAYBE2.BAT, the comments on the PAUSE lines inform you of the consequences of continuing with the batch session. If you want all files with a BAK file extension erased when the first PAUSE occurs, press any key except Ctrl-C. If you decide to continue batch processing, you reach another decision point at the second PAUSE command. Press Ctrl-C to stop or any other key to continue, depending on whether you want to delete all the files on A with a file extension of TXT.

1.41 PRINT

Forms Print One or More Files:

```
PRINT filespec_1 filespec_2 ...
PRINT filespec -d=device
```

Print a File on a Server Node:

```
PRINT filespec -n=node::
PRINT filespec -n=node:: -d=device
```

Display Printer Status:

```
PRINT -s
PRINT -s -n=node:: -d=device
```

Cancel Print Jobs:

```
PRINT -t
PRINT -c=job_id
```

Put Priority Job at Head of Print Queue:

```
PRINT filespec -h
```

Load Print Queue:

```
PRINT -l
```

Explanation

Note: You must have a print spooler to use this command. See the Configuration Guide for information about installing a print spooler. For information about using the print spooler, see the User's Guide. If your system does not include a spooler, use the COPY command to send files to your printer.

Also, you can only print files on nodes if your system is part of a network. See the FlexNET User's Guide for further information on network operations.

PRINT sends your files to a print spooler to be printed in the background, freeing your system for other work. You can combine any and all options listed above in any sequence. You can also have more than one filespec (but less than 20) on the command line. Wildcards may be used. A space must separate each element of the command line. Every character string that does not begin with a hyphen is treated as a filespec. For instance, "PRINT s" prints a file named "s" but "PRINT -s" shows printer status.

Use Form 1 to send one or more files to the local default printer's print queue. A print **queue** is a lineup of files waiting to be printed. PRINT confirms job entries by returning the status of the jobs entered. However, when you cancel print jobs or load print queues PRINT does not return confirmation.

You can override the system printer definitions and specify a local printer (Form 2) or a printer on another network node (Form 4). A **node** is a system in a network of systems. If you do not specify a printer, the job is sent to the local default printer or the default printer on the node you specified (Form 3).

If you do a lot of printing on a specific server node, use the DEFINE command to identify that node as the default so that you do not have to enter its name each time. Define the default printer (PRN) as follows: "DEFINE prn:=node::spldrv:". Replace "node" with your node name.

Form 5 is used to show the status of all print devices on the local node. You can also specify other nodes and devices with Form 6. The system displays a status report when either of these options are used. Form 7 is used to terminate all your print jobs. (If the -t option is used in any entry, it overrides all other options.) You can cancel specific print jobs by using the -c option and specifying the job id as shown in Form 8. The job id is listed in the status report; it is also displayed by the system when the job is entered.

Use Form 9 to enter a priority job at the head of the print queue so that it is next in line to print. (A priority job does not interrupt a job in progress.) Form 10 must be used to reload the print queue if a system halt or power failure has occurred. No new print jobs can be loaded before this is done. Jobs running when the interruption occurred are resumed when the queue is reloaded. The printer buffer may have been cleared, however, and part of the currently printing file may have been lost.

Table 1-17 presents the options for PRINT.

Table 1-17. PRINT Options

Option	Description
-c	Cancels single print job
-d	Device identifier
-h	Place job at head of queue
-I	Initialize print queue
-n	Node identifier (Network operations only)
-s	Displays printer status
-t	Terminate all printer operations from this logon

Examples

```
A>PRINT FIRST.DAT SECOND.TXT
```

Sends the files FIRST.DAT and SECOND.TXT to the local default printer. FlexOS shows the following status report on your print jobs, including the user name and the name of the default device:

Status	Job_id	User	Filespec	Node	Device
PRINTING	1	tim	FIRST.DAT		prn:
QUEUED	2	tim	SECOND.TXT		prn:

```
A>PRINT THIRD.DOC -d=lpt1:
```

Status	Job_id	User	Filespec	Node	Device
PRINTING	6	bob	THIRD.DOC		lpt1:

Specifies a printer other than the default. In this example, the file THIRD.DOC is being sent to device LPT1.

```
A>PRINT THIRD.DOC -n=AJAX -d=prn:
```

Status	Job_id	User	Filespec	Node	Device
QUEUED	4	bob	THIRD.DOC	AJAX::	prn:

Sends the file THIRD.DOC to a device on a remote server node named Ajax.

```
A>PRINT -s -n=ZEUS -d=lpt2:
```

*DEVICE: lpt2: on NODE ZEUS:: STATUS: inuse JOBS: 3

Status	Job_id	User	Filespec
PRINTING	7	jack	LETTER.TXT
QUEUED	8	jill	MEMO.DAT
QUEUED	9	john	REPORT.LST

