TOPS-20 User's Guide

AA-FP69B-TM

June 1988

This document introduces users to the TOPS-20 operating system. It describes how to use the system, obtain system information and run programs.

This document supersedes the document of the same name, order number, AA-FP69A-TM and also the document Getting Started with TOPS-20, order number AA-4187D-TM.

Change bars in margins indicate material that has been added or changed since the previous printing of this manual. Bullets indicate that material has been deleted.

Operating System: TOPS-20 (KL Model B) Version 7.0

Software: TOPS-20 EXEC Version 7.0

First Printing, September 1985
Revised, June 1988

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

CONTENTS

INDEX

FIGURES

TABLES

PREFACE

The TOPS-20 User's Guide describes the functions that you can perform with the TOPS-20 operating system. This manual is the first document of two TOPS-20 user-oriented manuals. The audience for the TOPS-20 User's Guide ranges from the entry level first-time user to the experienced higher level language programmer.

Descriptions of how to use the system, obtain system information, enter programs, run programs, and modify programs have been excerpted from Getting Started with TOPS-20 and incorporated into the TOPS-20 User's Guide. This information is not designated with change bars in the TOPS-20 User's Guide. Only new TOPS-20 operating system features are highlighted with change bars.

Once you learn about the functions described in the TOPS-20 User's Guide, you can refer to the second and more advanced manual, the TOPS-20 Commands Reference Manual, for complete descriptions of all of the TOPS-20 commands and how to use them.

The following suggests a list of chapters to read according to the level of information you need to do your job.

- If you are a first time user, such as a librarian, clerk, or data entry person, read Chapters 1, 2, 3, 8, 10.
- If you are a system administrator, or a new operator, read Chapters 4, 5, 6, 7.
- If you are a programmer, read Chapter 9.

Following is a list of manuals referenced in this manual:

- TOPS-20 Commands Reference Manual
- TOPS-20 User Utilities Guide
- TOPS-20 Tape Processing Manual
- TOPS-20 System Manager's Guide
- EDIT User's Guide
- EDIT Reference Manual
- TV Editor Manual
- EDT-20 Primer
- TOPS-10/TOPS-20 Batch Reference Manual
- TOPS-10/TOPS-20 DECmail/MS Manual

Conventions Used in This Manual

Underlined text	indicates what the user types in command examples.
^letter	means press the keys labeled CTRL and the specified letter simultaneously, for example ^C.
Ellipsis ...	means that items in a command line can be optionally repeated.
<RET>	is implied in command examples.
<ESC>	indicates when you should press the ESCape (or ALTmode) key.

CHAPTER 1

GETTING ON AND OFF THE SYSTEM

This chapter describes:

- Recognizing keyboard symbols (Section 1.1)
- Dialing In (Section 1.2)
- Getting the attention of the system (Section 1.3)
- Getting information about your terminal (Section 1.4)
- Declaring the terminal type (Section 1.5)
- Controlling terminal output (Section 1.5.1)
- Setting the terminal speed (Section 1.5.2)
- Starting a job with LOGIN (Section 1.6)
- Executing commands automatically during LOGIN (Section 1.7)
- Ending a job with LOGOUT (Section 1.8)
- Setting additional terminal parameters (Section 1.9)

1.1 RECOGNIZING KEYBOARD SYMBOLS

You use a terminal to communicate with the system. Although many different types and models of terminals exist, they all have similar keyboards, which resemble typewriter keyboards.

Before you begin using the system, become familiar with the keyboard on the terminal. In addition to the standard characters (letters, numbers, and punctuation) and the space bar, there are keys that perform special functions. Table 1-1 describes these keys and their functions.

Table 1-1: Special Function Keys

Key	Function
CTRL (Control)	The CTRL (or control) key initiates a number of system functions when it is used in conjunction with another character. To type a control character, hold down the CTRL key, and at the same time press the character you want. For example: to type a CTRL/C, hold down the CTRL key and at the same time press the letter C. In most cases this prints (echoes) on your terminal as ^C.
DELETE	The DELETE key erases characters. On some terminals this key is labeled DEL, RUBOUT, RUB CHAR OUT, or with a special symbol.
ESC (Escape)	The ESC (or escape) key initiates a variety of different functions. o Completes an abbreviated command and prompts you with a guideword o Completes an abbreviated argument o Ends input to some system programs o Causes special functions to be performed by some programs At TOPS-20 command level, the ESC key o Does not echo on your terminal o Displays an error message if you have made an error o Rings the terminal bell when you try to use it to complete a command and you have not typed sufficient information At system program level, depending upon the program you are running, the ESC key sometimes echoes on the terminal as a dollar sign. On some terminals this key is labeled ESCAPE, ALT, or ALTMODE.

	If there is no escape key on your terminal, use CTRL/[(press the CTRL and the left square bracket keys at the same time) to duplicate the function of the escape key.
RETURN	The RETURN key confirms to the system that you have completed a line and causes the terminal's cursor or printing head to go to the beginning of the next line. Unless you are told otherwise, terminate all command lines by pressing the RETURN key. On some terminals this key is labeled CR or RET.
SP (Space Bar)	Creates a blank space by moving the terminal printing head one space to the right.
TAB	The TAB key causes the cursor or printing head to move to the right to the next tab stop. Tab stops are normally every eight spaces. This is useful for aligning columns of data and for formatting programs. If there is no TAB key on your terminal, use CTRL/I to duplicate the function of the TAB Key.

1.2 DIALING IN

Some terminals are connected to the computer by telephone. If you are using such a terminal, find out the computer phone number and use the following procedure:

1. Turn on the terminal.
2. Check the speed setting. (Refer to Section 1.5.2, for information on setting your terminal speed.)
3. Dial the computer telephone number.
4. Wait for a steady tone or a high-pitched beep, which indicates that the telephone connection to the computer has been made.
5. Place the telephone receiver in the slots in either the terminal or the acoustic coupler. (An acoustic coupler is a device to connect the telephone with a terminal if the terminal does not have a built-in telephone receptacle.)
6. Wait for the carrier detect light to come on.

Your terminal is now connected to the computer. The system prints a system identification message similar to the following:

```
KL2102, TOPS-20 Development Sys., TOPS-20 Monitor 7(7)
@
```

The @ character, which is the TOPS-20 prompt, indicates that TOPS-20 is ready to accept a command.

1.3 GETTING THE ATTENTION OF THE SYSTEM

Press any key on the keyboard to signal the system that you want to log in. After you press a key, a system identification message and the TOPS-20 prompt, @, are printed on the terminal.

If you do not receive the system identification message, one of the following conditions exists:

- The system is down
- Your terminal is set at the wrong speed for the line you are connected to (refer to Section 1.5.2 for information on setting the terminal speed)
- The system is not available for your use
- The system is full
- Your terminal is not connected to the system

If the system is not available for your use, you receive a message similar to the following:

```
?LOGGING IN ON LOCAL TERMINALS IS CURRENTLY NOT ALLOWED
```

This message means that the operator has set the system to prevent timesharing. The system notifies you when it resumes its timesharing operation by printing a message similar to the following:

```
SYSTEM RESTARTING, WAIT...
```

and after a pause,

```
[FROM OPERATOR:  SYSTEM IN OPERATION]
```

If the system is full, you receive the following message:

```
?FULL reason
```

Wait a few minutes; then press a key. Repeat this until you receive the system identification message. The explanation that follows ?FULL is meaningful to the system manager and to system programmers. If you must wait an excessive length of time before successfully logging in, you might want to bring the error message to the attention of one of these people.

1.4 GETTING INFORMATION ABOUT YOUR TERMINAL

Terminals have different characteristics for printing information, depending on their type and speed. Because you have not yet told the system the kind of terminal you are using, the system automatically sets defaults for the terminal. These defaults are based on the most common type of terminal at your site. The defaults set parameters such as the terminal page length at 66 lines and the line width at 72 characters, in addition to setting lowercase and tabs. The INFORMATION TERMINAL command displays the settings of these parameters or values, along with other characteristics of your terminal.

After the system prints the system identification message and the TOPS-20 prompt (@), you are at TOPS-20 command level and you can give commands to the system. Type the TOPS-20 command INFORMATION TERMINAL-MODE and press RETURN. The system prints the information about your terminal.

```
@INFORMATION (ABOUT) TERMINAL-MODE (FOR TERMINAL)
 TERMINAL SYSTEM-DEFAULT
 TERMINAL SPEED 9600
 TERMINAL NO INHIBIT (NON-JOB OUTPUT)
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 RECEIVE USER-MESSAGES
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL NO PAUSE (ON) END-OF-PAGE
 TERMINAL LENGTH 66
 TERMINAL WIDTH 72
 TERMINAL LOWERCASE
 TERMINAL RAISE
 TERMINAL NO FLAG
 TERMINAL INDICATE
 TERMINAL NO FORMFEED
 TERMINAL NO TABS
 TERMINAL NO IMMEDIATE
 TERMINAL FULLDUPLEX
```

Note that you can specify a terminal line number after the (FOR TERMINAL) guidewords. This allows you to obtain information about another user's terminal. The system uses your terminal line number as the default when you do not specify one. The SYSTAT command (discussed in Section 3.1) shows the line numbers for all users on the system.

1.5 DECLARING THE TERMINAL TYPE

Once you are at TOPS-20 command level, you can inform the system of the type of terminal you are using.

Terminal Types Recognized by the System

HARD COPY	VIDEO
MODEL 33	H19
MODEL 35	TERMINET
MODEL 37	TI
EXECUPORT (TI)	VT05
LA30	VT50
LA36	VT52
LA38	VT100
LA120	VT102
	VT105
	VT200-SERIES
	VT300-SERIES

NOTE

Installations can add other terminals to their individual systems.

To declare the terminal type, give the TERMINAL command, and type in the type of your terminal. In this example, the terminal type is a VT100.

```
@TERMINAL (FEATURE OR TYPE) VT100
```

After you identify the terminal type to the system, all subsequent output conforms to preset terminal parameters for that type. The terminal type specifies the proper values for:

- Formfeed
- Tab
- Outputting lowercase characters
- Line width
- Page length

If you do not set the proper parameters for the terminal, you may find the output format undesirable for your work.

After you identify the terminal type, you can again give the INFORMATION TERMINAL-MODE command to see the parameters that were set as a result of your TERMINAL command.

Tell the system you are using a VT100 by giving the TERMINAL command; then give the INFORMATION TERMINAL-MODE command.

```
@TERMINAL (FEATURE OR TYPE) VT100
@INFORMATION (ABOUT) TERMINAL-MODE (FOR TERMINAL)
 TERMINAL VT100
 TERMINAL SPEED 9600
 TERMINAL NO INHIBIT (NON-JOB OUTPUT)
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 RECEIVE USER-MESSAGES
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL NO PAUSE (ON) END-OF-PAGE
 TERMINAL LENGTH 24
 TERMINAL WIDTH 80
 TERMINAL LOWERCASE
 TERMINAL NO RAISE
 TERMINAL NO FLAG
 TERMINAL INDICATE
 TERMINAL NO FORMFEED
 TERMINAL TABS
 TERMINAL NO IMMEDIATE
 TERMINAL FULLDUPLEX
```

Setting the terminal type changes only the following parameters: terminal type, length, width, lowercase, formfeed, and tab. Therefore, when you identify the terminal as a VT100, the output conforms to the parameters for that type of terminal, that is, a page length of 24 lines, a line width of 80 characters, lowercase letters, no mechanical formfeed, and no mechanical tabs.

Identifying the terminal type for a video terminal additionally allows more effective use of the DELETE key. The system erases the last character you typed on the screen rather than print the character followed by a backslash, as it does on a hard-copy terminal.

1.5.1 Controlling Terminal Output

The following commands control output to terminals:

TERMINAL PAUSE (ON) COMMAND

TERMINAL PAUSE (ON) END-OF-PAGE

TERMINAL PAUSE (ON) CHARACTER x (AND UNPAUSE ON) y

TERMINAL NO PAUSE (ON) END-OF-PAGE

The TERMINAL PAUSE COMMAND allows you to stop output to the terminal at any time by typing CTRL/S, and continue output by typing CTRL/Q. This command is the default for all terminal types. You can define your own characters to stop and continue output with the TERMINAL PAUSE CHARACTER command discussed below.

TERMINAL PAUSE END-OF-PAGE automatically stops output to the terminal when the output is equal to the current page length set for the terminal. When the system stops the output, it rings the terminal bell and waits for you to type CTRL/Q. The CTRL/Q resumes the output. This prevents the output from rolling off a video terminal screen so rapidly that you cannot read it. However, if you want to stop the output before the end of the page, type CTRL/S. This command is the default if you declare your terminal to be a video terminal, for example a VT100.

TERMINAL NO PAUSE END-OF-PAGE prevents the output from stopping at the end of the page. This command is the default if you declare your terminal to be a hard-copy terminal, for example an LA36.

If TERMINAL PAUSE END-OF-PAGE is not set, and you need the terminal output to stop at the end of a page, give the following command:

@TERMINAL PAUSE (ON) END-OF-PAGE

If TERMINAL PAUSE END-OF-PAGE is set, and you do not want the terminal to stop output at the end of the page, give the following command:

@TERMINAL NO PAUSE (ON) END-OF-PAGE

TERMINAL PAUSE CHARACTER x y allows you to choose your own pause and continue characters. These characters are alternatives to the CTRL/S and CTRL/Q default characters. (To specify your own pause and continue characters, TERMINAL PAUSE END-OF-PAGE and TERMINAL PAUSE COMMAND must be in effect.)

You can specify the pause and continue characters in several ways. Some of the more common forms are:

- an ASCII code in octal
- a character within double quotation marks (" ")
- the word SPACE to specify the space bar

Octal ASCII codes for the keyboard characters are listed in several TOPS-20 manuals. The TOPS-10/TOPS-20 Batch Reference Manual, for example, lists these codes.

To specify the space bar as both the pause and continue character, give the following command:

```
@TERMINAL PAUSE (ON) CHARACTER SPACE (AND UNPAUSE ON) SPACE
```

To see the characters that you may have specified in the TERMINAL PAUSE CHARACTER command, give the INFORMATION TERMINAL-MODE command:

```
@INFORMATION (ABOUT) TERMINAL-MODE (FOR TERMINAL)
 TERMINAL VT100
   .
   .
   .
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL PAUSE (ON) END-OF-PAGE
 TERMINAL PAUSE (ON) CHARACTER SPACE
   .
   .
   .
```

In this example, the continuation character is not displayed, because it is the same as the pause character (SPACE). Also, if you specify the TERMINAL NO PAUSE COMMAND or the TERMINAL NO PAUSE END-OF-PAGE command, or if the system default characters, CTRL/S and CTRL/Q, are in effect, the TERMINAL PAUSE CHARACTER line does not appear in the information display.

NOTES

Several terminal types require that you change the pause and continue characters to something other than CTRL/S and CTRL/Q. For example, the VT125 and the VT100 with the printer port option do not recognize these characters.

When you use the SET HOST command to log in to a remote system, CTRL/S and CTRL/Q are reserved by your host system; they are not passed to the remote system. CTRL/A is the default character for pausing and continuing output coming from a remote system.

1.5.2 Setting the Terminal Speed

Terminals can transmit and receive data at various speeds. This rate of speed is called a baud rate. Baud rates range from 10 to 960 characters per second: 10 characters per second is 110 baud; 960 characters per second is 9600 baud.

There are actually two different speeds: terminal speed and line speed. The terminal speed is the speed at which your terminal receives characters from and transmits characters to the system. This speed is set by switches or keys that are physically located on your terminal. The line speed is the speed at which the system receives characters from and transmits characters to your terminal. The line speed is set with the TERMINAL SPEED command. The terminal speed and the line speed must match for your terminal to communicate with the system.

Your system can have two types of terminal lines, those that are set to a certain speed and "autobaud" lines. An autobaud line automatically sets a line speed that matches the speed of your terminal when you initially type any key on the keyboard.

Your system manager presets line speeds when the type of terminal connected to the terminal line is constant. For example, a terminal line connected to a VT220 video terminal may be set to 9600 baud while a line connected to a slower LA100 hard-copy terminal may be set to 300 baud. Terminal lines are autobaud when the line can be connected to various types of terminals. For example, terminal lines which are reserved for telephone connections to the computer are usually autobaud.

NOTE

If your terminal is connected by telephone to an autobaud terminal line, an initial character enables the system to determine your terminal's baud rate, provided the rate is 300, 1200, 1800, 2400, or 9600. If the baud rate is 110 or 150, type a second character. If you press a character and fail to get the system identification message, press the BREAK key twice followed by another character.

Do not set the line speed to a speed your terminal (or modem) does not support. If you should do this by mistake, contact the operator for assistance.

To change your terminal and line speeds, first change your line speed with the TERMINAL SPEED command. Then, manually change the speed settings on your terminal.

For example, to change the line speed for input and output to 2400 baud, give the TERMINAL SPEED command:

@TERMINAL (FEATURE OR TYPE) SPEED (OF INPUT) 2400

NOTE

On some hard-copy terminals, the switch to change the baud rate is located at the left of the keyboard.

On some video terminals, the switch to change the baud rate is located on the underside or the back of the terminal. On others, special keys on the main keyboard are used to change the baud rate.

If you set only the input speed for the line and do not specify the output speed, the system assumes that the output speed is the same as the input speed.

If you are using a hard-copy terminal and accidentally set a line speed incompatible with your terminal, you cannot correct it. Contact the operator, give your terminal line number, and ask him to set your line at the speed you want.

If you are using a video terminal and accidentally set an incorrect line speed, you may be able to correct the speed by setting the terminal speed to the current line speed and then, resetting the line and terminal speeds.

After you start a job on the system, you may find there are more terminal parameters you need to set in addition to those already described. Section 1.6 describes starting a work session with LOGIN. Section 1.9 explains the additional parameters you can set.

1.6 STARTING A JOB WITH LOGIN

Before using TOPS-20 for the first time, you must obtain the following from the staff at your installation.

1. Your user name
2. Your password
3. Your account

Your user name, password, and account identify you so that you can use the computer and be charged appropriately.

To start working on the system, you must first identify yourself to the system by typing the LOGIN command, which validates you as a user, creates your job, and begins charging your account. The LOGIN command requires your user name, password, and account. The command also allows you to add remarks concerning the work session. This identification procedure is called logging in. After you give the LOGIN command, the system creates a job and prints a line containing the job number, the terminal number, the current date and time and the date and time of your last login. The system prints an @ on the next line; you are now at TOPS-20 command level.

TYPING ERRORS

> If, in the process of logging in, you make a typing error, type CTRL/U. This tells the system to ignore everything you have typed on that line, because you have made a mistake and want to start the line over. After you type a CTRL/U, the system prints XXX and then prints @ on the next line.

After the @ prompt, do the following:

1. Type LOGIN, and press the key labeled ESC (for ESCape).

```
          <ESC>
            |
@LOGIN (USER)
```

2. After you see the guideword (USER), type your user name and press the ESC key.

```
          <ESC>          <ESC>
            |              |
@LOGIN (USER) SARTINI (PASSWORD)
```

3. After you see the guideword (PASSWORD), type your password, and press the ESC key. Because your password is secret, it does not print on the terminal. This safeguard prevents other people from using your name and account. Even though your password is not printed, it is given to the system as part of your identification.

```
          <ESC>          <ESC>      <ESC>
            |              |          |
@LOGIN (USER) SARTINI (PASSWORD) (ACCOUNT)
```

NOTE

> On some terminals, the guideword (PASSWORD) may be followed by a nonsense word or message. If this is the case, when you type your password over this word, your password is illegible.

4. After you see the guideword (ACCOUNT), type your account and, instead of pressing the ESC key, press the key labeled RETURN. You use the RETURN key to tell the system you have finished typing the lines. TOPS-20 will print a message similar to the one below.

```
@LOGIN (USER) SARTINI (PASSWORD) (ACCOUNT) 341
 Job 40 on TTY127 6-Feb-88 08:42:47
@
```

This message gives you:

- Your system assigned job number (40).
- Your terminal number (127).
- The current date and time (6-Feb-88).
- A system message of the day, if any. Installations use the message of the day to inform users of new programs or system changes.

NOTE

Some systems do not require you to enter an account when logging in. If you don't have an account, press the RETURN key after you type your password. You will be logged in.

The following example shows the entire logging-in process:

```
AURORA, Research and Development, TOPS-20 Monitor 7(7)
@LOGIN (USER) SARTINI (PASSWORD) (ACCOUNT) 341
Job 57 on TTY127 23-Jul-88 09:48:40, Last Login 22-Jul-88
09:30:27
@
```

NOTE

You do not have to use the ESC key when logging in. However, the ESC key provides guidewords that prompt you for user name, password and account. Spaces between arguments are sufficient if you do not need the help of guidewords. For example: LOGIN SARTINI password 341.

1.6.1 User Names

Your user name identifies you to the system and to other users. A user name may contain up to 39 alphanumeric characters, as well as period (.) and hyphen (-). Your user name is also the name of your login directory.

1.6.2 Passwords

To provide security, you must give a password when logging in. Depending on the procedures at your site, you may be assigned a password or allowed to select one for your first login. When you type your password, it is not displayed on the terminal; this prevents others from learning it and logging into your area without your authorization.

1.6.2.1 Selecting Secure Passwords - Use these guidelines in selecting a password:

- Use a minimum of six characters. Unless your system manager sets a greater minimum password length, a password of at least six characters is recommended. Passwords can be up to 39 characters long and include hyphens.

- Use a password that cannot easily be guessed. Avoid passwords that have a personal association to you such as your name or initials, the name of a family member or pet, the make of your car, or any name associated with your work, such as your company or special project.

- Avoid words found in the dictionary. By avoiding words readily found in the dictionary your password choice is less subject to discovery by a program that successively enters the words in the dictionary, searching for one that produces a successful login. Use a nonsense word or a word from another language.

- Include digits in a password. The content of a password is more important than the length. Using digits as well as letters provides the most secure passwords. For example, for a six-character password using letters only, there are 300 million combinations, while a six-character password with digits has 2 billion combinations.

1.6.2.2 Keeping Your Password a Secret – Often illegal system accesses involving the use of a correct password can be traced to disclosure of the password by its owner. Do not be unconcerned about protecting your password because you do not keep any sensitive information on the system. A system breaker could use your password to gain more information about the system and break into other areas, or a malicious user could destroy your files or steal computer time.

Use these guidelines to prevent others from learning your password:

- Never write down your password.
- Do not include your password in any file, including the body of an electronic mail message. (If anyone else reveals their password to you in this fashion, be sure to delete the information promptly.)
- Never give your password to other users except under very unusual circumstances, and then be sure to change it immediately after the need for sharing has passed.
- Avoid using the same password for your accounts on multiple systems. The system breaker's first step after learning a password for one system is to try that username and password on other systems.
- Note the date and time of your last login. After you give the LOGIN command, the system displays the date and time of your last login. Check this message routinely. If you observe a login that you did not make, change your password immediately and notify your system manager.
- Change your password frequently. Changing your password every 3 to 6 months is sufficiently frequent on most systems where there have been no password compromises and no sharing of passwords. DIGITAL discourages sharing passwords; however, if passwords are shared, the frequency of password changes should be every month or two. To change your login password use the SET DIRECTORY PASSWORD command:

```
@SET DIRECTORY PASSWORD <login-directory-name>
```

1.6.3 Accounts

To log in to the system, you must give a valid account. Your account is billed for central processor unit (CPU) usage and for file storage.

Once you log in, all charges are made to the account you give in the LOGIN command unless you specify otherwise. If your login directory has a default account, you do not have to specify an account when you login. If you must change your account during a job, give the SET ACCOUNT command or include the ;A attribute in the file specification. (Refer to Section 4.2.7, file attributes.) However, you can change it only to another valid account.

1.6.4 Session-Remark

The LOGIN command allows for an optional argument following your account. If you press the ESC key after typing your account, the system prints the guidewords (SESSION-REMARK). You can then type one line of text to identify a specific work session for accounting purposes. This session remark cannot exceed 39 alphanumeric characters, including hyphens and spaces. If you need to change the SESSION-REMARK during a job, give the SET SESSION-REMARK command.

You can see the current session-remark for your job when you give the INFORMATION JOB-STATUS command.

1.7 EXECUTING COMMANDS AUTOMATICALLY DURING LOGIN

You can create a LOGIN.CMD file that contains the TOPS-20 commands you want executed when you log in. The system automatically reads this command file every time you log in. After executing these commands, the system prints any output from the commands followed by the message End of LOGIN.CMD and the TOPS-20 prompt (@).

For example, if you always use a VT100 terminal, you can include a TERMINAL VT100 command in a LOGIN.CMD file. Every time you log in, the system reads the LOGIN.CMD file and recognizes the terminal as a VT100. All output to the terminal conforms to the parameters set for a VT100. Below is an example of a typical LOGIN.CMD file. Note that comments are preceded by an exclamation mark (!). (Refer to Section 2.6 for information on adding comments.) The commands in this file are discussed in the following chapters.

```
TERMINAL VT100                          !Set the parameters for a VT100
TERMINAL PAUSE CHARACTER SPACE SPACE    !Set the space bar to stop and
                                        ! start terminal output
TERMINAL PAUSE END-OF-PAGE              !Stop scrolling output at end
                                        ! of page

DEFINE WK: WORK:<LEOPOLD>               !Define a logical name for a
                                        ! directory
DEFINE EDITOR: SYS:EDT.EXE              !Define a logical name for
                                        ! an editor
```

```
DEFINE PB: PS:PHONE.BOOK              !Define a logical name for a file
INFORMATION LOGICAL-NAMES             !Display the logical names defined
                                      ! in the three previous commands
MOUNT STRUCTURE WORK: /NOWAIT         !Mount the structure WORK:
SET PROGRAM DSR KEEP CONTINUE         !Keep program DSR when it's
                                      ! started
SET ALERT 16:25 VANPOOL IN 5 MINUTES  !Set a daily reminder
DAYTIME                               !Display the date and time
```

If there is an error with one of the commands, the system processes the commands up to the one in error. When the system encounters the error, it stops reading the file and prints the following message:

```
%Error while reading LOGIN.CMD.1, file aborted.
```

followed by the message produced by the command in error.

You can also create a COMAND.CMD file that contains any TOPS-20 commands you want executed when you log in. The COMAND.CMD file differs from the LOGIN.CMD file because the system automatically reads the COMAND.CMD file whenever you give a PUSH command as well as every time you log in. (Refer to Section 8.6 for an example using the PUSH command.) After executing the commands in the COMAND.CMD file, the system prints any output from the commands followed by the message End of COMAND.CMD and the TOPS-20 prompt.

Note that the system reads the LOGIN.CMD file before it reads the COMAND.CMD file. If there are conflicting commands in the two files, the last command executed (that is, the one in the COMAND.CMD file) takes precedence.

NOTE

> The system processes the LOGIN command line or the PUSH command before it reads the LOGIN.CMD file or the COMAND.CMD file. Therefore, you are still successfully logged into the system or the PUSH command is still in effect, even if the command file contains an error.

Your system manager can create system-wide LOGIN.CMD and COMAND.CMD files. Like your own command files, the system LOGIN.CMD and COMAND.CMD files are executed automatically when you login. Each system command file is executed before your own file of the same name:

1. SYSTEM:LOGIN.CMD
2. LOGIN.CMD
3. SYSTEM:COMAND.CMD
4. COMAND.CMD

If your site has system-wide LOGIN.CMD and COMAND.CMD files, you should examine the commands in these files to avoid putting duplicate commands in your own command files. To display the system LOGIN.CMD file give the command:

```
@TYPE SYSTEM:LOGIN.CMD
```

Refer to Chapter 5 for information on how to create files.

1.8 ENDING A JOB WITH LOGOUT

When you want to leave the system, you should not just turn off your terminal and walk away; you should tell the system you are leaving. To leave the system, type LOGOUT after the @, and press the RETURN key. This terminates your communication with the system. This procedure is called logging out.

```
@LOGOUT
```

After you press the RETURN key, you will see a message similar to:

```
Killed Job 57, User SARTINI, Account 341, TTY 127,
 at 23-Mar-88  09:49:36,  Used 0:0:14 in 1:25:56
```

This message indicates that you have successfully logged off the system. Your job number was 57, your user name was SARTINI, your account was 341, the terminal you were using was connected to terminal line 127. You left the system at 09:49:36 on March 23, 1988. The last part of the message indicates how long the system actually worked for you (14 seconds) and how long you were logged in (1 hour, 25 minutes, and 56 seconds).

If you do not log off the system, your terminal will not be free for another user. Also, someone can come along and do work on the system under your identification, and you will be charged for the computer use.

If you type a character to get the system's attention and fail to log in within 5 minutes, the system automatically logs you off the system and prints the LOGOUT message. This message is similar to the following:

```
 Autologout
Killed Job 8, TTY 26,
 at 23-Mar-88   10:50:35,   Used 0:0:0 in 5:15
```

If you are on a dial up line, the system hangs up the line.

1.9 SETTING ADDITIONAL TERMINAL PARAMETERS

After you log in to the system, you may find you need to set additional terminal parameters for your work. The following sections describe more parameters you can set. For a complete description of all parameters you can set with the TERMINAL command, refer to the TOPS-20 Commands Reference Manual. If you are reading this manual for the first time, you can skip these sections until later.

1.9.1 Setting the Terminal Page Length

When you declare the terminal type, the system sets a page length for the terminal. The length of the page varies depending on the type of terminal. To change the page length, give the TERMINAL LENGTH command.

The system uses the page length to determine where to stop terminal output when TERMINAL PAUSE END-OF-PAGE is set. The page length is also important when using formfeeds.

To change the page length to 30, give the following command.

```
@TERMINAL (FEATURE OR TYPE) LENGTH (OF PAGE IS) 30
```

1.9.2 Setting the Terminal Line Width

The system sets a line width for the terminal when you identify the terminal type. To change the line width, give the TERMINAL WIDTH command. The width can be set at a minimum of 8 characters per line to a maximum of 255 characters per line. To change the line width to 50, give the following command.

```
@TERMINAL (FEATURE OR TYPE) WIDTH (OF LINE IS) 50
```

If a line of input or output on your terminal exceeds the width set for the terminal, the system prints the maximum number of characters on one line and continues printing on the following lines. This can affect the number of lines the system prints when page mode is set.

1.9.3 Using Formfeeds

On a hard-copy terminal with a mechanical formfeed, the system advances the paper to the top of the next page by outputting a formfeed character (CTRL/L). On a hard-copy terminal without a formfeed mechanism, the system can simulate a formfeed by outputting the proper number of linefeeds. Usually the system prints ^L instead of advancing the paper.

To advance the paper to the top of the next page and prevent the ^L from printing, give the TERMINAL NO INDICATE command. Use this command to print a memo, report, or information that you want to appear on individual pages.

@TERMINAL (FEATURE OR TYPE) NO INDICATE (FORMFEED)

When you declare the terminal type, the system simulates formfeeds if they are required by the terminal. You can also use the TERMINAL NO FORMFEED command to force the system to simulate formfeeds regardless of the terminal type.

CHAPTER 2

COMMUNICATING WITH THE SYSTEM

This chapter describes:

- Using TOPS-20 commands (Section 2.1)
- Obtaining a list of TOPS-20 commands (Section 2.2)
- Obtaining information about the parts of a command (Section 2.3)
- Typing commands (Section 2.4)
- Continuing commands (Section 2.5)
- Adding comments to command lines (Section 2.6)
- Correcting input errors (Section 2.7)
- Setting alerts (Section 2.8)
- Operating system stoppage (Section 2.9)

2.1 USING TOPS-20 COMMANDS

A TOPS-20 command is an instruction that specifies the function you want the TOPS-20 operating system to perform. By giving TOPS-20 commands you accomplish your work through the operating system.

Each TOPS-20 command contains one or more of the following parts:

1. Command name
2. Guidewords
3. Arguments
4. Switches
5. Subcommands
6. Command terminator

The command name identifies the command and its function. Guidewords can assist you in identifying the argument you should type. (Guidewords are always printed within parentheses.) An argument is the response you enter after a guideword. This argument further identifies the information the system needs to process the command. Switches and subcommands allow you to select more precise options to a given command. Using a switch or a subcommand, you can also override default options that are part of the command. Use a carriage return to end a command.

Before doing anything more, try typing a few easy commands. TOPS-20 recognizes many commands, but this manual discusses only some commonly used commands. Appendix A contains a list of TOPS-20 commands, and their meanings. The TOPS-20 Commands Reference Manual describes all of the commands available to the nonprivileged user of TOPS-20.

You type a TOPS-20 command directly after the system prints the @ prompt; you end a TOPS-20 command by pressing the RETURN key. With some commands, you must type one or more arguments before you press the RETURN key. For example, the LOGIN command described earlier requires your user name, your password, and your account as arguments. The system tells you that it requires an argument by printing a guideword in parentheses after you press the ESC key. Some commands, such as DAYTIME, do not require arguments.

To find out today's date and time, type DAYTIME after you see the @, and then press the RETURN key.

```
@DAYTIME
 Thursday, May 26, 1988 08:41:21
@
```

The system prints the date in the format:

day-of-the-week, month day-of-the-month, year

The system prints the time of day in the format:

hours:minutes:seconds

The hours are given using a 24-hour clock. The time shown in the above example (08:41:21) is 21 seconds after 8:41 in the morning. Twelve midnight is displayed as 00:00:00, twelve noon is displayed as 12:00:00; and seven o'clock in the evening is displayed as 19:00:00.

