TOPS–20
User's Guide

AA–4179C–TM, AD–4179C–T1, AD–4179C–T2

December 1982

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

This document updates the document of the same name, Order No. AA–4179C–TM.

OPERATING SYSTEM: TOPS–20 V4.1 (KS/KL Model A)
TOPS–20 V5.1 (KL Model B)

SOFTWARE: TOPS–20 EXEC V5.1

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid–Atlantic Region

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884–6660

Central Region

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640–5612

Western Region

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734–4915

digital equipment corporation • marlboro, massachusetts

First Printing, March 1976
Revised, October 1976
Revised, May 1977
Updated, January 1978
Updated, April 1978
Revised, January 1980
Updated, April 1982
Updated, December 1982

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

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 middle document in a series of three 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.

Before you begin to use the TOPS-20 User's Guide, you should read the first document in the series, Getting Started With TOPS-20, to become familiar with the operating system. This manual provides sufficient information for you to create and edit files and run a few simple programs.

Once you learn about the functions described in the TOPS-20 User's Guide, you can refer to the third and most advanced manual in the series, 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.

All examples in the TOPS-20 User's Guide are produced on an LA36 hard copy terminal, using the system editor, EDIT.

The following list contains the titles and order numbers for the documentation referenced in this manual:

- Getting Started With TOPS-20 AA-4187D-TM
- TOPS-20 Commands Reference Manual AA-5115B-TM
- TOPS-20 User Utilities Guide AA-D850A-TM
- TOPS-20 Tape Processing Manual AA-H180A-TM
- TOPS-20 System Manager's Guide AA-4169F-TM
- EDIT Reference Manual AA-5415A-TM

- TV Editor Manual AA-H181A-TM
- TOPS-10/TOPS-20 Batch Reference Manual AA-H374A-TK

CHAPTER 1

GETTING ON AND OFF THE SYSTEM

This chapter describes:

- Recognizing keyboard symbols (Section 1.1)
- Getting the attention of the system (Section 1.2)
- Getting information about the terminal (Section 1.3)
- Declaring the terminal type (Section 1.4)
- Controlling terminal output (Section 1.4.1)
- Setting the terminal speed (Section 1.4.2)
- Starting a job with LOGIN (Section 1.5)
- Receiving messages (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)

The TOPS-20 commands and system programs introduced in this chapter are:

INFORMATION	RDMAIL
LOGIN	SET
LOGOUT	TERMINAL
PUSH	

1.1 RECOGNIZING KEYBOARD SYMBOLS

Although many different types and models of terminals exist, they all have similar keyboards that resemble typewriters.

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 RUBOUT.
ESC (Escape)	The ESC (or escape) key initiates a variety of different functions. • It completes a command and prompts you with a guideword. • It completes an argument. NOTE Using ESC to complete a command or request a guideword is called recognition. Refer to Section 2.2.2 for a complete description. • It ends input to some system programs. • It causes special functions to be performed by some programs. At TOPS-20 command level, the ESC key • does not echo on your terminal. • causes a question mark (?) to be printed if you made an error. • causes the terminal bell to ring 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 ALT or ALTMODE.

Table 1-1 (Cont.)
Special Function Keys

Key	Function
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.
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. (Refer to Section 1.9.4 for more information on tabs.)

1.2 GETTING THE ATTENTION OF THE SYSTEM

Type a CTRL/C or RETURN[1] to get the attention of the system. In this case (before logging in), CTRL/C does not echo on the terminal.

After you type one of these characters, a system identification message and the TOPS-20 prompt @ are printed on the terminal. If you type anything other than these characters, the system ignores the character and warns you by ringing the terminal bell.

If you do not receive the system identification message after typing CTRL/C or RETURN[1] , then 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 1.4.2 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 one of the following messages:

- ?LOGGING IN ON LOCAL TERMINALS IS CURRENTLY DISALLOWED
- ?LOGGING IN ON REMOTE TERMINALS IS CURRENTLY DISALLOWED

[1] This feature is not available with TOPS-20 monitors prior to Version 5.

If you receive one of these messages, the operator has set the system to prevent timesharing. The system notifies you when it resumes its timesharing operation by printing on your terminal 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 one of the following messages[1] :

```
          No more forks
?FULL     No more job slots
          No more SPT slots
          No more swapping space
```

Wait a few minutes; then type CTRL/C again. 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.

If you do not receive the system identification message because the terminal speed and the line speed are not compatible, you must reset one or both of the speeds. Refer to Section 1.4.2 for more information on setting terminal speed.

1.3 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 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. When you are at TOPS-20 command level, you can give commands to the system. Type the TOPS-20 command INFORMATION (ABOUT) TERMINAL-MODE and press RETURN. The system prints the information about your terminal.

---

[1] In TOPS-20 monitors prior to Version 5, you will receive only the message "?FULL".

- EXAMPLE -

```
@INFORMATION (ABOUT) TERMINAL-MODE (FOR TERMINAL)
 TERMINAL SYSTEM-DEFAULT
 TERMINAL SPEED 300
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-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.

1.4 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	VK100
MODEL 35	VT05
MODEL 37	VT50
EXECUPORT (TI)	VT52
LA30	VT100
LA36	VT125
LA38	
LA120	

NOTE

Installations can add other terminals to their individual systems.

To declare the terminal type, give the TERMINAL (MODE IS) command, and type in the exact type of your terminal. In this example, the terminal type is an LA36.

- EXAMPLE -

```
@TERMINAL (MODE IS) LA36
@
```

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 (ABOUT) TERMINAL-MODE command to see the parameters that were set as a result of your TERMINAL command.

Tell the system you are using an LA36 by giving the TERMINAL (MODE IS) command; then give the INFORMATION (ABOUT) TERMINAL-MODE command.

- EXAMPLE -

```
@TERMINAL (MODE IS) LA36
@INFORMATION (ABOUT) TERMINAL-MODE (FOR TERMINAL)
 TERMINAL LA36
 TERMINAL SPEED 300
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL NO PAUSE (ON) END-OF-PAGE
 TERMINAL LENGTH 66
 TERMINAL WIDTH 132
 TERMINAL LOWERCASE
 TERMINAL RAISE
 TERMINAL NO FLAG
 TERMINAL INDICATE
 TERMINAL NO FORMFEED
 TERMINAL NO 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 an LA36, the output conforms to the parameters for that type of terminal, that is, a page length of 66 lines, a line width of 132 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.