Shows the status of all print jobs on device LPT2 on node Zeus. (To get the status of all print jobs on the default device on the local node, enter the -s option alone.)

```
A>PRINT -c=job10
```

Cancels job 10 on the default printer on the local node. The command **PRINT -t** stops all of your jobs (jobs entered from that login) on all printers.

```
A>PRINT RUSH.LTR -h -n=JOVE -d=PRN:
```

Status	Job_id	User	Filespec	Node	Device
QUEUED	9	tad	RUSH.LTR	JOVE::	prn:

Puts the priority print job RUSH.LTR at the top of the printer queue for device PRN: on node Jove. The status report shows job 9 at the head of the queue, ready to print when the current job is finished.

```
A>PRINT -l -n=MARS -d=LPT1:
```

Reloads the print queue for device LPT1 on node MARS. To reload all devices, enter the -l option alone.

1.42 PROCESS

Forms
PROCESS
PROCESS VIEW
PROCESS VIEW -option
PROCESS CANCEL ID=##########

Explanation

The PROCESS command lets you view or delete processes currently running on the system. A process is defined as any command, program, or application running under FlexOS.

The PROCESS VIEW options, described in Table 1-18, can be used in any combination.

Table 1-18. PROCESS VIEW Options

Option	Description
-E	Provides extra information.
-F	Provides a Family filter, which displays only those processes that match this user's Family ID. Otherwise, all processes on the (local) system are displayed.
-M	Provides memory statistics.
-P	Provides paged output, as opposed to streamed output.
-U	Provides a User filter, which displays only those processes that match this user's Group and User IDs. Otherwise, all processes on the (local) system are displayed.

Examples

```
A>PROCESS
```

Describes the PROCESS syntax and options.

```
A>PROCESS VIEW
```

Displays information on each process running on the system. Information is displayed with the following headings:

name	Process name: a unique 8-character identifier.
pid	Process ID number: a unique decimal value up to 10 digits long.

wndw	Window number in which the process is running. A decimal value up to 3 digits long.
pcon	Physical console number on which the process is running.
fmly	Family ID number: a decimal value up to 5 digits long. Related processes (such as commands within a batch file, or commands chained together on the same command line) have the same family ID.
grp	Group ID of the person running the process: a decimal number 0–255.
user	User ID of the person running the process: a decimal number 0–255.
stat	Status of the process: running, waiting, terminating, or unknown. This heading also tells you if a process is using FlexOS's shared code facilities.

A typical PROCESS VIEW display looks like this:

```
PROCESS VIEW MODE

-------------------------------------------------------------------
name: PROCESS   pid : 4    wndw: 0       pcon: 0
fmly: 3         grp : 5    user: 2       stat: running     *
-------------------------------------------------------------------
name: COMMAND   pid : 2    wndw: 0       pcon: 0
fmly: 0         grp : 0    user: 0       stat: terminating
-------------------------------------------------------------------
name: CONFIG    pid : 1    wndw: 0       pcon: 0
fmly: 0         grp : 0    user: 0       stat: terminating
-------------------------------------------------------------------
```

The asterisk following "stat: running" indicates the PROCESS command itself is currently running.

A>PROCESS VIEW -E

Displays the following extra PROCESS VIEW information:

prty	Process priority. The priority for most processes is 200.
ppid	Parent process ID number: a decimal value up to 10 digits long. A parent process is a program or command that causes other processes to begin running automatically. For example, when you run the CONFIG program to start FlexOS, CONFIG automatically starts the COMMAND program. CONFIG is the parent process of COMMAND.
process type	If the process was started by a user, PROCESS displays the heading "user process"; if the process was started by the system, PROCESS displays the heading "system process."

context — If the process was started by a user, PROCESS displays the heading "main context"; if the process was started by the system, PROCESS displays the heading "system context."

evnt — Event flags: a 32-bit binary number. Event flags keep track of operations a process has not yet completed. If a process has completed all its operations, "evnt:" equals zero.

system manager — PROCESS displays this heading only if the process was started by the system manager.