Other commands require one or more arguments. Arguments can be letters, numbers, or a combination of both. A common argument is a file specification. (Refer to Section 4.2 for a description of file specifications.) To find out which kind of argument you should type, press ESC after you give the command. The system prints the guideword, prompting you for the kind of argument to type. If the command does not need an argument, when you press ESC, the system rings the terminal bell. The following example illustrates the DIRECTORY command followed by the guidewords (OF FILES) and the filename TEST.FOR as the argument:

```
@DIRECTORY (OF FILES) TEST.FOR
```

Some commands accept switches while others accept subcommands. With switches and subcommands, you can be more specific about what you want the command to do.

A switch is a slash followed by an option. The option may be followed by a colon and an argument. Switches specify details about the action of the given command. You can give one or more switches to a command by typing them on the same line as the command. To include a switch, type a slash (/), followed by the option. Some options require that a value, preceded by a colon, also be given. The following example shows the use of a single switch and its value to print four copies of the file TEST.FOR.3:

```
@PRINT (FILES) TEST.FOR.3/COPIES:4
[Job TEST Queued, Request-ID 41, Limit 27]
```

A subcommand resembles a switch in its function. The difference between switches and subcommands is the syntax. While you enter switches on the same line as the command, you enter each subcommand on a separate line following the command line.

To include subcommand(s), end the command line by typing a comma, and press RETURN. The system prints the subcommand level prompt, @@, to indicate that you can now type subcommands. Subcommands, like TOPS-20 commands, contain subcommand names, guidewords, and arguments of their own. You can give several subcommands, but each one must be typed on a separate line. To end each subcommand, press RETURN. After you type your last subcommand, press RETURN; the system prints @@; press RETURN again. The system then processes the command and its subcommand(s). When the system prints the single @ you are back at TOPS-20 command level. The following example demonstrates the use of a single subcommand to the DIRECTORY command:

```
@DIRECTORY (OF FILES),
@@DELETED (FILES ONLY)
@@

   PS:<PORADA>
  TEST.FOR.2
      .QOR.1
      .REL.3
Total of 3  files
```

Each part of a TOPS-20 command or subcommand is referred to as a field and is separated from each adjacent field by a space. Figure 2-1 shows the fields of the LOGIN command.

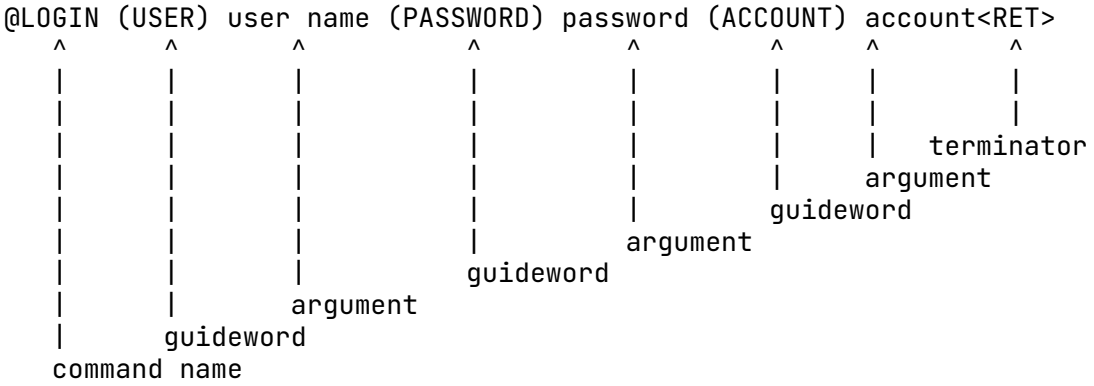


Figure 2-1: Fields of a Command

2.2 OBTAINING A LIST OF TOPS-20 COMMANDS

After the system outputs an @, you can type a question mark (?) to print the list of TOPS-20 commands.

```
 GIDNEY, TOPS-20 Development System, TOPS-20 Monitor (6012)
@? Command, one of the following:
 ACCESS        ADVISE         APPEND        ARCHIVE     ASSIGN
 ATTACH        BACKSPACE      BLANK         BREAK       BUILD
 CANCEL        CLOSE          COMPILE       CONNECT     CONTINUE
               .
               .
               .
 TAKE          TALK           TDIRECTORY    TERMINAL    TRANSLATE
 TYPE          UNATTACH       UNDECLARE     UNDELETE    UNKEEP
 UNLOAD        VDIRECTORY
  or system program name
```

To stop the printing of this list, type two CTRL/Cs, which returns you to TOPS-20 command level.

Appendix A gives a brief description of each command. The TOPS-20 Commands Reference Manual contains a complete description of all TOPS-20 Commands available to the nonprivileged user.

If you remember that a command begins with a certain letter or letters, type the letters that you recall, and then type ?. TOPS-20 prints the list of commands you could type using those letters. It then prints what you have typed so far and waits for you to finish the command. In the example below, you remember that the command you want begins with the letter A. You type A, followed by a question mark. TOPS-20 prints the names of all the commands beginning with the letter A and possible filenames from the (SYS:) system directory, and waits for you to complete the command or filename.

```
@A? Command, one of the following:
 ACCESS    ADVISE     APPEND     ARCHIVE    ASSIGN     ATTACH
  or system program name
 @ACCESS (TO DIRECTORY) <MORRILL>
 Password:
 @^C
```

When typing a question mark, you are not limited to just one letter; you may type as many as you need.

```
@CON? Command, one of the following:
 CONNECT    CONTINUE
  or system program name
 @CONNECT (TO DIRECTORY) <MORRILL>
 Password:
 @
```

The CONNECT command joins you to another user's files.

2.3 OBTAINING INFORMATION ABOUT THE PARTS OF A COMMAND

You can type a question mark following a command or subcommand to print a list of possible arguments for the command. For example, type the TERMINAL command followed by a question mark. You do not have to press the RETURN key. TOPS-20 lists the possible arguments, prints the command up to the point at which you typed ?, and waits for you to enter a valid argument.

```
@TERMINAL (FEATURE OR TYPE) ? one of the following:
 FLAG             FORMFEED          FULLDUPLEX      HALFDUPLEX
 HELP             IMMEDIATE         INDICATE        INHIBIT
 LENGTH           LINE-HALFDUPLEX   LOWERCASE       NO
 PAGE             PAUSE             RAISE           RECEIVE
 SPEED            TABS              TYPE            WIDTH
  or one of the following:
 33               35                37              EXECUPORT
 H19              LA120             LA30            LA36
 LA38             SYSTEM-DEFAULT    TERMINET        TI
 VK100            VT05              VT100           VT102
 VT125            VT131             VT200-SERIES    VT300-SERIES
 VT50             VT52
@TERMINAL (FEATURE OR TYPE)
```

Give the LENGTH argument, and press ESC. The system prints (OF PAGE IS):

```
@TERMINAL LENGTH (OF PAGE IS)
```

Type another question mark to find out which argument the system expects you to give. The system prints "Length of page in decimal" and reprints the command.

```
@TERMINAL LENGTH (OF PAGE IS) ?  Length of page in decimal
@TERMINAL LENGTH (OF PAGE IS)
```

Choose a number (the example uses 20); type it in and press RETURN.

```
@TERMINAL LENGTH (OF PAGE IS) 20
```

Some commands do not require arguments. If you type a command followed by a question mark and that command does not require further arguments, the system prints the message "Confirm with carriage return." This informs you that you are at the end of the command. Press RETURN to confirm the command and to have the system perform the function you requested.

```
@DAYTIME ? Confirm with carriage return
@DAYTIME
```

In addition, the question mark can be used to list the subcommands and switches of a command. To list the subcommands of a command, type a question mark at subcommand level (indicated by @@). The system prints the list of subcommands. For example, type the DIRECTORY command followed by a comma, and press RETURN. When you receive the @@, type a question mark.

```
@DIRECTORY (OF FILES) ,
@@? confirm with carriage return
  or one of the following:
 ACCOUNT                      ALPHABETICALLY
 ARCHIVE                      BEFORE
     .
     .
     .
 SMALLER                      TIMES
 USER
@@
```

To list the switches of a command, type the command; type a slash followed by a question mark. The system prints the list of switches for that command. Remember that all switches begin with a slash. For example, type the PRINT command, followed by a slash and a question mark.

```
@PRINT (FILES) /? /SPOOLED-OUTPUT
  or Job switch, one of the following:
 /ACCOUNT:          /AFTER:              /DESTINATION-NODE:
     .
     .
     .
 /REPORT:      /SPACING:
@PRINT (FILES)/
```

2.4 TYPING COMMANDS

You can type TOPS-20 commands to the system by using either full input, recognition input, abbreviated input, or a combination of these methods.

The LOGIN command, which identifies you to the system, is used in Sections 2.4.1 through 2.4.3 to demonstrate full, recognition, and abbreviated input.

2.4.1 Full Input

To give a command using full input, type the complete command, using a space to separate the fields. To log in using full input, type the complete LOGIN command line.

```
@LOGIN SARTINI ___ 341
```

2.4.2 Recognition Input

To give a command using recognition input, type a portion of the command and press ESC. In order for the system to distinguish this command from other commands, you must type enough of the command to make it unique. The system responds in one of the following ways:

1. Prints as much of the command as the system can recognize.
2. Prints the remainder of the command name.
3. Prints a guideword.
4. Prints the remainder of the argument.
5. Rings the terminal bell, indicating that you need to type more information.
6. Prints an error message.

Continue typing and pressing ESC until the command is complete. Recognition input requires less typing than full input, so you are less likely to make a mistake.

To log in using recognition input, type LOG and press ESC; the system finishes the LOGIN command by printing IN and the guideword (USER). You can also use recognition on your user name. (Here the user name is SARTINI.) Type SAR and press ESC; the system finishes the user name by printing TINI and the guideword (PASSWORD). Type the complete password (it is not printed) and press ESC; the system prints (ACCOUNT). Type the account (here it is 341) and press RETURN.

In the following example, type the underlined portions of the command. At the point where the underlining stops, press ESC.

```
       <ESC>        <ESC>             <ESC>
         |            |                 |
@LOGIN (USER) SARTINI (PASSWORD)___(ACCOUNT) 341
```

If you use recognition where it is ambiguous, the system rings the terminal bell. Type more information, or type a question mark to determine what the system wants you to type.

Use recognition with the INFORMATION command. Type INFO and press ESC; the system prints RMATION (ABOUT). Type a T and press ESC; the system rings the terminal bell because you did not give enough information. To find out what information the system needs, type a ?. The system prints TAPE-PARAMETERS and TERMINAL-MODE. This tells you that the system could not complete the argument beginning with the letter T because there are two possibilities to choose from, and you did not type enough of the argument to distinguish which one you wanted. Type an E and press ESC; this time the system prints RMINAL-MODE (FOR TERMINAL). Press RETURN to end the command.

```
        <ESC>            <ESC>
          |                |
     @INFORMATION (ABOUT) T? one of the following:
      TAPE-PARAMETERS  TERMINAL-MODE

                          <ESC>
                            |
     @INFORMATION (ABOUT) TERMINAL-MODE (FOR TERMINAL)
```

If you use recognition where it is not appropriate (such as at the end of a command), the system rings the terminal bell.

You can use recognition in typing arguments, subcommands, and file specifications. When typing file specifications, you can also use CTRL/F to complete individual portions of a file specification. (Refer to Chapter 4 for more information on using recognition with file specifications.)

Recognition input offers several advantages:

- You can double-check the accuracy of your typing. When TOPS-20 types the completed command, it verifies that it correctly interpreted your typing.
- You can minimize the amount of typing. When typing a filename you need to type only enough characters to uniquely identify that file.
- TOPS-20 prompts your next response by printing a guideword.

2.4.3 Abbreviated Input

To give a command using abbreviated input, type only enough of the command to distinguish it from any other command. Usually, typing the first three letters is sufficient to distinguish one command from another. Abbreviated input requires the least amount of typing of the various methods of input.

To log in using abbreviated input, type LOG and a space; type the full user name (here it is SARTINI) and a space; type the password (the password is not displayed); type the account (here it is 341) and press RETURN.

```
@LOG SARTINI ___ 341
```

There are a few cases where non-unique abbreviations stand for a frequently used command. For example, DIS is the abbreviation for DISABLE, even though other commands begin with the letters DIS - DISCARD and DISMOUNT.

Table 2-1: Special Command Abbreviations

Special Abbreviation	Command
C	CONTINUE
D	DEPOSIT
DIS	DISABLE
E	EXAMINE
INFORMATION F	INFORMATION FILE-STATUS
LOG	LOGIN (When not logged in)
LOG	LOGOUT (When logged in)

Some commands can be distinguished by typing only one or two letters. For example, several TOPS-20 commands begin with the letter A: ACCESS, ADVISE, APPEND, ASSIGN, and ATTACH. You can give any of these commands, by typing only the first two letters. To give the APPEND command you need only type AP; to give the ACCESS command, you need type only AC.

NOTE

When using one or two letters to distinguish commands, keep in mind that as the system develops, new commands will be added and existing abbreviations may require more letters to identify a unique command.

The same method of using abbreviated input for TOPS-20 commands applies for the arguments and subcommands to those commands. In the INFORMATION command, there are two arguments beginning with the letter T: TAPE-PARAMETERS and TERMINAL-MODE. To get information about the terminal parameters, just type E to complete the abbreviation TE.

```
@INFORMATION T? one of the following:
 TAPE-PARAMETERS  TERMINAL-MODE

@INFORMATION (ABOUT) TE
```

In the DIRECTORY command, there are four subcommands beginning with the letter S: SEPARATE, SINCE, SIZE, and SMALLER. To print a list of files in your directory, including the number of pages of each file, use the subcommand SIZE. Type DIRECTORY followed by a comma; the system prints the subcommand prompt, @@, ; type the abbreviation SIZ.

```
@DIRECTORY,
@@SIZ
@@

   PUBLIC:<LEOPOLD>
                  PGS

 PROG1.PAS         3
   .TXT.14         3
 LOGIN.CMD.2       1
 MAIL.TXT.1        2
 NATTACH.TST.1     1
 VERCBL.BAT.1      2
   .CBL.1          1

 Total of 13 pages in 7 files
```

You can type more letters than are required to uniquely identify a command. Abbreviated input simply makes the system more convenient to use.

2.4.4 Combined Recognition and Abbreviated Input

You can mix these two methods of typing commands. Use abbreviated input for the parts of the command you know, and use recognition for the parts of the command you are uncertain of. You can give the LOGIN command using the combination of input methods.

```
@LOG SARTINI (ACCOUNT) 341
```

To give this command, type LOG and a space; type the user name (here it is SARTINI) and a space; type the password and press ESC. After the system prints (ACCOUNT), type the account (here it is 341) and press RETURN.

2.4.5 What Are Defaults

A **default** is the value supplied by the operating system when you do not specify one yourself. For instance, if you do not specify the number of copies in a PRINT command, the system uses the default value of 1. By not explicitly stating the value, the system assumes you have chosen the default. TOPS-20 supplies default values in several areas. The defaults used with individual commands are specified in each command's description in the TOPS-20 Commands Reference Manual.

2.5 CONTINUING COMMANDS

Occasionally it is necessary to type a command that is longer than the maximum line length allowed by your terminal. You can continue typing commands past the end of the line and onto the next line, without pressing RETURN. The system accepts fields of a command that are split between two lines.

In the following example, note that the filename, MANUFACTURING, is split between two lines:

```
@PRINT (FILES) CONCERNS.TXT.1, DESIGN-REVIEWS.MEM.1, MANUFACT
URING.PLAN.1 /AFTER:18:00
```

If you want to avoid splitting a command field, type a space followed by the continuation character, a hyphen (-), at the end of the line and press RETURN. Then, continue typing the command on the next line.

```
@PRINT (FILES) CONCERNS.TXT.1, DESIGN-REVIEWS.MEM.1, -
MANUFACTURING.PLAN.1 /AFTER:18:00
```

Do not use the continuation character to divide a file specification. (Refer to Section 4.2, Complete Form of a File Specification, for a description of file specifications.)

2.6 ADDING COMMENTS TO COMMAND LINES

You can include comments on the command line or on a separate line by prefixing the comment with a comment character, either a semicolon or an exclamation point. The semicolon causes the remainder of the line to be considered as a comment; the exclamation point causes only the text up to the next exclamation point or the end of the line to be considered as a comment.

The following examples show the valid ways to add comments to the TERMINAL command:

```
@TERMINAL VT100  ;This comment follows the command

@!This comment precedes the command! TERMINAL VT100

@TERMINAL !This comment is within the command!  VT100

@;This entire line is a comment
```

If a comment exceeds one line, the same rules applied to continuing commands (refer to the previous section) apply to continuing comments.

The comment character is useful for placing comments in your LOGIN.CMD and COMAND.CMD files. (Refer to Section 1.7 for an example of a LOGIN.CMD file with comments.)

The comment character is also useful when conversing with another user while linked via the TALK command. (Refer to Section 3.2 for information on using the TALK command.)

2.7 CORRECTING INPUT ERRORS

Five keys help you correct input mistakes. These keys are DELETE, CTRL/R, CTRL/U, CTRL/W, and CTRL/H. Except for CTRL/H, these keys are effective only before you press RETURN to end the command. If you press the RETURN key before noticing that a command is incorrect, the system tries to execute it. Usually the command is invalid and the system prints:

```
?Unrecognized command
@
```

This allows you to try again. If you typed a valid command by mistake, you can halt its execution by various means:

1. Stop EDIT by pressing ESC, typing EQ and pressing the RETURN key.

2. Stop printout on your terminal by typing CTRL/O.

3. Stop system programs (such as FILCOM) by typing CTRL/C.

4. Stop any program or command by typing two CTRL/C's.

In each case you are returned to the TOPS-20 operating system. You can then give any valid TOPS-20 command.

2.7.1 DELETE - Erasing a Character

The DELETE key moves the cursor back one character and deletes that character. Most video terminals actually move the **cursor** (an underline or block that marks your position) backward and erases the character when you press DELETE. Hardcopy terminals print the deleted character followed by the backslash character /.

2.7.2 CTRL/U - Erasing an Entire Line

To erase the current command line, type CTRL/U. CTRL/U deletes the line and performs a RETURN so that you can reenter an entire line. Most video terminals erase the line when you press CTRL/U. Hardcopy terminals print three Xs at the end of the command.

2.7.3 CTRL/W - Erasing a Word

To erase a word, type CTRL/W. Most video terminals actually move the cursor backward and erase the last word when you type CTRL/W. Hardcopy terminals print an underscore after the word to indicate that the word has been deleted.

2.7.4 CTRL/R - Reprinting a Command Line

CTRL/R reprints the current command line. You commonly use CTRL/R when editing with CTRL/W and DELETE on a hardcopy terminal has made the command difficult to read.

In this example of the TERMINAL command, you mistakenly type WIDHT instead of WIDTH and correct the mistake with DELETE. To make the command more readable by incorporating the correction, type CTRL/R.

```
@TERMINAL (FEATURE OR TYPE) WIDHT\T\HTH^R
@TERMINAL (FEATURE OR TYPE) WIDTH
```

2.7.5 CTRL/H - Reprinting Part of an Erroneous Command Line

If you make an error in a command line and press RETURN, the system prints a question mark (?) followed by an error message. To reprint the command line up to the erroneous field, type CTRL/H or the BACKSPACE key. The system reprints the command line up to the field that is in error, and you can now complete the command correctly. (CTRL/H or BACKSPACE must be the very next character pressed after pressing RETURN. Also note that both CTRL/H and BACKSPACE print ^H on the terminal.)

The following example illustrates the use of CTRL/H or BACKSPACE with the TERMINAL command:

```
@TERMINAL (FEATURE OR TYPE) LENGTH-WIDTH
?Does not match switch or keyword - "LENGTH-WIDTH"
@^H
@TERMINAL (FEATURE OR TYPE) LENGTH 66
```

To try this example, type TER and press ESC; the system prints MINAL (FEATURE or TYPE). Type LENGTH-WIDTH and press RETURN. The system prints the error message ?Does not match switch or keyword - "LENGTH-WIDTH". (There is no TERMINAL command argument LENGTH-WIDTH. The argument is LENGTH or WIDTH but not both.) Type CTRL/H or BACKSPACE; the system reprints the command line up to the erroneous field. You can finish the command correctly by typing LENGTH 66.

2.8 SETTING ALERTS

You can arrange for the system to ring your terminal bell and issue a one-line message at any future time. You do this by giving the SET ALERT command.

```
@SET ALERT (AT TIME) 9:45:00 (MESSAGE) PREPARE FOR 10:00 MEETING

[09:45:00 alert - PREPARE FOR 10:00 MEETING]
```

You can also be notified at a time that is relative to the current time. The following example sends an alert 10 minutes from the current time:

```
@SET ALERT (AT TIME) +00:10:00 (MESSAGE) END OF COFFEE BREAK!

[10:02:26 alert - END OF COFFEE BREAK!]
```

If you wish to be alerted at the same times, include the appropriate SET ALERT commands in your LOGIN.CMD file. This file is discussed in Section 1.7. Refer to the TOPS-20 Commands Reference Manual for complete information on SET ALERT.

To obtain a listing of all outstanding alert requests, give the INFORMATION ALERTS command.

```
@INFORMATION (ABOUT) ALERTS (PENDING)
 Next alert at 1-Mar-88 11:25:00 - Project meeting 5 mins
 Other alerts set for:
   1-Mar-88 13:00:00 - Call for ajax specs
   1-Mar-88 16:55:00 - Almost time to go home!
   2-Mar-88 00:09:00 - Submit weekly report by noon
   14-Mar-88 09:00:00 - Going away luncheon for manager today

 Alerts are automatic
```

The line "Alerts are automatic" indicates that alerts are issued whether or not you are running a program. Your issuing of the SET AUTOMATIC or the SET NO AUTOMATIC command determines whether or not the system interrupts programs to issue you alerts. If SET NO AUTOMATIC is in effect, you are notified only when your terminal is at TOPS-20 command level.

Note that when you log out, all pending alerts are cleared. You have to reset them when you log in again, unless they are specified in your LOGIN.CMD or COMAND.CMD command file.

2.9 OPERATING SYSTEM STOPPAGE

The TOPS-20 Operating System may stop unexpectedly because of a malfunction. When the operating system stops, the terminal does not print or receive any characters you type. This indicates that the part of the computer controlling input from and output to the terminal is malfunctioning. If the system can recover from this error, it prints:

```
[DECSYSTEM-20 continued]
```

You may lose a few seconds of typing, but after this message prints on the terminal, you can continue your work.

When a fatal error occurs (the entire computer stops working), the system outputs the message:

```
%DECSYSTEM-20 not running
```

When the system resumes operation, it outputs the message:

```
System restarting, wait...
```

and after a few moments, it prints another message, similar to the following:

```
[From OPERATOR on line 6:   SYSTEM IN OPERATION]
```

Once the system restarts after a fatal error, you must log in to the system again. If you have changed the speed of your line with the TERMINAL SPEED command, you may have to reset the speed, depending upon the default speed set by the system manager.

After a fatal error, some of your files may be missing or incomplete. Contact the operator to have these files restored from the system backup tapes.

CHAPTER 3

COMMUNICATING WITH OTHER USERS

This chapter describes:

- Getting a list of users on the system (Section 3.1)
- Linking with other terminals (Section 3.2)
- Reading mail (Section 3.3)
- Sending mail (Section 3.4)
- Communicating with the operator (Section 3.6)
- Controlling messages and terminal links (Section 3.7)

3.1 GETTING A LIST OF USERS ON THE SYSTEM

To get a list of users currently on the system, type the command SYSTAT, and press the RETURN key. The SYSTAT command reports on the status of the system:

```
@SYSTAT
 Mon 26-May-88 15:25:55  Up 6:09:39
 12+5 Jobs   Load av   0.13   0.10   0.06

Job  Line Program  User
  9   120  EMACS   TAMBUR
 11   251  MACRO   GUNN
 12   131  FILDDT  MARTIN
 13   176  EXEC    GREEN
 14   140  MS      SULLIVAN
 26    63  SYSDPY  DEUFEL
 27   173  EXEC    BERRY
 33   DET  EXEC    MORIL
 34    65  EMACS   WORLEY
 45   142  EXEC    HARAMUND
```

```
 50*  210  SYSTAT  MORIL
 51   105  EXEC    BRANNON

  1   232  PTYCON  OPERATOR
  2   233  JNPGPD  OPERATOR
  3   234  EXEC    OPERATOR
  4   235  NMLT20  OPERATOR
  5   236  MCBNRT  OPERATOR
```

The first line of output gives the day of the week, date, time, and the length of time since the system was last started. In the above example, the date is Monday, May 26, 1988 at 3:25:55 PM. The system has been up just over six hours.

The second line gives the number of user jobs plus the number of operator jobs. There are 12 timesharing jobs, plus the operator of the system who is running 5 programs. The last three numbers on this line indicate the load average on the system over a one, five, and fifteen minute period. The load average is a measure of system demand.

The third line contains the column headings for the job number, the line number, the program, the user, and the user's originating system. The number of the job attached to your own terminal (in this case you are job 50) appears with an asterisk (*) next to it in the job column.

To display information about the jobs on a specified node in the TOPS-20 cluster, include the NODE keyword and node name argument.

```
@SYSTAT NODE KL2102
 Thu 13-Aug-88 13:08:12
 THEP Up    0:10:33 17+6 Jobs   Load av   0.11   0.12
 Job  Line Program  Node    User            Origin
 231   DET  DTRSRV  KL2102 Not logged in
 232   DET  RMSFAL  KL2102 Not logged in
 233   DET  RMSFAL  KL2102 Not logged in
 234   434  EXEC    KL2102 LOMARTIRE        LAT1(LAT)
  .
  .
  .
 228   235  MAILS   KL2102 OPERATOR
 229   236  WATCH   KL2102 OPERATOR
 230   237  EXEC    KL2102 OPERATOR
```

If you specify an asterisk as the node name, the SYSTAT command displays information on all nodes in the TOPS-20 cluster.

```
@SYSTAT SYSTEM NODE *
 Thu 13-Aug-88 13:02:00
 DISNEY Up 223:12:12 17+6 Jobs  Load av  0.3    0.27   0.14
 THUP   Up   0:10:33 11+5 Jobs  Load av 10.36  10.27  10.14
 CLYDE  Up  26:34:31  6+8 Jobs  Load av  1.33   1.21   0.99
 CONRO  UP  12:13:14  2+5 Jobs  Load av  5.01   4.95   4.99
```

3.2 LINKING WITH OTHER TERMINALS

One way to communicate with a user that is logged-in to the system is by linking terminals. This allows you to conduct a two-way conversation. To link terminals, give the TALK command followed by the name of the user you want to talk to. The system prints a message informing you that the terminals are linked, and prints the @ sign on the following line. Now, everything you type, or the system prints on your terminal is also printed on the terminal you are linked with.

```
@TALK (TO) MAYO

LINK FROM SARTINI, TTY26
```

After you see the @ sign, you can conduct your conversation using one of the following options: an exclamation mark, the REMARK command, or a combination of both options.

Begin each line you type with an exclamation point (!). After you press RETURN, the system prints an @ sign on the following line and you can continue typing, beginning each line with an exclamation point. If you do not begin the line you type with an !, after you press RETURN, the system prints the message ?UNRECOGNIZED COMMAND.

```
@TALK (TO) MAYO

LINK FROM SARTINI, TTY26
@! This is a test.
```

To avoid typing the exclamation point on each line when you have several lines of text, give the REMARK command. After you give the REMARK command, the system prints a message advising you to type the remark, and end it with CTRL/Z. The system does not print an @ sign when you use REMARK. After you type the message and end with CTRL/Z, the system prints the @ sign on the next line.

```
@TALK (TO) MAYO

LINK FROM SARTINI, TTY26
@REMARK
Type remark. End with CTRL/Z

PER YOUR REQUEST, A NEW COPY OF THE
UPDATED LIST OF MANUALS IS AVAILABLE
IN THE DIRECTORY <NEW-MANUALS>. ^Z
```

You can use a combination of the exclamation point and the REMARK command when you TALK with another user. Use REMARK for a several line comment and the ! for a shorter comment. To end the link with another user's terminal, give the BREAK command. The other user can also give the BREAK command to end the link with your terminal.

```
@TALK (TO) MAYO

LINK FROM SARTINI, TTY26
@REMARK
Type remark. End with CTRL/Z.

PER YOUR REQUEST, A NEW COPY OF THE
UPDATED LIST OF MANUALS IS AVAILABLE
IN THE DIRECTORY <NEW-MANUALS>. ^Z
@!THANKS, I HAVE SEVERAL ITEMS TO ADD TO THE LIST.
@!SEND MAIL TO HOLLAND WITH THE INFO.
@BREAK (LINKS)
```

When you are linked to another user's terminal, the other user's job is not affected by what you type. For example, if another user is running a program that is waiting for a command, and you TALK to that user, the system does not interpret what you type as a command to that user's program. Anything output to one terminal is output to the other as well. Thus you can show another user the output from a program by running it, or the contents of a file by typing it.

If the user you want to TALK to does not want to receive links from another terminal, the system rings the bells on both terminals five times, then prints the following message on your terminal:

```
?Refused, Send mail to user instead
```

Refer to Section 3.7, Controlling Messages And Terminal Links, for information on refusing and receiving links.

If the user you want to TALK to is not logged in, the system prints the following message:

```
?User is not logged in
 Send mail to the user instead
```

3.3 READING MAIL

There are two types of mail that you can receive at your terminal: mail from the system and mail from other users. You can receive system and user mail when you are logged in or logged off the system.

3.3.1 System Mail

System mail is sent to all users on the system by the operator or a privileged user. This type of mail automatically prints on your terminal when you log in.

```
TOSCA, Computer Engineering, TOPS-20 Monitor 7.0(7)
@LOGIN (USER) SARTINI (PASSWORD)___ (ACCOUNT) 341
 Job 57 on TTY127 16-MAR-88 09:49:24
 Date:  16-MAR-1988 0842-EST
 From:  OPERATOR at TOSCA
 To:  SYSTEM
 Subject:  SYSTEM SHUTDOWN

The system will not be available tomorrow from noon to 2:00  p.m.
due to scheduled maintenance.
```

When system mail is sent while you are logged in, you are notified with the message:

```
[New message-of-the day available]
```

To read the new message of the day, use the INFORMATION MAIL SYSTEM command:

```
@INFORMATION MAIL SYSTEM
Sender: OPERATOR
Date: 23 Jul 88, 1033-EST
From: OPERATOR
To: SYSTEM at KL2102
Subject:  Lineprinter paper

A new shipment of lineprinter paper is now available  for  anyone
who needs to replenish paper.

   ========
```

3.3.2 User Mail

User mail is mail sent to you by another user on your system or a system in your network. When you log in, you are notified of new mail with a message similar to:

```
You have mail from COMBS at 08:18:13
```

When user mail arrives while you are logged in, you are notified with a message similar to:

```
[You have netmail from COMBS@GIDNEY at 14:40:56]
```

The program that you can use to read messages sent to you by another user is DECmail/MS. (For complete information on the DECmail/MS program, refer to the TOPS-10/TOPS-20 DECmail/MS Manual).

To start the DECmail/MS program, type MS and press RETURN. The system prints:

```
@MS
 Last read:23-Apr-88 13:00.  24 messages, 5 pages.
 Message 19 flagged.
MS>
```

The lines that appear between the MS command and the MS> prompt give you the status of your mail file.

To read any unread messages in the current message file, use the READ NEW command:

```
MS>READ (MESSAGE SEQUENCE) NEW
Message 24 (261 chars), received 23-Apr-88 22:46:35
Date: 23 Apr 1988 2248-EDT
From: MORRILL at KL2102
To: RANDERSON at KL2102
Subject: Project Meeting
Message-ID: <"MS10(2055)+GLXLIB1(1056)" 11818792562.11.542.18243
at KL2102)

There will be a project meeting today at 4 p.m. in the
Engineering Conference Room.
   ========
MS read>>
```

This command displays all messages in the current message file that you have not read. After the READ command displays a message, it leaves your terminal at read-command level, as indicated by the "MS read>>" prompt. Press the RETURN key to read the next new message (or to return to the MS> prompt if there are no other new messages).

To read any of the messages again, use the READ command at the MS> prompt or at read-command level.

```
MS>READ (MESSAGE SEQUENCE) message sequence

or

MS READ>>READ
```

where:

message sequence specifies the messages you want to read. At read-command level, it is assumed that you want to reread the current message.

Mail you receive from other users is contained in a file called MAIL.TXT. Although the mail program locates this file automatically, you should be aware of its location. This is described in Section 4.6.2, The Device POBOX:.

3.4 SENDING MAIL

Another way to communicate with a user is to send mail with the DECmail/MS program. DECmail/MS handles local and network mail, which goes to users of different (remote) computers. DECmail/MS also provides facilities for filing, retrieving, editing and deleting mail messages.

You can send mail to a user currently on the system, or to a user who is not logged in. The DECmail/MS program can also send mail to a group of users. To start the DECmail/MS program, type MS and press RETURN; the system prints MS>. Type the SEND command. After you give the SEND command, the DECmail/MS system prompts you for the "To". Type the user name or names (if you type a group of user names, separate them with commas); the system prints cc:. Type the name(s) of the user or users you want to receive a copy of the mail; the system prints Subject:. Type a one-line heading for the message.

```
MS>SEND
To: PORADA, MORRILL, MCELMOYLE
cc: BROPHY
Subject: SYSTEM CHANGES
```

Now, the DECmail/MS system types the following help message on your terminal:

```
    Message (ESC to enter Send level, ctrl/Z to send, ctrl/K to
             redisplay, ctrl/B to insert file, ctrl/E to enter editor):
```

After the DECmail/MS system types the help message, anything you type (other than ESC and the control characters) is assumed to be the text of your message.