1.4.1 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[1]

TERMINAL NO PAUSE (ON) END-OF-PAGE

TERMINAL PAUSE (ON) 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.

TERMINAL PAUSE (ON) 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 VT52.

NOTE

You can also stop and continue output with the characters you specified in a TERMINAL PAUSE (ON) CHARACTER command. [1] The system default characters are CTRL/S and CTRL/Q, however. The TERMINAL PAUSE (ON) CHARACTER command [1] is discussed below.

TERMINAL NO PAUSE (ON) 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 (ON) END-OF-PAGE is not set, and you need the terminal output to stop at the end of a page, give the following command:

- EXAMPLE -

```
@TERMINAL PAUSE (ON) END-OF-PAGE
@
```

If TERMINAL PAUSE (ON) 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:

- EXAMPLE -

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

[1] This command is not available with TOPS-20 monitors prior to Version 5.

TERMINAL PAUSE (ON) CHARACTER x (AND UNPAUSE ON) y[1] allows you to choose your own pause and continue characters. These characters are alternatives to the CTRL/S and CTRL/Q default characters. You can specify the x and y arguments of the command 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:

-EXAMPLE-

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

To see the characters that you may have specified in the TERMINAL PAUSE (ON) CHARACTER x (AND UNPAUSE ON) y command,[1] give the INFORMATION (ABOUT) TERMINAL-MODE command:

- EXAMPLE -

```
@INFORMATION (ABOUT) TERMINAL-MODE (FOR TERMINAL) 145
 TERMINAL VT52
 TERMINAL SPEED 300 9600
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL PAUSE (ON) END-OF-PAGE
 TERMINAL PAUSE (ON) CHARACTER SPACE
 TERMINAL LENGTH 24
 TERMINAL WIDTH 77
 TERMINAL LOWERCASE
 TERMINAL NO RAISE
 TERMINAL NO FLAG
 TERMINAL INDICATE
 TERMINAL NO FORMFEED
 TERMINAL TABS
 TERMINAL NO IMMEDIATE
 TERMINAL FULLDUPLEX
@
```

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 (ON) COMMAND or the TERMINAL NO PAUSE (ON) END-OF-PAGE command, or if the system default characters, CTRL/S and CTRL/Q, are in effect, the TERMINAL PAUSE (ON) CHARACTER[1] line does not appear in the information display.

---

[1] This command is not available with TOPS-20 monitors prior to Version 5.

NOTE

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. Also note that CTRL/A is the default pause and continue character in a TOPS-20 network environment.

1.4.2 Setting the Terminal Speed

To change the rate at which data is input from or output to your terminal, set the terminal speed. This rate is called a baud rate.

When you set the terminal speed, you affect two different speeds, the line speed and the terminal speed. Although you use the TERMINAL (MODE IS) SPEED command to set the terminal speed, you actually change the speed of the line to which your terminal is connected. You must then manually change the switch on your terminal to be compatible with the changed line speed.

Terminals operate at speeds ranging from 10 to 960 characters per second. The speeds are stated in baud rates: 10 characters per second is 110 baud; 30 characters per second is 300 baud.

The system manager determines the speed for each line when the system starts up. Usually 300 baud is the slowest speed at which a line is set.

NOTE

Do not use a line speed greater than 300 baud if you are connected to the computer via a telephone unless you have special 1200 baud equipment. Also, do not set the line speed to greater than 300 baud if you are using an LA36 hard-copy terminal.

To change the speed of the line you are using, give the TERMINAL (MODE IS) SPEED command. After you press RETURN to end the command, manually change the setting on the terminal. (Note that you set line speeds with a command; you set terminal speeds by manually changing switches on the terminal.)

NOTE

On some hard-copy terminals, the switch to change the baud rate is located at the left of the keyboard. If you have set your hard-copy terminal to a speed greater than 300, contact the operator.

On some video terminals, the switch to change the baud rate is located on the underside or the back of the terminal. On others it is located on the keyboard.

To change the line speed for input and output to 2400 baud, give the TERMINAL (MODE IS) SPEED command.

- EXAMPLE -

@TERMINAL (MODE IS) SPEED (OF INPUT) 2400 (AND OUTPUT) 2400

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 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 using the switches located on the bottom of the terminal. Set the terminal switches to be consistent with the incorrect line speed; then correct the line speed. Reset the terminal switches to a valid position to resume input and output. (Refer to Appendix C for a list of compatible line speeds 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 1.5 describes starting a work session with LOGIN. Section 1.9 explains the additional parameters you can set.

1.5 STARTING A JOB WITH LOGIN

To start working on the system, you must give the LOGIN command. The LOGIN command validates you as a user, creates your job, and begins charging your account.

The LOGIN command requires a user name, a password, and an account. The command also allows you to add remarks concerning the work session. After you give the LOGIN command, the system creates a job and prints a line containing the job number, terminal number, date and time. The system prints a @ on the next line; you are now at TOPS-20 command level.

- EXAMPLE -

```
 2102 Development System, TOPS-20 Monitor 4(2505)
@LOGIN (USER) SARTINI (PASSWORD)__(ACCOUNT) 341
 Job 57 on TTY127 23-Jul-79 09:48:40
@
```

1.5.1 User Names

Your user name identifies you to the system and to other users. A user name comprises up to 39 alphanumeric characters, including hyphens and periods.

1.5.2 Passwords

To provide the security necessary in a large timesharing system, each user selects a password that must be given when logging in. A password comprises up to 39 alphanumeric characters, including hyphens.

Only you and the system manager know your password. When you type your password, it is not printed on the terminal; this prevents others from learning it and logging in to your area without your authorization.

1.5.3 Accounts

To log in to the system, you must give a valid account. An account comprises up to 39 alphanumeric characters including hyphens. 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 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 4.1.7, file attributes.) However, you can change it only to another valid account.

NOTE

Not all installations require valid accounts. Ask your system manager, if you have any question on your account.

1.5.4 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 (ABOUT) JOB-STATUS command.

1.6 RECEIVING MESSAGES

1.6.1 Ordinary Messages

Many types of messages can be printed on your terminal; one is a message of the day; another is a notice that you have a message from another user. If there is a message for you, the system informs you by printing on the next line: YOU HAVE A MESSAGE. A message of the day (also called a system message) is sent to all users on the system. This type of message automatically prints on the terminal before you receive the TOPS-20 prompt. You do not have to type a command to receive it.