The PROCESS VIEW -E display looks like this:

```
PROCESS VIEW MODE

-------------------------------------------------------------------------
name: PROCESS    pid : 4    wndw: 0          pcon: 0
fmly: 3          grp : 5    user: 2          stat: running      *
prty: 200        ppid: 2    (user process)   (main context)
evnt: 0000 0000 0000 0000 0000 0000 0000 0000
-------------------------------------------------------------------------
name: COMMAND    pid : 2    wndw: 0          pcon: 0
fmly: 0          grp : 0    user: 0          stat: terminating
prty: 200        ppid: 1    (user process)   (main context)
evnt: 0000 0000 0000 0000 0000 0000 0000 0000
-------------------------------------------------------------------------
name: CONFIG     pid : 1    wndw: 0          pcon: 0
fmly: 0          grp : 0    user: 0          stat: terminating
prty: 200        ppid: 0    (user process)   (main context)
evnt: 0000 0000 0000 0000 0000 0000 0000 0000
-------------------------------------------------------------------------
```

A>PROCESS VIEW -F

Displays only those processes whose family IDs match your own. The information displayed is the same as for PROCESS VIEW.

A>PROCESS VIEW -M

Displays memory statistics with the following headings:

maxm — Maximum memory: a hexadecimal number up to eight digits long. This is the maximum amount of memory allocated to the process.

code — Code start: a hexadecimal number up to eight digits long. This is the location in memory where the code of the process begins. The size of the code is displayed in the "size:" heading below "code:."

data — Data start: a hexadecimal number up to eight digits long. This is the location in memory where the data associated with the process. The size of the data is displayed in the "size:" heading below "data:."

heap — Heap start: a hexadecimal number up to eight digits long. A heap is a storage area for data. "heap:" is the location in memory where this storage area begins. The size of the heap is displayed in the "size:" heading below "heap:."

memory status — If the memory used by a process can be transferred to and from secondary storage, PROCESS displays the heading "swap." If memory cannot be transferred, PROCESS displays the heading "locked."

The PROCESS VIEW -M display looks like this:

```
PROCESS VIEW MODE

-------------------------------------------------------------------
name: PROCESS   pid : 4         wndw: 0         pcon: 0
fmly: 3         grp : 5         user: 2         stat: running     *
maxm: 0H        code: 41DC0000H data: 44040000H heap: 0H
(swap)          size: 2280H     size: FE0H      size: 0H
-------------------------------------------------------------------
name: COMMAND   pid : 2         wndw: 0         pcon: 0
fmly: 0         grp : 0         user: 0         stat: terminating
maxm: 0H        code: 2CD90000H data: 37850000H heap: 0H
(swap)          size: AAC0H     size: 4140H     size: 0H
-------------------------------------------------------------------
name: CONFIG    pid : 1         wndw: 0         pcon: 0
fmly: 0         grp : 0         user: 0         stat: terminating
maxm: 0H        code: 0H        data: 0H        heap: 0H
(swap)          size: 0H        size: 0H        size: 0H
-------------------------------------------------------------------
```

A>PROCESS VIEW -E-M

Displays the information generated by both options.

A>PROCESS VIEW -E-M-P

The -P option displays the information one screenful at a time. At the end of each screen, you are prompted to press any key to continue. If you do not specify -P, process information scrolls on your screen until it reaches the end of the information.

A>PROCESS VIEW -U

Displays only those processes whose User and Group IDs match your own.

A>PROCESS CANCEL ID=4

Cancels the program whose process ID is 4. You can only cancel processes you have created, and you must know its process ID number. After the process is canceled, PROCESS displays "Process deleted." and you are returned to the system prompt.

1.43 PROMPT

Forms PROMPT
PROMPT text

Explanation

PROMPT sets a new system prompt or restores the default setting. The system prompt may not exceed 128 characters.

The first non-blank character after the word PROMPT is the first character of the prompt. FlexOS considers all the text on the PROMPT command line to be the new system prompt.

If you enter the PROMPT command by itself, FlexOS restores the default prompt. You can include special characters in the text in the form $c, where c is one of the following:

t	time	b	| character
d	date	q	= character
g	> character	$	dollar sign
l	< character	e	ESCape character
p	current directory path	n	current drive

h backspace and erasure of previous character
_ go to beginning of new line on screen
v FlexOS version number

Any other character is treated as a null character. The PROMPT command takes no action on it.

Examples

```
CAM!PROMPT

A>_
```

Restores the current prompt CAM! for drive A: to its default form A>.

```
A>PROMPT XYZ

XYZ_
```

Changes your current prompt (default or otherwise) to the specified form, in this case "XYZ".

```
XYZPROMPT $n$g

A>_
```

Sets the system prompt to its default value by explicitly stating its form.

```
A>PROMPT Time=$t$_$p$g
```

```
Time=11:17:33.76
A:/COMMANDS/>_
```

Sets a two-line prompt that displays the current time and the current directory path of the default drive.

1.44 RECDIR

Form RECDIR -d:[pathname]

Explanation

RECDIR is used to analyze and repair disk directories with bad sectors. If you receive repeated read error messages when trying to access a directory, the problem is most likely a defective sector. You can usually recover the directory that contains the bad sector, minus the file entries in the bad sector.