Type a line of text and issue CTRL/Z to send the message:

```
THERE IS A LIST OF THE NEW SYSTEM
CHANGES AVAILABLE IN THE PROJECT
ROOM.
^Z
```

The DECmail/MS system types information similar to the following, letting you know that the message was successfully sent:

```
Processing mail...
Mail queued for delivery by MX
MS>
```

If you send mail frequently to a group of users, store the list of names in a file. Then, when you run the DECmail/MS program, instead of typing the entire list of names after the To:, you can type the name of the file, preceded by an @ sign. (Refer to Chapter 4 for information on specifying files and to Chapter 5 for information on creating files.)

```
MS>SEND
To: @USERS.LST
CC:
Subject:
```

For a complete description of the DECmail/MS program, refer to the TOPS-10/TOPS-20 DECmail/MS Manual.

3.5 SENDING QUICK MESSAGES

Another way to communicate with a user who is logged in to the system is to send a message with the SEND command. To send a message, give the SEND command followed by the user name and a message with up to six 80-character lines of text. The system prints your user name, terminal line number and message on the receiving terminal.

SEND does not detect the status of the receiving terminal. So, if the receiving terminal is turned off or the user is not logged in, the message cannot be received. Before you SEND a message, use the SYSTAT command to verify that the receiver is logged in to the system.

The following example illustrates the SYSTAT command and the SEND command:

```
@SYSTAT KISTLER
  10    11  EXEC     KISTLER
@SEND KISTLER Are you on the North project interest list?
```

To type a multiple line message, just keep typing past the end of the line and onto the next line without typing RETURN. SEND reorganizes your message so that words split between two lines appear correctly formatted on the receiver's terminal.

```
@SEND KISTLER The North project team meets every Friday at 9 in
the Lunar Conference Room.
```

The message appears on the receiver's terminal as:

```
From LEOPOLD on line 11:
[The North project team meets every Friday at 9 in the Lunar
 Conference Room.]
```

To send a message to a user on a remote node in the TOPS-20 cluster, specify the /NODE: switch:

```
@SEND /NODE:THUP ANDERSON Don't forget the meeting!
```

3.6 COMMUNICATING WITH THE OPERATOR

To communicate with the operator on your system, use the PLEASE program. This program allows you to conduct a two-way conversation with the operator or send the operator a one-way message.

To use the PLEASE program, type PLEASE and press RETURN. PLEASE then prints a message instructing you to type your message and end it with CTRL/Z or ESC. Now, type your message. If you need a response from the operator, end your message by typing CTRL/Z. If you just want to send a one-way message to the operator and do not need a response, end your message by pressing ESC.

In the following example, you need a response from the operator, so you end your message with CTRL/Z. Then, when your dialog with the operator is finished, press ESC.

```
@PLEASE
Enter text, terminate with CTRL/Z to wait for response,
or ESCape to send message and exit
What happened to the RP07?<CTRL/Z>
[PLSOPN Operator at GIDNEY has been notified at 11:18:32]

11:36:04 From Operator at terminal 2
     => Just aligning the heads - back up in 10 minutes

Enter new text (Same terminators)

Thanks<ESC>
```

In this example you don't need a response from the operator so you press ESC after your message:

```
@PLEASE
Enter text, terminate with CTRL/Z to wait for response,
or ESCape to send message and exit
The laser printer is out of paper<ESC>
[PLSOPN Operator at GIDNEY has been notified at 11:18:32]
@
```

If your PLEASE message exceeds one line, press RETURN at the end of the line and continue typing on the next line.

If no operator is in attendance, PLEASE warns you before you can type your message. Your message is still sent and can be answered by the operator when he returns. However you should end your message with ESC, since it may be a long wait before it is answered. To find out if the operator is in attendance before you use the PLEASE program, give the INFORMATION SYSTEM-STATUS command.

For a complete description of the PLEASE program, refer to the TOPS-20 User Utilities Guide.

3.7 CONTROLLING MESSAGES AND TERMINAL LINKS

Several types of messages can appear on your terminal while you are running a program or executing a TOPS-20 command. In addition, another user can link his terminal to yours with an ADVISE or TALK command. You can allow or suppress types of messages and terminal links. This lets you work without interruption or print a clean copy of a file on a hard copy terminal.

3.7.1 System Messages

System messages are messages of general interest to all users. These messages are sent by the system, by the operator, or by a privileged user. Some examples of system messages are:

```
[Caution -- disk space is low]
[System going down in 1 minute!]
[Deleted files will be expunged in 30 seconds]
[System expunge completed]
```

You can specify if you want to receive or refuse system messages on your terminal with the RECEIVE or REFUSE SYSTEM-MESSAGES commands. Note that these commands also control the notice of new mail.

To see if your terminal is set to RECEIVE or REFUSE SYSTEM-MESSAGES, give the INFORMATION TERMINAL command. Then give the REFUSE SYSTEM-MESSAGES command to suppress system messages.

```
@INFORMATION (ABOUT) TERMINAL-MODE
    .
    .
 REFUSE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 RECEIVE USER-MESSAGES
    .
    .
@REFUSE SYSTEM-MESSAGES
```

CAUTION

Since some system messages report important events, you should use the REFUSE SYSTEM-MESSAGES command only when you need to produce uninterrupted output (such as on a hard-copy terminal). Remember to set your terminal back to RECEIVE SYSTEM-MESSAGES after the output is complete.

3.7.2 User Messages

User messages occur when another user issues a SEND command to send a message to your terminal:

```
From SMITTY on line 24:
[Going to lunch?]
```

You can specify if you want to receive or refuse user messages on your terminal with the RECEIVE or REFUSE USER-MESSAGES commands. In the following example, check to see if your terminal is set to RECEIVE or REFUSE USER-MESSAGES with the INFORMATION TERMINAL command. Then give the RECEIVE USER-MESSAGES command to accept user messages.

```
@INFORMATION (ABOUT) TERMINAL-MODE
 TERMINAL VT100
    .
    .
    .
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 REFUSE USER-MESSAGES
    .
    .
    .
 TERMINAL FULLDUPLEX
@RECEIVE USER-MESSAGES
```

3.7.3 Terminal Links

Terminal links occur when another user gives a TALK or ADVISE command to link his terminal to yours:

```
LINK FROM PRATT, TTY 123
!Do you still have my pack?
```

You can stop another user from linking his terminal to yours with the REFUSE LINKS command. In the following example, check to see if your terminal is set to RECEIVE or REFUSE LINKS with the INFORMATION TERMINAL command. Then give the REFUSE LINKS command.

```
@INFORMATION (ABOUT) TERMINAL-MODE
 TERMINAL VT100
    .
    .
    .
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 REFUSE USER-MESSAGES
    .
    .
    .
 TERMINAL FULLDUPLEX
@REFUSE LINKS
```

Note that if you set your terminal to REFUSE LINKS and another user attempts to TALK to you, the system signals you by ringing bells on your terminal five times.

3.7.4 Inhibiting All Non-Job Output

The TERMINAL INHIBIT command stops your terminal from accepting links, system-messages and user-messages; in other words, all output that does not originate from your own job. Use this command when you need to protect your terminal from unwanted output, for example, when printing a file on a hard copy terminal.

TERMINAL INHIBIT essentially has the same function as REFUSE LINKS, SYSTEM-MESSAGES and USER-MESSAGES. However, TERMINAL INHIBIT blocks all links and messages before they can be rejected or accepted by your REFUSE and RECEIVE settings. Therefore, when TERMINAL INHIBIT is in effect, your REFUSE and RECEIVE settings are disabled. Note that in the INFORMATION TERMINAL-MODE display below, a "IS DISABLED" comment follows each REFUSE and RECEIVE setting.

To block all terminal output that does not originate with your job, give the TERMINAL INHIBIT command. Then, check the result with the INFORMATION TERMINAL command.

```
@TERMINAL INHIBIT
@INFORMATION (ABOUT) TERMINAL-MODE
 TERMINAL VT100
    .
    .
    .
```

```
TERMINAL INHIBIT (NON-JOB OUTPUT)
REFUSE LINKS IS DISABLED
REFUSE ADVICE IS DISABLED
RECEIVE SYSTEM-MESSAGES IS DISABLED
RECEIVE USER-MESSAGES IS DISABLED
   .
   .
   .
TERMINAL FULLDUPLEX
```

Use the TERMINAL NO INHIBIT command to restore your REFUSE and RECEIVE settings.

3.7.5 Mail Messages

Mail messages appear on your terminal when another user sends you mail or when you have unread mail. These messages come from two different sources. The first type of mail message is a notice of new mail. This message comes from the mail program and is printed whenever new mail arrives:

```
[You have a message from PRATT]
```

You can specify if you want to receive notice of new mail on your terminal with the RECEIVE or REFUSE SYSTEM-MESSAGES commands.

The second type of mail message results from your giving the SET MAIL-WATCH command.

```
[You have mail from PRATT at 16:07:05]
```

SET MAIL-WATCH causes the system to check your MAIL file for unread mail every five minutes. If the system finds unread mail it prints a message when your terminal is at TOPS-20 command level. This means that if, for example, you are using an editor, the notice of unread mail is not printed until you exit the editor and return to TOPS-20 command level.

You can control the notice of unread mail with the SET MAIL-WATCH and SET NO MAIL-WATCH commands. SET NO MAIL-WATCH is the default.

The SET AUTOMATIC command allows the SET MAIL-WATCH command to send you a message any time, no matter what you are doing at your terminal. The SET NO AUTOMATIC command is the default.

If you want to be reminded of unread mail no matter what you are doing at your terminal, give the SET MAIL-WATCH and SET AUTOMATIC commands.

```
@SET MAIL-WATCH
@SET AUTOMATIC
```

To see if you have any new mail, give the INFORMATION MAIL command. The system lists the name of the sender and the time received for the last unread message in your MAIL file.

```
@INFORMATION MAIL
 Mail from PRATT at 16:07:05
```

3.7.6 Alerts

An alert results from your giving a SET ALERT command:

```
[08:55:00 alert - Group meeting in 5 minutes]
```

Unless you have given the SET AUTOMATIC command, alerts are issued only when your terminal is at TOPS-20 command level. If you do give a SET AUTOMATIC command, alerts will interrupt you no matter what you are doing at your terminal.

You can cancel alerts with the SET NO ALERTS command or you can stop alerts from appearing when you are running a program with SET NO AUTOMATIC.

This command cancels alerts for the next hour:

```
@SET NO ALERT +01:00
```

Check pending alerts with the INFORMATION ALERTS command.

CHAPTER 4

FILE SPECIFICATIONS

This chapter describes:

- TOPS-20 File System Organization (Section 4.1)
- Complete form of a file specification (Section 4.2)
- Using wildcards to specify files (Section 4.3)
- Specifying special characters - CTRL/V (Section 4.4)
- Typing file specifications (Section 4.5)
- Using logical names (Section 4.6)

4.1 TOPS-20 FILE SYSTEM ORGANIZATION

When you enter a program, data or text into the computer, it is stored in a file. A computer file system has an organization similar to that of an office file cabinet system. You create and label a file then store the file in your drawer of the file cabinet. Your drawer of the file cabinet is called your directory on TOPS-20.

4.2 COMPLETE FORM OF A FILE SPECIFICATION

The "label" on a file is called a file specification. A file's specification tells the system where to locate and identify the file. The complete form of a file specification is:

```
dev:<dir>name.typ.gen;attribute
```

where:

dev:	is a device name, a file structure name, or a defined logical name. (A file structure is a name used to reference specific disk devices. Logical names are described in Section 4.6.)
<dir>	is a directory name, or in special cases, a project-programmer number that specifies an area on the disk. You must enclose the directory name in angle brackets <> or square brackets [].
name	is a filename that specifies a particular file in the directory.
.typ	is a file type that helps identify the contents of a file.
.gen	is a generation number that specifies the number of times the file has been changed.
;attribute	is a modifier for the file and specifies a distinctive characteristic for the file.

4.2.1 Device Names - dev:

A device name designates the location of the file on a particular device or file structure. (Refer to Section 6.1 for a description of file structures.)

A device name consists of alphabetic characters that indicate the type of device, a number specifying a particular device (when more than one of a particular device is available), and a colon terminating the name of the device. Table 4-1 lists some common DECSYSTEM-20 devices and their device names.

Table 4-1: System Device Names

Device	Device Name
Your Connected Structure and Directory	DSK:
Your Terminal	TTY:
The structure that receives your mail messages	POBOX:
A Particular Terminal	TTYn:
A Particular Magnetic Tape	MTAn:
Any Line Printer	LPT:
A Particular Line Printer	LPTn:
Any Card Reader	CDR:
A Particular Card Reader	CDRn:
Receptacle for unwanted program output or supplier* of null input	NUL:

The number n indicates a particular unit when the device has multiple units.

* For example, COPY (FROM) NUL: (TO) TEST.FIL erases the contents of the file TEST.FIL.

A colon terminates the device name. With some TOPS-20 commands, the colon following the device name is optional. Refer to the TOPS-20 Command Reference Manual.

Examples of device names are:

TTY20:	the terminal connected to line 20
MTA0:	the magnetic tape unit numbered 0
LPT:	a line printer
ADMIN:	a file structure

If you omit a device name from a file specification, the system uses, as a default, the device or file structure you are presently using.

4.2.2 Directory Names - <DIR>

One area of disk storage allocated for your use is your log-in directory. You reference your log-in directory by using a directory name, which is your user name, enclosed in angle brackets <> or square brackets []. Therefore, if your user name is KIRSCHEN, you have a directory named <KIRSCHEN>. You can use other directories in addition to your log-in directory.

If you have access to the directory of another user and you want to use a file from that directory, insert the directory name enclosed by angle brackets, immediately before the filename. The illustration below shows how to access the file LIZARD.DAT from the directory of user HODGES.

```
<HODGES>LIZARD.DAT.3
```

A directory name consists of up to 39 alphanumeric characters, including the special characters dollar sign, period, hyphen, and underline. You can use the * and % wildcard characters to specify a group of directories, though it is not actually part of a directory name. (Refer to Section 4.3 for more information on using wildcard characters.) Directory names are always enclosed in brackets and are used only when the device is a disk. Examples of directory names are:

```
<PORADA>
<MCELMOYLE>
<MORRILL>
<NEXT-RELEASE>
```

4.2.3 Project-Programmer Numbers - [PPN]

Most programs and commands allow you to type a directory name, but a few require a similar designator called a project-programmer number. Table 4-2 lists the TOPS-20 system programs that require you to type a project-programmer number instead of a directory name when you reference files in directories. Your installation may also have other system programs with this requirement.

Table 4-2: Special System Programs

ALGOL	LIBARY
CREF	LINK
FILCOM	MAKLIB
ISAM	

A project-programmer number consists of two numbers separated by a comma and enclosed in square brackets. To find the project-programmer number corresponding to a particular directory name, give the TRANSLATE command. The following example shows how to find the project-programmer number associated with the directory <KIRSCHEN>:

```
@TRANSLATE (DIRECTORY) <KIRSCHEN>
PS:<KIRSCHEN> (IS) PS:[4,516]
```

The FILCOM program, for example, requires a project-programmer number. If you want to compare your version of the file PLEASE.MAC with the version of the same file in user KIRSCHEN's directory, give the following commands:

```
@FILCOM

*TTY:=PLEASE.MAC[4,516],PLEASE.MAC/A
```

Refer to the TOPS-20 User Utilities Guide for a complete description of the FILCOM program.

4.2.4 Filenames - name

Each file in your directory is identified by a filename consisting of up to 39 alphanumeric characters, including hyphen, dollar sign, and underline. You choose the filename. A filename that reflects the contents of the file will help you remember what is in the file. Examples of filenames are:

```
TEST
COMPUT
ACCTS
DATA-ITEM
10-MEM
```

Although most programs and commands allow filenames up to 39 characters long, some programs do not support this extended length. If you are using any of the programs listed in Table 4-2, the maximum length of a filename is six characters; $, -, and _ characters are invalid in a filename; and the wildcard characters * and % are used for specifying a group of filenames where permitted by the program.

4.2.5 File Types - .typ

To help identify the contents of a file or give the same filename to more than one file, specify a file type consisting of a period followed by up to 39 alphanumeric characters, including $, - and _. The wildcard characters can be used to specify a group of files with the same file type, but is not actually part of the file type. Examples of some standard file types that contain programs are:

File Type	Program Language
.ALG	ALGOL
.BAS	BASIC
.CBL	COBOL
.FOR	FORTRAN
.MAC	MACRO

Other file types include:

- DAT - a data file
- RNO - the input file to the system program RUNOFF
- MEM - the output file for the system program RUNOFF

Refer to Appendix B for a list of standard file types.

In addition to the standard file types, you may use your own file types.

Although most programs and commands allow file types up to 39 characters in length, some software programs do not recognize this extended length. If you are using any of the programs listed in Table 4-2, the maximum length of a file type is three characters; the $, -, and _ characters are invalid in a file type; and the wildcard characters are used for specifying a group of file types where permitted by the program.

4.2.6 Generation Numbers - .gen

The generation number identifies modified or additional versions of the same file. The operating system increases the generation number by one when you change the file. You can create a new file and assign a generation number to it.

When you type a file specification, you can include a generation number. At times you may have more than one generation of a file, especially if you previously gave the SET FILE GENERATION-RETENTION COUNT command. The system always assumes that the most recent file is the one with the highest generation number. If you create a new file with a generation number lower than an existing file with the same filename and type, you may have trouble saving and restoring it on tape using DUMPER or using it with the LOAD-class commands (unless you delete the version with the higher generation number). Refer to the TOPS-20 User Utilities Guide for a description of the DUMPER program, and to Section 9.3 of this manual for information on the LOAD-class commands.

When you do not specify a generation number, the system selects one according to the way you use the file:

1. If you create a new file, the system gives the new file a generation number of 1.

2. If you use an existing file, the system selects the one with the highest generation number.

3. If you create a new version of an existing file, the system adds one to the highest generation number for that file.

4. If you delete or restore a file, the system deletes or restores all versions of the file.

When you do specify a particular generation number, the system uses the file with that generation number. You can give a generation number as a positive number or as a symbol. There are four symbolic generation numbers. Refer to Table 4-3 for a list and description of the four symbolic generation numbers.

Table 4-3: Symbolic Generation Numbers

Generation Number	Represents
.0	the highest existing generation number.
.-1	one greater than the highest existing generation number.
.-2	the lowest existing generation number.
.-3 or *	all existing generations.

For example, if you have three generations (.1,.2,.3) of the file BACKUP.DAT, .0 is the symbolic generation number for BACKUP.DAT.3, .-2 is the symbolic generation number for BACKUP.DAT.1, and .-1 is the symbolic generation for BACKUP.DAT.4. Refer to Section 6.5 for an example of how the system uses symbolic generation numbers.

Some installations limit the number of generations of any one file you can keep. Therefore, if the limit is 3 and you create a fourth generation of the file, the system deletes the file with the lowest generation number. If you have the files BRKING.CBL.3,4,5, and you create BRKING.CBL.6, the system deletes the oldest file (BRKING.CBL.3). The system always assumes that the oldest file is the one with the lowest generation number, and the most recent file is the one with the highest generation number.

If you are using a file with any of the programs listed in Table 4-2, you cannot include a generation number in the file specification. These programs always use the highest existing generation number for files if you are reading or the generation number 1 if you are creating a file.

4.2.7 File Attributes - ;A, ;P, ;T

File attributes specify distinctive characteristics for a file specification. More than one attribute may appear in a file specification. The three most common attributes are: ;A for account, ;P for protection, and ;T for temporary.

The account descriptor takes the form:

```
;Adescriptor
```

The descriptor is an account consisting of up to 39 alphanumeric characters. All charges for file storage are billed to this account. If you do not specify an account for your file specification, the system uses the account you specified in your LOGIN command or your last SET ACCOUNT command.

The file protection code takes the form:

```
;Pprotection
```

Protection is a TOPS-20 protection code. (Refer to Section 6.2, Protecting Directories and Files.)

A temporary file specification contains the file descriptor ;T and a generation number of 100000 plus the number of the job that created the file. (Refer to Section 6.11 for more information on temporary files.) Temporary files are deleted from your login and connected directories when you log off the system.

You can display a list of files with the same attribute by using the DIRECTORY command. This command prints a list of all files with an account of 17:

```
@DIRECTORY (OF FILES) *.*;A17
```

NOTE

You can specify other file attributes when working in a DECnet or magnetic tape environment. Refer to Appendix C of the TOPS-20 Commands Reference Manual for the complete list of attributes. The DECnet-20 User's Guide further describes the DECnet-related file attributes.

4.3 USING WILDCARDS TO SPECIFY FILES

You can use a wildcard character in a file specification to specify files that have part or all of a directory name, filename, file type or generation number that is the same in each file specification. The characters are valid wildcard characters.

The * wildcard matches any number of characters in a field of a file specification that uniquely identifies the file. The following example illustrates using the wildcard character * to list all files in the directory <SMITH> with the file type .TXT:

```
@DIRECTORY (OF FILES) *.TXT

   PS:<SMITH>
 DATA.TXT.7
 MAIL.TXT.5
 TEST.TXT.1

 Total of 3 files
```

If you give the command DIRECTORY L*, the system lists all the filenames beginning with the letter L.

```
@DIRECTORY (OF FILES) L*

   PS:<SMITH>
 LAST.TXT.10
 LEVEL.DAT.1
 LOOP.TXT.6
 LOST.DAT.4

 Total of 4 files
```

If you give the command DIRECTORY *T, the system lists all the filenames ending with the letter T.

```
@DIRECTORY (OF FILES) *T

   PS:<SMITH>
 ACCTST.FOR.1
 CKACCT.FOR.1
 NEWACT.FOR.1
 TEST.FIL.1

 Total of 4 files
```

The % wildcard matches a single character in a field of a file specification that uniquely identifies the file. You cannot use % in a generation number. The following example illustrates using % to list all files in the directory <SMITH> containing four letters, and beginning with the letter L and ending with the letters ST:

```
@DIRECTORY (OF FILES) L%ST

   PS:<SMITH>
 LAST.TXT.10
 LIST.FOR.3
 LOST.DAT.4

 Total of 3 files
```

If you are using a file with any of the programs listed in Table 4-2, you must use a different convention for specifying groups of files. The * wildcard designates a group of filenames or file types, but must either entirely replace the filename or file type, or occur at the end of the filename or file type. Therefore, the construction TEST* is valid but the construction *TEST is not.

NOTE

Not all programs in Table 4-2 accept wildcard characters in a file specification. Also, the commands, COMPILE, DEBUG, EXECUTE, and LOAD do not accept wildcard characters in file specifications.

4.4 SPECIFYING SPECIAL CHARACTERS - CTRL/V

If you need to include a special character, that is, any character other than an alphanumeric, $, - or _ in a file specification, type CTRL/V directly before the special character.

If you are using a file with any one of the programs listed in Table 4-2, do not use the CTRL/V feature.

4.5 TYPING FILE SPECIFICATIONS

There are two methods of typing a file specification in a command: full input and recognition input. For full input, you type the complete file specification. If you are using any of the programs listed in Table 4-2, you must always use full input; recognition is not available.

When you are unsure of a file specification, type a question mark to obtain a list of possible file names, extensions (including nulls), and file versions. For example:

```
@DIRECTORY E? FILE NAME
     EXTRA
     EXTUSR
     EMACS
@DIRECTORY EMACS.? FILE NAME
     INIT
     VARS
@DIRECTORY EMACS.INIT.? FILE NAME
     1
     2
```

Recognition input makes it easier for you to type file specifications. You can make the system recognize file specifications by using either CTRL/F or ESC. For file specifications, CTRL/F recognizes only the current field of the specification, for example it completes a directory name, filename, file type, generation number. The ESC key recognizes as many subsequent fields as possible, including any defaults. Many commands set up defaults so that you can press the ESC key at the beginning of a file specification, causing the system to print the full default file specification on your terminal.

The following example illustrates a way to use CTRL/F to recognize a portion of a default file specification. If you want to change the file type of the file PROG1.OAS, give the RENAME command followed by the file name PROG1 and press the ESC key; the system prints .OAS.* (TO BE). Now type CTRL/F; the system prints PROG1. Type the new file type, .PAS.

```
                        <ESC>          <CTRL/F>
                          |              |
     @RENAME (FILE) PROG1.OAS.* (TO BE) PROG1.PAS
```

The system considers generation numbers in specific ways. When you are using an existing file, the system selects the highest generation number; when you are creating a file and a file with that same name and type already exists, the system assigns a generation number one higher than the highest existing generation number.

The following examples illustrate the way the system considers generation numbers. If you have two files in your directory, TEST.TXT.2 and TEST.TXT.3, and you give the TYPE command to print the TEST.TXT file, the system selects the file with the highest generation number. Give the TYPE command, followed by the filename TEST.TXT and press ESC. The system prints .3 (the highest generation number).

```
                        <ESC>
                          |
@TYPE (FILE) TEST.TXT.3
```

If you want to copy the file NEW.FIL.1 to the destination file TEST.TXT.3, give the COPY command, followed by the filename NEW.FIL and press ESC; the system prints .1 and (TO). Type the filename TEST.TXT and press ESC; the system assigns a generation number one higher than the existing generation number. In this case, the destination file becomes TEST.TXT.4.

```
                       <ESC>           <ESC>
                         |               |
@COPY (FROM) NEW.FIL.1 (TO) TEST.TXT.4 !New generation!
```

NOTE

> You can use recognition on any part of the file specification except the device name field. When you use a device name, you must always type this name in full. If you do not type a device name, the system uses DSK: (your connected file structure), but does not print it on your terminal.

When you type more than one file specification, you can incorporate recognition input when typing each file specification. You can also incorporate wildcards with recognition input when you type a group of files.

When you type more than one file specification on a line, separate each file specification with a comma. The following example illustrates using commas to separate file specifications in a PRINT command.

```
@PRINT (FILES) ATEST.LOG, BTEST.LOG, CTEST.LOG
[Printer job ATEST queued, request #18, limit 9, 3 files]
```

4.6 USING LOGICAL NAMES

A logical name is a descriptive word used to establish a search route for locating files in other directories or on other structures. When you define a logical name, you tell the system where, and in which order, to search for a file.

A logical name comprises up to 39 alphanumeric characters, including -, $, and _. followed by a colon. However, you can use an abbreviated word for the logical name when you define the search list.

For example, you are a member of a team working on a project. Your team has a directory called <TEAM> on the structure PS: where the members store all the completed programs for the project. When you are looking for a project file and you are not sure where it is, you must look through your own directory, and then through the team's directory to find it. Instead of giving two separate DIRECTORY commands for each directory, you can give one DIRECTORY command using a logical name that will automatically search through both directories until it finds the file. The example below illustrates defining a logical name to search your directory, (here your user name is KONEN), and then the team's directory. Include the structure name with the directory names.

```
@DEFINE (LOGICAL NAME) ALL: (AS) PS:<KONEN>,PS:<TEAM>
```

You now have the logical name ALL: defined as PS:<KONEN> and PS:<TEAM>. If you want to search for the file TEST.FOR in either directory, give the following command:

```
@DIRECTORY (OF FILES) ALL:TEST.FOR

   PS:<TEAM>
 TEST.FOR.5
```

The system searches first in the directory <KONEN> where it does not find the file, and then in the directory <TEAM> where it does find the file. If the file TEST.FOR exists in <KONEN> and in <TEAM>, the system searches only until it finds the first file. In this case, finding the file in <KONEN>, it does not continue the search in the directory <TEAM>. When you give the DIRECTORY command, the system always prints the name of the directory and the structure in which it finds the file.

The logical name you define applies only to your current job. It remains in effect until you either remove it, or end your job by logging out. If you want the same defined logical name every time you log in, you can put the definition in your LOGIN.CMD file. (Refer to Section 1.7 for information on LOGIN.CMD files.)

To find out what logical name you are using, you can give the INFORMATION LOGICAL-NAMES JOB command.

```
@INFORMATION (ABOUT) LOGICAL-NAMES (OF) JOB
ALL: => PS:<KONEN>,PS:<TEAM>
```

There are also systemwide logical names that all users can give without having to define them for each job. A systemwide logical name, like SYS:, is usually defined by each installation and includes the directories that contain standard system software. To print a list of systemwide logical names, give the INFORMATION LOGICAL-NAMES SYSTEM command.

```
@INFORMATION (ABOUT) LOGICAL-NAMES (OF) SYSTEM
ACCOUNT: => GIDNEY:<ACCOUNTS>
DEFAULT-EXEC: => SYSTEM:EXEC.EXE
    .
    .
    .
SYS: => PS:<SUBSYS>,PS:<NEW>
TOOLS: => SNARK:<TOOLS>
```

When you define a logical name, you can include an existing systemwide logical name in your definition. Each directory name, device name, or other logical name you use in defining the logical name must be separated by a comma. For example, you can set up a search route to look for a file in the system directories, SYS:, then in <TEAM> and <KONEN>.

```
@DEFINE (LOGICAL-NAME) TEST: SYS:,PS:<TEAM>,PS:<KONEN>
```

By defining the logical name TEST:, the system searches SYS: first, because that was the first area you specified, and if it does not find the file there, continues its search through <TEAM> next, and finally through <KONEN>.

If you copy a file to a logical name, the system places the file in the first area defined in the logical name. For example, if you copy the file CHECK.TST to the logical name ALL:, the system places the file in the directory <KONEN>, because that directory was the first area defined in ALL:.

```
@COPY (FILE) CHECK.TST.1 (TO) ALL:CHECK.TST.1 !New file!
 CHECK.TST.1 => <KONEN>CHECK.TST.1 [OK]
```

If you are defining a logical name for a program listed in Table 4-2, you cannot include the characters - $ or _ in the logical name. Also the logical name cannot exceed six characters, excluding the colon.

To remove a logical name you have defined, give the DEFINE command, but do not type any definition. After the DEFINE command, type only the logical name. The following example shows how to remove the logical name TEST::

```
@DEFINE (LOGICAL-NAME) TEST:
```

You can also use the logical name as an abbreviation for all or part of a file specification. Using a logical name saves you typing if your file specification is lengthy.

The following example shows defining a logical name for a directory name, and then giving the DIRECTORY command using the logical name:

```
@DEFINE (LOGICAL NAME) TS: (AS) PS:<TEST-SPECS>
@DIRECTORY (OF FILES) TS:
```

The following example shows defining a logical name for a filename, and then giving the EDIT command followed by the logical name to get the file.

```
@DEFINE (LOGICAL NAME) PP:(AS) R4-PROJECT-PLAN.RNO
@EDIT PP:
```

Logical names can be used to define physical device names. For example, suppose you have a program that uses one tape drive to input data and another to receive output. These tape drives, physically named MTA0: and MTA1:, can be given the logical names IN: and OUT:. By placing logical names for devices in your programs and defining them at runtime, you can eliminate the need to modify the program to refer to the devices that are currently available.

4.6.1 The Device DSK:

The system defines DSK: to be your connected structure and connected directory. Any time a command or program wants to use a file in your connected directory, it follows the definition of the logical name DSK: to locate the file. Thus, if you want to alter the way each system command and program searches for files, change the definition of the logical name DSK:. The following type of definition:

```
@DEFINE (LOGICAL NAME) DSK: (AS) DSK:,ADMIN:<TESTER>
```

is most common and tells the system to search in your connected directory first; then, if the file is not found, look in the alternate directory <TESTER> on your connected structure.

NOTE

Make sure you do not inadvertently leave out the comma. If you do, DSK: is defined as DSK:<TESTER>, and programs and commands will look only in this directory on the connected structure.

Another example is:

```
@DEFINE (LOGICAL NAME) DSK: (AS) DSK:,ADMIN:<RECORD>, -
ADMIN:<GENLED>
```

The system searches your connected structure and directory first. Then, if the file is not found, it looks on structure ADMIN: in directories <RECORD> and <GENLED>.

When you create files, they are stored in your connected directory or in the first item in your definition of the logical name DSK:.

4.6.2 The Device POBOX:

Every user has his own personal message file, called a mail file. All your incoming messages go into your mail file. This file is named MAIL.TXT. The system defines the logical name POBOX: which defines a search list that points to structures where your mail files reside. When another user sends you mail or when you use a mail program to read your mail, the mail program follows the definition of POBOX: to locate MAIL.TXT. To learn the name of the structure that contains your directory with your MAIL.TXT file, give the command INFORMATION LOGICAL-NAMES POBOX:.

```
@INFORMATION (ABOUT) LOGICAL-NAMES (OF) POBOX:
System-wide:

POBOX: => RANDOM:
```

The directory name of your directory on the POBOX: structure is your user name, for example RANDOM:<DOE>.

CHAPTER 5

CREATING AND EDITING FILES

This chapter describes:

- Selecting an editor (Section 5.1)
- Defining the logical name EDITOR: (Section 5.2)
- Correcting Typing Errors (Section 5.3)

5.1 SELECTING AN EDITOR

The TOPS-20 Operating System allows you to create or change files by using a system editor program. DIGITAL supports three editors for TOPS-20: EDIT, TV, and EDT-20. Other editors which are not supported by DIGITAL, such as EMACS and SED, may be installed on your system.