- EXAMPLE -

```
 2102 Development System, TOPS-20 Monitor 4(2505)
@LOGIN (USER) SARTINI (PASSWORD)__(ACCOUNT) 341
 Job 57 on TTY127 23-Jul-79 09:49:24
 --------
 DATE:  23-JAN-79
 FROM:  OPERATOR
 TO:  SYSTEM
 --------
 SUBJECT:  SYSTEM SHUTDOWN

The system will not be available tomorrow from noon to 2:00  P.M.
due to scheduled maintenance.

 --------


@
```

When you receive notice that you have a message from another user, use the RDMAIL program to read it. To start the RDMAIL program, type RDMAIL and press RETURN. The system prints a message to insert the date and time. To read the message for the first time you do not need to type in the date and time; just press RETURN. The RDMAIL program then prints the message on the terminal.

- EXAMPLE -

```
 2102 Development System, TOPS-20 Monitor 4(2505)
@LOGIN (USER) SARTINI (PASSWORD)__(ACCOUNT)  341
 Job 57 on TTY127 23-Jul-79 09:49:57
 YOU HAVE A MESSAGE
@RDMAIL
 DATE AND TIME (/H FOR HELP)

 --------

 DATE:  23-JUL-79
 FROM:  COMBS
 TO:  SARTINI

 --------
 SUBJECT:  Project Meeting

There will be a project meeting today at 4 p.m. in the
Engineering Conference Room.

 --------
 --------

@
```

To stop the message from printing on the terminal, type CTRL/O. To return to TOPS-20 command level immediately, type two CTRL/Cs.

You can receive new messages while you are logged in. If someone sends a message to you while you are logged in, the system rings the terminal bell and prints the line:

```
[YOU HAVE A MESSAGE FROM sender]
```

Use the RDMAIL program to read the message.

If a new message of the day is sent after you log in, the system notifies you by printing the following:

```
[NEW MESSAGE-OF-THE DAY AVAILABLE]
```

To read the new message of the day, use the RDMAIL program and type /M on the same line as Date and Time (/H for help). The /M switch instructs RDMAIL to look at the message of the day file and print the new message on your terminal.

- EXAMPLE -

```
@RDMAIL
DATE AND TIME (/H FOR HELP) /M
--------
DATE:  23-JUL-79
FROM:  OPERATOR
TO:  SYSTEM
  -----
SUBJECT:  Lineprinter paper

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

--------
--------

@
```

To read any of the messages again, use the RDMAIL program and type the date and time using the format

mmm/dd/yyyy hh:mm

where:

mmm	is the first three letters of the month
dd	is the day
yyyy	is the year
hh	is the hour
mm	is the minute

If you type only the date, the system uses the time of 00:01 on that date.

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

1.6.2 Alerts

Another type of message that can be printed on your terminal is known as an alert. You can arrange for the system to buzz your terminal and issue a one-line message at any future time. You do this by giving the SET ALERT command.

- EXAMPLE -

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

[09:45:00 alert - GET READY FOR 10:00:00 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:

- EXAMPLE -

```
@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 (ABOUT) ALERTS (PENDING) command.

- EXAMPLE -

```
@INFORMATION (ABOUT) ALERTS (PENDING)
 Next alert at 1-Jul-82 11:25:00
 Other alerts set for:
   1-Jul-82 11:45:00
   1-Jul-82 12:00:00
   1-Jul-82 13:00:00
   1-Jul-82 16:30:00
   1-Jul-82 16:55:00 - ALMOST TIME TO GO HOME!
   1-Jul-82 17:00:00
   1-Jul-82 17:11:00
   2-Jul-82 00:00:00 - SUBMIT WEEKLY REPORT BY NOON
   14-Jul-82 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 about to type the TOPS-20 prompt (@), that is, upon the completion of a TOPS-20 command.

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.

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 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 (@).

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
```

If you receive the above message, correct the error and try again.

NOTE

The system processes the LOGIN command line before it reads the LOGIN.CMD file. Therefore, you are still successfully logged in to the system, even if the LOGIN.CMD file contains an error.

For example, if you always use a VT52 terminal, you can include a TERMINAL (MODE IS) VT52 command in a LOGIN.CMD file. Every time you log in, the system reads the LOGIN.CMD file and recognizes the terminal as a VT52. All output from the terminal conforms to the parameters set for a VT52.

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.

If there is an error with one of the commands in the COMAND.CMD file, 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 COMAND.CMD.1, FILE ABORTED.
```

If you receive the above message, correct the error and try again.

NOTE

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

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

1.8 ENDING A JOB WITH LOGOUT

When you complete your work, give the LOGOUT command. The LOGOUT command ends the job and prints a message similar to the following:

```
@LOGOUT
Killed Job 57, User SARTINI, Account 341, TTY 127,
 at 23-Jul-79 09:49:36, Used 0:0:00 in 0:00:12
```

If you type CTRL/C 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-Jul-79   10:50:35,   Used 0:0:0 in 5:15
```

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

To start the LOGIN process again, type another CTRL/C.

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. Sections 1.9.1 through 1.9.5 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 (MODE IS) LENGTH command.

The system uses the page length to determine where to stop terminal output when TERMINAL PAUSE (ON) 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.

- EXAMPLE -

```
@TERMINAL (MODE IS) 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 (MODE IS) 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.

- EXAMPLE -

```
@TERMINAL (MODE IS) 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 (MODE IS) NO INDICATE command. Use this command to print a memo, report, or information that you want to appear on individual pages.

- EXAMPLE -

```
@TERMINAL (MODE IS) 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.

1.9.4 Using Tab Stops

Many terminals have automatic tab stops set at every eighth column. Pressing the tab key moves the print head or cursor just beyond the column whose number is the next larger multiple of eight. This is desirable for quicker output at slow line speeds. When the terminal does not have mechanical tab stops, the system simulates them by moving the printing head or cursor the proper number of spaces to the right. When you declare the terminal type, the system simulates tab stops if they are required by the terminal.

To simulate tab stops regardless of terminal type, give the TERMINAL (MODE IS) NO TABS command.

- EXAMPLE -

```
@TERMINAL (MODE IS) NO TABS (EXIST ON TERMINAL)
```

1.9.5 Using Uppercase and Lowercase Letters

You can control the way the system handles uppercase and lowercase letters as they are sent to and from your terminal. If you prepare text files, or include comments in your programs, you probably want to use both uppercase and lowercase letters.