RECDIR analyzes the media in the drive for suitability and then analyzes the disk. RECDIR copies any directory cluster with defective sectors to a new cluster and fills the bad sector with null directory entries. You can recover the affected files by using the -F option of the CHKDSK command and responding Yes when prompted. The lost data is then written to files in the format FILEnnnn.CHK, where nnnn is a sequential four-digit number.

Note: A cluster is the minimum allocation unit of a directory file. This varies with the media and is a multiple of the sector size.

RECDIR requires that you be able to read from and write to the root directory of the disk you are attempting to access. Any I/O directory access or memory allocation errors cause RECDIR to terminate prematurely and an error message appears.

Example

```
A>RECDIR -B:
```

Analyzes the media in drive B for suitability before attempting repair. RECDIR locates bad sectors on the entire disk, fills them with nulls, and copies each repaired directory to a new cluster. RECDIR must be entered with a disk drive designator followed by a colon.

```
A>RECDIR -B:/PROPERTY/TAXES
```

Restricts the analysis to the directories at the lowest directory level specified by the path.

1.45 RECFILE

Forms RECFILE filespec_1 filespec_2 ...filespec_last
RECFILE -option(s) filespec_1 filespec_2 ...filespec_last

Explanation

RECFILE analyzes and repairs files with hard read errors; it recovers files from a disk which has defective sectors. It also reports the number of clusters and extents for each file. If you receive repeated read or write error messages when trying to access a file, the problem is most likely a defective sector. You can usually recover the file that contains the bad sector, minus the data in the bad sector.

Note: A **cluster** is the minimum allocation unit of a file. This varies with the media and is a multiple of the sector size. An **extent** is a set of contiguous sectors for a file. The more extents that exist for a file, the more fragmented the file is.

Table 1-19 describes the options you can use with RECFILE.

Table 1-19. RECFILE Options

Option	Description
-t	Disables the File Allocation Table (FAT) trace. Displays only the total number of clusters for a file. If not specified, a list of cluster numbers that each file occupies is generated for each file specified.
-i	Translates all filespecs to uppercase before comparing. Normally, RECFILE only converts filespecs to uppercase before comparing them if the media default does not support upper- and lowercase letters. The -i option ensures that translation is performed regardless of the media default.
-f	Causes RECFILE to attempt to repair the file by rereading the bad clusters sector by sector, filling in unreadable sectors with fill data (0x2e), and moving the cluster data to the next free cluster. Bad clusters are then marked as flawed in the File Allocation Table (FAT) so they will not be allocated to other files.

RECFILE displays the name of each file being processed. When it encounters a directory file, as can happen when you use wildcards, RECFILE displays a message that it cannot repair directory files, but will continue to process any other files specified. (Use RECDIR to repair directory files, including the root directory.)

If you do not specify a drive, the current or "default" drive is used. If you do not specify a path, the default path is used. You may use wildcards with RECFILE.

Text files that have been repaired usually require editing before they can be used for normal processing.

Examples

```
A>RECFILE EXPENSES.DAT TRAVEL.DAT INCOME.REC
```

Reads the specified files, cluster by cluster, and displays a list of cluster numbers that the files occupy along with the total number of clusters and number of extents. If any bad clusters are found, their numbers are displayed with an error message.

```
A>RECFILE -f -t -i EXPENSES.DAT INCOME.REC
```

Analyzes both files specified and then does the following:

- Displays the total number of clusters for each file.
- Translates all filespecs to uppercase.
- Attempts to repair the files.

```
A>B:RECFILE -f *.*
```

Attempts to repair all files in the current directory of drive A.

1.46 RENAME

Form RENAME old_filespec new_filespec

Abbreviation REN

Explanation

The RENAME command lets you change the name and/or extension of a file. RENAME does not change the information in the file.

You can specify a directory path with RENAME. If you specify a path with new_filespec that is different from the path with old_filespec, the file is renamed and moved to the second directory path. (The second directory path must already exist.) If you don't name a directory path or if both paths are the same, the file stays in the same directory after it is renamed.

You cannot rename a directory, and you cannot rename a file to a different drive.

You can use wildcards in both filespecs.

Examples

```
A>RENAME IN.TXT OUT.TXT
```

Renames IN.TXT to OUT.TXT. OUT.TXT remains in the current directory.

```
A>REN B:REVIEW.Q1 B:*.Q2
```

Renames the file REVIEW.Q1 on drive B to REVIEW.Q2, also on drive B. The wildcard tells FlexOS not to change the filename.

```
A>REN B:/WORKFILE/LIST.TXT B:/LEISURE/LISTLESS.TXT
```

Renames the file LIST.TXT in subdirectory /WORKFILE on drive B to LISTLESS.TXT and moves it to subdirectory /LEISURE.