5.1.1 EDIT

EDIT is a line-oriented editor. With a line-oriented editor, you can change a line by referencing the line number, then substituting characters, or by retyping the line. Some computer programming languages use line numbers when giving error messages. Line numbers are also used with some debuggers.

EDIT has an easy-to-learn and simple-to-use command language. You can use EDIT effectively on either a hard-copy or video terminal.

You can use EDIT to create a program and enter it into a file. There are two commands that call the EDIT program:

- The CREATE command - to create a file.
- The EDIT command - to change a file.

The following sequence shows how to use EDIT to create an ALGOL program that calculates the square root of a number. (If ALGOL is not available on your system, refer to Chapter 9, Producing And Running Your Own Programs, and use the FORTRAN program for the examples).

1. Type CREATE and press the ESC key. The system prints (FILE).

```
     <ESC>
       |
@CREATE (FILE)
```

2. Type the filename and file type that you have chosen for your file. For this example, use SQRT.ALG.

3. Press the RETURN key. EDIT prints the name of the input file and the first line number.

```
     <ESC>
       |
@CREATE (FILE) SQRT.ALG<RET>
Input:  SQRT.ALG.1
00100
```

NOTE

If you already have a file with this name and type, the generation number will not be 1. To change the filename, press the ESC key; the editor (EDIT) prints an asterisk. Type EQ (End and Quit), and press the RETURN key. The system prints the @. You can then CREATE a new file with a different filename and file type.

4. Begin typing your program. (If you make a mistake, refer to Section 5.3, Correcting Typing Errors, for assistance.) Press the return key after each line of the program. EDIT automatically types the next line number. The line numbers that EDIT supplies give you reference points to use when you want to edit your file. (See Section C.2.1 in Appendix C).

```
     <ESC>
       |
@CREATE (FILE) SQRT.ALG<RET>
Input: SQRT.ALG.1
00100   BEGIN<RET>
00200   REAL X,Y;<RET>
00300   WRITE ("[2C] TYPE THE VALUE OF X: [B]");<RET>
00400   <TAB>  READ (X);<RET>
00500   <TAB>  Y :=SQRT(X);<RET>
00600   WRITE ("[C] THE SQUAREROOT OF ");<RET>
00700   <TAB>  PRINT (X,3,3);<RET>
00800   <TAB>  WRITE (" IS ");<RET>
00900   <TAB>  PRINT (Y,3,3);<RET>
```

5. Press the ESC key after you type the last character in the last line of your program. This indicates that your file is complete. TOPS-20 returns the dollar sign then an asterisk.

```
01000   END<ESC>$
```

6. Type E (for End) and press the RETURN key. EDIT prints the name of your file, saves the file and returns you to the TOPS-20 operating system.

```
*E<RET>

[SQRT.ALG.1]
@
```

EDIT is fully described in the EDIT User's Guide and the EDIT Reference Manual.

5.1.2 TV

TV is a character-oriented editor. With a character-oriented editor, you can change one or more characters in a line without retyping the line.

TV has a more powerful command language than EDIT. With this command language, you can accomplish complex editing functions with fewer commands.

For the most effective use of TV, you should use a video terminal. TV is described in the TV Editor Manual. .hl2 EDT-20 EDT is DIGITAL's standard text editor. It is available on many DIGITAL operating systems, for example, TOPS-20, VAX/VMS, RSTS/E, RSX-11M, and RSX-11M-PLUS. There are only minor differences in the features of EDT found on each of these operating systems.

EDT-20 has three editing modes: keypad, nokeypad, and line. Keypad and nokeypad modes are character-oriented editors for use on video terminals. Line mode can be used on either video or hardcopy terminals but is best used with hardcopy terminals.

EDT provides many features that are not available in EDIT or TV. To name a few, EDT has an on-line help facility, it is customizable, and it allows you to work with several files during a single editing session.

To learn how to use EDT on TOPS-20, refer to the EDT-20 Primer. For a complete description of EDT-20 commands and functions, refer to the EDT-20 Reference Manual. Once you have begun using EDT, the EDT Quick Reference Guide is a summary of EDT commands and functions.

5.2 DEFINING THE LOGICAL NAME EDITOR

To run an editor, type the name of the editor (EDIT, TV or EDT) and press RETURN. TOPS-20 also has three commands for running editors: EDIT, CREATE and PERUSE. These commands run the editor that is defined by the logical name EDITOR:. To determine your system's definition of EDITOR: give the INFORMATION LOGICAL-NAMES command.

```
@INFORMATION (ABOUT) LOGICAL-NAMES (OF) EDITOR:
System-wide:

EDITOR: => SYS:EDIT.EXE
```

If the system definition of EDITOR: is not the editor you have chosen to use, make your own definition of EDITOR: with the DEFINE command:

```
@DEFINE (LOGICAL NAME) EDITOR:  (AS) SYS:EDT.EXE
```

Because this command is only in effect until you LOGOUT, you should place it in your LOGIN.CMD file so that it will take effect every time you log in.

5.3 CORRECTING TYPING ERRORS

As you type your program, you may need to correct typing errors. You can correct your program lines by typing CTRL/U or by pressing the DELETE key.

- CTRL/U - Use CTRL/U when you want to delete the line that you are currently typing. CTRL/U deletes the line and allows you to start over again.

  When you type CTRL/U, EDIT responds with the number of the line you just deleted. Retype the line, and press the return key.

- The DELETE Key - Use the DELETE key to erase incorrect characters in the current line.

  NOTE

  On some terminals, the DELETE key is labeled RUBOUT or DEL.

  Each time you press this key, you erase the last character that you typed. When the system deletes a character, it responds with the deleted character, followed by a backslash for each deleted character.

  Correcting a Mistake When You Make It -
  Suppose that while typing the word READ, you press the E key twice. If you notice your mistake right away, you can erase the second E by pressing the DELETE key once. The system responds by printing the deleted character (E) and a backslash. You can then continue typing the line.

  ```
  00400   REEE\AD (X)
  ```

  Correcting a Mistake After You Make It -
  In the example below, you notice that you misspelled SQUAREROOT, after you typed the word OF. To correct the error, delete the last six characters by pressing the DELETE key six times. (A space counts as a character.) You can then continue typing the line.

  ```
  00600   WRITE ("[C] THE OFF\O\ T\O\OEROOT OF ")
  ```

  You CANNOT use the DELETE key to correct characters on a line once you have pressed the RETURN key. The manual for your editor explains how to correct errors on previous lines.

Section 2.7 describes other ways to correct typing errors.

You CAN use the DELETE key to correct mistyped TOPS-20 commands. If you incorrectly type LOGOUT, for example, but notice the mistake before you press the RETURN key, you can use the DELETE key to fix the error. Section 2.7.1 contains further information about correcting commands with the DELETE key.

CHAPTER 6

USING DISK FILES

This chapter describes:

- Using file structures (Section 6.1)
- Protecting directories and files (Section 6.2)
- Connecting to directories (Section 6.3)
- Accessing directories (Section 6.4)
- Copying files (Section 6.5)
- Renaming files (Section 6.6)
- Appending files (Section 6.7)
- Listing files (Section 6.8)
- Printing files (Section 6.9)
- Deleting and restoring files (Section 6.10)
- Creating temporary files (Section 6.11)
- Regulating disk file storage (Section 6.12)
- Long term off-line file storage (Section 6.13)
- Visible and invisible files (Section 6.14)

6.1 USING FILE STRUCTURES

A file structure comprises one or more disk packs containing your files and other user files. A file structure name consists of alphanumeric characters followed by a colon. Even if a file structure is made up of several disk packs, it is referenced by one name. You create and reference files on a structure by specifying the structure name in the device field (dev:) of a file specification.

One file structure, the public structure (PS:), also known as the system structure, is the boot structure (BS:) by default and always remains on line during system operation. This public structure contains a directory for every user of the system, and the necessary accounting information to allow the users to log in. When you log in, you are connected to your directory on the public structure. This directory is referred to as your log-in directory and, in addition to the accounting information, contains some or all of your files.

Once you have entered your programs into the computer and executed them, you have several files in your directory. To obtain a listing of the filenames, type the DIRECTORY command, and press the return key.

```
@DIRECTORY<RET>      Request a list of your filenames.
    PS:<SARTINI>     The structure on which your directory resides
                     and your directory name.
 ADDTWO.FOR.2        The edited version of your FORTRAN file.
   .QOR.1            An unedited backup file, for protection.
   .REL.2            Translation of your edited FORTRAN file.
 SQRT.ALG.1          Your original ALGOL program.
   .REL.1            Translation of your ALGOL file.

 TOTAL OF 5 FILES
@
```

The files in your directory are listed in alphabetical order. When two or more files have the same name but different types, the name is listed only once, for the first file. Subsequent entries for the name are indented, and only the file types and generation numbers are listed.

If you use EDIT to edit an existing file for the first time, EDIT changes your original, unedited file into a backup file. In the process of making this backup file, EDIT changes the file type to Qxx, where xx are the last two letters of the original file type. EDIT creates this backup file so that you have an unaltered copy of your file. Every time you edit the file after the first time, the Qxx file is given a new generation number. In the example above, ADDTWO.QOR.1 is the backup file.

You can have and use files on structures other than the public structure. Like the public structure, these structures also contain directories and files. Unlike the public structure, you cannot log in to these structures. Although the public structure (PS:) and boot structure (BS:) remains on line during system operation, other structures may be mounted (put on line) and dismounted by the operator according to users' requests. To request the mounting and dismounting of structures, use the MOUNT STRUCTURE and DISMOUNT STRUCTURE commands.

The MOUNT STRUCTURE command informs the system that you require the use of a specific file structure (other than the public one). It causes the system to increment a count, called the mount count. The mount count for a structure is the number of users who have given the MOUNT command for that structure. This count assures you that a structure will remain mounted until you no longer need it. You usually have to give the MOUNT command before using files on any structure other than the public one. (Structures that require a MOUNT command are termed "regulated;" other structures are termed "unregulated.")

```
@MOUNT STRUCTURE (NAME) MISC:
Structure MISC: mounted
```

The DISMOUNT STRUCTURE command informs the system that you no longer require the use of a structure and decrements the mount count for that structure.

```
@DISMOUNT STRUCTURE (NAME) MISC:
Structure MISC: dismounted
```

After a structure is mounted, you can use the directories and files on that structure, depending on the protection codes set for those directories and files. (Refer to Section 6.2 for more information on directory and file protection codes and Section 6.3 and 6.4 for more information on connecting to directories and accessing files).

To find out which structures are presently mounted, give the INFORMATION STRUCTURE * command.

```
@INFORMATION (ABOUT) STRUCTURE (NAME) *
Status of structure BOSTON:
Mount count: 4, open file count: 227, units in structure: 2
Public Domestic
Users who have MOUNTed BOSTON: SUSSMAN, TOMCZAK, LNEFF, DNEFF
Users ACCESSing BOSTON: OPERATOR, R.ACE, SAMBERG, COMBS, SYLOR,
        KONEN, COHEN, ZIMA, JENNESS, BLOUNT, SUSSMAN, REILLY,
        CIRINO,

           .
           .
           .
```

```
Status of structure PMH:
Mount count: 1, open file count: 0, units in structure: 1
Domestic
Users who have MOUNTed PMH: HALL
No users are ACCESSing PMH:
Users CONNECTed to PMH: HALL
```

6.2 PROTECTING DIRECTORIES AND FILES

The TOPS-20 file system allows flexibility in sharing some or all of your files with other users. Files and directories are protected at three levels: owner, group member, and all users. Usually files are protected to prevent access from non-owners who are not group members. When you want to share files among a known set of users, you can arrange to share files by asking your system manager to establish a group. Members of a group can access directories belonging to the group, and use files in those directories. (For a complete description of groups, refer to the TOPS-20 System Managers Guide.)

The access to each directory and file is determined by a protection number. You may have some files in your directory that you do not want to share. By setting the proper file protection you can prevent users from accessing these files, while allowing them to use other files in your directory.

Each directory protection number and file protection number comprises six digits, divided into three distinct sections that contain two digits each. The first two digits specify the owner's access; the second two digits specify the group members' access; and the third two digits specify all other users' (also called world) access.

PROTECTION CODE

```
        dd        dd        dd

      Owner     Group    All Users
```

6.2.1 Directory Protection Numbers

The digits in a protection number have different meanings, depending on whether they are in a directory protection number or in a file protection number. Table 6-1 lists the directory protection digits.

Table 6-1: Directory Protection Digits

Digits	Permit
77	Full access to the directory is permitted.
40	Access to files in the directory according to the protection number on the individual files is permitted. To delete and expunge the entire directory (though these digits permit expunging files on an individual basis), you must also assign the digit 10. To create files, you must also assign the digit 04.
10	Connecting to the directory without giving a password, undeleting files, expunging the entire directory, changing times, dates and accounting information for files is permitted. All other access is governed by the protection on the individual file.
04	Create files in the directory.
00	Access to the directory is not permitted.

You can add directory protection digits together. For example, if your directory protection number is 774000, you have full access as the owner of the directory, you allow members of the group to access the directory according to the protection on individual files, and you prohibit all other users from accessing the directory. If you want to allow members of the group not only to access the directory, but also to create files in your directory, you can add the directory protection code 04 to the 40 to get 44. Your entire directory protection code then becomes 774400.

6.2.2 File Protection Numbers

Table 6-2 lists the file protection digits.

Table 6-2: File Protection Digits

Digits	Permit
77	Full access to the file.
40	Read the file.
20	Write or delete the file.
10	Execute the file.
04	Append to the file.
02	Find the file specification using wildcarding.
00	Find the file specification only if the file is specified explicitly and completely. No other access is allowed.

The system default protection number for files is generally 777700. This means that the owner of a file and members of the owner's group have full access, and all other users have no access to the file.

As with directory protection codes, you can construct file protection codes by adding the protection digits together. For example, a code of 44 allows reading and appending, but prohibits modifying or deleting the file, or listing the file in a DIRECTORY command.

6.2.3 Checking Protection Numbers

To validate access to directories and files, the system scans the protection code beginning with the two digits to the right, and moves to the left until it has reached the highest level of access.

The system scans a file or directory protection number in the following way:

1. It scans the two digits to the far right in the protection code to see if all users have access.

2. If all users have access, you can access the file or directory.

3. If all users do not have access, the system moves to the two digits in the center of the protection number to see if members of the group have access.

4. If members of a group have access, you can access the file or directory if you are in the group.

5. If members of a group do not have access, the system moves to the two digits to the far left of the protection code to see if the owner has access.

6. If the owner has access, you can access the file or directory if you are the owner.

7. If the owner does not have access, the system prints an error message.

The protection system works in the following way. For example, you want to type the file TEST.TXT in user HOLLAND'S directory on your terminal. Before printing the file you requested, the system scans the protection code on the directory <HOLLAND> to validate that you have access. If you are not allowed to access the directory, the system prints an error message and cancels the command.

```
@TYPE (FILE) <HOLLAND>TEST.TXT
?Directory access privileges required - "<HOLLAND>TEST.TXT"
```

If the directory protection allows you the access, the system scans the protection on the individual file TEST.TXT. If you are not allowed to access the file, the system prints an error message and cancels the command.

```
@TYPE (FILE) <HOLLAND>TEST.TXT
?READ protection violation for:  "<HOLLAND>TEST.TXT.2"
```

If the file protection allows you to access the file, the system prints the file on your terminal.

To print a directory protection number, use the INFORMATION DIRECTORY command with the VERBOSE subcommand. The directory protection number is in the field "Protection of directory". To print the file protection number, use the VDIRECTORY command (or the DIRECTORY command with the PROTECTION subcommand).

```
@VDIRECTORY (OF FILES) TEST.FIL
   PS:<PORADA>
 TEST.FIL.1; P777700   1 110(7)   21-Mar-88 11:44:25 PORADA
```

6.2.4 Changing a Directory Protection Number

To change a directory protection number, use the SET DIRECTORY PROTECTION command.

```
@SET DIRECTORY PROTECTION (OF DIRECTORY) <EMORRILL> (TO) 770000
```

6.2.5 Changing a File Protection Number

The system assigns a default file protection number to all files created in a directory. This default is usually 777700. To change the default file protection number for a directory, use the SET DIRECTORY FILE-PROTECTION-DEFAULT command.

```
@SET DIRECTORY FILE-PROTECTION-DEFAULT <BLACK> 770000
```

To change a file protection number, use the SET FILE PROTECTION command.

```
@SET FILE PROTECTION (OF FILES) TEST.FIL (TO) 774400
 TEST.FIL.1 [OK]
```

To print a directory's default file protection number use the INFORMATION DIRECTORY command, with the VERBOSE subcommand.

6.3 CONNECTING TO DIRECTORIES

When you log in, you are automatically connected to the directory on the public structure that has the same name as your user name. For example, user McElmoyle is connected to <MCELMOYLE> on the public structure:

```
@LOGIN (USER) MCELMOYLE (PASSWORD)___(ACCOUNT) 341
```

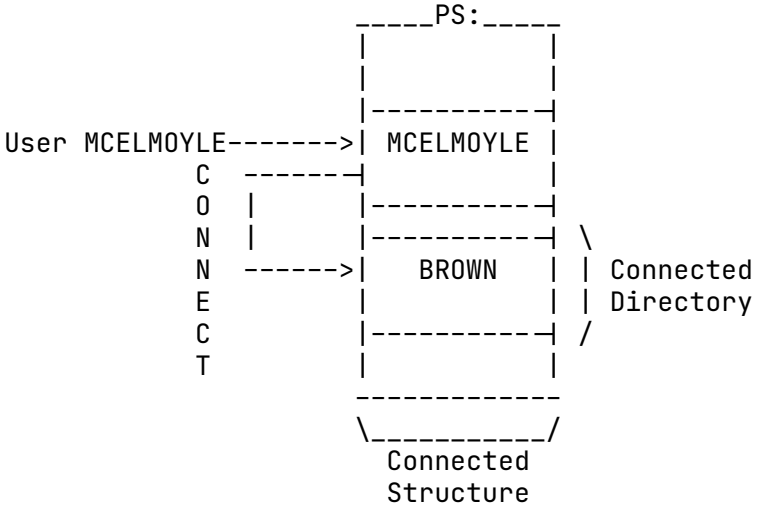
If you need to work in another directory, you can connect to that directory. When you connect to a directory, the system automatically disconnects you from the directory you are presently in and uses the new directory as your default directory. Your default directory is the one the system assumes when you omit a directory name in a file specification.

In addition, you have owner rights for that directory, just as if you logged in to it. The owner rights for a directory are valid as long as you are connected to that directory; the rights terminate when you connect to another directory. You always retain the owner rights to the files in your log-in directory.

You can connect to a directory on the public structure or on another on-line structure. To connect to another directory, give the CONNECT command and the name of the directory you want to use. You are prompted for a password for the directory depending on your ownership and group rights for the directory.

The example below illustrates the effects of logging in, then connecting to another directory on the public structure. When you (user MCELMOYLE) log in to the system, you are connected to your own directory on the public structure. When you omit a directory name and/or structure name in a file specification, the system assumes your logged-in directory <MCELMOYLE> on the public structure. After you log in, connect to the directory <BROWN> on the public structure. Now, if you omit the directory name and/or structure name in a file specification, the system assumes your connected directory <BROWN> on the public structure.

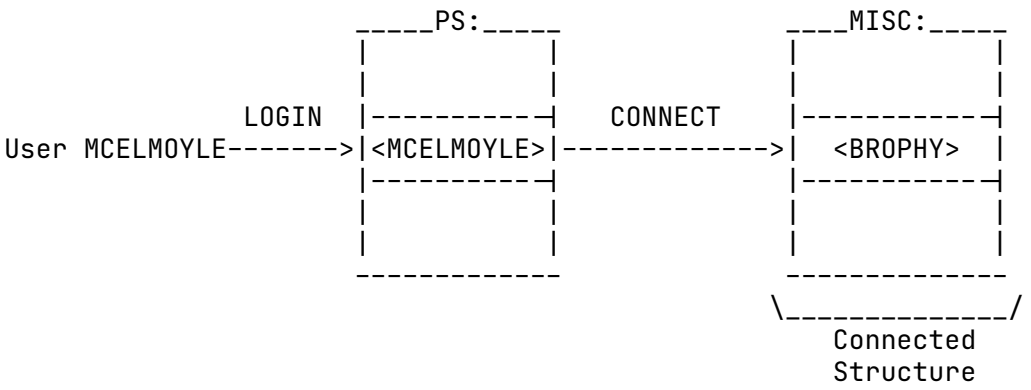
```
@LOGIN (USER) MCELMOYLE (PASSWORD)___(ACCOUNT) 341
 Job 5 on TTY26 31-Mar-88 14:56:24, Last Login 30-Mar-88 08:24:13
@CONNECT (TO DIRECTORY) <BROWN>
Password:___
```



When you give the CONNECT command for a directory that is located on a different structure, your default structure also changes. The system assumes both the connected structure and the connected directory when you omit them in a file specification.

The example below illustrates the effects of logging in on the public structure and then connecting to a directory on another structure named MISC:. When you (user MCELMOYLE) log in, you are connected to your directory on the public structure. After you log in, connect to the directory <BROPHY> on the structure MISC:.

```
@LOGIN (USER) MCELMOYLE (PASSWORD)___(ACCOUNT) 341
 Job 28 on TTY26 31-Mar-88 12:02:46, Last Login 30-Mar-88 08:32:26
@CONNECT (TO DIRECTORY) MISC:<BROPHY>
Password:___
```



If you later omit a structure name or a directory name from a file specification, the system assumes the structure MISC: and the directory <BROPHY>.

If you forget which directory or structure you are connected to, give the INFORMATION JOB-STATUS command. If no directory name is printed, then you are connected to your logged in directory.

```
@INFORMATION (ABOUT) JOB-STATUS
 Host AURORA
 Job 105, TTY46, User HIGGINS, SUMMIT:<HIGGINS>
 Account 341
```

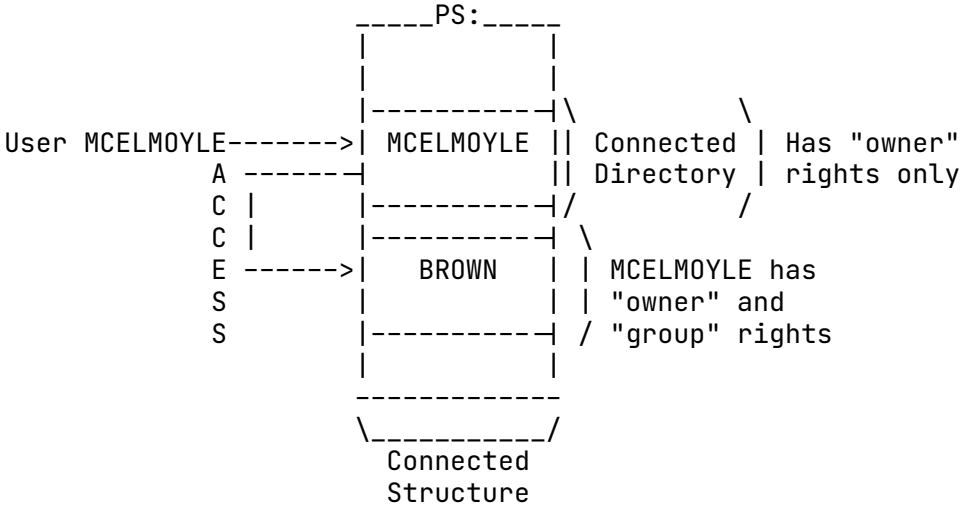
6.4 ACCESSING DIRECTORIES

To access another directory and remain connected to your present directory, give the ACCESS command.

When you access a directory, you are actually working in your connected directory but you also have owner and group rights to the other directory. This means that you can use the files in the directory you have accessed by specifying that directory in the file specification. Unless you specify otherwise, any file you create appears in your connected directory. If you want the file to be written into the directory you have accessed, you must specify the directory name in the file specification. If the directory you access is located on a different structure than your connected directory, you must specify the structure and directory names in any file specification.

The example below illustrates the effects of logging in, then accessing another directory on the public structure. When you (user MCELMOYLE) log in to the system, you are connected to your login directory. After you log in, access the directory <BROWN> on the public structure. You now have owner and group rights for directory <BROWN>.

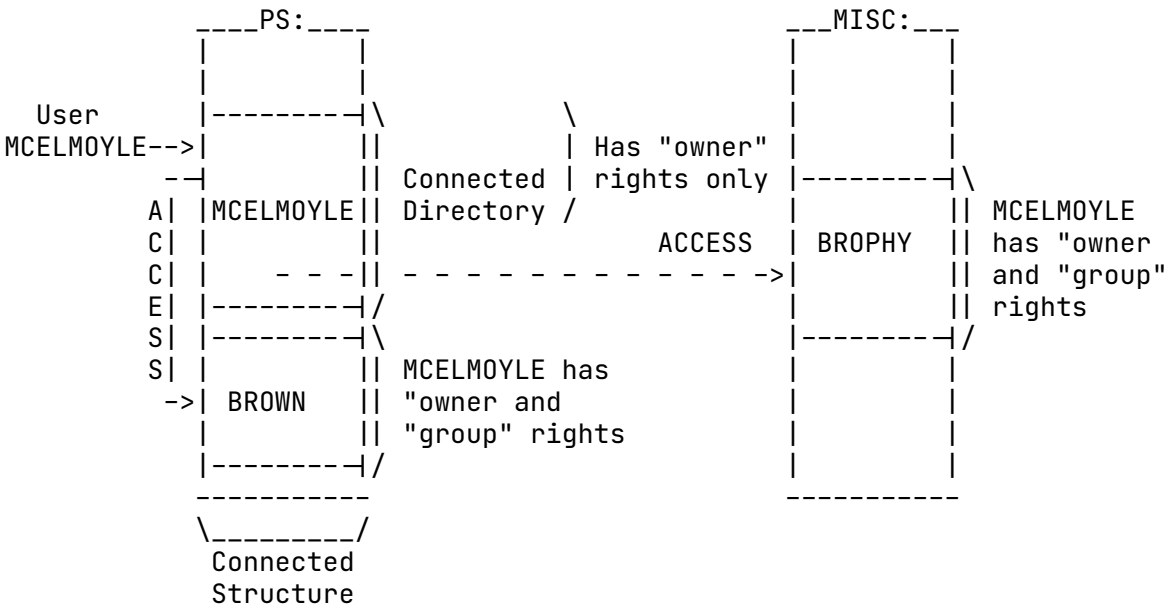
```
@LOGIN (USER) MCELMOYLE (PASSWORD)___(ACCOUNT) 341
 Job 32 on TTY26 31-Mar-88 10:08:16, Last Login 30-Mar-88 11:36:44
@ACCESS (TO DIRECTORY) <BROWN>
Password:___
```



You can access more than one directory during a job session. You can access a directory on one structure and also access a directory on a different structure. If each directory you access is located on a different structure, the owner and group rights for these directories remain in effect throughout your entire job session (from LOGIN to LOGOUT) or until a structure is dismounted. You can access only one directory per structure, however. If you access a second directory on the same structure, your access to the first directory is cancelled. You always retain your owner rights to your log-in directory on the public structure. However, when you give the ACCESS command to a different directory on the public structure, you lose the group privileges of your log-in directory.

You can log in, access another directory on the public structure, then access a directory on another structure, MISC:, as in the following example:

```
@LOGIN (USER) MCELMOYLE (PASSWORD)___(ACCOUNT) 341
 Job 32 TTY26 31-Mar-88 10:08:14, Last Login 30-Mar-88 11:16:02
@ACCESS (TO DIRECTORY) <BROWN>
Password:___
@ACCESS (TO DIRECTORY) MISC:<BROPHY>
Password:___
```



6.5 COPYING FILES

You can use the COPY command to reproduce one of your files. This procedure is useful if you want to change a file without altering the original file.

To copy a file to another file, give the COPY command. The COPY command copies the contents of an existing file (called a source file) to a destination file, and keeps the original file. The following example shows how to copy the existing file TEST1.DAT to the destination file 2TEST.DAT.

```
@COPY (FROM) TEST1.DAT.1 (TO) 2TEST.DAT.2 !New generation!
 TEST1.DAT.1 => 2TEST.DAT.2 [OK]
```

You can also use the COPY command to copy a file from another user's directory. First give the ACCESS command with the other user's directory name and password. (The password does not print on the terminal.) Then type COPY, and press the ESC key. The system prints (FROM). Type the other user's directory name (enclosed in angle brackets), the name of the file you want to copy and the ESC key. The system prints the generation number and the guideword (TO). Press the RETURN key. The other user's file is copied to your disk area. The file keeps the same name.

```
@ACCESS<ESC>(TO DIRECTORY) <PORADA><RET>
Password: <RET>

@COPY<ESC>(FROM) <PORADA>TODAY.EXE<ESC>.3 (TO)<RET>
 <PORADA>TODAY.EXE.3  => TODAY.EXE.1 [OK]
```

You can also copy a file from another user's directory and give the file a different filename. To do this, use the procedure described above. The system prints the generation number and guideword (TO):

```
@COPY<ESC>(FROM) <PORADA>TODAY.EXE<ESC>.3 (TO)
```

Instead of pressing the RETURN key, as in the previous example, type the name that you want to give the new file. When you have typed the new name, press the RETURN key. The system prints a message telling you that it has copied the file.

```
@COPY<ESC>(FROM) <PORADA>TODAY.EXE<ESC>.3 (TO) TEST.EXE<RET>
 <PORADA>TODAY.EXE.3  => TODAY.EXE.1 [OK]
@
```

You can copy multiple files by using a wildcard. For example, if you type COPY (FROM) *.FOR, the system places each file with the file type .FOR into a destination file. If you type COPY (FROM) TEST.*, the system places each file with the filename TEST into a destination file.

```
@COPY (FROM) TEST.* (TO) NEWTST.*.-1
 TEST.FOR.1 => NEWTST.FOR.1 [OK]
 TEST.TXT.2 => NEWTST.TXT.1 [OK]
```

If you use recognition input in the above example, when you press ESC after the filename NEWTST, the system rings the terminal bell, asking you to type more information. In this example, type a period after the filename, indicating to the system the end of the filename; and press ESC. The system prints the wildcard character, *, and a .-1 generation number. The -1 generation number is a symbolic generation number and indicates to you that when the system processes the command line, it will use one greater than the highest number of each file. (Refer to Section 4.2.6, Generation Numbers - .gen, for more information on symbolic generation numbers.)

6.6 RENAMING FILES

You can use the RENAME command to change the name of a file or to put a file into another directory on the same structure. When you use RENAME, the system simply changes the file specification instead of actually duplicating the file.

```
@RENAME (EXISTING FILE) TEST1.DAT.* (TO BE) TESTAL.DAT.-1
 TEST1.DAT.1 => TESTAL.DAT.2 [OK]
```

To move files from one structure to another, use the COPY command. RENAME will not work across structures. After copying the file, you can delete the original.

```
@COPY (FROM) MISC:TEST.FIL.5 (TO) TEST.FIL.1 !New file!
 MISC:TEST.FIL.5 => TEST.FIL.1 [OK]
@DELETE MISC:TEST.FIL.5
```

6.7 APPENDING FILES

To add the contents of one or more source files to the end of a destination file, give the APPEND command. The destination file can be an existing file or a new file. The following example shows how to add the contents of the source file STAT.TXT.5 to the end of the file CHECK.TXT:

```
@APPEND (SOURCE FILE) STAT.TXT.5 (TO) CHECK.TXT
 STAT.TXT.5 [OK]
```

You can append a series of files with the same filename or file type using a wildcard. The following example shows how to append all files with the file type .FOR. Notice that these files are appended in alphabetical order when using a wildcard for the filename.

```
@APPEND (SOURCE FILE) *.FOR (TO) ATEST.FOR.1 !New file!
 ACCOUN.FOR.2 [OK]
 ACCTST.FOR.1 [OK]
 NEWTST.FOR.1 [OK]
 TEST.FOR.1 [OK]
```

You can append files from a directory on one structure to a directory on another structure. The system prints the structure name, the directory name and the filename of the source file, followed by the message [OK] when the file has been appended.

```
@APPEND (SOURCE FILE) PS:<DOE>SMALL.FOR (TO) MISC:<DOE>LARGE.FOR
   PS:<LATTA>SMALL.FOR.2 [OK]
```

NOTE

Some programs, such as COBOL and SORT, cannot use appended files.

6.8 LISTING FILES

To display a copy of your file on your terminal, type the TYPE command, and press the ESC key. After (FILE), type the filename and file type of your file. Press the RETURN key.

To see a copy of ADDTWO.FOR type the following command:

```
@TYPE (FILE) ADDTWO.FOR
```

If you want TOPS-20 to stop printing a file after it begins, type a CTRL/O. CTRL/O stops the printout. You can resume the printing by typing a second CTRL/O.

6.9 PRINTING FILES

To print a file or files, give the PRINT command. The PRINT command places entries into the line printer output queue.

```
@PRINT (FILES) UPDATE.CBL
[Printer job UPDATE queued, request  57, limit 27]
```

To see that your job is in the line printer output queue, give the INFORMATION OUTPUT-REQUESTS command. The system lists all the jobs in the queue. If you want only the entries of your job(s), include the /USER switch.

```
@INFORMATION (ABOUT) OUTPUT-REQUESTS

Printer Queue:
Job Name  Req#  Limit            User
--------  ----  -----  ------------------------
* BOX       53    270  LYONS                      On Unit:0
   Started at 14:29:29, printed 122 of 270 pages
* UPDATE    57     27  SARTINI                    On Unit:1
   Started at 14:38:18, printed 0 of 27 pages
  MIDAS     34     27  REILLY                     /Forms:NARROW
There are 3 Jobs in the Queue (2 in Progress)
```

You can control several conditions of your print request by using switches with the PRINT command.