The system controls case shifting separately on input and output. Therefore, you may type lowercase letters on a terminal that can only print uppercase letters. The lowercase letters you type are sent to the system as lowercase, but they appear on your terminal as uppercase.

The examples in the following sections contain a line beginning with an exclamation mark. Using an exclamation mark at the beginning of a line indicates to the system that the information you type on that line is a comment. Refer to Section 2.6 for more information on adding comments.

1.9.5.1 **Testing for Lowercase Letters** - You can determine if your terminal is capable of sending lowercase letters by giving the TERMINAL (MODE IS) LOWERCASE command. Type a few words to see if the characters are lowercase.

- EXAMPLE -

```
@TERMINAL (MODE IS) LOWERCASE (EXISTS ON TERMINAL)
@! THIS IS A TEST
```

If the terminal is capable of doing so, it will send lowercase letters. If the terminal sends uppercase letters, the system may be raising the lowercase letters you type to the corresponding uppercase letters. Give the command TERMINAL (MODE IS) NO RAISE, and type again.

- EXAMPLE -

```
@TERMINAL (MODE IS) NO RAISE (TERMINAL INPUT)
@! This is a test
```

If your terminal still prints uppercase letters, it is not capable of printing lowercase letters. However, the terminal may be able to send lowercase letters to the system. To find out, give the TERMINAL (MODE IS) NO LOWERCASE and TERMINAL (MODE IS) FLAG commands. When you give the TERMINAL (MODE IS) FLAG command, the system flags each uppercase letter by printing an apostrophe before it. To flag uppercase letters, you must also set TERMINAL (MODE IS) NO LOWERCASE, and type a third line.

```
@TERMINAL (MODE IS) NO LOWERCASE (EXISTS ON TERMINAL)
@TERMINAL (MODE IS) FLAG (UPPERCASE CHARACTERS)
@!  'THIS IS A TEST
```

If only the uppercase letters you typed are flagged (the uppercase letter preceded by a '), the terminal can send lowercase letters but is printing them as corresponding uppercase letters. If all the letters you typed are flagged (as in the following example) the terminal can send only uppercase letters.

```
@TERMINAL (MODE IS) NO LOWERCASE (EXISTS ON TERMINAL)
@TERMINAL (MODE IS) FLAG (UPPERCASE CHARACTERS)
@!  'I'H'I'S 'I'S 'A 'I'E'S'I
```

1.9.5.2 **Raising Lowercase Letters in Input** - To raise any lowercase letter you type to the corresponding uppercase letter, give the TERMINAL RAISE command. To restore the ability to type lowercase letters give the TERMINAL NO RAISE command. TERMINAL RAISE is the default.

The following example illustrates typing in lowercase, converting to uppercase and returning to lowercase.

```
@TERMINAL (MODE IS) LOWERCASE
@!  This is a line in uppercase and lowercase
@terminal (MODE IS) raise (TERMINAL INPUT)
@!NOW ALL THE TERMINAL CAN TYPE IS UPPERCASE
@TERMINAL (MODE IS) NO RAISE (TERMINAL INPUT)
@!  This is a return to uppercase and lowercase.
```

1.9.5.3 **Printing Lowercase Letters in Output** - To print lowercase letters (on a terminal capable of printing lowercase letters), give the TERMINAL (MODE IS) LOWERCASE command.

```
@TERMINAL (MODE IS) LOWERCASE (EXISTS ON TERMINAL)
```

To print all uppercase letters (on a terminal capable of printing lowercase letters), give the TERMINAL (MODE IS) NO LOWERCASE command.

```
@TERMINAL (MODE IS) NO LOWERCASE (EXISTS ON TERMINAL)
```

If the terminal cannot print lowercase letters and you want to know which letters are uppercase and which are lowercase, give the TERMINAL (MODE IS) FLAG command. (Remember to set TERMINAL (MODE IS) NO LOWERCASE as well.)

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 (Section 2.6)
- Correcting input errors (Section 2.7)
- Operating system stoppage (Section 2.8)

The TOPS-20 commands mentioned in this chapter are:

CONTINUE	EXECUTE	LOGOUT
CONNECT	EXPUNGE	PRINT
DAYTIME	INFORMATION	TALK
DIRECTORY	LOGIN	TERMINAL

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. (Guide words 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. A command terminator ends the command and is a carriage return (RET or CR key) or a line feed (Line Feed or LF key).

Some commands require no arguments. The following example illustrates the DAYTIME command, which does not require an argument.

- EXAMPLE -

```
@DAYTIME
 Tuesday, July 23, 1979 09:50:41
@
```

Other commands require one or more arguments. Arguments can be letters, numbers, or a combination of both. A common argument is a filename. (Refer to Section 4.1.4 for a description of filenames.) 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:

- EXAMPLE -

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

   PS:<PORADA>
 TEST.FOR.3
@
```

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:

- EXAMPLE -

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

If you try to give a switch on a TOPS-20 command that does not accept switches, the system prints an error message. The format of the error message differs depending upon the command you attempted.

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:

- EXAMPLE -

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

   PS:<PORADA>
  TEST.FOR.2
      .OOR.1
      .REL.3
TOTAL OF 3 FILES
@
```

If you try to enter subcommand level in a TOPS-20 command that does not accept subcommands, the system prints a ? and/or an error message. The format of the error message differs depending upon the command you attempted, but is similar to the one in the following example.

- EXAMPLE -

```
@PRINT (FILES) ,
?Invalid PRINT command - Not a switch - does not begin with slash
@
```

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.

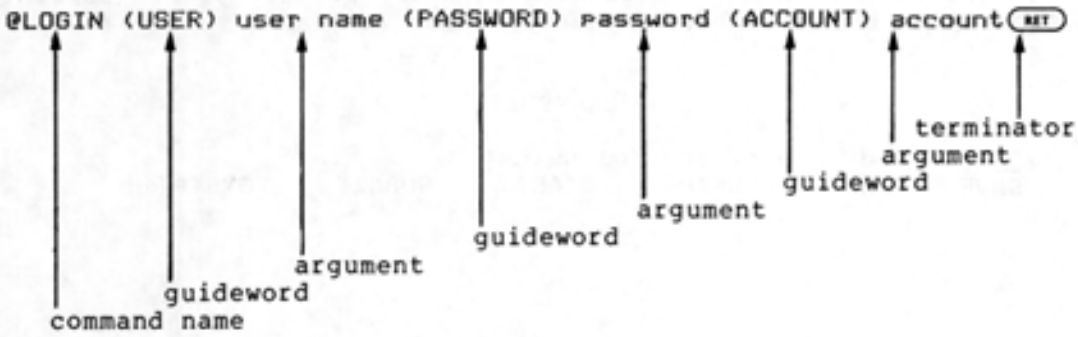


Figure 2-1 Fields of A Command

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.