1.47 REM

Forms REM
REM remark

Explanation

The REM command lets you put remarks in your batch file. These remarks are displayed on the screen when FlexOS encounters them in running batch files. Processing is not interrupted by a REM command; the remark is merely displayed and processing continues on the line following the REM command.

REM lines are not displayed when ECHO is OFF.

Examples

Form 1 of the REM command inserts a blank line in your batch file. Careful use of blank lines makes batch files easy to read because of the visual (and hopefully logical) separation of one block of text from another. The following example uses the REM command to separate the COPY commands from the ERASE commands.

```
REM
COPY A:*.BAK C:
COPY B:*.BAK C:
REM
REM
ERASE A:*.BAK
ERASE B:*.BAK
REM
DIR C:*.BAK
```

Form 2 lets you put remarks in your batch file. A remark is a string up to 123 characters. Adding text to your REM line helps keep you informed of actions your batch file took or is about to take. For example, suppose ENDOFDAY.BAT contains the following commands:

```
REM     copy *.BAK files to drive C
COPY A:*.BAK C:
COPY B:*.BAK C:
REM
REM     Erase *.BAK files on A and B
REM
ERASE A:*.BAK
ERASE B:*.BAK
REM     Show current set of *.BAK files
REM     on drive C
DIR C:*.BAK
```

The meaning of each portion of text becomes clear. Using remarks in your batch files makes the files easier to read and to work with.

1.48 RESTORE

Forms
RESTORE backup:
RESTORE backup: source:
RESTORE backup: source: filespec
RESTORE backup: source: filespec -option

Explanation

The RESTORE command lets you restore to the source disk one or more files that have been backed up to diskettes. For instructions on how to back up files, see the BACKUP command in this section.

In the forms shown above, "backup:" identifies the drive containing the backup disks; "source:" identifies the drive to which you are restoring your files.

If you do not specify a path for the source drive, the files are restored to the current directory.

Table 1-20 summarizes the options you can use with RESTORE. You can use both options on the same command line.

Table 1-20. RESTORE Options

Option	Description
-P	Prompts you before restoring Read-Only files and files that have changed since the last backup. FlexOS system files are marked Read-Only when they are created by the FORMAT and SYS commands. If you want to restore a diskette which has these files on it, use -P.
-S	Restores backed up files to their original directories on the source disk. If a directory no longer exists on the source disk, restoring files to it recreates the directory.

RESTORE reads the backup disks in the order they were created. For that reason, when RESTORE prompts you to insert the backup disk, insert the first disk that might contain the file you want to restore. (If you are not sure, insert backup disk number 1.) If the file is not on the disk you inserted, RESTORE prompts you to insert the next backup disk.

If you specify a wildcard character in the filename, all files matching that filename are restored. RESTORE then prompts you to insert the next backup disk.

Examples

```
C>RESTORE A:
```

Restores all files on the diskette in drive A to the current directory of the disk in drive C (in this case, the root directory). Because the source disk is not specified, FlexOS assumes it to be the current drive.

```
D>RESTORE B: C:
```

Restores all files in the current directory of the backup disk in drive B to the current directory of the source disk (drive C).

```
C>RESTORE A: /CUSTOMER/LETTER.TXT
```

Restores the file LETTER.TXT to the subdirectory CUSTOMER on drive C. Note that it is not necessary to specify drive C because drive C is the current drive.

```
D>RESTORE A: C:*.MEM
```

Restores each file with an extension of MEM from the backup disk to the current directory on drive C.

1.49 RMDIR

Form RMDIR directory_path
RMDIR drive: directory_path

Abbreviation RD

Explanation

RMDIR removes a subdirectory from the current or specified disk. You cannot remove the root or current directories. RMDIR removes the last directory name in the specified path.

You must delete all files from a directory before you remove it. If you try to remove a directory containing files, you receive an error message.

Example

```
A>RD /FOREST/TREES
```

Removes the directory TREES from the directory path root/FOREST/TREES.

```
C>RD A:/FOREST/TREES
```

From drive C, removes the directory TREES from the directory path root/FOREST/TREES on drive A.

1.50 SECURITY

Forms View your current file access privileges:

SECURITY

Change your current file access privileges:

SECURITY -O=[R][W][E][D]
SECURITY -G=[R][W][E][D]
SECURITY -W=[R][W][E][D]

Explanation

SECURITY lets you view or change current file access privileges. These privileges are Read (R), Write (W), Execute (E), and Delete (D). You can change the level for all three user classes: Owner (-O), Group (-G), and World (-W). See the FSET command for an explanation of file access privileges and user classes.