To simply print a file, it is not necessary to include switches. However, you can include switches with the PRINT command. To obtain a list of valid switches, type PRINT, followed by a ?. The list of switches the system prints contains both job switches and file switches.

```
@PRINT ?  /SPOOLED-OUTPUT
  or Job switch, one of the following:
| /ACCOUNT:            /AFTER:              /CHARACTERISTIC:
  /DESTINATION-NODE:   /FORMS:              /GENERIC
  /JOBNAME:            /LIMIT:              /LOWERCASE
  /NOTE:               /NOTIFY:             /PRIORITY:
| /REMOTE-PRINTER:     /SEQUENCE:           /UNIT:
  /UPPERCASE           /USER:
  or File switch, one of the following:
  /BEGIN:          /COPIES:         /DELETE           /FILE:
  /HEADER          /MODE:           /NOHEADER         /PRESERVE
  /REPORT:         /SPACING:
| or ","
| or File specification
@PRINT
```

If you include a job switch with the PRINT command, the entire job is affected by the switch. For example, if you print three files and you add the /AFTER: switch, all three files will be printed after the time you specify.

```
@PRINT (FILES) LARGE.DAT, MYTEST.DAT, TEST1.DAT /AFTER:15-MAR-88
[Printer job LARGE queued, request #58, limit 27]
```

If you include a file switch with the PRINT command, only the file directly before the switch is affected. For example, if you print three files and you add the /COPIES:6 switch after the first filename, the system prints six copies of the first file only.

```
@PRINT (FILES) LARGE.DAT/COPIES:6, MYTEST.DAT, TEST1.DAT
```

A file switch can act as a job switch when placed before all files in a command. For example, if you print three files and you add the /COPIES:6 switch before the first filename, the system prints six copies of each of the three files.

```
@PRINT (FILES)/COPIES:6,LARGE.DAT,MYTEST.DAT,TEST1.DAT
```

You can direct a PRINT request to a remote destination by including the /REMOTE-PRINTER switch. The destination is either a VMS printer queue for DQS printers or a LATserver PORT or SERVICE for LAT printers. To specify a string that communicates file features such as layout or lettering type, include the /CHARACTERISTIC switch.

The following example shows how to print a job with a PORTRAIT 90 characteristic on a XEROX 8700 printer on a VMS system.

```
@PRINT FILE4.MEM/REMOTE-PRINTER:XEROX/CHARACTERISTIC:P90
[Printer job FILE4 queued, request #33, limit 1 files]
```

You can direct a PRINT request to a remote node by specifying the /DESTINATION-NODE switch. The remote node can be either an IBM remote station, a node in a TOPS-20 cluster, a VMS remote node or a LATserver.

The following example shows how to PRINT a job on a printer service named XEROX on a LATserver named LAT97.

```
@PRINT FILE.DAT/REMOTE-PRINTER:XEROX/DESTINATION-NODE:LAT97
[Printer job FILE queued, request #45, limit 1 files]
```

You can specify the SET REMOTE-PRINTING PRINTER command to establish the /REMOTE-PRINTER queue and characteristic parameters. The SET REMOTE-PRINTING command can be invoked at command level or within a command file.

The following example shows how to define the name of a remote printer queue on node OURVAX.

```
@SET REMOTE-PRINTING PRINTER XEROX SI$8700 OURVAX
@
```

Now, to direct a print request to the remote printer queue:

```
@PRINT MYFILE.MEM/REMOTE-PRINTER:XEROX
@
```

For more information about directing print requests to remote destinations, refer to the TOPS-20 Commands Reference Manual.

6.9.1 Modifying a PRINT Request

To change and/or add one or more switches to a previously issued PRINT command, give the MODIFY command. After you give the MODIFY command, type PRINT, followed by the first six letters of the jobname, or the request ID, then type the switch you want to change or add.

You can modify almost all PRINT command switches. To obtain a list of switches you can modify, give the MODIFY PRINT/ command, followed by a question mark (?).

The following example shows how to modify the PRINT request for LARGE.DAT by including the /AFTER: switch:

```
@MODIFY (REQUEST TYPE) PRINT (ID) LARGE /AFTER:25-MAR-88
[1 Job modified]
```

After you give the command, the system prints a message informing you that the job was modified. If the system is processing the entry when you give the MODIFY command, it does not modify the job and prints the message [No Jobs modified].

6.9.2 Canceling a PRINT Request

To cancel or remove entries you have previously placed in the line printer output queue, give the CANCEL command. After you give the CANCEL command, type PRINT, followed by the first six letters of the jobname or the request ID of the job you want to remove.

Once the CANCEL command removes the entry from the line printer output queue, the system prints the message [1 Job Canceled]. If the system is processing the entry when you give the CANCEL command, it stops the job and prints the message, [1 Job Canceled (1 was in progress)].

The following example shows how to cancel the PRINT request for TEST.FOR.

```
@CANCEL (REQUEST TYPE) PRINT (ID) TEST
[1 Job canceled]
```

If you have several PRINT jobs in the lineprinter output queue, you can cancel them all by using an asterisk.

```
@CANCEL (REQUEST TYPE) PRINT (ID) *
[3 Jobs canceled]
```

You can cancel a PRINT request to a remote printer in the same TOPS-20 cluster as the requesting node by including the /DESTINATION-NODE: switch in the command. This switch cancels only the print requests that were made from the local node. Other print requests made on the remote node are not affected.

The following example shows how to cancel a remote print request.

```
@CANCEL PRINT SUM7/DESTINATION-NODE:KL2102
[1 print request cancelled]
```

Note that PRINT requests directed to a remote node not in the same cluster as the requesting node cannot be cancelled from the requesting node.

6.9.3 Setting Defaults for the PRINT Command

If you want the PRINT command to always contain certain switches, give the SET DEFAULT PRINT command, followed by the switch or switches. Whenever you give a PRINT command, the switches you specified in the SET DEFAULT command are automatically included in the PRINT command.

To give the /NOTE switch with PRINT commands, place the following command in COMAND.CMD.

```
@SET DEFAULT (FOR) PRINT /NOTE:FLOOR4
```

Every time you give the PRINT command, the system includes the switch /NOTE:FLOOR4 in the command.

To avoid having to type the SET DEFAULT PRINT command every time you log in to the system, put this command in a COMAND.CMD file. (Refer to Section 1.7 for information about a COMAND.CMD file.)

To see which defaults you set for the PRINT command, give the INFORMATION DEFAULTS PRINT command.

```
@INFORMATION (ABOUT) DEFAULTS (FOR) PRINT
 SET DEFAULT PRINT /NOTE:FLOOR4
```

6.10 DELETING AND RESTORING FILES

When you no longer need to keep a file, you can delete it by giving the DELETE command. The DELETE command marks the file for automatic deletion; it does not actually erase the file.

The deleted files in your logged-in or connected directory are erased (expunged) when one of the following occurs:

- You give the EXPUNGE command for the directory.
- The operator gives the EXPUNGE command for all directories in the structure.
- You (or another user connected to your directory) log off the system.

The EXPUNGE command erases all files marked for deletion since the last time the directory was expunged. Deleting and erasing files are separate operations. Therefore, once you delete a file, it does not immediately disappear. If you delete a file by mistake, you can type the UNDELETE command to restore the file to your directory. Type this command as soon as you detect your mistake; otherwise, you may not be able to restore the file. You cannot restore a file once you log off the system.

To delete the file TEST.FIL from your directory, give the following command:

```
@DELETE (FILES) TEST.FIL
 TEST.FIL.5 [OK]
```

You can give the DIRECTORY command with the deleted subcommand to list all the files that have been deleted but not yet expunged.

```
@DIRECTORY (OF FILES) TEST.FIL.5,
@@DELETED
@@

   PS:<PORADA>
 TEST.FIL.5
```

To restore TEST.FIL, give the UNDELETE command.

```
@UNDELETE (FILES) TEST.FIL.5
 TEST.FIL.5 [OK]
```

If you give the DIRECTORY command again, you will see that the file has been restored in your directory.

If you delete a file and give the EXPUNGE command, the file is erased immediately.

```
@DELETE (FILES) TEST.FIL
 TEST.FIL.5 [OK]
@EXPUNGE (DELETED FILES)
   PS:<PORADA> [3 pages freed]
```

If you expunge a file by mistake, contact the operator. Most systems keep backup tapes from which you can obtain an older version of the file. If you expunge a newly-created file, one that has not been backed-up on tape, you cannot recover it.

CAUTION

Do not delete files and plan to undelete them at a later time, because deleted files may be expunged by the system at any time.

6.11 CREATING TEMPORARY FILES

When you have a file that you need only for the current terminal session, such as a scratch file, give the file the ;T attribute. The ;T attribute indicates that the file is temporary. When you log off the system, the system deletes and expunges any temporary files in your logged-in and/or connected directories.

One way to create a temporary file is to use the COPY TTY: command. This command simulates the action of the CREATE command by copying the text you type on your terminal (device TTY:) to a file.

Give the COPY TTY: command, type the contents of the file and end your input with a CTRL/Z:

```
@COPY (FROM) TTY:  (TO) TEMP.FIL;T
 TTY:  => TEMP.FIL.100160;T

ESCAPE 031
EXTENDED
OPAQUE
PAGE
^Z
```

To give an existing file the ;T attribute, use the RENAME command.

```
@RENAME (EXISTING FILE) SCRATCH.FIL (TO BE) SCRATCH.FIL;T
 SCRATCH.FIL.1 => SCRATCH.FIL.100014;T [OK]
```

Do not use recognition input to print the second file name in the RENAME command. Recognition prints the comment !New generation! after the file specification and causes the ;T attribute to be ignored.

You can assign any generation number to a temporary file. If you do not specify a generation number, the system assigns the file a generation number of 100000 plus your job number. In the above example, the user's job number is 14; the system added 100000 for a generation number of 100014. Two users connected to the same directory can both create temporary files; however, if one user logs off, the other user's temporary files are not deleted, because the files are identified by different job numbers.

Refer to Appendix C of the TOPS-20 Commands Reference Manual for a complete list of file attributes.

6.12 REGULATING DISK FILE STORAGE

The system manager sets an upper limit on the amount of disk space for each directory on the system. This disk space, referred to as directory storage allocation, is allotted as a number of pages.

Each directory receives a specific number of pages. To see the number of pages allocated to your directory, and the number of pages you are using, give the INFORMATION DISK-USAGE command.

```
@INFORMATION (ABOUT) DISK-USAGE (OF DIRECTORY) <SARTINI>
 PS:<SARTINI>
 37 Pages assigned
 50 Working pages, 50 Permanent pages allowed
 34142 Pages free on PS:
```

In the example above, user SARTINI has 37 pages assigned to his directory, and a working storage allocation and permanent storage allocation of 50 pages. There are 34142 free pages remaining on this file structure.

The system automatically checks your working storage allocation whenever you create a new file page. If you are over that allocation, it prints the message "?Disk or directory full, or quota exceeded" and does not let you continue writing to your file. You can delete any unimportant or temporary files and expunge the directory to get under your working allocation.

Whenever you give a LOGIN or LOGOUT command or connect to another directory, the system checks the permanent disk storage allocation of your connected directory. If it is exceeded, the system prints a message in the form:

```
<directory> Over permanent storage allocation by n page(s)
```

CAUTION

> If you exceed your working storage allocation, the system programs listed in Table 4-2 expunge any deleted files. When a system program expunges deleted files, it prints a message; however, once you see the message, you cannot halt the expunging process.

Depending upon the policy at your installation, if you do not regulate your own disk storage allocation, the operator may regulate it for you by running a system program to move some of your disk files to magnetic tape for short-term off-line storage. This program looks for directories that are over quota and moves files from the directories until they are under quota. The operator runs this program as often as required to bring directories under quota. This forced migration of files from disk to tape is used to keep the system disk space free.

The system manager determines which type of files the program moves to tape storage. However, if you want to specify a particular order in which you want the files moved when the operator runs the program, you can include a MIGRATION.ORDER file in your log-in directory. In the MIGRATION.ORDER file, you can list the files you want moved first. For example, to request that temporary files and files with the .LST file type be migrated before your other files, place this line in the MIGRATION.ORDER file:

```
*.TMP, *.LST
```

The SET FILE RESIST command also gives you some control over involuntary file removal. It delays migration of the specified files for as long as possible.

```
@SET FILE RESIST (MIGRATION OF FILES) MEMO.INI
 MEMO.INI.1 [OK]
```

The file MEMO.INI will be among the last files to be removed from the disk.

To see the files that will "resist" migration, give the DIRECTORY command with the RESIST-MIGRATION subcommand:

```
@DIRECTORY (OF FILES) ,
@@RESIST-MIGRATION (FILES ONLY)
@@

   PS:<TUCKER.USER>
 MEMO.INI.1
 USEDOC.DST.3
 USEPLN.DST.2

 Total of 3 files
```

To see the files that were moved to off-line storage by the system program, give the DIRECTORY command. Next to the names of the files that were moved, the system prints ;OFFLINE.

```
@DIRECTORY (OF FILES)

   PS:<SARTINI>
 2TEST.DAT.3
 NEWACCT.LST.1;OFFLINE
 OVERVIEW.LST.10;OFFLINE
 SQUARE.B20.1

 Total of 4 files
```

If you need to use the file, give the RETRIEVE command followed by the name of the file. The RETRIEVE command notifies the system that you are requesting the restoration of the file from off-line storage.

```
@RETRIEVE (FILES) MYTEST.DAT.1

 MYTEST.DAT.1 [OK]
```

To see your retrieval request, give the INFORMATION RETRIEVAL-REQUESTS command. The system prints a list of requests in the retrieval queue.

```
@INFORMATION (ABOUT) RETRIEVAL-REQUESTS

Retrieval Queue:
 Name     Req#    Tape 1   Tape 2   User
 ------   ----    ------   ------   --------------------
 ADVENT      6      5845     5641   ENGEL
 CHESS       7      5845     5641   ENGEL
 MYTEST     68      5854     5852   SARTINI
 OTHELL      9      5641     8459   ENGEL
 There are 4 jobs in the Queue (None in Progress)
```

You can remove any retrieval requests before the contents of the off-line file are restored to disk by using the CANCEL command.

```
@CANCEL (REQUEST TYPE) RETRIEVE (ID) MYTEST
[1 Job canceled]
```

6.13 LONG TERM OFF-LINE FILE STORAGE

If you have disk files that you do not use, but want to keep, you can mark these files for extended off-line storage by using the ARCHIVE command. The operator periodically runs a program that moves the files marked for archiving from disk to magnetic tape for off-line storage. After the program moves the files to tape, it sends a message through the MAIL program telling you the file has been archived and its contents deleted from the disk. Your system manager can tell you which files you should archive, and how long they will be stored. The system manager can also tell you how often the operator runs the program to move the files marked for archiving.

You can also use DUMPER for off-line storage. Refer to the DUMPER description in the TOPS-20 User Utilities Guide for more information.

6.13.1 Archiving Files

To mark a file for archiving, give the ARCHIVE command, followed by the name of the file you want archived.

```
@ARCHIVE (FILES) CHECK.TXT
 CHECK.TXT.1 [Requested]
```

6.13.2 Getting Information about Archive Status of Files

To see that the file is marked for archiving, give the INFORMATION ARCHIVE-STATUS command, followed by the name of the file.

```
@INFORMATION (ABOUT) ARCHIVE-STATUS (OF FILES) CHECK.TXT
 CHECK.TXT.1 Archive requested
```

You can also give the INFORMATION ARCHIVE-STATUS command without any argument. The system prints a list of your files that are archived, and files for which archiving has been requested.

Once you mark a file for archiving, the name of the file no longer appears when you give the DIRECTORY command. To see which files are archived, and which files are marked for archiving, give the subcommand ARCHIVE to the DIRECTORY command. The files that are already archived will have the comment ;OFFLINE next to the filename.

```
@DIRECTORY,
@@ARCHIVE
@@

   PS:<SARTINI>

 CHAPT21.TCT.1;OFFLINE
 CHECK.TXT.1

 Total of 2 files
```

When you mark a file for archiving, you cannot modify, delete, or copy the file. The file does not appear in your directory unless you include the ARCHIVE subcommand in the DIRECTORY command.

6.13.3 Canceling an Archive Request

If you decide that you do not want to archive the file, give the CANCEL command to remove the archival request. You can give the CANCEL command as long as the file is still in archival request status, that is, as long as the INFORMATION ARCHIVE-STATUS command shows that archive is requested but not completed.

```
@CANCEL (REQUEST TYPE) ARCHIVE (FOR FILES) CHECK.TXT
 CHECK.TXT.1 [OK]
```

6.13.4 Retrieving an Archived File

Once a file is archived, it is stored off-line on magnetic tape. If you need to use the file again, give the RETRIEVE command. The RETRIEVE command notifies the system that you are requesting the restoration of the file from off-line storage. To actually restore the file, the operator mounts the magnetic tape containing the archived file, and moves the file to your directory on disk.

```
@RETRIEVE (FILES) CHAP21.TCT
 CHAP21.TCT.1 [OK]
```

To see your retrieval request, give the INFORMATION RETRIEVAL-REQUESTS command. The system prints a list of the requests in the retrieval queue.

```
@INFORMATION (ABOUT) RETRIEVAL-REQUESTS

Retrieval Queue:
 Name   Req#    Tape 1  Tape 2  User
------  ----    ------  ------  --------------------
CHAP21    48      5520    5543  SARTINI
There is 1 job in the queue (none in progress)
```

Once your archived file is restored to disk, you must copy its contents to a new file before you modify it. You must use a copy of the file because you cannot alter an archived file in any way, even after it is restored to disk.

You can cancel any retrieval requests before the archived file contents are restored to disk, by using the CANCEL RETRIEVE command.

```
@CANCEL (REQUEST TYPE) RETRIEVE (ID) CHAP21
[1 Job Canceled]
```

6.13.5 Deleting an Archived File

If you decide that you will never need the tape copy of an archived file, delete the file with the DISCARD command. The DISCARD command does not delete the file itself, but it deletes the pointer from your directory to the file copy on tape. The tape copy of the file is actually deleted when the operator recycles tapes that contain files that have passed their expiration dates and/or have their pointers deleted.

After you give the DISCARD command, the operator sends you a mail message that contains information about the discarded file. If you wish to use the tape copy, you may be able to recover it using this information, as long as the tape has not yet been recycled.

If you have a disk copy of an archived file, the DISCARD command restores this file to its normal status.

6.13.6 Archiving Expired Files Automatically

There are several dates associated with each file you create. One of these dates is the on-line expiration date, which determines when a file's disk contents may be automatically moved to off-line storage. The SET DIRECTORY ARCHIVE-ONLINE-EXPIRED-FILES command enables this automatic archiving. This command is discussed at the end of the section.

On-line expiration dates are displayed with the DIRECTORY command:

```
@DIRECTORY (OF FILES) ,
@@DATES (OF) ONLINE-EXPIRATION
@@

  PS:<TUCKER.USER>
  Online expiration

  ARCHIV.MEM.4      3-May-88
    .QNO.15         5-May-88
    .RNO.15         21-Nov-88
  COMAND.CMD.5      21-Nov-88
  MEMO.INI.1        8-Apr-88
  USER.RNO.2        8-Apr-88

  Total of 6 files
```

The system manager establishes a systemwide on-line expiration date, but you can override the system default with the SET DIRECTORY ONLINE-EXPIRATION-DEFAULT command:

```
@SET DIRECTORY ONLINE-EXPIRATION-DEFAULT (OF DIRECTORY) -
<TUCKER> (TO) 26-NOV-88
```

You can specify a time interval rather than a specific date:

```
@SET DIRECTORY ONLINE-EXPIRATION-DEFAULT (OF DIRECTORY) -
<TUCKER> (TO) +30
```

The command above sets the on-line expiration date to 30 days from the creation date.

You can also establish on-line expiration dates for individual files:

```
@SET FILE ONLINE-EXPIRATION (OF FILES) MEMO.INI (TO) +120
MEMO.INI.1 [OK]
```

If you want a file to be immediately available for archiving, give the SET FILE EXPIRED command:

```
@SET FILE EXPIRED (FILES) PENDING.Q
PENDING.Q.11 [OK]
```

The command above sets the expiration date to today's date.

When you are satisfied with the on-line expiration dates for your files, you can indicate that the system is to mark them for archiving when the expiration dates are reached:

```
@SET DIRECTORY ARCHIVE-ONLINE-EXPIRED-FILES (OF DIRECTORY) -
 <TUCKER>
```

You also have the choice of leaving expired files in your directory until a possible forced migration:

```
@SET DIRECTORY NO ARCHIVE-ONLINE-EXPIRED-FILES (OF DIRECTORY) -
 <TUCKER>
```

This is the default setting for directories.

To see if expired files in your directory will be automatically archived, give the INFORMATION DIRECTORY command:

```
@INFORMATION (ABOUT) DIRECTORY (DIRECTORY NAME) <TUCKER>
Name PS:<TUCKER>
     .
     .
Archive online expired files
     .
     .
```

The line "Archive online expired files" indicates that automatic archiving will take place. If the SET DIRECTORY NO ARCHIVE-ONLINE-EXPIRED-FILES command is in effect, this line does not appear in the information display.

6.14 VISIBLE AND INVISIBLE FILES

Typically, you will have files that are not currently in use and cluttering your directory. You can clean up a directory by moving the files to tape (archiving), or to other directories, or by making your infrequently used files invisible.

An invisible file is not displayed by a simple DIRECTORY command and is not accessible to programs and EXEC commands. The ARCHIVE command automatically makes files invisible. When you RETRIEVE archived files, they will remain invisible when restored to disk.

To make a file invisible, use the SET FILE INVISIBLE command. To make an invisible file visible again, use the SET FILE VISIBLE command. To display your invisible files, use the DIRECTORY command with the INVISIBLE subcommand.

CHAPTER 7

USING MAGNETIC TAPE

This chapter describes:

- Using magnetic tape storage (Section 7.1)
- Using unlabelled tapes (Section 7.2)
- Using labelled tapes (Section 7.3)

7.1 USING MAGNETIC TAPE STORAGE

Magnetic tape provides off-line storage for data. You put data onto tape for storage using the COPY command, DUMPER program, or a program of your own. (For a complete description of the DUMPER program, refer to the TOPS-20 User Utilities Guide.) Tapes can be labelled or unlabelled. An unlabelled tape is identified only by a gummed label on the outside of the tape reel. A labelled tape is identified by the information contained internally on the tape as well as a gummed label on the outside of the tape reel. Refer to the TOPS-20 Tape Processing Manual for more information on labelled and unlabelled tapes.

7.2 USING UNLABELLED TAPES

Before you use an unlabelled tape, give the INFORMATION SYSTEM-STATUS command to find out if the tape allocation facility of TOPS-20 is enabled. The process to gain and release access to a tape differs, depending upon whether this tape allocation facility is in use. (Refer to the TOPS-20 System Manager's Guide for an explanation of tape allocation.)

7.2.1 Using Unlabelled Tapes with Tape Allocation Enabled

If tape allocation is enabled on your system, you can mount an unlabelled tape by giving the MOUNT TAPE command followed by the name of the tape (the name that appears on the gummed label). Before you give the MOUNT TAPE command, tell the operator the name you selected for your tape or ask him to get the tape from the tape library. After you give the MOUNT TAPE command, you must wait until the operator mounts the tape, and the system prints a message telling you that the tape is mounted.

```
@MOUNT TAPE (NAME) ACE1:
[Tape set ACE1, volume ACE1 mounted]
[ACE1:  defined as MT0:]
```

You can include the /NOWAIT switch with your MOUNT TAPE command. By including this switch, you do not have to wait for a response from the operator and you can continue working until the tape is mounted. When you use the /NOWAIT switch, you can also check on your mount request by giving the INFORMATION MOUNT-REQUESTS command.

```
@MOUNT TAPE (NAME) ACE1: /NOWAIT
```

If you want to remove the request from the queue before the tape is mounted, type a CTRL/C to return to command level, then give the CANCEL MOUNT command. If you included a /NOWAIT switch with the MOUNT TAPE command, you can simply give the CANCEL MOUNT command.

After the operator mounts the tape, the system sends a message advising you that the tape is ready for your use. You can now run your program.

When you complete your work, give the DISMOUNT TAPE command, followed by the name of the tape. The system prints a message telling you that the tape is dismounted.

```
@DISMOUNT TAPE (NAME) ACE1:
[Tape dismounted, logical name ACE1: deleted]
```

7.2.2 Using Unlabeled Tapes with Tape Allocation Disabled

If tape allocation is not enabled on your system, you must first assign a tape drive for your job. To find out which tape devices are available, give the INFORMATION AVAILABLE-DEVICES command.

```
@INFORMATION (ABOUT) AVAILABLE-DEVICES
Devices available to this job:
 DSK, PS, ADMIN, MTA1, MTA2, LPT, CDR, PTY15, NUL
Devices assigned to/opened by this job: TTY23
```

Assign one of the devices beginning with 'MTA'. The example shows assigning drive 2.

```
@ASSIGN (DEVICE) MTA2:
```

After assigning the drive to your job, you can run the PLEASE program and ask the operator to mount your tape.

```
@PLEASE
Enter text, terminate with CTRL/Z to wait for response
Or ESC to send message and Exit
Please mount tape TEST:<CTRL/Z>
[PLSOPN Operator at GIDNEY has been notified at 11:18:32]

11:36:04 From Operator at terminal 2
     =>Your tape is mounted

Enter new text (Same terminators)

Thanks<ESC>
```

When you complete your work, give the UNLOAD command. This command unloads the magnetic tape by rewinding it entirely onto the source reel.

After you give the UNLOAD command, give the DEASSIGN command. The DEASSIGN command returns the device you had previously ASSIGNed back to the pool of available devices. If you forget to do this, no other user can use the device until you log out.

7.2.3 Setting Tape Parameters

You must make sure that you read and write the data on the tape with the proper tape parameters set. Give the INFORMATION TAPE-PARAMETERS command.

```
@INFORMATION (ABOUT) TAPE-PARAMETERS
 SET TAPE DENSITY 1600
 SET TAPE PARITY ODD
 SET TAPE FORMAT CORE-DUMP
 SET TAPE RECORD-LENGTH 512
```

These parameters work for most tape transfers; if you have to change any of the parameters, give the SET TAPE command.

```
@SET TAPE DENSITY (TO) 800
```

These changed parameters remain in effect until you log off, or change the parameters.

NOTE

Not every tape drive supports every parameter. Check with your system manager to find out what drive types are available on your system and which parameters work with each drive type.

If you set a parameter by giving a DUMPER command, that parameter affects only the DUMPER operations and does not change your job defaults. For a complete description of DUMPER, refer to the TOPS-20 User Utilities Guide.

7.2.4 Positioning the Tape

There are commands that position a magnetic tape: BACKSPACE, REWIND, and SKIP. The BACKSPACE command backspaces the tape over a certain number of records or files on unlabeled tapes, and over a certain number of files on labeled tapes; the REWIND command rewinds the tape to the beginning of the tape; the SKIP command advances the magnetic tape a certain number of records or files on unlabeled tapes, and a certain number of files on labeled tapes.

```
@SKIP (DEVICE) MTA2: 4 FILES
```

7.3 USING LABELLED TAPES

The operator creates the labelled tapes for you through a process called initialization. When a tape is initialized, the system actually writes specific information on the tape. Included in this information is a volume identifier, also called a VOLID. The VOLID is a unique number assigned to the tape.

Once the operator creates the labelled tape, you can give the MOUNT TAPE command followed by the tape volid or the setname you selected for your tape(s). In the following example, the /NEW switch specifies that you are creating a new tape with the tape setname ABCD:. For a complete list of switches to use with the MOUNT TAPE command, refer to the TOPS-20 Commands Reference Manual.

```
@MOUNT TAPE (NAME) ABCD:/NEW
[Tape set ABCD, volume 002001 mounted]
[TEST:  defined as MT2:]
```

After the operator mounts the tape, the system sends a message advising you that the tape is ready for your use and which drive you have been assigned. You can now run your program.

If your program requires additional tapes to complete the job, the operator will automatically mount the additional tapes. The system does not notify you of the volids of the additional tapes. To find out the volids of the additional tapes you can give the INFORMATION VOLUMES command, followed by the tape set name to obtain a list of the volume identifiers for each tape in the tape set. In the following example, the tape set name ABCD: contains three tapes with the volids of 002001, 002002, and 002003:

```
@INFORMATION (ABOUT) VOLUMES (OF TAPE) ABCD:
Volumes of tape set ABCD:  002001, 002002, 002003
```

To read an existing tape set containing several volumes, include the tape setname and the /VOLIDS: switch in the MOUNT TAPE command.

```
@MOUNT TAPE (NAME) ABCD:/VOLIDS: 002001,002002,002003
[Tape set ABCD, volume 002001 mounted]
[ABCD: defined as MT2:]
```

You can also mount a specific volume in the tape set by specifying the /START switch followed by the volid for that specific volume. For example, if you want to mount the second volume in the tape set name ABCD:, give the following command.

```
@MOUNT TAPE (NAME) ABCD:/VOLIDS:002001,002002,002003  -
/START:VOLUME 002002
[Tape set ABCD, volume 002002 mounted]
[ABCD:  defined as MT0:]
```

The operator mounts the tape, and the system prints a message telling you that the tape that you requested is mounted.

If you include the /NOWAIT switch in the MOUNT TAPE command you can check on your request to mount the tape, by giving the INFORMATION MOUNT-REQUESTS command. The system prints a list of mount requests in the queue, and indicates the status of the request.

```
@INFORMATION (ABOUT) MOUNT-REQUESTS

Mount Queue:
Volume     Status   Type  Dens   Write    Req#   Job#   User
-------   --------  ----  ----  -------  ------  ----  ---------
MCBFT2    MTA2      Tape  1600             32      18  SROBINSON
ASDF      MTA3      Tape  defa  Enabled    73      36  KONEN
002002    MTA0      Tape  6250  Enabled    74       7  SARTINI

There are 3 requests in the queue
```

If you want to remove your mount request from the queue, give the CANCEL MOUNT command, followed by the tape setname. You must first give a CTRL/C to get out of the MOUNT command before you can cancel the mount request. If you included the /NOWAIT switch, you can simply give the CANCEL MOUNT command. You can give the CANCEL MOUNT command as long as the request is in waiting status, that is, as long as the operator has not mounted the tape.

```
@CANCEL (REQUEST TYPE) MOUNT ABCD:
[1 mount request canceled]
```

When you no longer need to access the tape, give the DISMOUNT TAPE command, followed by the tape setname.

```
@DISMOUNT TAPE ABCD:
[Tape dismounted, Logical name ABCD:  deleted]
```

CHAPTER 8

RUNNING SYSTEM PROGRAMS AND OTHER USERS' PROGRAMS

This chapter describes:

- Running system programs (Section 8.1)
- Giving commands to system programs (Section 8.2)
- Getting information about system features (Section 8.3)
- Running user programs (Section 8.4)
- Controlling programs (Section 8.5)
- Running programs without destroying memory (Section 8.6)
- Running multiple programs (Section 8.7)

8.1 RUNNING SYSTEM PROGRAMS

The TOPS-20 system has many system programs. To get a complete list of the programs available, contact your system manager. The HELP ? command prints a list of the programs explained by the HELP program.

In general, a system program produces an output file by performing some operation on an input file. Some programs perform different functions, depending on the file type of the input file; however, unless you specifically request it, the program does not destroy your input file. You can give a particular name to your output file or let it take a default name. The program creates default names by keeping the name of the input file and changing the file type. For instance, the default output name used by the RUNOFF program is the input filename with the file type .MEM.

To run any of the system programs provided with TOPS-20, type the name of the program, and press RETURN. The following example shows how to start the DUMPER program:

```
@DUMPER                 !Type DUMPER and press RETURN.
DUMPER>                 !DUMPER starts
                        !And waits for a command
```

8.2 GIVING COMMANDS TO SYSTEM PROGRAMS

Once the system program responds with its prompt, you can give the program a command. There are two types of prompts from the system program.

Some programs respond by printing an asterisk on the terminal. You can then type a command in the following format:

```
desination-filespec = source-filespec

destination file specification = source file specification(s)
/switch(es)
```

You cannot use recognition on file specifications or switches when you run any of the programs listed in Table 4-2, Special System Programs.

Other system programs respond by printing a prompt that identifies the program, such as the prompt for the DUMPER program.

```
@DUMPER
DUMPER>
```

You can use recognition on commands and arguments to these programs.