NOTE

Refer to Section 1.4.1, Controlling Terminal Output, if you have a video terminal and want to prevent output from rolling off the screen too rapidly.

- EXAMPLE -

(CTRL/C)

```
 2102 Development System, TOPS-20 Monitor 4(2505)
@? Command, one of the following:

 ACCESS          ADVISE          APPEND          ARCHIVE         ASSIGN
 ATTACH          BACKSPACE       BREAK           BUILD           CANCEL
 CLOSE           COMPILE         CONNECT         CONTINUE        COPY
 CREATE          CREF            CSAVE           DAYTIME         DDT
 DEASSIGN        DEBUG           DEFINE          DELETE          DEPOSIT
 DETACH          DIRECTORY       DISABLE         DISCARD         DISMOUNT
 EDIT            ENABLE          END-ACCESS      EOF             EXAMINE
 EXECUTE         EXPUNGE         FDIRECTORY      FORK            GET
 HELP            INFORMATION     LOAD            LOGIN           LOGOUT
 MERGE           MODIFY          MOUNT           POP             PRINT
 PUNCH           PUSH            R               RECEIVE         REENTER
 REFUSE          REMARK          RENAME          RESET           RETRIEVE
 REWIND          RUN             SAVE            SET             SKIP
 START           SUBMIT          SYSTAT          TAKE            TALK
 TDIRECTORY      TERMINAL        TRANSLATE       TYPE            UNATTACH
 UNDELETE        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.

You can use the question mark to list the TOPS-20 commands beginning with a specific letter or letters. For example, to list commands beginning with the letter S, type S followed by a question mark. The system prints the commands beginning with the letter S, then reprints the line up to the point where you typed the ?.

- EXAMPLE -

```
@S? Command, one of the following:
 SAVE     SET     SKIP     START     SUBMIT     SYSTAT
@S
```

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. The system prints all the possible arguments and reprints the command.

- EXAMPLE -

```
@TERMINAL (MODE IS) ? one of the followins:
 33                 35              37              EXECUPORT
 FLAG               FORMFEED        FULLDUPLEX      HALFDUPLEX
 HELP               IMMEDIATE       INDICATE        LA120
 LA30               LA36            LA38            LENGTH
 LINE-HALFDUPLEX    LOWERCASE       NO              PAGE
 PAUSE              RAISE           SPEED           SYSTEM-DEFAULT
 TABS               TERMINET        TI              TYPE
 VT05               VT100           VT50            VT52
 WIDTH
@TERMINAL (MODE IS)
```

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

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

Type another question mark to find out which argument the system expects you to give. The system prints NUMBER and reprints the command.

```
@TERMINAL (MODE IS) LENGTH (OF PAGE IS)? NUMBER
@TERMINAL (MODE IS) LENGTH (OF PAGE IS)
```

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

```
@TERMINAL (MODE IS) 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.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@DIRECTORY (OF FILES) ;
@@? confirm with carriage return
  or one of the following:
 ACCOUNT                          ALPHABETICALLY
 ARCHIVE                          BEFORE
 CHECKSUM                         CHRONOLOGICAL
 CRAM                             DATES
 DELETED                          DOUBLESPACE
 EVERYTHING                       FIND
 GENERATION-RETENTION-COUNT       HEADING
 INVISIBLE                        LARGER
 LENGTH                           LPT
 NO                               OFFLINE
 OUTPUT                           PROTECTION
 REVERSE                          SEPARATE
 SINCE                            SIZE
 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.

- EXAMPLE -

```
@PRINT (FILES) /? /SPOOLED-OUTPUT
  or Job switch, one of the following:
 /ACCOUNT:              /AFTER:                /DESTINATION-NODE:
 /FORMS:                /GENERIC               /JOBNAME:
 /LIMIT:                /LOWERCASE             /NOTE:
 /NOTIFY:               /PRIORITY:             /UNIT:
 /UPPERCASE             /USER:
  or File switch, one of the following:
 /BEGIN:     /COPIES:      /DELETE       /FILE:        /HEADER
 /MODE:      /NOHEADER     /PRESERVE     /PRINT:       /REPORT:
 /SEQUENCE: /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 name, guidewords, arguments and subcommands or switches (if any), using a space to separate the fields. To log in using full input, type the

complete LOGIN command line. In this example, the system does not print the guideword (PASSWORD) or the actual password although both were typed by the user.

- EXAMPLE -

@LOGIN (USER) SARTINI (ACCOUNT) 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 the remainder of the command name.
2. prints a guideword.
3. prints the remainder of the argument.
4. rings the terminal bell, indicating that you need to type more information.
5. prints ?, indicating that you made an error.

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.

- EXAMPLE -

@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 IS). Press RETURN to end the command.

- EXAMPLE -

```
@INFORMATION (ABOUT) I? ONE OF THE FOLLOWING:
 TAPE-PARAMETERS
 TERMINAL-MODE


@INFORMATION (ABOUT) TERMINAL-MODE
 TERMINAL LA36
 TERMINAL SPEED 300
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL NO PAUSE (ON) END-OF-PAGE
 TERMINAL LENGTH 66
 TERMINAL WIDTH 132
 TERMINAL LOWERCASE
 TERMINAL RAISE
 TERMINAL NO FLAG
 TERMINAL INDICATE
 TERMINAL NO FORMFEED
 TERMINAL NO TABS
 TERMINAL NO IMMEDIATE
 TERMINAL FULLDUPLEX
@
```

If you use recognition where it is not appropriate (such as at the end of a command or when a password is required), the system prints ? or 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 the rest of a partial file specification. (Refer to Chapter 4 for more information on using recognition with file specifications.)

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 leave a space; type the full user name (here it is SARTINI) and leave a space; type the password (it is not printed) and leave a space; type the account (here it is 341) and press RETURN.

- EXAMPLE -