Privileges set with SECURITY are in effect for the duration of a terminal session. We suggest that you include a SECURITY command in your AUTOEXEC.BAT file to set file access privileges automatically each time you start your computer. You can also invoke SECURITY from the command line.

Example

```
A>SECURITY -G -W R
```

Sets the file access privileges for Group and World users to Read access. Note that you can combine user classes by separating them with spaces.

1.51 SHIFT

Form SHIFT

Explanation

The SHIFT command lets you use more than 10 parameters on one command line. SHIFT does this by shifting all the words on a command line one word to the left. SHIFT can only be used with batch files.

Remember that you can only put 10 placeholders in a batch file (%0 through %9). However, SHIFT lets you use different parameters for the same placeholders in a given batch file.

Because placeholders in a batch file depend on a parameter's position on a command line for placeholder substitution, SHIFT changes the relationship of the parameters to the batch file placeholders in the following way:

- The parameter leftmost on the line before SHIFT was run is no longer available to the running batch file after a SHIFT is run.
- The remaining command line parameters decrement their ordinal values by one, so that the 11th parameter becomes the 10th, the 10th becomes the 9th, and so on.

Compare the relationship of the command line parameters to the batch file placeholders in the following example:

```
A>SHIFT1 FIRST SECOND THIRD

SHIFT1.BAT:

ECHO %0 %1 %2
SHIFT
ECHO %0 %1 %2
SHIFT
ECHO %0 %1 %2
```

FIRST, SECOND, and THIRD are the first, second, and third parameters to be used with SHIFT1.BAT.

The batch file has three placeholders: %0, %1, and %2. Consequently, SHIFT1 substitutes for %0, FIRST for %1, SECOND for %2, and THIRD for %3 before any shifting occurs. When you run SHIFT1 using the following command line, this output is displayed on the screen:

```
A>SHIFT1 FIRST SECOND THIRD

A>ECHO SHIFT1 FIRST SECOND
SHIFT1 FIRST SECOND

A>ECHO FIRST SECOND THIRD
FIRST SECOND THIRD

A>ECHO SECOND THIRD
SECOND THIRD

A>ECHO THIRD
THIRD
```

Each leftmost parameter from the original command line disappears after each SHIFT execution. Notice also how the values for %0, %1, and %2 change after each SHIFT: FIRST substitutes for %0, SECOND for %1, and THIRD for %2 after the first SHIFT. Finally, notice that the parameter THIRD was not available to the running batch file until the parameter list had shrunk by one word -- that is, after the first SHIFT.

By structuring your parameters and synchronizing them with the placeholders in your batch file, SHIFT lets you use as many parameters as you can fit on a command line.

1.52 SORT

Forms Sort the input from a file:

```
SORT < filespec
SORT -option < filespec
```

Sort the output from a command:

```
COMMAND | SORT
COMMAND | SORT -option
```

Explanation

The SORT filter rearranges the input from a file or the output from a command. SORT does not change the contents of the file; SORT merely changes the way the information is displayed.

Data is sorted using the ASCII collating sequence.

Table 1-21 summarizes the options you can use with SORT.

Table 1-21. SORT Options

Option	Description
-R	Sorts lines in the reverse order of the ASCII collating sequence. Z comes before Y which comes before X, and so on.
-+N	Sorts lines into the ASCII collating sequence based on the character in column N. N must be a whole number.

Examples

The following examples sort a file called GUEST.LST. Note that the heading "Guest List," the line under the heading, and the blank line before the names are also sorted.

```
        Guest List
        ----------

Larry     Irwin
Peter     Glass
Becky     Savage
Meryle    Edwards
Terry     Gibson
Sheila    Escovedo
Tom       Andrews
```

1 sorts the input from a text file into the ASCII collating sequence and displays it on your screen. Lines are sorted beginning with the character in column 1.

```
A>SORT < GUEST.LST
```

Sorts the list into alphabetical order by the first name of each guest:

```
          ----------
          Guest List

Becky     Savage
Larry     Irwin
Meryle    Edwards
Peter     Glass
Sheila    Escovedo
Terry     Gibson
Tom       Andrews
```

```
A>SORT -+10 < GUEST.LST
```

Uses the +N option to sort the list alphabetically by last name. (The guest list was prepared so that the last names began in column 10.)

```
          ----------
Tom       Andrews
Meryle    Edwards
Sheila    Escovedo
Terry     Gibson
Peter     Glass
Larry     Irwin
Becky     Savage
          Guest List
```

The following examples use the output of the DIR command to sort the root directory of drive A. The first example shows the unsorted output of the command.

```
FOG       TXT        64   2-01-1986   2:53a
SAND      TXT      3768   6-18-1986  12:17p
BEACH     LST        81   3-01-1986   3:08p

  Directory of  A:

  Volume in drive A: has no label

           3 Files     241947 bytes free
```

Note that the lines "Directory of...", "Volume in drive...", and "3 Files..." are part of the ASCII collating sequence.