8.2.1 Example: Using a System Program

The FILCOM (for FILe COMparison) program which compares two files and indicates the differences between them, works as follows:

1. Create two files that are similar but not identical. You may create two files of your own or use the files created in the following example:

```
@CREATE (FILE) FIRST.FIL
Input:  FIRST.FIL.1
00100           TYPE 101
00200   101     FORMAT ('THIS PROGRAM WAS WRITTEN FIRST.')
00300           TYPE 102
00400   102     FORMAT ('BUT THE TWO PROGRAMS ARE SIMILAR.')
00500           END
^E

[FIRST.FIL.1]
@

@CREATE (FILE) SECOND.FIL
Input:  SECOND.FIL.1
00100           TYPE 101
00200   101     FORMAT ('THIS PROGRAM WAS WRITTEN SECOND.')
00300           TYPE 102
00400   102     FORMAT ('BUT THE TWO PROGRAMS ARE SIMILAR.')
00500           END
*E

[SECOND.FIL.1]
@
```

2. Start the FILCOM program by typing FILCOM and pressing the RETURN key. When FILCOM is ready, it prints an asterisk on your terminal:

```
@FILCOM

*
```

3. Tell FILCOM which files to compare and what to do with the results of the comparison. For this example, type the line:

```
*TTY:=FIRST.FIL,SECOND.FIL
```

   This line tells FILCOM to compare the two files and print the results on your terminal (TTY stands for terminal.) If, instead, you want to store the results in the file PROG.DIF, type the line:

```
*PROG.DIF=FIRST.FIL,SECOND.FIL
```

4. Press RETURN at the end of the line to execute the command

```
*TTY:=FIRST.FIL,SECOND.FIL
File 1) DSK:FIRST.FIL[4,16]     created: 0837 10-Jun-1988
File 2) DSK:SECOND.FIL[4,16]    created: 0839 10-Jun-1988

1)1   00200   101 FORMAT ('THIS PROGRAM WAS WRITTEN FIRST.')
1)    00300    TYPE 102
****
2)1   00200   101 FORMAT ('THIS PROGRAM WAS WRITTEN SECOND.')
2)    00300    TYPE 102
**************

%files are different

*
```

In the comparison, lines preceded by a 1) are from the first file, FIRST.FIL. Lines preceded by a 2) are from the second file, SECOND.FIL. FILCOM puts an extra number beside the lines that differ, and then prints the line. After each of the differing lines, FILCOM prints the next line (for example, TYPE 102), so that you can easily find your place in the files.

After the first comparison, FILCOM prints another asterisk to show that it is ready to do more work. This time, let FILCOM compare the files but print only the second file. If there are any differences between the second file and the first, request FILCOM to put a vertical bar in the left column beside any such line. The switch /U does this.

```
*TTY:=FIRST.FIL,SECOND.FIL/U

    00100         TYPE 101
:   00200 101     FORMAT ('THIS PROGRAM WAS WRITTEN SECOND')
    00300         TYPE 102
    00400 102     FORMAT ('BUT THE TWO PROGRAMS ARE SIMILAR.')
    00500         END

%files are different

*
```

Now, to exit FILCOM, type a CTRL/C. The system prints the @.

```
*^C
@
```

You can run many system programs in this manner. Some programs behave differently. For help, type HELP and the program name. If you cannot obtain any information, contact your system manager.

8.3 GETTING INFORMATION ABOUT SYSTEM FEATURES

The HELP program gives you useful information about the commands for various programs of the TOPS-20 system. The simplest way to run the HELP program is to type HELP and press the RETURN key. TOPS-20 then responds with the general instructions for obtaining information.

```
@HELP

HELP Command ====
The HELP command prints helpful documentation on various system
features, The command

 @HELP
will print this message on your terminal.

 @HELP NAME
will look for, and print out information about the system feature
names in "NAME".  For example,

 @HELP EDIT
will print out information about the EDIT program.

 @HELP ?
will give a list of features for which HELP is available and
retype to wait for any additional input.

[End of HELP.HLP]
@
```

To get information about a system feature, type HELP, followed by a space and a question mark. The system prints a list of features for which it has information.

```
@HELP ? one of the following:
 68274     8700       ACCT20  ACL     ACTGEN  ADJPSX  ALGDDT
 ALGOL     APL        APLSF   ASTROL  BLAST   BLIS10  BLIS11
 BLISS     BLSCRF     BOX     CALC    CALMNT  CALN    CBL74
 CHANGE    CHECKD     CHESS   CHKPNT  CMPTXT  CMS     CN
 CNVDSK    COBDDT     COBOL   CONGEN  CONTEN  CONTNT  CONV20
    .
    .
    .
 SNOBOL    SORT       SOUP    STEP    SYSERR  SYSJOB  TAR
 TCX       TERMINAL   TGHA    TMSTAP  TOC     TRAK20  TRANSF
 TRANSL    TTYINI     TUTIO   TV      TYPVF7  ULIST   UNITS
 US        USAG20     USAH20  VAXTAP  VTECO   WATCH   WATCH-NEW
 XEROX     XOUT
  or confirm with carriage return
```

To get help on a specific feature, type HELP and the name of a system program as an argument. TOPS-20 then responds with the information available about that program.

```
@HELP FILCOM
FILCOM V21B(60)

FILCOM compares two files in either ASCII mode
or binary depending upon switches or file name extensions.
All standard binary extensions are recognized as binary by
default.
Switches are :-
/A  compare in ASCII mode
/B  allow compare of Blank lines
/C  ignore Comments and spacing
/E  file is in .EXE format
/S  ignore Spacing
/H  type this Help text
/#L Lower limit for partial compare
         or number of Lines to be matched
         ( # represents an octal number)
/#U Upper limit for partial compare
/Q  quick compare only, give error message if files differ
/U  compare in ASCII Update mode
/W  compare in Word mode but don't expand files
/X  expand files before word mode compare

@
```

Note that many programs also have a HELP command. /H is the help command for programs that have an * prompt, while HELP is the command for programs using the program name and > prompt; for example, DUMPER.

8.4 RUNNING USER PROGRAMS

To run your own executable program in your connected directory, give the RUN command. In the following example, run the program LESTSQ:

```
@RUN (PROGRAM) LESTSQ
```

Files with the file type .EXE contain executable programs. An executable program is a program that has already been compiled, loaded, and saved. (Refer to Section 9.1.)

To run another user's program, give the file specification with the RUN command:

```
@RUN (PROGRAM) <HOLLAND>TEST
```

You must have read and/or execute access to the file and access to the directory.

8.5 CONTROLLING PROGRAMS

You can control programs by using three control characters: CTRL/C, CTRL/O and CTRL/T. CTRL/C halts the execution of a program; CTRL/O controls output to your terminal; CTRL/T checks the status of a running program.

8.5.1 Typing CTRL/C to Halt Execution

You may want to stop your program for several reasons.

- Unexpected things may happen in your program and it does not complete execution.
- You may write your program to get information from another file, and during execution of the program find that the other file does not exist.
- You may want to perform some other task.

To stop an executing program or command, type two CTRL/Cs. Only one CTRL/C echoes on the terminal. The program (or command) stops and returns you to command level. In the following example, you decide to stop your program.

```
@EXECUTE (FROM) SQRT.ALG
ALGOL:  SQRT
LINK:  Loading

    ALGOL  Running at 701105 Used 0:00:04.5 in 0:01:49
^C
@
```

You can now give any command that does not change the contents of memory; for example, the TERMINAL command. (You can give commands that change memory if you have "kept" forks in memory. Refer to Section 8.7 Running Multiple Programs). When you are finished, give the CONTINUE command and the program resumes where it left off. (The CONTINUE command will not continue a TOPS-20 command that you interrupted.)

Some programs (such as APL, BASIC, and EDIT) intercept the CTRL/C and do not return you to TOPS-20 command level. In these special cases, refer to the description of the particular program to return to TOPS-20 command level.

The system does not respond immediately to a single CTRL/C, but waits for the time when you would normally give input to the program. However, the system processes two CTRL/Cs immediately.

8.5.2 Typing CTRL/O to Stop Output to Your Terminal

To stop terminal output but not execution, type CTRL/O. The system prints:

```
^O...
```

and stops all output to the terminal. The program (or command) still executes, but no output appears on the terminal. When the program (or command) finishes, the system prints the TOPS-20 prompt.

```
@DIRECTORY (OF FILES) *.FOR

   PS:<MILLER>
 ARDVRK.FOR.1
 BASTST.FOR.3
 ^O...
```

If you stop output on the terminal and want to resume printing later during the execution of the same program or command, type another CTRL/O.

```
@DIRECTORY (OF FILES) *.CBL

   PS:<MILLER>
 ANDTST.CBL.6
 BEHIND.CBL.2
 DEVCHR.CBL.4
 ^O...
 WOBBLE.CBL.3
 XTMP.CBL.9
 Total of 34 files
```

Each successive pair of CTRL/Os stops and resumes terminal output.

The effect of CTRL/O is cancelled when the program requests terminal input.

8.5.3 Typing CTRL/T to Print the Run Status

You can check the progress of your program even while it is running. To do this, type CTRL/T.

The response from CTRL/T shows:

1. The current time
2. The status of your program
3. The amount of computer time used
4. The time elapsed since you logged in

In the example below, you type a CTRL/T immediately after the computer prints ALGOL:SQRT. At that time, the program is executing the instruction stored in memory location 540016. Up to this point, you have used 15.9 seconds of computer time while being logged in for 17 minutes and 2 seconds.

```
@EXECUTE (FROM) SQRT.ALG
ALGOL:  SQRT<CTRL/T>
09:36:35 SQRT Running at 540016  Used 00:00:15.9  in 0:17:02,
Load  2.08
LINK:  Loading
[LNKXCT SQRT Execution]

 TYPE THE VALUE OF X:  4

 THE SQUAREROOT OF     4.000  IS     2.000

End of execution.
@
```

Depending on when you press CTRL/T, other possible responses are:

IO WAIT AT location	This means that your program is probably waiting for you to type something.
HALT AT location	This means that your program has finished.

The symbol "location" is a 6-digit octal number that tells you which instruction in computer memory is currently being executed.

Typing a CTRL/T does not interfere with the running of your program in any way. However, if your program is printing information on your terminal at the same time that you type a CTRL/T, the response from CTRL/T is mixed with the information from your program.

The information is in the form:

```
time name status Used CPU-time in logged-in-time, Load average
```

The status message tells you the status of the program. Table 8-1 lists some of the common status messages.

Table 8-1: CTRL/T Status Messages

Message	Means the Process is:
RUNNING AT pc	Running
IO WAIT AT pc	Doing input or output
HALT AT pc	Stopped
FORK WAIT AT pc	Waiting for a process to terminate
SLEEP AT pc	Temporarily suspended

pc is the memory location of the current instruction being executed. You can cause this location to be displayed as either a symbol or an octal address by using the SET TYPEOUT MODE command. Refer to the TOPS-20 Commands Reference Manual for information on SET TYPEOUT MODE.

The load average gives a rough indication of current system use, and thus helps you estimate the length of time your program will take to run. Higher load averages tend to indicate heavy use and slow system response. Refer to the TOPS-20 WATCH document for further information on load averages.

If you stop the program by typing a CTRL/C, the system may precede any of the messages in Table 8-1 with ^C FROM. If a process terminates unexpectedly, the CTRL/T message prints in the form:

```
HALT: reason
```

where reason can be one of the messages listed in Table 8-2.

Table 8-2: Unexpected Process Termination Messages

CHANNEL n INTERRUPT AT pc There is a software interrupt on channel n when executing the instruction located at pc.
OVERFLOW AT pc There is an integer overflow when executing the instruction at location pc.
FLOATING OVERFLOW AT pc There is a floating point overflow when performing a floating point operation at location pc.
PUSHDOWN OVERFLOW AT pc There is an overflow during a pushdown stack operation at location pc.
END-OF-FILE AT pc There is an unexpected end-of-file encountered while executing the instruction at location pc.
IO DATA ERROR AT pc There is an input or output data error when executing the instruction at location pc.
FILE ERROR 3 INTERRUPT AT pc FILE ERROR 4 INTERRUPT AT pc There is a file error while executing the instruction at location pc.
ILLEGAL MEMORY READ AT pc ILLEGAL MEMORY WRITE AT pc ILLEGAL EXECUTE AT pc There is an illegal attempt to access memory at location pc.
FORK TERMINATION INTERRUPT AT pc There is a software interrupt that terminated another fork (process) while executing the instruction at location pc.
FILE OR SWAPPING SPACE EXCEEDED AT pc There is no more room in the system memory or disk storage while executing the instruction at location pc.

8.6 RUNNING PROGRAMS WITHOUT DESTROYING MEMORY

If you are executing a long-running program and find a file missing, you can stop the program without destroying the contents of memory, run another program (such as an editor) to create the missing file, and return to continue your original program. Before running another program to create the file, type two CTRL/Cs to halt the program and then give a PUSH command. The PUSH command creates a new, inferior TOPS-20 command level and a fresh copy of memory. You can now run a program without affecting the program in the superior TOPS-20 command level. When you finish, give the POP command to return to the previous memory and command level. Finally, give the CONTINUE command to resume the execution of your program.

NOTE

If you run another program without giving the PUSH command, the new program will replace the old program in memory, and you will not be able to continue the old program.

The following example illustrates how to run a FORTRAN program. As it nears completion, the program requires a file you forgot to create. Stop the program; give the PUSH command; create the file; give the POP command; and continue the program.

```
@EXECUTE (FROM) RANK.FOR      !Execute the program
FORTRAN: RANK
LINK:   Loading
[LNKXCT RANK Execution]

%FRSOPN File was not found    !The file was not found
Unit=1 DSK:NUMBER.DAT/ACCESS=SEQIN/MODE:ASCII

Enter new file specs. End with $(ALT)
*^C                           !type CTRL/C to stop
@PUSH (COMMAND LEVEL)         !Save the program and set up
                               a new copy of memory
 TOPS-20 Command processor 6.1(7)
@CREATE (FILE) NUMBER.DAT
     .
     .
     .
@POP (COMMAND LEVEL)          !Return to the last command level
@CONTINUE                     !Resume execution
NUMBER.DAT                    !Type the name of the file
STOP                          !The program finishes

END OF EXECUTION
CPU TIME: 0.38  ELAPSED TIME: 3.87:49
EXIT
```

When you need to run a program and do not want to destroy the current contents of memory, give the PUSH command, run the appropriate program, give the POP command and continue the first program. The POP command returns you to the preceding level. You can give as many pairs of the PUSH and POP commands as you need. If the system temporarily does not have enough resources to give you a new level of TOPS-20, it cancels the PUSH command and prints the message:

```
?Insufficient resources available
```

Reissue the command, and if you still get errors, you may have given too many PUSH commands without any intervening POP commands. Give a POP command. If the system cannot execute a POP command, it cancels the command and prints the message:

```
?No higher command level
```

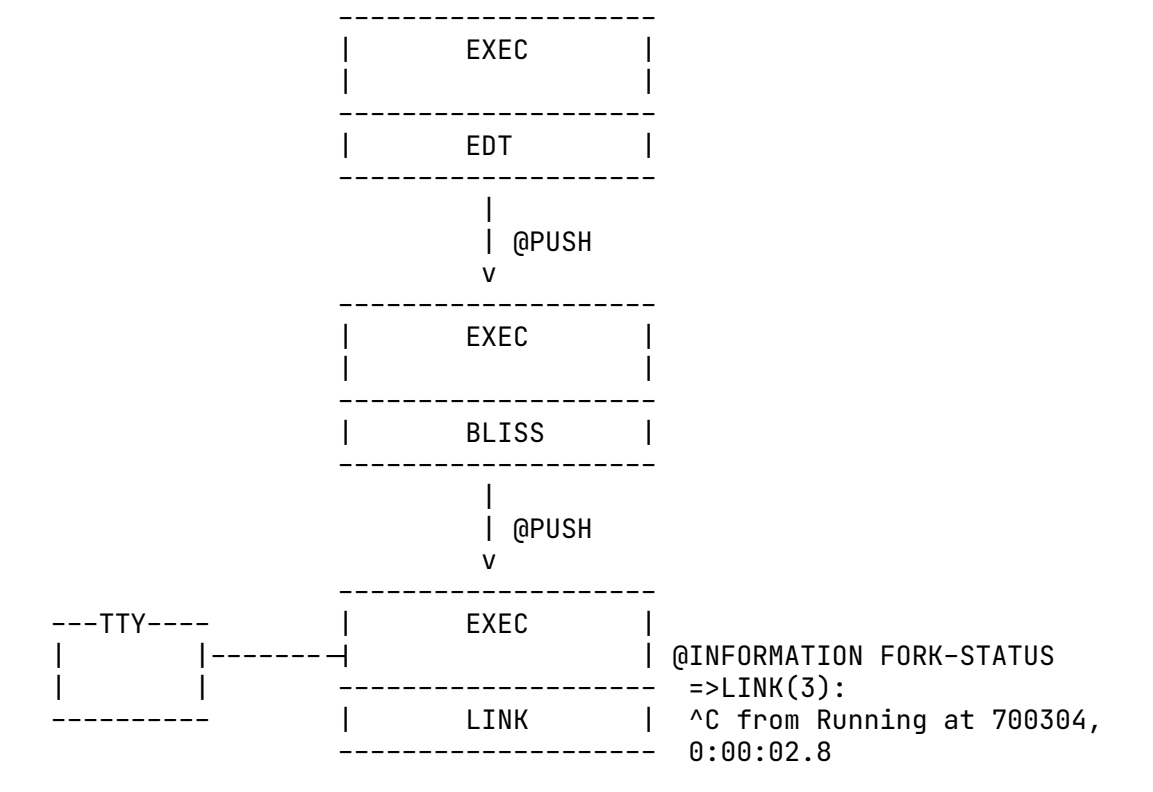
When you give a PUSH command, the contents of memory are preserved in their exact state and cannot be changed until you give a POP command to return to that level.

8.7 RUNNING MULTIPLE PROGRAMS

In addition to the PUSH and POP commands, TOPS-20 provides another method of running multiple programs without destroying memory. This feature, called "Multiforking," allows you to have multiple programs at the same TOPS-20 command level (EXEC). Each program resides in its own address space. This space is called a "fork" or a "process." Multiforking allows you to go from an editing program to a compiler and back again without reloading either program. Furthermore, you can run multiple programs and leave your terminal free for other work.

Figure 8-1 illustrates the structure of multiple forks created with the PUSH command. Note that the forks are organized in a hierarchy. Each PUSH command creates an inferior EXEC. To address a higher fork, you must POP back up the hierarchy. Each time you POP, you erase the inferior EXEC and its forks.

1.Using PUSH and POP



2.Using Multiforking

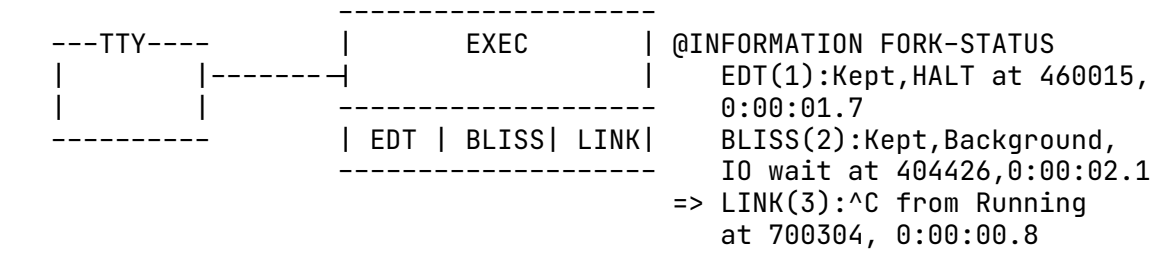


Figure 8-1: Methods of Running Multiple Programs

Now look at the structure of multiple forks created with multiforking in Figure 8-1. Note that there is only one EXEC command level, and the forks are organized parallel to each other. Because of this structure, any fork can be addressed without erasing any existing forks. Since all the forks belong to the same EXEC, the INFORMATION FORK-STATUS command displays the status of all forks. With the PUSH and POP method, you can only see the status of the "current" fork. The current fork is the fork that TOPS-20 commands refer to when you do not give a fork name as a command argument. In the fork status display, an arrow (=>) points to the current fork.

You can control forks with the multiforking-class commands discussed in the following sections.

8.7.1 Saving Forks

Normally, any time you load a program, the new program takes the place of or "resets" the program in the current fork. You can preserve the contents of a fork with the KEEP command. The KEEP command gives a fork a "kept" status. A kept fork is not cleared from memory when you run another program. Instead, a new fork is created for the new program.

In the following example, you have the EDIT program loaded, and you need to run the BLISS program while preserving the state of EDIT. First, display the fork status with the INFORMATION FORK-STATUS command. Then, make the EDIT fork a kept fork to protect it from being reset by BLISS. Next, redisplay the fork status:

```
@INFORMATION FORK-STATUS
 => EDIT (1): HALT at 6254, 0:00:22.8
@KEEP (FORK)
@INFORMATION FORK-STATUS
 => EDIT (1): Kept, HALT at 6254, 0:00:22.8
```

Now, load the BLISS program and exit BLISS to check the fork status:

```
@BLISS
BLISS>/EXIT
@INFORMATION FORK-STATUS
    EDIT (1): Kept, HALT at 6254, 0:00:22.8
 => BLISS (2): HALT at 6065, 0:00:00.2
```

Note that the arrow indicates that BLISS is now the current fork.

Forks are named after the program they contain and numbered in the order they were created. In the multiforking class commands, the fork name and number are interchangeable.

You can execute a program in a kept fork by typing only the fork name or enough letters of the fork name to distinguish it from any other fork name or TOPS-20 command. For information on automatically KEEPing forks, see the SET PROGRAM command in the TOPS-20 Commands Reference Manual.

8.7.2 Changing the Current Fork

Multiforking-class commands always refer to the current fork unless you specify a fork name as a command argument. Other EXEC commands always refer to the current fork and do not accept a fork name argument. The FORK command changes the current fork so that the EXEC commands refer to a new current fork. Use the FORK command before any EXEC command that only refers to the current fork, such as EXAMINE, DEPOSIT, and INFORMATION MEMORY-USAGE.

In the next example, you need to know how many pages are being used by the EDIT program. Since the command INFORMATION MEMORY-USAGE provides memory information about the current fork, which is now BLISS, you must first make EDIT the current fork. Give the command FORK EDIT, and check the fork status to note that EDIT is the new current fork. Then give the INFORMATION MEMORY-USAGE command.

```
@FORK (IS) EDIT
@INFORMATION FORK-STATUS
 => EDIT (1): Kept, ^C from IO wait at 2476, 0:00:00.5
    BLISS (2): HALT at 3744, 0:00:00.9
@INFORMATION MEMORY-USAGE
```

8.7.3 Creating Background Forks

A fork that is running while your terminal is at EXEC command level or at another program command level is called a "background" fork. To place the BLISS program in a background fork, type the command CONTINUE BLISS to make BLISS the current fork, and enter BLISS command level. (Note that if BLISS was a kept fork, typing only BLISS would invoke the BLISS fork.)

At the BLISS> prompt, enter a filename to start the BLISS program. Then, type two CTRL/Cs (the first CTRL/C does not appear on your terminal) to halt BLISS and bring you back to EXEC command level. Check the status of BLISS with the INFORMATION FORK-STATUS command.

```
@CONTINUE BLISS
BLISS>PROBE.BLI
;File: PUBLIC:<DBONIN.FORK>PROBE.BLI.3
^C
@INFORMATION FORK-STATUS
    EDIT (1): Kept, HALT at 6254, 0:00:22.8
 => BLISS (2): ^C from Running at 155471, 0:00:54.2
```

Now, continue BLISS with the CONTINUE command and the /BACKGROUND switch. The /BACKGROUND switch places the program in the background and lets you stay at EXEC command level.

```
@CONTINUE /BACKGROUND
@INFORMATION FORK-STATUS
    EDIT (1): Kept, HALT at 6254, 0:00:22.8
 => BLISS (2): Background, Running at 6065, 0:00:54.2
```

With BLISS running in a background fork, your terminal is now free for other work. You can give other EXEC commands, run EDIT or a new program. A new program does not clear the unkept BLISS fork while BLISS is in the background. The system notifies you when BLISS wants input by ringing the terminal bell and printing the message [BLISS: wants the TTY].

Because EDIT is in a kept fork, you can continue EDIT at its start address by typing EDIT.

```
@EDIT
[Starting]
Edit: CHECK.TXT
*
```

The [Starting] message indicates that the kept fork was continued at its start address. You can set kept forks to continue at their continue, reenter, or start address with the command, SET PROGRAM. (For more information on the SET PROGRAM command, see the TOPS-20 Commands Reference Manual).

8.7.4 Deleting Forks

Forks are valuable system resources. The maximum number available on any system is usually 512. When all the system's forks are in use, new users cannot log in and the system displays the message ?Full No more forks. Also, when users already on the system attempt to create new forks the system displays the message ?Insufficient system resources.

Therefore, you should always return your idle forks to the system. The RESET command clears forks from memory and makes them available to other users.

```
@INFORMATION FORK-STATUS
 => EDIT (1): Kept, HALT at 6254, 0:00:22.8
    BLISS (2): Background, Running at 6065, 0:02:54.2
@RESET EDIT
@INFORMATION FORK-STATUS
    BLISS (2): Background, Running at 6065, 0:02:54.2
```

Your system manager can restrict the number of forks allowed to each job. Attempting to exceed this limit also results in the message ?Insufficient system resources.

CHAPTER 9

PRODUCING AND RUNNING YOUR OWN PROGRAMS

This chapter describes:

- Producing a simple program (Section 9.1)
- Preparing a multi-module program (Section 9.2)
- Using the LOAD-class commands (Section 9.3)

9.1 PRODUCING A SIMPLE PROGRAM

To produce a simple program:

- Write the source program in a programming language
- Enter the source program into a file
- Execute (compile, load, and start) the program

If you find errors after executing the program, change the source program to eliminate the errors, and re-execute the program.

9.1.1 The Source Program

A source program is the program you input, in a programming language, to the system. The file containing your program has a file type indicating the language in which the program is written. After the system translates your program, it creates a new file containing the translation. The new file has the same file name as the source file, but it has a file type of .REL (which stands for relocatable binary). This translated program is called an object program.

To write the source program, choose one of the programming languages: ALGOL, BLISS, COBOL, FORTRAN, MACRO, or PASCAL. The languages BASIC, APL and CPL do not produce object programs (.REL files). To write a program in one of these languages, follow the procedures described in the appropriate language manual. (Refer to Appendix D, USING BASIC for an explanation of how to enter and run a BASIC program.)

The following example shows a FORTRAN program that requires you to type a number; the program then prints two times that number. Enter this program into a file.

```
C       THIS IS A SMALL FORTRAN PROGRAM
        TYPE 101
101     FORMAT (' TYPE A NUMBER:  '$)
        ACCEPT 102,X
102     FORMAT (F)
        Y=2*X
        TYPE 103,X,Y
103     FORMAT (' TWO TIMES ',F,' IS ',F)
        STOP
        END
```

9.1.2 Executing the Program

Once you enter the source program into a file, do the following:

- Compile the source program to produce an object program.
- Load the object program into memory and combine it with any routines required from the appropriate system library.
- Start the program in memory.

The language compiler or assembler translates the source program, producing an object program. The LINK program places the object program in memory, and the START command starts the program. You do not have to give all these commands to perform the individual functions. Instead, you can give the EXECUTE command, which performs the functions collectively. The COMPILE, LOAD, DEBUG, and EXECUTE commands are referred to as LOAD-class commands.

```
@EXECUTE (FROM) SMALL.FOR
FORTRAN: SMALL
MAIN.
LINK:   Loading
[LNKXCT SMALL Execution]

TYPE A NUMBER: 5
```

```
TWO TIMES      5.0000000  IS      10.0000000
STOP

END OF EXECUTION
CPU TIME: 0.07  ELAPSED TIME: 3.00
EXIT
```

9.1.3 Debugging the Program

If your program does not run correctly the first time, check for:

- Syntax errors
- Execution errors

To eliminate syntax errors, examine the line or lines for which the compiler or assembler prints errors. Edit the source program to correct the errors and re-execute it. Continue until your program is successfully translated.

If your program does not give the correct answer after it executes, check for a logic error in the program. To do this, you can carefully review the source program for any errors or you can use one of the system debugging programs: COBDDT for COBOL programs; FORDDT for FORTRAN programs and DDT for most other programs. These debugging programs allow you to stop at certain points in your program, examine the contents of the program, make changes, and then continue the program. (For more information refer to the appropriate TOPS-20 language manual.)

To get a listing of your compiled program, give the COMPILE command with the /LIST switch; the listing file has the same name as your last source file and is output directly to the line printer. When you give the COMPILE command, the system scans the list of files to be compiled. Only those files that are current (a source program not changed since the last compilation) are not recompiled. If you have a current object program, you must include the /COMPILE switch to force the compiler to recompile your source file. The following example shows how to recompile the program SMALL and get a listing:

```
@COMPILE (FROM) SMALL/LIST/COMPILE
FORTRAN: SMALL
MAIN.
@
```

To see the location of your program in the line printer output queue, give the INFORMATION OUTPUT-REQUESTS command.

```
@INFORMATION (ABOUT) OUTPUT-REQUESTS

Printer Queue:
Job Name  Req#   Limit             User
--------  ----   -----    -------------------------
* SMALL   3891      52    SARTINI                     /Unit:1
   Started at 11:02:34, Printed 0 of 52 Pages
There is 1 Job in the Queue (1 in Progress)
```

The SMALL program is the only job listed and the only job being printed.

9.1.4 Saving the Program for Future Use

Once you debug the program, load it into memory (using the LOAD command) and save the loaded program in an .EXE file (using the SAVE command). Refer to the following example. The .EXE file is an executable memory image file.

```
@LOAD (FROM) SMALL
LINK:   Loading
@SAVE (ON FILE)
 SMALL.EXE.1 Saved
```

To run the program, give a RUN command.

```
@RUN SMALL

TYPE A NUMBER: 25

TWO TIMES       25.0000000 IS    50.0000000
THREE TIMES       25.0000000 IS    75.0000000
STOP

END OF EXECUTION
CPU TIME: 0.08 ELAPSED TIME: 6.42
EXIT
```

Using the .EXE file and a RUN command saves the system from checking to see that the object file is current and loading it into memory. Make an .EXE file only when your program is running correctly. RUN is not a LOAD-class command. Therefore, if the source program for SMALL changes, giving the command RUN SMALL will not compile the program SMALL.

9.2 PREPARING A MULTI-MODULE PROGRAM

To produce a program consisting of a number of modules, do the following:

- Write the modules in a programming language and enter the modules into files
- Translate the modules, load them into memory, and then run the program

Sections 9.2.1 through 9.2.7 describe some helpful functions:

- Writing and entering modules into files
- Producing listings with cross-references to labels
- Creating and accessing subroutine libraries
- Saving the program for future use
- Saving arguments in indirect files
- Comparing files with the FILCOM program

9.2.1 Writing and Entering Modules into Files

Design the program and write the modules in a programming language. Using separate files for the modules gives you flexibility in debugging the program. If there is an error in one module, you do not have to recompile the other modules. If you do not enter each module into a separate file and an error occurs in one of the modules, you must recompile all modules in that file.

The following example illustrates entering each module into a separate file:

File COMP.FOR

```
      TYPE 101
101   FORMAT (' TYPE TWO NUMBERS: '$)
      ACCEPT 102,A,B
102   FORMAT (2F)
      CALL ADDEM(A,B)
      CALL DIFFER(A,B)
      STOP
      END
```

File ADDEM.FOR

```
              SUBROUTINE ADDEM(A,B)
              C = A + B
              TYPE 101,C
        101   FORMAT (' THE SUM IS: ',F)
              RETURN
              END
```

File DIFFER.FOR

```
              SUBROUTINE DIFFER(A,B)
              C = ABS(A - B)
              TYPE 101,C
        101   FORMAT (' THE DIFFERENCE IS: ',F)
              RETURN
              END
```

9.2.2 Executing the Program

You can run the program by giving the EXECUTE command. The FORTRAN compiler processes all three source modules and produces the three object programs; then the LINK program loads them into memory and starts them.

```
@EXECUTE (FROM) COMP,ADDEM,DIFFER
FORTRAN: COMP
MAIN.
FORTRAN: ADDEM
     .
     .
     .
END OF EXECUTION
CPU TIME: 0.16 ELAPSED TIME: 2.00
EXIT
```

9.2.3 Producing a Cross-Reference Listing

Many programs contain numerous modules that are significantly larger than those shown in the previous examples. If you want to find the place where a variable is defined or used, you must search each module line by line. However, the system can help you by creating a cross-reference listing that you can print on the line printer. The cross-reference listing shows where each variable is defined and used.

The CREF (for Cross-REFerence) program produces the listing. To use the CREF program, give the /CREF switch, along with a LOAD-class command that compiles your source program. After the program is compiled, your directory will contain a .CRF file in addition to your .REL file. Thus, if you have the file TEST.FOR and give the command:

```
@COMPILE (FROM) /CREF TEST
```

your directory will contain the files TEST.FOR, TEST.REL and TEST.CRF.

The .CRF file contains information for the CREF program. When you are ready to produce the listing, give the CREF command. This command produces listings for all the .CRF files in your connected directory that were created since you logged in. The program sends the listings to the printer. The following example produces a cross reference listing for the COMP, ADDEM, and DIFFER programs.

```
@EXECUTE (FROM) /CREF COMP,ADDEM,DIFFER  !Include /CREF
FORTRAN: COMP
MAIN.
FORTRAN: ADDEM
     .
     .
     .
END OF EXECUTION
CPU TIME: 0.15 ELAPSED TIME: 1.52
EXIT
@CREF                                    !Then run CREF
CREF:   COMP
CREF:   ADDEM
CREF:   DIFFER
```

If you already have object files for the programs, give the COMPILE command with the /CREF, /NOBINARY, and /COMPILE switches. The system produces just the .CRF file, without producing an object file.