```
@LOG SARTINI 341
```

Certain commands have accepted abbreviations, even though the abbreviation is not unique. Table 2-1 lists some of the commands that require special abbreviations.

Table 2-1
Accepted Command Abbreviations

Command	Accepted Abbreviation
CONTINUE	CON
CONNECT	CONN
EXECUTE	EX
EXPUNGE	EXP
LOGIN	LOG
LOGOUT	LOGO

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 IS). To get information about the terminal parameters, just type the abbreviation TE.

- EXAMPLE -

```
@INFORMATION T? ONE OF THE FOLLOWING:
 TAPE-PARAMETERS
 TERMINAL-MODE


@INFORMATION (ABOUT) TE
 TERMINAL LA36
 TERMINAL SPEED 300
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL NO PAUSE (ON) END-OF-PAGE
 TERMINAL LENGTH 66
 TERMINAL WIDTH 132
 TERMINAL LOWERCASE
 TERMINAL RAISE
 TERMINAL NO FLAG
 TERMINAL INDICATE
 TERMINAL NO FORMFEED
 TERMINAL NO TABS
 TERMINAL NO IMMEDIATE
 TERMINAL FULLDUPLEX
@
```

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.

- EXAMPLE -

```
@DIRECTORY,
@@SIZ
@@
                    PGS

   PS:<SARTINI>
 CHAPT2.QXT.13      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
@
```

NOTE

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.

- EXAMPLE -

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

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

2.5 CONTINUING COMMANDS

If a command is longer than one line, type a hyphen at the end of the line, press RETURN, and continue typing the command on the next line.

Do not type a hyphen in the middle of a word or the middle of a file specification. (Refer to Section 4.1 for a description of file specifications.)

2.6 ADDING COMMENTS

To add a comment at the end of a command line, type an exclamation mark (!) followed by the text. End the comment by typing another ! or, if you are at the end of a line, press RETURN. The text you type between the exclamation marks or between an exclamation mark and the end of the line does not affect the command.

The following example shows how to add comments with the PRINT command:

- EXAMPLE -

```
@PRINT (FILES) TEST.FOR !OLD FILE
[Job TEST Queued, Request-ID 47, Limit 27]
@
```

If the comment you include exceeds one line, type a hyphen at the end of the line, press RETURN, and continue typing.

To type a complete line as a comment, start the line with ! and end it with ! or RETURN.

The ! 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.

2.7.1 DELETE - Erasing a Character

To erase a character, press DELETE. Each time you press DELETE, the system erases the last character you typed and prints the erased character, followed by a backslash (\). When you delete all characters and spaces back to the @ sign, the system rings the terminal bell.

NOTE

If you are using a video terminal that you have declared as a video terminal, the system erases the characters from the screen rather than printing the character followed by a backslash, (\) as it does on a hard-copy terminal.

The following example illustrates the use of DELETE when typing the TERMINAL command.

- EXAMPLE -

```
@TERMINAL (MODE IS) LO366\3\O\A36
```

To try this example, type TER and press ESC; the system prints MINAL (MODE IS). Type LO36; then press DELETE three times to erase the incorrect characters. The system prints 6\3\O\. Type A36 and press RETURN.

2.7.2 CTRL/R - Reprinting a Line

To reprint a command line, incorporating the editing performed by DELETE, type CTRL/R. CTRL/R prints a clean copy of the command line for you to check.

NOTE

If you are using a video terminal that you have declared as a video terminal, the system overprints the current line instead of printing the current command on the next line. In this case, CTRL/R does not produce a line different from the current line.

The following example illustrates the use of CTRL/R with the TERMINAL command.

- EXAMPLE -

```
@TERMINAL (MODE IS) LO36\6\3\O\A36 CTRL/R
@TERMINAL (MODE IS) LA36
```

To try this example, type TER and press ESC; the system prints MINAL (MODE IS). Type LO36 and press DELETE three times to erase the three characters; then type A36. Type a CTRL/R on the same line. This control character does not echo on the terminal. The system reprints the command line, incorporating the typing corrections you made using DELETE.

2.7.3 CTRL/U - Erasing an Entire Line

To erase the current command line, type CTRL/U. CTRL/U deletes the line and prevents the system from processing it. Although CTRL/U does not echo on the terminal, the system prints three Xs at the end of the command line and a @ sign on the following line.

NOTE

If you are using a video terminal that you have declared as a video terminal, the system erases the line from the screen rather than printing the three Xs.

The following example illustrates the use of CTRL/U with the TERMINAL command.

- EXAMPLE -

```
@TERMINAL (MODE IS) LENGTH XXX
@
```

To try this example, type TER and press ESC; the system prints MINAL (MODE IS). Type LENGTH, followed by CTRL/U. The system prints three Xs on the same line and the @ prompt on the next line.

2.7.4 CTRL/W - Erasing a Word

To erase a word, type CTRL/W. Although CTRL/W does not echo on the terminal, the system prints an underscore and you can continue typing the command.

NOTE

If you are using a video terminal that you have declared as a video terminal, the system removes the characters from the screen instead of printing the underscore.

The following example illustrates the use of CTRL/W with the TERMINAL command. To see what was deleted by CTRL/W, use CTRL/R to reprint the line.

- EXAMPLE -

```
@TERMINAL (MODE IS) LENGTH (CTRL/W)
@TERMINAL (MODE IS)
```

To try this example, type TER and press ESC. The system prints MINAL (MODE IS). Type LENGTH and then type CTRL/W. The system prints an underscore, and you can continue to type the command. To see what was deleted by CTRL/W, type a CTRL/R to reprint the line.

If you are using CTRL/W in commands that contain punctuation, the system prints an underscore and deletes all letters and numbers (alphanumerics) back to a punctuation character. The next CTRL/W you type deletes the punctuation character and all letters and numbers back to the next punctuation character. A punctuation character in this case (using CTRL/W) is defined as any character other than letter or digit; (such as space, period, comma, hyphen, parenthesis).

The example below illustrates the use of CTRL/W with a command line that contains punctuation characters. To do this example, type PRI and press ESC; the system prints NT (FILES). Type TEST.FOR.3.

```
@PRINT (FILES) TEST.FOR.3
```

Type CTRL/W; the system prints an underscore and deletes back to the first period it finds. Type CTRL/R to reprint the line.

```
@PRINT (FILES) TEST.FOR.3 (CTRL/W)
@PRINT (FILES) TEST.FOR.
```

Type CTRL/W again; the system prints an underscore and deletes back to the next period. Type CTRL/R to reprint the line.