A>DIR | SORT

Sorts the directory alphabetically by filename:

```
          5 Files     241947 bytes free
 Directory of  A:
 Volume in drive A: has no label
%PIPE1    $$$          0   6-10-1986   4:30p
%PIPE2    $$$          0   6-10-1986   4:30p
BEACH     LST         81   3-01-1986   3:08p
FG        TXT         64   2-01-1986   2:53a
SAND      TXT       3768   6-18-1986  12:17p
```

FlexOS creates the $$$ temporary files for piping data between commands. The next time you use DIR, they will be gone from the directory. When piping data between commands, do not run programs that erase of modify these files. This can cause your data to be piped incorrectly.

Note: FlexOS does not create these temporary files if you pipe data between exclusively external commands.

A>DIR | SORT-R

Sorts the directory in reverse alphabetical order:

```
SAND      TXT       3768   6-18-1986  12:17p
FOG       TXT         64   2-01-1986   2:53a
BEACH     LST         81   3-01-1986   3:08p
%PIPE2    $$$          0   6-10-1986   4:30p
%PIPE1    $$$          0   6-10-1986   4:30p
  Directory of  A:
  Volume in drive A: has no label

          5 Files     241947 bytes free
```

A>DIR | SORT-+10

Sorts by the character at column 10 (the column the file extension starts in), and sends the output to the screen. The directory is now sorted alphabetically by file extension:

```
          5 Files     241947 bytes free
%PIPE1    $$$          0   6-10-1986   4:30p
%PIPE2    $$$          0   6-10-1986   4:30p
BEACH     LST         81   3-01-1986   3:08p
FOG       TXT         64   2-01-1986   2:53a
SAND      TXT       3768   6-18-1986  12:17p
  Volume in drive A: has no label
  Directory of  A:
```

A>DIR | SORT -R-+10

Combines the -R and -+N options to do a reverse listing of the characters in the specified column number, in this case sorting the directory by file extension in reverse order:

```
  Directory of  A:
  Volume in drive A: has no label
SAND       TXT      3768   6-18-1986  12:17p
FOG        TXT        64   2-01-1986   2:53a
BEACH      LST        81   3-01-1986   3:08p
%PIPE2     $$$         0   6-10-1986   4:30p
%PIPE1     $$$         0   6-10-1986   4:30p
          5 Files     241947 bytes free
```

A>SORT < PARTTIME.LST >> FULLTIME.LST > PRN:

This example combines redirection symbols and a filter to perform a more sophisticated task. PARTTIME.LST is a list of part-time employees' names; FULLTIME.LST is a list of full-time employees' names. The command sorts PARTTIME.LST alphabetically and adds it to the end of FULLTIME.LST. FULLTIME.LST now contains a list of all the employees' names. This file is redirected to the printer device PRN:.

1.53 SYS

Forms SYS destination_drive:
SYS destination_drive: source_filespec

Explanation

The SYS command copies the FlexOS operating system to a diskette or a hard disk. The destination disk must be bootable. That is, it must be formatted with the FORMAT utility using the -B or -S options, and the directory must have enough contiguous space starting on the first data sector to contain the system image file.

SYS copies BOOTLOAD.SYS (or BOOTLOAD.IMG), the FlexOS **loader image** file and FLEXOS.SYS, the FlexOS **system image** file, and any file that has the System, Hidden, and Read-Only attributes to the specified drive. BOOTLOAD.SYS must contain the proper header information, such as code start and length, and data start and length.

SYS performs the following actions:

1. Reads BOOTLOAD.SYS from the default directory and writes BOOTLOAD.IMG to the first data sectors of the **destination_drive**. BOOTLOAD.SYS is assumed to be a newly created or modified version of the system loader image file. If BOOTLOAD.SYS is not found, SYS copies BOOTLOAD.IMG (the current sector-aligned file image) from the root directory. If SYS cannot find BOOTLOAD.SYS or BOOTLOAD.IMG, you receive an error message.

2. Copies the **source_filespec** to the **destination_drive** and if necessary, renames the file to FLEXOS.SYS. If you do not specify a source filespec, SYS copies FLEXOS.SYS from the default directory to the destination drive. If SYS cannot find the filespec you specify or FLEXOS.SYS, you receive an error message.