The following example shows how to produce only cross-reference listings:

```
@COMPILE (FROM) /CREF /NOBINARY /COMPILE COMP,ADDEM,DIFFER
FORTRAN: COMP
MAIN.
FORTRAN: ADDEM
ADDEM
FORTRAN: DIFFER
DIFFER
@CREF
CREF:   COMP
CREF:   ADDEM
CREF:   DIFFER
```

If you have a COBOL program, the /CREF switch puts the cross references in the listing file that it normally produces; you do not need to run the CREF program.

Refer to the TOPS-20 User Utilities Guide for a complete description of CREF.

9.2.4 Using Subroutine Libraries

If you have a set of frequently used subroutines, you can group them in a single object file called a library file, rather than keep the object files separate. Then when you give a LOAD-class command, all you need type is the one library file specification instead of a list of subroutine file specifications. In addition, it is easier to keep track of one file, especially if a group of users is sharing the subroutines.

For example, if you have the subroutines OPREAD, OPWRIT, CLREAD, and CLWRIT, which may be called by the main program WRITER, your LOAD-class command is:

```
@LOAD (FROM) WRITER,OPREAD,OPWRIT,CLREAD,CLWRIT
```

If you place the four subroutines in a library, DOFILE, your command is shortened to:

```
@LOAD (FROM) WRITER,DOFILE/LIBRARY
```

The /LIBRARY switch causes the system to load only those subroutines that are actually called. If you use the library file and the /LIBRARY switch, after writing a main program that calls the subroutines, you do not have to remember which subroutines the program calls to include the proper file specifications in the LOAD-class command.

A library file is produced by compiling the subroutines separately and then running the MAKLIB program to construct the library file. MAKLIB is a program that manipulates .REL files. If you need to modify any one of the library files, edit the source file, recompile, and use MAKLIB to replace the subroutine in the library file.

Sections 9.2.4.1 through 9.2.4.5 show how to create a library containing four subroutines, use the library, change a subroutine, then replace the old subroutine in the library with the new one. Four subroutines: OPREAD, OPWRIT, CLREAD, and CLWRIT are entered into files, compiled, then stored in the library, DOFILE.

Refer to the TOPS-20 User Utilities Guide for a complete description of MAKLIB.

9.2.4.1 Entering the Subroutines into Files - Enter the subroutines into separate files.

File OPREAD.FOR

```
      SUBROUTINE OPREAD(NAME)
      OPREAD - OPENS A FILE FOR READING
      DOUBLE PRECISION NAME
      OPEN(UNIT=21,ACCESS='SEQIN',FILE=NAME)
      RETURN
      END
```

File OPWRIT.FOR

```
      SUBROUTINE OPWRIT(NAME)
      OPWRIT - OPENS A FILE FOR WRITING
      DOUBLE PRECISION NAME
      OPEN(UNIT=21,ACCESS='SEQOUT',FILE=NAME)
      RETURN
      END
```

File CLREAD.FOR

```
      SUBROUTINE CLREAD(NAME)
      CLREAD - CLOSES A FILE OPENED FOR READING
      DOUBLE PRECISION NAME
      CLOSE(UNIT=21,FILE=NAME)
      RETURN
      END
```

File CLWRIT.FOR

```
      SUBROUTINE CLWRIT(NAME)
      CLWRIT - CLOSES A FILE OPENED FOR WRITING
      DOUBLE PRECISION NAME
      CLOSE(UNIT=21,FILE=NAME)
      RETURN
      END
```

9.2.4.2 Compiling the Subroutines - After entering the subroutines into files, compile them to produce four separate object files.

```
@COMPILE (FROM) OPREAD,OPWRIT,CLREAD,CLWRIT
FORTRAN: OPREAD
OPREAD
FORTRAN: OPWRIT
OPWRIT
FORTRAN: CLREAD
CLREAD
FORTRAN: CLWRIT
CLWRIT
```

9.2.4.3 Creating the Library File - Create the library file by running the MAKLIB program. After starting MAKLIB, type the name of the library file, followed by an equal sign. Then type the name of each object file, followed by the /APPEND switch.

```
@MAKLIB
*DOFILE=OPREAD/APPEND,OPWRIT/APPEND,CLREAD/APPEND,CLWRIT/APPEND
```

If you want some switches to be in effect every time you run the MAKLIB program, you can create a SWITCH.INI file and include the switches. When you issue a MAKLIB command line, MAKLIB reads the SWITCH.INI file in your **connected directory** and uses the switches specified in that file. (Note that the EDIT program, on the other hand, reads the SWITCH.INI file in your logged-in directory.)

The format of the line in the SWITCH.INI file is:

```
MAKLIB/switch(es)
```

Thus, if you always want to give the /LIST switch (which lists the names of the modules that are contained in the master library) with MAKLIB, insert in the SWITCH.INI file the line

```
MAKLIB/LIST
```

Now, instead of typing the command

```
@MAKLIB
*MASTER=NEW/LIST
```

you can type the following command, and the /LIST switch is automatically included in the command:

```
@MAKLIB
*MASTER=NEW
```

If the switches occupy more than one line, use a hyphen at the end of the first line and continue on the next line.

Once you create the library file, you can list its contents on your terminal by giving a MAKLIB command with the /LIST switch in the command below. The first number following the subroutine name is the highest relocatable address it occupies, and the second number indicates its length; both numbers are octal.

```
*TTY:=DOFILE/LIST
        Listing of Modules
Produced by MAKLIB Version 2.2(104) on 26-Mar-88 at 15:00:48

        **************************
```

```
     DSK:DOFILE.REL[4,164] Created on 26-Mar-88 at 15:00:00

     OPREAD  400016  000007
     OPWRIT  400016  000010
     CLREAD  400016  000007
     CLWRIT  400016  000010
```

To end MAKLIB, type a CTRL/C.

9.2.4.4 Using the Library File - To use the library file, first create a main program that uses the subroutines. LOAD this main program and the library file into memory. Notice that the WRITER program in the example below does not use all the subroutines. When you give the LOAD command with the /LIBRARY switch, the system loads only the subroutines, OPWRIT and CLWRIT.

File WRITER.FOR

```
            DOUBLE PRECISION NAME,DAY
            CALL DATE(DAY)
            CALL OPWRIT('DATE.FIL')
            TYPE 101,DAY
      101   FORMAT (' UPDATING AS OF: ',A10)
            WRITE (21,102) DAY
      102   FORMAT (' => UPDATED ON: ', A10)
            CALL CLWRIT('DATE.FIL')
            STOP
            END
```

After entering the main program, load it with the library file and start it. Remember to include the /LIBRARY switch.

```
     @LOAD (FROM) WRITER,DOFILE/LIBRARY
     FORTRAN: WRITER
     MAIN.
     LINK:   Loading

     EXIT
     @START
     UPDATING AS OF:  26-Mar-88
     STOP

     END OF EXECUTION
     CPU TIME: 0.41 ELAPSED TIME: 1.33
     EXIT
```

9.2.4.5 Changing a Subroutine in the Library - To change a subroutine in the library, edit the source file, recompile the subroutine and use MAKLIB to update the library file. After editing the file, compile a new object file.

```
@COMPILE (FROM) OPWRIT.FOR
FORTRAN: OPWRIT
OPWRIT
```

Now, run the MAKLIB program. First, check the contents of the library file to be sure you are updating the proper file.

```
@MAKLIB
*TTY:=DOFILE/LIST
        Listing of Modules
Produced by MAKLIB Version 2A(67) on 26-Sep-88 at 15:05:06

        *************************

DSK:DOFILE.REL[4,164] Created on 26-Sep-88 at 15:00:00

OPREAD  400016  000007
OPWRIT  400016  000010
CLREAD  400016  000007
CLWRIT  400016  000010
```

Second, update the library file. Type the name of the new library file followed by an equal sign. Type the name of the library file you want to update and the /MASTER: switch. After /MASTER: type the name of the subroutine you are replacing and a comma. Last, type the name of the file containing the new subroutine followed by the /REPLACE switch. Press RETURN. When the system completes the update, it prints an asterisk.

```
*DOFILE=DOFILE/MASTER:OPWRIT,OPWRIT/REPLACE
```

You can now check the new library to be sure that the new subroutine is included. As you can see, the length of the OPWRIT subroutine has changed to include the additional statements.

```
*TTY:=DOFILE/LIST
        Listing of Modules
Produced by MAKLIB Version 2A(67) on 26-Sep-88 at 15:10:10

        **************************

DSK:DOFILE.REL[4,164] Created on 26-sep-88 at 15:09:00

OPREAD  400020  000007
OPWRIT  400035  000015
CLREAD  400020  000007
CLWRIT  400020  000010
*^C
```

Load the main program with the new library. You do not have to recompile the main program or any of the other subroutines to change OPWRIT. After loading the program, save it for future use, then start the program.

```
@LOAD (FROM) WRITER,DOFILE/LIBRARY
LINK:  Loading
@SAVE
 WRITER.EXE.1 Saved
@START
[DATE.FIL    OPENED]
UPDATING AS OF:   26-Sep-88
STOP

END OF EXECUTION
CPU TIME: 0.18 ELAPSED TIME: 0.86
EXIT
```

Refer to the TOPS-20 User Utilities Guide for more information on the MAKLIB program.

9.2.5 Loading and Saving the Program for Future Use

The example below shows how to load the main program and the library file. Instead of loading all four subroutines in DOFILE, the system loads only the two that the program actually uses (OPWRIT and CLWRIT).

```
@LOAD (FROM) WRITER,DOFILE/LIBRARY
LINK:   Loading

EXIT
```

Give the SAVE command to save the program. To run the program later, give the RUN command. Note that if you specified a name in your program by using the PROGRAM statement, the name of the saved file will reflect that name.

```
@RUN (PROGRAM) WRITER
```

Never save a program after you have started it; some storage areas may not get properly cleared during restarting.

9.2.6 Saving Arguments in Indirect Files

If the arguments for a LOAD-class command are complex, you can store them in a file called an indirect file. Later, when you give the LOAD-class command, specify the file where the arguments are stored, rather than typing the entire line. Instead of receiving the arguments directly from your terminal, the system receives them indirectly from the file. In this case precede the indirect filename with an @ sign.

If you give the indirect command file a file type of .CMD, you do not have to include a file type when giving its file specification. This example shows the line in a command file that will compile the four subroutines:

```
OPREAD,OPWRIT,CLREAD,CLWRIT
```

To use the file in a LOAD-class command, precede it with an @. You can use recognition in typing the file specification. If you do not give a file type, the system uses file type .CMD.

```
@COMPILE (FROM) @D
FORTRAN: OPREAD
OPREAD
FORTRAN: OPWRIT
OPWRIT
FORTRAN: CLREAD
CLREAD
FORTRAN: CLWRIT
CLWRIT
```

This example shows an indirect file you can use to create the program WRITER and to search the library:

```
WRITER.FOR,DOFILE.REL/LIBRARY
```

9.2.7 Comparing Changes in Files

To run the FILCOM program, type FILCOM and press RETURN; the system prints an asterisk. Type a command to FILCOM in the form:

```
destination-filespec = source-filespec, source-filespec2,/switches
```

The destination file is the file that contains the differences. It can be printed in a file or on your terminal (TTY:). The first file is the one that will be listed first in the list of differences, and the second file is the one that will be listed second. The list of switches specifies any special parameters for properly performing the comparison.

First, change one line in the file WRITER.FOR and save the new file in UPDATE.FOR. This example uses the EDIT editor.

```
@EDIT (FILE) WRITER.FOR.1 (OUTPUT AS) UPDATE.FOR
Edit: WRITER.FOR.1
*F=>$
00700   102   FORMAT (' => UPDATED ON: ', A10)
*SUPDATED$ADDED NEW DATA$.
00700   102   FORMAT (' => ADDED NEW DATA ON: ', A10)
*E

[UPDATE.FOR.1]
```

There are now two files: WRITER.FOR, which contains the original line, and UPDATE.FOR, which contains the modified line. The example below shows how to compare the two files and output the differences to your terminal. Type a CTRL/C to end FILCOM.

```
@FILCOM                              !Start FILCOM

*TTY:=WRITER.FOR,UPDATE.FOR          !Type the command
FILE 1) DSK:WRITER.FOR  CREATED: 1554 2-MAR-88
FILE 2) DSK:UPDATE.FOR  CREATED: 1556 24-MAR-88

1)1     00700   102   FORMAT (' => UPDATED ON: ', A10)
1)      00800         CALL CLWRIT('DATE.FIL')
****
2)1     00700   102   FORMAT (' => ADDED NEW DATA ON: ', A10)
2)      00800         CALL CLWRIT('DATE.FIL')
**************

%files are different

*^C                                  !Type a CTRL/C to
                                     !end FILCOM
```

For more information on the FILCOM program, see the TOPS-20 User Utilities Guide.

9.3 USING THE LOAD-CLASS COMMANDS

The LOAD-class (COMPILE, LOAD, EXECUTE, DEBUG) commands help you produce programs easily and correctly. The four commands perform all the functions you need to compile (or assemble) and debug a program:

COMPILE — The COMPILE command causes the appropriate language processor to produce object programs from source programs.

LOAD	The LOAD command causes the appropriate language processor to produce an object program and then load it into memory.
EXECUTE	The EXECUTE command causes the appropriate language processor and LINK to produce an object program, load it into memory, and then start its execution.
DEBUG	The DEBUG command causes the appropriate language processor and LINK to produce an object program, load it and the appropriate debugging program into memory, then start execution of the debugging program.

In addition to the functions listed above, the LOAD-class commands perform some helpful and timesaving functions by:

1. Recognizing the programming language in which you write your program(s) if you use the standard file types

2. Recompiling only out-of-date source programs

3. Remembering arguments of the last LOAD-class command when you omit the arguments to a current command

4. Taking arguments from an indirect file

5. Concatenating files to produce one source program

6. Passing switches to the LINK program

7. Specifying special actions with switches

Sections 9.3.1 through 9.3.6 describe some useful ways you can use these features.

Section 9.3.1 describes object programs and their uses. You may skip this section, but the information is valuable in understanding the flexibility that relocatable programs provide.

9.3.1 Object (Relocatable) and Executable Programs

The main function of any LOAD-class command is to produce an object program. (Refer to Figure 9-1.) The source program is stored in a source file with a file type that indicates the programming language. (Table 9-1 contains a list of the standard file types.) By compiling the source program with a LOAD-class command, you produce the object program stored in a file having a filename the same as the source filename. The object program is relocatable, which means you can load it into memory with subroutines, or as a subroutine, without recompiling. Hence, the object file has a file type of .REL (for relocatable) and is often called a .REL file. To run the program, you

must load the object program into memory. At that time, the various subroutines and main programs are linked. The loaded program is now executable; it may be saved in a disk file with the same name as the main source program and the file type .EXE (for executable).

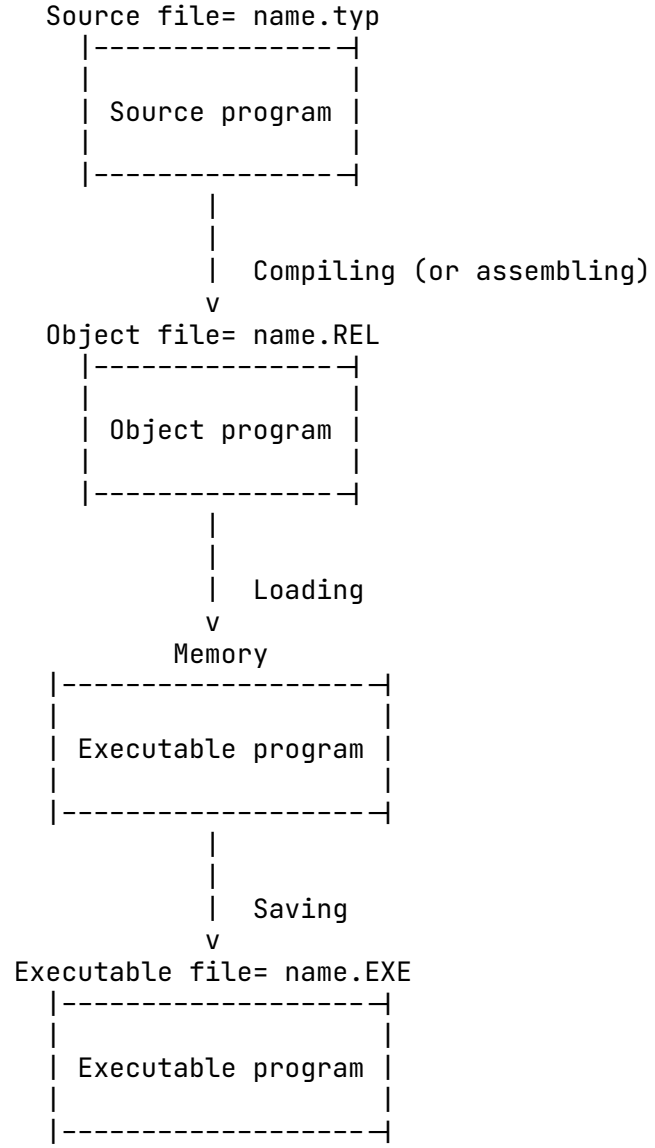


Figure 9-1: Source, Object, and Executable Programs

Any program you run must be in executable form. To form an executable program, you must compile the source program, then load the object program into memory. After you have the executable program in memory, you can save it for future use or start its execution.

In creating an executable program, you must go through the process of compiling and loading. Should you use the same subroutine in more than one program, you can reuse the object program when loading the modules into memory. By eliminating the needless compilation, you save both time and computer charges.

9.3.1.1 Using Relocatable Object Programs - Once you compile a source program into an object program, you can load that object program into memory with any combination of cooperating programs and produce an executable program. (The word program, as it is used here, refers to both main programs and subroutines.)

The examples below show how to use the FILLER subroutine in three different programs, without having to recompile it each time.

In the first example, FILLER is used with the main program, TESTER. To run TESTER, give the command:

```
@EXECUTE (FROM) TESTER,FILLER
```

The system compiles TESTER and FILLER, loads them into memory, and then starts the execution of TESTER.

The second program, LAYOUT also has another subroutine, TTYOUT, that you must include in the EXECUTE command.

```
@EXECUTE (FROM) LAYOUT,FILLER,TTYOUT
```

The system compiles LAYOUT and TTYOUT, loads them into memory with FILLER and executes LAYOUT.

The third program, GAMMA, has a POLAR subroutine that is included in the EXECUTE command.

```
@EXECUTE (FROM) GAMMA,POLAR,FILLER
```

When typing the file specifications, you do not have to place them in any specific order.

9.3.2 Selecting a File and Recognizing the Programming Language

When you give a filename as an argument to a LOAD-class command, you do not have to include the file type. For example, you can give the command:

```
@COMPILE (FROM) SMALL
FORTRAN:  SMALL
MAIN.
```

The system found the file SMALL.FOR and compiled it using FORTRAN. The file type .FOR identifies to the system that the file contains FORTRAN source code and should be compiled using FORTRAN.

When you do not include a file type in a LOAD-class command, the system searches for a file name with a file type that matches a file type in Table 9-1. The order in Table 9-1, is the order in which the system searches for a matching file.

Upon finding a matching file, the system (if necessary) compiles it using the language compiler specified by the file type. For example, if the file type is .CBL, the system uses the COBOL compiler. If there is no file type, or if the file type is not one of the file types in Table 9-1, the system defaults to the FORTRAN compiler. Note that your installation may modify this list to include other language processors.

Table 9-1: LOAD-Class Command Standard File Types

File Type	Language Compiler
MAC	MACRO
CBL	COBOL-68 or COBOL-74
C74	COBOL-74
C68	COBOL-68
74C	COBOL-74
68C	COBOL-68
ALG	ALGOL
B10	BLISS-10
BLI	BLISS-10
B36	BLISS-36
SIM	SIMULA
PAS	PASCAL
SNO	SNOBOL
FAI	FAIL
SAI	SAIL
FOR	FORTRAN
REL	Object Program, do not compile

For example, if you type the file name PAYROL, the system looks for PAYROL., PAYROL.MAC, PAYROL.CBL, PAYROL.C74, PAYROL.C68, and so on. If none of those files exists, the system prints the message: %Source file missing - PAYROL. If PAYROL.CBL exists; the system would compile PAYROL.CBL using COBOL.

If you have the files PAYROL.CBL and PAYROL.MAC and give a LOAD-class command listing the name PAYROL, the system uses the file PAYROL.MAC. If you also have the file PAYROL..1, the system uses it instead of using PAYROL.MAC. If PAYROL..1 needed compiling, the system would use the FORTRAN compiler.

9.3.2.1 Using Nonstandard File Types - If you include a file type in your file specification, the system examines the file type to select the proper translator. If the file type is not one of the standard file types shown in Table 9-1, the system uses the FORTRAN compiler.

```
@COMPILE (FROM) TEST.REF
FORTRAN: TEST
MAIN.
```

If you want to use a nonstandard file type on a non-FORTRAN program, include one of the compiler switches after the file specification.

```
@COMPILE (FROM) ENABLE.MON/MACRO
MACRO: ENABLE
```

9.3.2.2 Setting a Default Compiler - You can set a default compiler with the SET DEFAULT COMPILER-SWITCHES command. For example, this command tells the system to use the PASCAL compiler whenever you give a filename without a file type:

```
@SET DEFAULT COMPILER-SWITCHES /PASCAL
```

You can also define a file type to mean another compiler. For example, this command tells the system that a file with the type .C68 should compile with the COBOL-74 compiler instead of COBOL-68.

```
@SET DEFAULT COMPILER-SWITCHES C68 /COBOL-74
```

To display your default settings, give the INFORMATION DEFAULT COMPILER-SWITCHES command. It is recommended that you put SET DEFAULT commands in your COMAND.CMD file.

9.3.2.3 Using the File Type .REL - If you want to use a particular object file, type the filename and the file type .REL. The system does not attempt to compile this file; it simply loads it into memory.

```
@LOAD (FROM) START.REL
LINK:   Loading

EXIT
```

If you have an object program stored in a file with a file type other than .REL (this is highly discouraged), include the /RELOCATABLE switch after the file specification. Otherwise, the system attempts to compile the object program as a source program.

```
@LOAD (FROM) MIDDLE.OBJ/RELOCATABLE
LINK:   Loading

EXIT
```

9.3.2.4 Examples - If you have the file TRYIT.FOR.1 and you give the following command:

```
@EXECUTE (FROM) TRYIT
```

the system uses the file TRYIT.FOR.1. If you have the files NXTONE.MAC and NXTONE.CBL, and give the following command:

```
@EXECUTE (FROM) NXTONE
```

the system searches Table 9-1 and finds .MAC before .CBL. Therefore, the system uses the file NXTONE.MAC.

If you have the files TABLE and TABLE.FOR, and give the command:

```
@EXECUTE (FROM) TABLE
```

the system uses the file TABLE as the source program and compiles it with FORTRAN (as the default).

9.3.3 Compiling Only Out-of-Date Object Programs

Whenever you give a LOAD-class command that requires a .REL file, the system compiles an object program only if one or more of the following occurs:

1. There is no existing .REL file with the same filename.
2. The .REL file is out of date (which means that the .REL file is older than the corresponding source file).
3. You give a /COMPILE switch to the LOAD-class command.

9.3.4 Remembering Arguments to LOAD-Class Commands

If you omit the arguments to a LOAD-class command, the system supplies the arguments you specified in the last LOAD-class command containing a file specification or LINK switch. For example, if you give the following sequence of commands:

```
@COMPILE (FROM) TEST.FOR,SUB1.FOR
@EXECUTE (FROM)
```

the COMPILE command stores its arguments; then, when you omit the arguments to the EXECUTE command, the system uses the arguments you gave to the COMPILE command.

Whenever you give a LOAD-class command, the system saves its arguments only if it contains a source or object file specification or a LINK switch. Otherwise, the system appends the saved arguments from a previous command to your current command. The system does not change the saved arguments to include the contents of your current command. Suppose you give the command:

```
@COMPILE (FROM) /CREF/COBOL MANCOB,TTYIN,TTYOUT,LPOUT
```

then the command:

```
@LOAD (FROM) /MAP
```

The arguments from the COMPILE command are appended to the single switch you gave in the LOAD command. The system really executes the command:

```
@LOAD (FROM) /MAP/CREF/COBOL MANCOB,TTYIN,TTYOUT,LPOUT
```

If your next command is:

```
@COMPILE (FROM) /COMPILE
```

the system executes the command:

```
@COMPILE (FROM) /COMPILE/CREF/COBOL MANCOB,TTYIN,TTYOUT,LPOUT
```

Notice this command does not include the /MAP switch. The command:

```
@EXECUTE (FROM) LINER.MAC
```

would change the saved arguments to just the file specification LINER.MAC. If you give a command without a source file specification and there are no saved arguments to LOAD-class commands, the system prints "?No saved arguments" and cancels the command.

```
@EXECUTE
?No saved arguments
```

9.3.5 Concatenating Files to Produce One Source Program

Frequently it is useful to combine a parameter definition file or a small subroutine library with a main program. The + sign appends the file following it to the file before it to produce one source program. The example below shows how you might use a + to produce a MACRO program. The DEFS file contains parameter and storage definitions and the PROMPT file contains the main logic of the program.

File DEFS.MAC

```
        SEARCH MONSYM,MACSYM
PRMTXT: ASCIZ/NEXT COMMAND>/
        T1==1
        T2==2
```

File PROMPT.MAC

```
        TITLE PROMPT
PROMPT: HRROI T1,PRMTXT    !Get address of string
        PSOUT              !Print it
        HALTF              !Stop
        END PROMPT
```

```
@COMPILE (FROM) DEFS+PROMPT
MACRO:  PROMPT
```

9.3.6 Specifying Special Actions with Switches

You can supply various switches with the LOAD-class commands. Refer to the TOPS-20 Commands Reference Manual for a complete description of the LOAD-class commands.

Many switches have a global effect if you type them before any file specifications. For instance, the command:

```
@COMPILE (FROM) /CREF TAB,SIFT,WOB
```

produces a cross-reference listing for each file and requires significantly less typing than if you had to type:

```
@COMPILE (FROM) TAB/CREF,SIFT/CREF,WOB/CREF
```

It may be easier to set some global switches and turn them off for a particular file. If you have a list of source files with nonstandard file types that you want to compile with FORTRAN, you might use the command:

```
@COMPILE (FROM) /FORTRAN SCHED.R1,ENA.R1,DIS.R1
```

Now suppose you add the routine MONINT.R1, which is a COBOL file; you could modify your command as follows:

```
@LOAD (FROM) /FORTRAN SCHED.R1,ENA.R1,MONINT.R1/COBOL,DIS.R1
```

As a result of this command, all the files are compiled with FORTRAN except MONINT.R1, which is compiled with COBOL. The /COBOL switch located after the file affects only the file it follows.

However, if you add two COBOL programs, MON1 and MON2, your command is:

```
@LOAD /FORTRAN SCHED.R1,ENA.R1,DIS.R1,/COBOL MON1.R1,MON2.R1
```

In that case, you have changed the global /FORTRAN switch to /COBOL, and each succeeding file is compiled using COBOL.

CHAPTER 10

USING BATCH

This chapter describes:

- Preparing a batch job (Section 10.1)
- Creating a control file (Section 10.1.1)
- Monitoring your batch job (Section 10.1.2)
- Submitting a control file (Section 10.1.3)
- Setting defaults for the SUBMIT command (Section 10.1.3.1)
- Checking a batch job (Section 10.1.4)
- Examining the output from a batch job (Section 10.1.5)
- Modifying a batch job (Section 10.2)
- Canceling a batch job (Section 10.3)

10.1 PREPARING A BATCH JOB

If you have a procedure that you execute frequently, you can submit it as a batch job rather than repeatedly executing it from your terminal.

To prepare to submit a batch job, enter the commands you would normally type on a terminal into a file called a batch control file. You can submit a control file to the batch system via a punched card deck or your terminal. Submitting this file creates a request for the system to run your job. The batch system logs your job in, executes the commands stored in the batch control file, and after executing the last command in the file, ends the job by logging it off. The batch system records the input and output of the job in a log file.

When you create a control file, use any filename and a file type of .CTL. Type each command and argument in full into the control file instead of abbreviated input. You must precede each TOPS-20 command and subcommand with an @. You must precede each program command with an *.

NOTE

If you are including subcommands in a control file:

- place only one @ before a subcommand.
- place an @ before the RETURN that terminates the entire command.

You can create a BATCH.CMD file that is read by the system when your batch job is run. This file contains any TOPS-20 system commands you want executed every time you run the batch program. The BATCH.CMD file is similar to the LOGIN.CMD file the system reads every time you log in. Like the LOGIN.CMD file, a BATCH.CMD file usually contains commands such as the DEFINE command (to define logical names). Once the batch job is logged in, the system reads the BATCH.CMD file and executes the commands contained in it.

Your system manager can create a sytem-wide BATCH.CMD file. The file SYSTEM:BATCH.CMD is read by the system before reading your own BATCH.CMD file.

NOTE

Do not include TERMINAL commands in a BATCH.CMD file.

The batch program does not recognize the TOPS-20 commands listed in Table 10-1. If you include them, the system issues a fatal error message. Be certain you do not include these commands in your control file, BATCH.CMD file, or COMAND.CMD file.

Table 10-1: Illegal Commands in Batch Jobs

ATTACH
SET CONTROL-C-CAPABILITY
SET TIME-LIMIT
TALK

10.1.1 Creating a Control File

To create a control file, place all the commands you usually type on your terminal into the file. The following example shows how to create a control file that runs the FILCOM program to compare two files and prints a file containing the comparisons:

File TEST.CTL

```
@FILCOM
*SAMPLE.SCM=DATA.OLD,DATA.NEW
@PRINT SAMPLE.SCM
$
```

10.1.2 Monitoring Your Batch Job

You can include the SEND command in your batch control file to send a message informing you when the batch job is done. Use SEND's line number argument and not the user name argument for this purpose.(Refer to Chapter 3 for more information on using the SEND command). You can also include a command to run one of the mail programs. (Refer to the TOPS-20 User Utilities Guide for information on the MAIL program or, if you are using the DECmail/MS mail program, refer to the TOPS-10/TOPS-20 DECmail/MS Manual).

10.1.3 Submitting a Control File to Batch

To submit a control file to batch, give the SUBMIT command followed by the name of the control file. The SUBMIT command places the job in a waiting line called the batch input queue. When batch can accommodate another job, it selects one from the input queue.

The example below shows how to submit the TEST.CTL control file. Because the control file has the file type .CTL, you do not need to include the file type in the command.

```
@SUBMIT (BATCH JOB) TEST
[Job TEST Queued, Request-ID 105, Limit 0:05:00]
```

You can submit more than one control file to batch with the same SUBMIT command. The following example shows how to submit TEST.CTL and DATA.CTL:

```
@SUBMIT (BATCH JOB) TEST,DATA
[Job TEST Queued, Request-ID 106, Limit 0:05:00]
[Job DATA Queued, Request-ID 107, Limit 0:05:00]
```

Where you place switches in a SUBMIT command line determines the files affected by the switch.

If you place a switch after the command but before you give the filenames, all the files are affected by the switch. A switch that affects all files is called a global switch. In the following example submit TEST.CTL and DATA.CTL using a global switch /AFTER:.

```
@SUBMIT (BATCH JOB)/AFTER:8-Jun-88 TEST,DATA
[Job TEST Queued, Request-ID 108, Limit 0:05:00]
[Job DATA Queued, Request-ID 109, Limit 0:05:00]
```

If you type a command followed by a filename, a switch, and another filename, only the file preceding the switch is affected. A switch that affects only one file is called a local switch. The following example shows how to submit TEST.CTL using a local /AFTER: switch and DATA.CTL:

```
@SUBMIT (BATCH JOB) TEST/AFTER:10-Jun-88,DATA
[Job TEST Queued, Request-ID 110, Limit 0:05:00]
[Job DATA Queued, Request-ID 111, Limit 0:05:00]
```

10.1.3.1 Setting Defaults for the SUBMIT Command – If you want the SUBMIT command to always contain certain switches, give the SET DEFAULT SUBMIT command, followed by the switch or switches. To give the /OUTPUT: switch with SUBMIT commands, place the following command in COMAND.CMD:

```
@SET DEFAULT (FOR) SUBMIT /OUTPUT:NOLOG
```

To avoid having to type the SET DEFAULT SUBMIT and its arguments every time you log in to the system, put this command in a COMAND.CMD file. (Refer to Section 1.7 for information about a COMAND.CMD file.) Whenever you give a SUBMIT command, the switches you specify in the SET DEFAULT command are automatically included in the SUBMIT command. To see the defaults you have set for the SUBMIT command, give the INFORMATION (ABOUT) DEFAULTS (FOR) SUBMIT command.

```
@INFORMATION (ABOUT) DEFAULTS (FOR) SUBMIT
 SET DEFAULT SUBMIT /OUTPUT:NOLOG
```

Every time you give the SUBMIT command, the system includes the switch /OUTPUT:NOLOG in the command.

10.1.4 Checking a Batch Job

To check the progress of the batch job, give the INFORMATION BATCH-REQUESTS command. The system prints a list of all the jobs in the batch queue and their status. Certain switches specified in the SUBMIT command appear in the queue listing. The system lists these switches if their value is not the default.