```
@PRINT (FILES) TEST.FOR._(CTRL/R)
@PRINT (FILES) TEST.
```

Type CTRL/W a third time; the system prints an underscore and deletes back to the space. Type CTRL/R to reprint the line.

```
@PRINT (FILES) TEST._(CTRL/R)
@PRINT (FILES)
```

Type CTRL/W once more; the system prints an underscore and deletes only the space, because there are two adjacent punctuation characters (the space and the right parenthesis). Type a second CTRL/W; the system prints another underscore and deletes back to the left parenthesis. Type CTRL/R to reprint the line.

```
@PRINT (FILES)_(CTRL/R)
@PRINT (
```

Type CTRL/W; the system prints an underscore and deletes back to the next space, because again there are two adjacent punctuation characters (left-hand parenthesis and space). Type a second CTRL/W; the system prints another underscore and deletes back to the @ sign. Type CTRL/R to reprint the line.

```
@PRINT (_(CTRL/R)
@
```

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 ?, and sometimes an error message. To reprint the command line up to the erroneous field, type CTRL/H. (It will echo on the terminal.) The system reprints the command line up to the field that is in error, and you can now complete the command correctly.

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

- EXAMPLE -

```
@TERMINAL (MODE IS) LENGTH-WIDTH
?
@^H
@TERMINAL (MODE IS) LENGTH 66
```

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

2.8 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:SYSTEM IN OPERATION]
```

Once the system restarts after a fatal error, you must type a CTRL/C and 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)
- Sending mail (Section 3.3)
- Communicating with the operator (Section 3.4)
- Controlling messages from users (Section 3.5)
- Controlling system messages (Section 3.6)

The TOPS-20 commands and programs mentioned in this chapter are:

BREAK	REFUSE
INFORMATION	REMARK
MAIL	SYSTAT
PLEASE	TALK
RECEIVE	

3.1 GETTING A LIST OF USERS ON THE SYSTEM

To get a list of users currently on the system, give the SYSTAT command. The SYSTAT command reports on the status of the system:

The following example shows typical output from a SYSTAT command:

- EXAMPLE -

```
@SYSTAT
 Tue 31-Jul-79 9:45:27  Up 2:09:08
 15+9 Jobs   Load av (class 0)   8.74  7.10  4.36

  Job  Line Program User
   6   152  EXEC    SAMBERG
   7    42  EXEC    DBELL
   8   174  EXEC    WALLACE
   9   123  EXEC    DRUEKE
  10   150  EXEC    WHEELER
  11    75  VTECO   HURLEY
  12     3  MONMED  R.ACE
  13   175  EXEC    SYLOR
  14    37  EXEC    WEISS
```

```
 15    117  EXEC     LEAPLINE
 16    102  EXEC     WEBBER
 17     50  EXEC     GUNN
 18    121  EXEC     ORAN
 19    215  NRUNOF   LNEFF
 20*    41  SYSTAT   SARTINI

  1    205  PTYCON   OPERATOR
  2    207  MFORK    OPERATOR
  3    210  NETCON   OPERATOR
  4    211  OPR      OPERATOR
  5      2  TV       OPERATOR
 30    212  ACJ      OPERATOR
 32    213  EXEC     OPERATOR
 33    DET  PERF     OPERATOR
 38    214  IBMSPL   OPERATOR
 56    DET  EXEC     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 Tuesday, July 31, 1979 at 9:45:27 AM. The system has been up just over two hours.

The second line gives the number of user jobs plus the number of operator jobs. 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 and the user.

If you want to talk to a user on the system, you can communicate by linking terminals, or by sending mail.

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.

- EXAMPLE -

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

- EXAMPLE -

```
@TALK (TO) MAYO

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

To avoid typing the exclamation mark 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 following line.

- EXAMPLE -

```
@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 mark 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.

- EXAMPLE -

```
@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. However, if the user you are linked to is outputting, whatever you type will appear on that output on the user's terminal (but not in the user's output file).

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, use 'MAIL' to send mail to user
```

Refer to Section 3.5 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

Use 'MAIL' to send mail to user
```

3.3 SENDING MAIL

Another way to communicate with a user is to send mail with the MAIL program. You can send mail to a user currently on the system, or to a user who is not logged in. The MAIL program can also send messages to a group of users. To start the MAIL program, type MAIL and press RETURN; the system prints 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; the system prints MESSAGE (TERMINATE WITH ESC OR CTRL/Z):. Type your message and end it with an ESC or CTRL/Z. If there are no errors, the system prints a three-line message confirming that the mail was sent.

- EXAMPLE -

```
@MAIL
TO: PORADA, D.CROWLEY, MCELMOYLE
CC: BROPHY
SUBJECT: SYSTEM CHANGES
MESSAGE (TERMINATE WITH ESC OR CTRL/Z):

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

PROCESSING MAIL...
NO ERRORS
-DONE-
@
```

If you send mail frequently to a group of users, store the list of names in a file. Then, when you run the MAIL 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.)

- EXAMPLE -

```
@MAIL
TO: @USERS.LST
CC:
SUBJECT:
```

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

3.4 COMMUNICATING WITH THE OPERATOR

To communicate with the operator use the PLEASE program. This program allows you to conduct a two-way conversation with the operator.

To start the PLEASE program, type PLEASE, followed by the message. If the message exceeds one line, type a hyphen at the end of the line and continue typing on the next line. When you complete your message, press RETURN. The system then notifies you that the message was sent, the time it was sent, and advises you to wait for the operator to reply.

- EXAMPLE -

```
@PLEASE OPERATOR, WHAT TIME TODAY IS THE FRONT-END RELOAD?
[MESSAGE SENT AT 11:20:30 WAITING FOR OPERATOR RESPONSE]

11:20:40 From Operator Terminal 3:
=>AT NOON
@
```

If you want to send a one-way message without a reply from the operator, use the /MESSAGE switch after you type PLEASE.

If the operator is not available, the system prints a message telling you that no operator is in attendance.

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

3.5 CONTROLLING MESSAGES FROM USERS

You can specify if you want to receive links from other users. To see if your terminal is set to RECEIVE or REFUSE LINKS, give the INFORMATION (ABOUT) TERMINAL command. The RECEIVE LINKS command is the default. If you do not want to allow other users to TALK to you, give the REFUSE LINKS command.