Examples

```
A>SYS C:
```

Transfers the FlexOS system image file and all files with the System, Hidden, and Read-Only attributes from the FlexOS system disk in drive A to drive C.

```
A>SYS C: SYSTEM:/OSFILES/FLEXOS.SYS
```

Identifies the FlexOS system file with a filespec. This example copies the FlexOS system file from the filespec A:SYSTEM:/OSFILES/FLEXOS.SYS to drive C. Files with the System, Hidden, and Read-Only attributes are also copied.

1.54 TIME

Forms Set the time:

TIME
TIME hh:mm:ss

Display the time:

TIME -D

Explanation

TIME displays, sets, or changes the time known to the system. When you create or modify a file, FlexOS records the current time in the file's directory entry.

Note: If you are on a multi-user system, only the system manager can set the time. If you are on a single-user system, you can set the time, as well as display it.

Time is expressed in terms of a 24-hour clock where:

- hh is a number 0-23 for the hour.
- mm is a number 0-59 for the minute.
- ss is a number 0-59 for the second.
- xx is a number 0-99 for the hundredths of a second.

Use colons (:) to separate the hours, minutes, and seconds. Use a period (.) to separate the seconds and the hundredths of a second. If you do not use the proper separators, TIME displays an error message.

As long as the digits are within the defined ranges, any time is acceptable to FlexOS. If you enter a valid time, FlexOS accepts it and displays the system prompt. If you enter an invalid time, TIME displays an error message.

Examples

```
A>TIME

Current time is 08:30:00.00.00
Enter new time : 9:53:26
```

Displays the time known to the system. To change the time, enter a new time at the prompt and press Enter. If you do not want to change the time, just press Enter. FlexOS records the time as 09:53:26.00. You do not have to include a leading zero for A.M. settings.

```
A>TIME 13:25
```

Specifies the time from the command line. If you omit any of the TIME components (hours, minutes, and so on), FlexOS fills them in with zeros.

```
A>TIME -D

Current time is 13:25:05.11
```

The -D option only displays the current time; it does not prompt for a new time.

1.55 TREE

Forms
TREE
TREE drive:
TREE -F
TREE drive: -F

Explanation

The TREE command displays all the directory paths on the current or specified drive. TREE also gives you the option of listing the files in each subdirectory.

Examples

A>TREE

Displays the full path of each directory name on the current drive, along with the names of any directories defined within it. The display takes this form:

```
Directory Path Listing.  The volume is INVEST

Path: A:/

Sub-directories: STOCKS
                 BONDS
                 COMMODITIES

Path: A:/STOCKS/

Sub-directories: BLUECHIP

Path: A:/STOCKS/BLUECHIP/

Sub-directories: None

Path: A:/BONDS/

Sub-directories: None

Path: A:/COMMODITIES/

Sub-directories: None
```

A>TREE C:

Specifies the drive from which to read the tree.

A>TREE -F

Lists all the subdirectories on the current drive and lists the files in each subdirectory. In the following display, there is one subdirectory called MYTEST on drive A. MYTEST contains the files GEORGE, THOMAS, and ABRAHAM.

```
Directory Path Listing.  The volume is NAMES

Path: A:/

Sub-directories: MYTEST

Files          : NAMES
                 DATES
                 SCORES


Path: A:/MYTEST/

Sub-directories: None

Files          : GEORGE
                 THOMAS
                 ABRAHAM
```

1.56 TYPE

Form TYPE filespec

Explanation

TYPE displays the contents of a specified text file on the screen. You must enter the filename and the file extension, if there is one. You cannot specify wildcards with TYPE. If you do, you receive an error message.

Typing non-ASCII files, such as object program files, may produce an unreadable display because they contain special characters.

Example

```
A>TYPE B:FUNCTION.LST
```

Displays the contents of FUNCTION.LST on drive B.

1.57 VER

Form VER

Explanation

VER displays the version number of your operating system and the hardware environment you are working with. This information is set by your computer manufacturer.

Note: Do not confuse VER with the VERIFY command.

1.58 VERIFY

Forms
VERIFY
VERIFY ON
VERIFY OFF

Explanation

The VERIFY command makes sure that data is correctly written to the disk and that the data can be read without error.

VERIFY does this whether it is on or off.

1.59 VOL

Forms VOL
VOL drive:

Explanation

VOL displays the volume label of a disk in the current or specified drive. You can write a volume label on your disk using either the DISKSET or FORMAT commands.

Examples

```
A>VOL
```

Displays the volume label of the disk in the current drive (drive A). If the disk has a volume label, VOL displays it like this:

```
Volume in drive A: is RECORDS
```

If there is no volume label on the disk, VOL displays:

```
Volume in drive A: has no label
```

```
A>VOL B:
```

Displays the volume label of the disk in drive B.

End of Section 1

Index

X

Y

Z