To print only the status of your job, use the /USER switch with the INFORMATION BATCH-REQUESTS command. To print the status of another user's job, use the /USER: switch, followed by the user's name.

```
@INFORMATION (ABOUT) BATCH-REQUESTS

Batch Queue:
Job Name  Req#  Run Time            User
--------  -----  --------  -----------------
* VNP20    102  00:07:00  SROBINSON               In Stream:1
    Job# 32 Running EXEC Runtime 0:00:00
* CROSS    103  00:05:00  SROBINSON               In Stream:2
   Started at 08:31:09
  FOO        3  00:05:00  RETI                    /Proc:CALL37
  DATA     111  00:05:00  SARTINI
  GALAXY   104  00:10:00  SAMBERG
There are 5 Jobs in the Queue (2 in Progress)
```

10.1.5 Examining the Output from a Batch Job

The system places the output from a batch job into a log file. A log file has a filename that is the same as the job name, and a file type of .LOG. Unless you specify otherwise, the system automatically sends the log file to the line printer, but also leaves a copy of it in your directory.

Give the DIRECTORY command to see that the log file is in your directory with the control file.

```
@DIRECTORY (OF FILES) TEST

   AURORA:<HIGGINS>
   TEST.CTL.1
     .LOG.1

Total of 2 files
```

The following example contains the log file from the batch job, TEST.CTL.1.

```
                        17-Oct-88 13:20:34

BATCON Version  5(6057)               GLXLIB Version  5(1247)

           Job TEST Req #88 for EMORRILL in Stream 1

        OUTPUT:  Log                          TIME-LIMIT: 0:05:00
        UNIQUE:  Yes                          BATCH-LOG:  Append
        RESTART: No                           ASSISTANCE: Yes
        ACCOUNT: 341                          SEQUENCE:   1435
```

```
        Input from => PUBLIC:<EMORRILL>TEST.CTL.1
        Output to  => PUBLIC:<EMORRILL>TEST.LOG

13:20:36 USER    TEAL, Accounting Dept., TOPS-20 Monitor 7(7)
13:20:36 MONTR   Job 290 on TTY246 17-Oct-88 13:20:36
13:20:39 MONTR  [PUBLIC Mounted]
13:20:39 MONTR
13:20:39 MONTR  [CONNECTED TO PUBLIC:<EMORRILL>]
13:20:39 MONTR  @FILCOM
13:20:41 USER
13:20:41 USER   **SAMPLE.SCM=DATA.OLD, DATA.NEW
13:20:43 USER
13:20:43 USER   No differences encountered
13:20:43 USER
13:20:43 USER   *^C
13:20:43 MONTR  @@PRINT SAMPLE.SCM
13:20:44 MONTR  [Printer job SAMPLE queued, request #89, limit 3]
13:20:44 MONTR  @
13:20:46 MONTR  Killed by OPERATOR, TTY 233
13:20:46 MONTR  Killed Job 290, User EMORRILL, Account 341, TTY 246,
13:20:46 MONTR    at 17-Oct-88 13:20:46,  Used 0:00:02 in 0:00:12
```

The system begins each line in the log file with the time the line was processed. The system prints a code following the time that indicates the job state: at TOPS-20 command level (MONTR) or at program command level (USER). Other codes may appear as well. The remainder of the line contains system output and the lines in the control file.

The system checks that the job is at TOPS-20 command level before it processes a TOPS-20 command in the control file. Since the first command in the control file is FILCOM, the job enters FILCOM command level. The next TOPS-20 command in the control file is PRINT. Because the job is at FILCOM command level, the system must give a CTRL/C to return to TOPS-20 command level before it processes the PRINT command.

For a detailed description of batch, refer to the TOPS-10/TOPS-20 Batch Reference Manual.

10.2 MODIFYING A BATCH JOB

To change and/or add one or more switches to a previously issued SUBMIT command, give the MODIFY command. After you give the MODIFY command, type BATCH, followed by the first six letters of the jobname, or the request ID; then type the switch you want to change or add.

You can modify almost all SUBMIT command switches. To obtain a list of switches you can modify, give the MODIFY BATCH command, followed by a slash (/) and a question mark (?). The system prints a list of switches you can modify, and reprints the command line.

```
@MODIFY (REQUEST TYPE) BATCH/? Switch, or parameter to modify, one
of the following:

 /AFTER:              /BEGIN:              /CARDS:
 /DEPENDENCY-COUNT:   /DESTINATION-NODE:   /FEET:
 /JOBNAME:            /OUTPUT:             /PAGES:
 /PRESERVE            /PRIORITY:           /PROCESSING-NODE:
 /RESTARTABLE:        /SEQUENCE:           /TIME:
 /TPLOT:              /UNIQUE:             /USER:
@MODIFY (REQUEST TYPE) BATCH/
```

In the following example, modify the batch job TEST.CTL by adding the /AFTER: switch and the date August 15, 1988:

```
@MODIFY (REQUEST TYPE) BATCH (ID) TEST/AFTER:15-AUG-88
[1 Job Modified]
```

10.3 CANCELING A BATCH JOB

To remove entries you have previously placed in the batch input queue, give the CANCEL command. After you give the CANCEL command, type BATCH, followed by the first six letters of the jobname or the request ID of the job you want to remove.

Once the CANCEL command removes the entry you specify from the batch queue, the system notifies you of the removal by printing the message [1 Job Canceled]. If the system is processing the entry in the queue when you give the CANCEL command, it stops the job and prints the message, [1 Job Canceled, (1 was in progress)].

In the following example, remove the batch job TEST.CTL.

```
@CANCEL (REQUEST TYPE) BATCH (ID) TEST
[1 Job Canceled]
```

If you have several batch jobs running, you can cancel them all by using an *. Give the CANCEL command, followed by the request type you want to cancel; then type an * instead of a job name. The following example shows how to remove all of your batch jobs:

```
@CANCEL (REQUEST TYPE) BATCH *
[2 Jobs Canceled]
```

APPENDIX A

TOPS-20 COMMANDS

This appendix contains a brief explanation of the commands in the TOPS-20 Command Language. The commands are grouped in categories of similar use. Although some of these commands are not described in this manual, the purpose of this list is to make you aware of the full extent and capability of the TOPS-20 Command Language. For a complete description of all TOPS-20 commands, refer to the TOPS-20 Commands Reference Manual.

A.1 SYSTEM ACCESS COMMANDS

These commands allow you to gain and relinquish access to the system, to change jobs, and to release and connect terminals to your job.

ATTACH	Connects your terminal to a designated job.
DETACH	Disconnects your terminal from the current job without affecting the job.
DISABLE	Returns a privileged user to normal status.
ENABLE	Permits privileged users to access and change confidential system information.
LOGIN	Gains access to the TOPS-20 system.
LOGOUT	Relinquishes access to the TOPS-20 system.
UNATTACH	Disconnects a terminal from a job; it does not have to be the terminal you are using.

A.2 FILE SYSTEM COMMANDS

The file system commands allow you to create and delete files, to specify where they are to be stored, to copy them, and to output them on any device.

ACCESS	Grants ownership and group rights to a specified directory.
APPEND	Adds information from one or more source files to a new or existing disk file.
ARCHIVE	Marks a file for long-term off-line storage.
BUILD	Allows you to create, change, and delete subdirectories.
CANCEL	Removes files from any of several system queues.
CLOSE	Closes a file or files left open by a program.
CONNECT	Removes you from your current directory and connects you to a specified directory.
COPY	Duplicates a file in a destination file.
CREATE	Invokes your defined editor to create a file.
DELETE	Marks the specified file(s) for eventual deletion (disk files only).
DEFINE	Associates a logical name with one or more file, directory, or structure names.
DIRECTORY	Lists the names of files residing in the specified directory and information relating to those files.
DISMOUNT	Notifies the system that the given structure or magnetic tape is no longer needed.
EDIT	Invokes your defined editor to modify a file.
EXPUNGE	Permanently removes any deleted files from the disk.
END-ACCESS	Relinquishes ownership and group rights to a specified directory.
FDIRECTORY	Lists all the information about a file or files.
MODIFY	Changes and/or adds switches to a previously issued PRINT or SUBMIT command.

MOUNT	Requests that a structure or a magnetic tape be made available to the user.
PERUSE	Invokes your defined editor to read an existing file in read-only mode.
PRINT	Enters one or more files in the line printer queue.
RENAME	Changes one or more descriptors of an existing file specification.
RETRIEVE	Requests restoration of a file stored off-line.
TDIRECTORY	Lists the names of all files in the order of the date and time they were last written.
TYPE	Types one or more files on your terminal.
UNDELETE	Restores one or more disk files marked for deletion.
VDIRECTORY	Lists the names of all files, as well as their protection, size, and date and time they were last written.

A.3 DEVICE HANDLING COMMANDS

These commands allow you to reserve a device prior to using it, to manipulate the device, and to release it once it is no longer needed.

ASSIGN	Reserves a device for use by your job.
BACKSPACE	Moves a magnetic tape drive back any number of records or files.
DEASSIGN	Releases a previously assigned device.
EOF	Writes an end-of-file mark on a magnetic tape.
REWIND	Positions a magnetic tape backward to its load point.
SKIP	Advances a magnetic tape one or more records or files.
UNLOAD	Rewinds a magnetic tape until the tape is wound completely on the source reel.

A.4 PROGRAM CONTROL COMMANDS

The following commands help you create, run, edit, and debug your own programs.

COMPILE	Translates a source module using the appropriate compiler.
CONTINUE	Resumes execution of a program interrupted by a CTRL/C or the FREEZE command.
CREF	Runs the CREF program which produces a cross-reference listing and automatically sends it to the line printer.
CSAVE	Saves the program currently in memory so that it may be used by giving a RUN command. The program is saved in a compressed format.
DDT	Merges the debugging program, DDT, with the current program and then starts DDT.
DEBUG	Takes a source program, compiles it, loads it with the appropriate debugger and starts the debugger.
DEPOSIT	Places a value in an address in memory.
ERUN	Runs an executable program in an ephemeral (transitory) fork.
EXAMINE	Allows you to examine an address in memory.
EXECUTE	Translates, loads, and begins execution of a program.
FREEZE	Stops a running program.
FORK	Selects the fork to which TOPS-20 commands apply.
GET	Loads an executable program from the specified file into memory, but does not start it.
KEEP	Protects a fork from being cleared from memory.
LOAD	Translates a program (if necessary) and loads it into memory.
MERGE	Merges an executable program with the current contents of memory.

POP	Stops the current active copy of the TOPS-20 command level and returns control to the previous command level.
PUSH	Preserves the contents of memory at the current command level and creates a new TOPS-20 command level.
R	Runs a system program.
REENTER	Starts the program currently in memory at an alternate entry point specified by the program.
RESET	Clears the current job.
RUN	Loads an executable program from a file and starts it at the location specified in the program.
SAVE	Copies the contents of memory into a file in executable format. If memory contains a program, you may now execute the program by giving the RUN command.
SET	Sets various parameters for your job, a directory, a file, or a device.
START	Begins execution of a program previously loaded into memory.
TRANSLATE	Translates a project-programmer number to a directory name or a directory name to a project-programmer number.
UNKEEP	Cancels the kept status of a fork.

A.5 INFORMATION COMMANDS

These commands return information about TOPS-20 commands, your job, and the system as a whole.

DAYTIME	Prints the current date and time of day.
HELP	Prints information about system features.
INFORMATION	Provides information about your job, files, memory, errors, system status, queue requests, and other parameters.

SYSTAT	Outputs a summary of system users and available computing resources.

A.6 TERMINAL COMMANDS

The terminal commands allow you to clear your video terminal screen, to declare the characteristics of your terminal, and to control linking to another user's terminal.

ADVISE	Sends whatever you type on your terminal as input to a job connected to another terminal.
BLANK	Clears the video terminal screen and moves the cursor to the first line.
BREAK	Clears a terminal links made with the TALK command.
RECEIVE	Allows your terminal to receive links and advice from other users.
REFUSE	Denies links and advice to your terminal.
REMARK	Allows you to type many lines of text when using the TALK command.
SEND	Sends a message to another user's terminal.
SET	Establishes certain job-wide characteristics for the terminal.
SET HOST	Connects the terminal to another system.
TAKE	Accepts TOPS-20 commands from a file, just as if you had typed them on your terminal.
TALK	Links two terminals so that each user can observe what the other user is doing, yet does not affect either user's job.
TERMINAL	Declares the type of terminal you have, and lets you inform TOPS-20 of any special characteristics of the terminal.

A.7 BATCH COMMAND

The TOPS-20 operating system also has a Batch System to which you may submit jobs for later execution.

SUBMIT	Enters a file into the Batch waiting queue. When it is your job's turn, the commands contained in the file are executed.

APPENDIX B

STANDARD FILE TYPES

Table B-1 lists the file types that have a specific meaning to the system or to certain programs. When you create a file for use with a particular program, you should assign the correct file type. If you do, the system has more information about the file and can attempt to perform the correct function after you type a minimum set of commands or switches. Normally, no penalty arises from assigning an undefined file type, but if you assign an incorrect file type, the system may incorrectly interpret the file, especially when you use the LOAD-class commands.

Table B-1: Standard File Types

File Type	Kind of File	Meaning
68C	Source	Source file in the COBOL-68 language
74C	Source	Source file in the COBOL-74 language
A10	ASCII	ASCII version of a DECSYSTEM-20 program loaded by the PDP-11
A11	ASCII	ASCII version of a PDP-11 program loaded by the PDP-11
ABS	Object	Absolute (nonrelocatable) program
AID	Source	Source file in AID language
ALG	Source	Source file in ALGOL language

ALP	ASCII	Printer forms alignment
ATO	ASCII	PTYCON automatic command file
ATR	Binary	Attribute file in SIMULA language
AWT	Binary	Data for automatic wire tester
BAK	Source	Backup file from TECO
B10	Source	Source file in the BLISS-10 language
B20	Source	Source file in the BASIC-PLUS-2/20 language
B36	Source	Source file in the BLISS-36 language
BAS	Source	Source file in BASIC language
BCM	ASCII	Listing file created by FILCOM (binary compare)
BCP	Source	Source file in BCPL language
BFR	ASCII	Copy of VTECO buffer
BIN	Binary	Binary file
BIX	ASCII	Output of the DSR program for input to the TCX program
BLB	ASCII	Blurb file
BLI	Source	Source file in BLISS-10 language
BOX	ASCII	Output of BOX program - picture for use in specifications and manuals
BTC	ASCII	Output of the DSR program for input to the TOC program
BUG	Object	Saved to show a program error

BWR	ASCII	Beware file listing warnings about a file or program
C68	Source	Source file in the COBOL-68 language
C74	Source	Source file in the COBOL-74 language
CAL	Object	CAL data and program files
CBL	Source	Source file in COBOL-68 or COBOL-74 language
CCL	ASCII	Command file for LINK
CDP	ASCII, Binary	Spooled output for card punch
CED	ASCII	Input to COPYED
CFL	ASCII	RUNFIL command file
CKP	Binary	Checkpoint core image file created by COBOL operating system
CHN	Object	CHAIN file
CMD	ASCII	Command file
COB	ASCII	COBOL Source File
COR	ASCII	Correction file for SOUP
CPY	Binary	Copy of a crash written by SETSPD
CRF	ASCII	CREF (cross-reference) input file
CTL	ASCII	Batch control file
DAT	ASCII, Binary	Data (FORTRAN) file, output file with ANSI carriage control
DCT	ASCII	Dictionary of words
DIR	ASCII	Directory from DIRECTORY command

DMP	ASCII	COBOL compiler dump file
DOC	ASCII	Listing of modifications to the most recent version of the software
DRW	Binary	Drawing for VB10C drawing system
ERR	ASCII	Error message file
EXE	Object	Executable program
FAI	Source	Source file in FAIL language
FCL	Source	Source file in FOCAL language
FLO	ASCII	English language flowchart
FOR	Source	Source file in FORTRAN language
FRM	ASCII	Blank form for handwritten records
FTP	Source	FORTRAN test programs
GND	ASCII	List of ground pins for automatic wirewrap
HGH	Object	Nonsharable high segment of a TOPS-10 two-segment program
HLP	ASCII	Help text files
IDA	ASCII, Binary	COBOL ISAM data file
IDX	ASCII,SIXBIT	Index file of a COBOL ISAM file
INI	ASCII, Binary	Initialization file
L36	Object	LIBRARY object file for the BLISS-36 language
LAP	ASCII	Output from the LISP compiler
LIB	ASCII	COBOL source library

LOG	ASCII	Batch, PTYCON or LINK log file
LOW	Object	Low segment of a TOPS-10 two-segment program
LPT	ASCII	Spooled output for line printer
LSP	Source	Source file in LISP language
LST	ASCII	Listing data created by assemblers and compilers
MAC	Source	Source file in MACRO language
MAN	ASCII	Manual (documentation) file
MAP	ASCII	LINK map file
MEM	ASCII	Formatted text file produced by the DSR program
MID	Source	Source file in MIDAS (MIT Assembler) language
MIM	Binary	Snapshot of MIMIC simulator
MSB	Object	MUSIC compiler binary output
MUS	Source	MUSIC compiler input
N	Source	Source file in NELIAC language
NEW	All	New version of a program or file
OBJ	Object	PDP-11 relocatable binary file
OLD	Source, Object	Backup source program
OPR	ASCII	Installation and assembly instructions
OVR	Object	COBOL overlay file
P11	Source	Source program in MACX11 language

PAL	Source	Source file in PAL 10 (PDP-8 assembler)
PAS	Source	Source file in the PASCAL language
PCO	ASCII	Program change order
PL1	Source	Source file in PL1 language
PLM	ASCII	Program logic manual
PLO	Binary	Compressed plot output
PLT	ASCII	Spooled output for plotter
PPL	Source	Source file in PPL language
PTP	ASCII, Binary	Spooled output for paper-tape punch
Qxx	ASCII	Edit backup file
R36	Source	LIBARY source file for the BLISS-36 language
RAM	ASCII	DECSYSTEM-20 microcode
REL	Object	Relocatable binary file
REQ	Source	LIBARY source file for the BLISS-36 language
RIM	Object	RIM loader file
RNO	ASCII	Programming specifications in DSR input
RSP	ASCII	SCRIPT response time log file
RSX	All	Files for RSX-11
RUN	ASCII	Command file for SYSJOB
SAI	Source	Source file in SAIL language
SAV	Object	Low segment from a one-segment TOPS-10 program
SCD	ASCII	Differences in directory

SCM	ASCII	Listing file created by FILCOM (source compare)
SCP	ASCII	SCRIPT control file
SEQ	ASCII, SIXBIT	Sequential COBOL data file, input to ISAM program
SHR	Object	A TOPS-10 sharable program
SIM	ASCII	Source file in SIMULA language
SMP	Source	Source file in SIMPLE language
SNO	Source	Source file in SNOBOL language
SPC	ASCII	Corrected file for SPELL program
SPD	ASCII	Dictionary for SPELL program
SPM	ASCII	File of mispelled words for SPELL program
SPT	ASCII	SPRINT - created files
SPU	ASCII	File of uppercase words for SPELL program
SPX	ASCII	File of exception (error) lines for SPELL program
SRC	ASCII	Source files
STB		Symbol table file
STD	ASCII	Standards
SYM	Binary	LINK symbol file
SYS	Binary	Special system files
TEC	ASCII	TECO macro
TEM	ASCII, Binary	Temporary files
TMP	ASCII, Binary	Temporary files

TPB	ASCII	Typeset input for producing a .BLB file
TPC	ASCII	Typeset input for producing a .CCO file
TPD	ASCII	Typeset input for producing a .DOC file
TPE	ASCII	Typeset input for producing error message text
TPH	ASCII	Typeset input for producing a .HLP file
TPL	ASCII	Typeset input for producing a logic manual
TPM	ASCII	Typeset input for producing a .MAN file
TPO	ASCII	Typeset input for producing a programming specification
TPP	ASCII	Typeset input for producing an .OPR file
TSK	Object	An RSX-11 task image
TST	All	Test data
TV	ASCII	Command file for TV
TXT	ASCII	Text file
UPD	ASCII	Updates flagged in margin (FILCOM)
WCH	ASCII	SCRIPT monitor (WATCH) file
WRL	ASCII	Wirelist
XOR	Binary	Module data for XOR tester
XPN	Object	Expanded save file (FILEX and LINK)
Zxx	ASCII	EDIT original file (all xx)

APPENDIX C

CHANGING YOUR PROGRAM USING EDIT

This appendix shows you how to enter, run, edit, rerun and print a FORTRAN program.

C.1 ENTERING YOUR FORTRAN PROGRAM

Type the FORTRAN program show below into a file called ADDTWO.FOR. (See Chapter 5, Section 5.1.1, EDIT.) This program contains errors that you will learn to correct in sections C.2.2 through C.2.6 of this appendix.

```
@CREATE (FILE) ADDTWO.FOR
Input: ADDTWO.FOR.1
00100        C  THIS PROGRAM ADDS TWO NUMBERS.
00200        <TAB>   TYPE 191
00300        101<TAB>FORMAT  ('TYPE TWO NUMBERS.')
00400        <TAB>   ACCEPT 102,X,Y
00500        <TAB>   Z=C+Y
00600        <TAB>   Z=X+Y
00700        <TAB>   TYPE 103,X,Y,Z
00800        103<TAB>FIRMAT  (' ADDING  ',F,' TO ',F,' GIVES ',F)
00900        <TAB>   END
00100        <ESC>
*E

[ADDTWO.FOR.1]
@
```

Now that your program is in the file, run it to find any errors.

1. Type EXECUTE, and press the ESC key.

2. The system prints (FROM).

3. Type the filename and file type of your program.

4. Press the RETURN key.

The file type .FOR causes the FORTRAN compiler to translate your program.

```
@EXECUTE (FROM) ADDTWO.FOR
FORTRAN: ADDTWO
00800   103     FIRMAT  (' ADDING ',F,' TO ',F,' GIVES ',F)
?FTNNRC Line:00800 Statement not recognized

Underlined labels

  103      102      191

?FTNFTL    MAIN.        4 fatal errors and no warnings
LINK:    ?LNKSUP Loading suppressed

EXIT
@
```

The program contained the following errors:

1. In line 200, 191 should be 101.

2. In line 800, the word FORMAT is misspelled as FIRMAT.

3. Between lines 400 and 500, there should be FORMAT statement 102.

4. Line 500 should be deleted.

C.2 EDITING YOUR FORTRAN PROGRAM

You can remove the errors in your program by using the EDIT program.

C.2.1 Starting EDIT

You can start edit after you see the @ prompt:

1. Type EDIT and press the ESC key.

2. The system prints (FILE).

3. Type the name of your file.

4. Press the RETURN key.

EDIT prints the word "Edit:"; repeats the name of your file, and then prints an asterisk.

```
@EDIT (FILE) ADDTWO.FOR
Edit: ADDTWO.FOR.1
*
```

NOTE

If you type the name of a file that does not exist, EDIT prints:

```
?FILE NOT FOUND, CREATING NEW FILE INPUT:
name,type,generation 00100
```

This allows you to create a new file. If you mistyped the filename or file type, you should end this EDIT session and start over:

1. Press the ESC key and type EQ (for End and Quit).
2. Press the RETURN key.
3. The system prints the @.
4. Type a new EDIT command with an existing filename.

After you see the asterisk, you can use any of the commands described in this chapter to change your file.

C.2.2 Printing a Line

The first error in ADDTWO.FOR occurs in line 200. To view the error, print line 200 on your terminal.

To print a line:

1. Type P.
2. Type the number of the line that you want to print.
3. Press the RETURN key.

The command P200 prints line 200.

```
*P200
00200       TYPE 191
*
```

To check the other errors, you can print a group of lines:

1. Type P.
2. Type the number of the first line that you want to print.
3. Type a colon.
4. Type the number of the last line that you want to print.
5. Press the RETURN key.

The command P300:900 prints lines 300 through 900, inclusive.

```
*P300:900
00300  101  FORMAT (' TYPE TWO NUMBERS.')
00400       ACCEPT 102,X,Y
00500       Z=C+Y
00600       Z=X+Y
00700       TYPE 103,X,Y,Z
00800  103  FIRMAT ('ADDING ',F,' TO ',F,' GIVES ',F)
00900       END
*
```

C.2.3 Inserting a Line

To insert a line into your file:

1. Type I.
2. Type the number that you want to give to your new line.
3. Press the RETURN key.
4. EDIT prints the line number.
5. Type the line.
6. Press the RETURN key again.

The command I450 inserts the new line number 450 into you program.

```
*I450
00450  102<TAB> FORMAT (2F)
*
```

After you press the RETURN key at the end of the inserted line, EDIT may print another line number. You can then type another line. If you don't want to type another, press the ESC key. EDIT prints an asterisk; you can now give any EDIT command.

```
*I1000
01000  C THIS IS A COMMENT AT THE END OF A PROGRAM -
01100  C WHEN EDIT PRINTS THE NEXT NUMBER, YOU CAN TYPE A LINE.
01200  C TO STOP INSERTING, PRESS THE ESC KEY.
01300<ESC>
*
```

C.2.4 Deleting a Line

To delete a line in your file:

1. Type D.
2. Type the number of the line that you want to delete.
3. Press the RETURN key.
4. EDIT confirms that it deleted the line.

Line 500 of ADDTWO.FOR should be deleted; the variable C should be an X. Line 600 contains the correction. The command D500 deletes line 500 on page 1 of the file.

```
*D500
1 LINES (00500/1)  deleted
*
```

To delete more than one line:

1. Type D.
2. Type the number of the first line.
3. Type a colon.
4. Type the number of the last line.
5. Press the RETURN key.

The command D1000:1200 deleted lines 1000 through 1200, inclusive.

```
*D1000:1200
3 LINES (01000/1:01200)  deleted
*
```

C.2.5 Replacing a Line

To replace a line:

1. Type R.
2. Type the number of the line that you want to replace.
3. Press the RETURN key.
4. EDIT prints the line number.
5. Type the new line.
6. Press the RETURN key.
7. EDIT prints a message giving the line numbers of any deleted lines.

The error in line 200 of ADDTWO.FOR occurs because 191 should be 101. The command R200 replaces line 200.

```
*R200
00200      <TAB>      TYPE 101
1 LINES (00200/1)  deleted
*
```

C.2.6 Changing a Line Without Completely Retyping It

To replace an existing group of characters on a line with a new group of characters:

1. Type S.
2. Type the existing group of characters.
3. Press the ESC key (EDIT prints a $).
4. Type the new characters.
5. Press the ESC key.
6. Type the line number that contains the existing group of characters.
7. Press the RETURN key.
8. EDIT prints the revised line on your terminal.

In line 800 of ADDTWO.FOR, the word FIRMAT should be corrected to FORMAT. The command SFIRMANT<$>FORMAT<>800 replaces the word FIRMAT with FORMAT.

```
*SFIRMAT<ESC>FORMAT<ESC>800<RET>
00800  103     FORMAT   (' ADDING ',F,' TO ',F,' GIVES ',F)
*
```

C.2.7 Saving a File

To finish your EDIT session and save the edited file, type E and press the RETURN key. EDIT prints the name of your file and returns you to TOPS-20 command levels.

```
*E

[ADDTWO.FOR.2]
@
```

C.3 RERUNNING A FORTRAN PROGRAM

After editing your program, run it again to find out if it works.

To reexecute a FORTRAN program, type EXECUTE and press the RETURN key. You do not have to give the name of the file. If you omit the filename, TOPS-20 executes the file that you named in your last EXECUTE command. In this case, TOPS-20 executes ADDTWO.FOR.

```
@EXECUTE
FORTRAN: ADDTWO
MAIN.
LINK:   Loading
(LNKXCT ADDTWO execution)
TYPE TWO NUMBERS.
34,78
ADDING      34.0000000 TO     78,0000000 GIVES    112,0000000
CPU time 0.25   Elapsed time 10.03
@
```

C.3.1 Typing Out Your Program

To see a final copy of your FORTRAN program printed on your terminal

1. Type TYPE.
2. Press the ESC key.
3. The system prints (FILE).
4. Type the filename and file type of your file.
5. Press the RETURN key.

```
@TYPE (FILE) ADDTWO.FOR
00100    C THIS PROGRAM ADDS TWO NUMBERS.
00200            TYPE 101
00300    101     FORMAT (' TYPE TWO NUMBERS.')
00400            ACCEPT 102,X,Y
00450    102     FORMAT  (2F)
00600            Z=X+Y
00700            TYPE 103,X,Y,Z
00800    103     FORMAT (' ADDING ',F,' TO ',F,' GIVES ',F)
09000            END
@
```

APPENDIX D

USING BASIC

If you want to enter and run a BASIC program using BASIC-PLUS-2, type in your program directly to BASIC; you should not use EDIT. The following sections show how to start BASIC, and then how to enter, save, run, edit, rerun, and list your program. Section D.10 shows you how to leave BASIC.

D.1 STARTING BASIC

After you see the @, type BASIC, and press the RETURN key. BASIC prints READY.

```
@BASIC

READY
```

Once BASIC prints READY, you can use any of the BASIC commands discussed in this section. You must not, however, use any TOPS-20 command or recognition input.

D.2 ENTERING YOUR PROGRAM

1. After you see READY, type NEW to enter a new file into your working area in the computer. Press the RETURN key.

```
READY
NEW
New Program Name--
```

2. After you see New Program Name--, type a name for your program, and press the RETURN key. The program name can have up to 39 letters and numbers. The following example shows using the name SQUARE for the new program name:

```
New program name--SQUARE

READY
```

3. After you see READY, begin typing your program. Type line numbers at the beginning of each line of your program. Press the RETURN key at the end of each line.

The example shows a BASIC program as it was originally entered; the program contains an error that is corrected later in this section.

```
100!SQUARE.B20 - THIS PROGRAM CALCULATES A SQUAREROOT.
200 PRINT "TYPE A NUMBER."
300 INPUT X
400 Y = SQR(X)
500 PRINT "THE SQUAROOT OF ";X;" IS ";Y
600 STOP
700 END
```

D.3 SAVING YOUR PROGRAM

After you enter your program into your working area, type SAVE, and press the RETURN key. When you see READY, it means that BASIC saved the program you just typed into your storage area. By saving your program, you can run it at a later time without having to reenter it into the computer. (Refer to Section D.9).

```
SAVE

READY
```

To see a list of the files saved in your storage area, type CATALOG, and press the RETURN key.

```
CATALOG

ADDTWO.FOR.2
FACTOR.B20.1
RANDOM.B20.2
SQRT.ALG.2
SQUARE.B20.1

READY
```

D.4 RUNNING YOUR PROGRAM

To run the program in your working area, type RUN, and press the RETURN key. You wil see the name of your program, the time, and the date. When you see READY, your program has finished running.

```
RUN

SQART.B20
Friday, June 10, 1988 09:13:16

TYPE A NUMBER.
 ? 45.668
THE SQUAREOOT OF  45.668  IS  6,75781
STOP at line 00600 of MAIN PROGRAM

Compile time: 0.054 secs
Run time: 0.138 secs            Elapsed time: 0:00:08

READY
```

D.5 EDITING YOUR PROGRAM

To edit the program in your working area:

1. Type the number of the line that you want to change.
2. Type the new contents of the line.
3. Press the RETURN key.

```
500 PRINT "THE SQUAREROOT OF ";X;" IS ";Y
```

D.6 RENAMING YOUR PROGRAM

To rename an existing BASIC program in your storage area:

1. Type RENAME.
2. Type the new program name.
3. Press the RETURN key.

```
RENAME SQART

READY
```

D.7 RERUNNING YOUR PROGRAM

Now that you have changed your program, run it to make sure that it works properly.

```
RUN

SQUARE.B20
Friday, June 10, 1988 09:11:44

TYPE A NUMBER.
 ? 45.668
THE SQUAREROOT OF 45.668  IS  6.75781
STOP at line 00600 of MAIN PROGRAM

Compile time: 0:086 secs
Run time: 0.229 secs            Elapsed time: 0:00:10

READY
```

D.8 LISTING YOUR PROGRAM

To get a final copy of your corrected program, type LIST, and press the RETURN key.

```
LIST

SQART.B20
Tuesday, February 6, 1988 11:03:31

00100!SQUARE.B20 - THIS PROGRAM CALCULATES A SQUAREROOT.
00200 PRINT "TYPE A NUMBER."
00300 INPUT X
00400 Y = SQR(X)
00500 PRINT "THE SQUAREROOT OF ";X;"  IS ";Y
00600 STOP
00700 END

READY
```

To list a single line of your program, type LIST, type the line number of the line, and press the RETURN key.

```
LIST 500

SQART.B20
Tuesday, February 6, 1988 11:03:40

00500 PRINT "THE SQUAREROOT OF ";X;"  IS ";Y

READY
```

D.9 RUNNING AN EXISTING PROGRAM

To run an existing BASIC program, you must do two things:

1. Move a copy of the program from your storage area into your working area.

2. Type the RUN command.

To move a copy of an existing program into your working area and run the program:

1. Type OLD.

2. Type the program name.

3. Press the RETURN key.

4. Type RUN.

5. Press the RETURN key.

```
OLD FACTOR

READY
RUN<RET>

FACTOR
Tuesday, February 6, 1988 11:28:46

FIND THE FACTORIAL OF:
 ? 6<RET>
THE FACTORIAL OF 6 IS 720

RUNTIME: 0.563 SECS                ELAPSED TIME: 0:00:51

READY
```

D.10 LEAVING BASIC

To leave BASIC, type MONITOR, and press the RETURN key. When you see the @, you can type any TOPS-20 command.

```
MONITOR
@
```

To leave BASIC and log out automatically, type BYE.

INDEX