- EXAMPLE -

```
@INFORMATION (ABOUT) TERMINAL-MODE
 TERMINAL LA36
 TERMINAL SPEED 300
 RECEIVE LINKS
 REFUSE ADVICE
 RECEIVE SYSTEM-MESSAGES
 TERMINAL PAUSE (ON) COMMAND
 TERMINAL NO PAUSE (ON) END-OF-PAGE
 TERMINAL LENGTH 66
 TERMINAL WIDTH 132
 TERMINAL LOWERCASE
 TERMINAL RAISE
 TERMINAL NO FLAG
 TERMINAL INDICATE
 TERMINAL NO FORMFEED
 TERMINAL NO TABS
 TERMINAL NO IMMEDIATE
 TERMINAL FULLDUPLEX
@
```

3.6 CONTROLLING SYSTEM MESSAGES

You can specify if you want to receive systemwide messages on your terminal. To see if your terminal is set to RECEIVE or REFUSE SYSTEM-MESSAGES, give the INFORMATION (ABOUT) TERMINAL command. The RECEIVE SYSTEM-MESSAGES command is the default. If you elect to REFUSE SYSTEM-MESSAGES, you will no longer see such messages as:

```
[CAUTION..DISK SPACE LOW]
[SYSTEM GOING DOWN IN 1 MINUTE!!]
[DELETED FILES WILL BE EXPUNGED IN 30 SECONDS]
[SYSTEM EXPUNGE COMPLETED]
```

CAUTION

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.

CHAPTER 4

FILE SPECIFICATIONS

This chapter describes:

- Complete form of a file specification (Section 4.1)
- Using wildcards to specify files (Section 4.2)
- Specifying special characters (Section 4.3)
- Typing file specifications (Section 4.4)
- Using logical names (Section 4.5)

The TOPS-20 commands and programs mentioned in this chapter are:

ALGOL	DIRECTORY	INFORMATION	RERUN
COBOL	DUMPER	ISAM	SET
COPY	EDIT	LIBRARY	TRANSLATE
CREF	FILCOM	LINK	TYPE
DEFINE	FORTRAN	MAKLIB	

4.1 COMPLETE FORM OF A FILE SPECIFICATION

File specifications tell 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 4.5.)

<dir> is a directory name, or in special cases, a project-programmer number that specifies an area on the disk. You must always include the angle brackets around a directory name.

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.

4.1.1 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 descripion 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
Public File Structure	usually PS:
Your Connected Structure and Directory	DSK:
Your Terminal	TTY:
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 must terminate the device name. 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.

4.1.2 Directory Names - <DIR>

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

A directory name consists of up to 39 alphanumeric characters, including period, hyphen, and underline. You can use the asterisk (*) to specify a group of directories, though it is not actually part of a directory name. (Refer to Section 4.2 for more information on using an asterisk.) Directory names are always enclosed in brackets and are used only when the device is a disk. Examples of directory names are:

```
<PORADA>
<MCELMOYLE>
<D.CROWLEY>
<NEXT-RELEASE>
```

4.1.3 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 program	ISAM program
COBOL program	LIBARY program
CREF program	LINK program
FILCOM program	MAKLIB program
FORTRAN program	RERUN program

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>:

- EXAMPLE -

```
@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:

- EXAMPLE -

```
@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.

4.1.4 Filenames - name

Each file has a filename consisting of up to 39 alphanumeric characters, including hyphen, dollar sign, and underline. The character * may be used to specify a group of files with the same filename, but is not actually part of the filename. Examples of filenames are:

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

Although most programs and commands allow filenames up to 39 characters long, some software components 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; the characters - and are invalid in a filename; and the characters * and ? are used for specifying a group of filenames where permitted by the program.

4.1.5 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 hyphen and underline. The character * can be used to specify a group of files with the same file types, but is not actually part of the file type. Examples of file types are:

```
.FOR
.DAT-PROGRAM-1
.ALG
.CBL
```

Refer to Appendix B for a list of standard 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 characters - and are invalid in a file type; and the characters * and ? are used for specifying a group of file types where permitted by the program.

4.1.6 Generation Numbers - .gen

A generation number identifies the version of 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 6.6 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. If you are deleting a file, the program deletes all generations.

4.1.7 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 valid 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 6.3 for more information on temporary files.) Temporary files are deleted when you log off the system.

A file attribute never distinguishes one file from another; it defines a certain characteristic of the file. For example, if you give the command:

- EXAMPLE -

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

the system prints a list of all the files with an account named 'account'.

NOTE

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

4.2 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 asterisk (*) and the percent sign (%) are valid wildcard characters.

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

- EXAMPLE -

```
@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 (OF FILES) L*, the system lists all the filenames beginning with the letter L.

- EXAMPLE -

```
@DIRECTORY (OF FILES) L*

   PS:<SMITH>
 LAST.TXT.10
 LEVEL.DAT.1
 LIST.FOR.3
 LL.TST.2
 LOGIN.CMD.1
 LOOP.TXT.6
 LOST.DAT.4

 TOTAL OF 7 FILES
@
```

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

- EXAMPLE -

```
@DIRECTORY (OF FILES) *T

   PS:<SMITH>
 ACCTST.FOR.1
 CKACCT.FOR.1
 LAST.TXT.10
 LIST.FOR.3
 LOST.DAT.4
 NEWACT.FOR.1
 TEST.FIL.1

 TOTAL OF 7 FILES
@
```

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

- EXAMPLE -

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

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

 TOTAL OF 3 FILES
@
```

If you give the command DIRECTORY (OF FILES) L%, the system lists all two-character filenames beginning with the letter L.

- EXAMPLE -

```
@DIRECTORY (OF FILES) L%

   PS:<SMITH>
 LL.TST.1
@
```

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 * 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 * in a filename or file type.

4.3 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 it.

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

4.4 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. You always use full input when you are creating a file. If you are using a file with any of the programs listed in Table 4-2, you must always use full input; recognition is not available.

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 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).

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@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 COPY command.

- EXAMPLE -

```
@COPY (FROM) BCHECK.TST,CCHECK.TST,DCHECK.TST.1 (TO) CHECK.TST.5 !NEW GENERATION!
      BCHECK.TST.1 => CHECK.TST.5 [OK]
      CCHECK.TST.1 => CHECK.TST.6 [OK]
      DCHECK.TST.1 => CHECK.TST.7 [OK]
```

4.5 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 hyphen, dollar sign, and underline, 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 of where it is, you must look through your directory on PS:, 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.

- EXAMPLE -

```
@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:

- EXAMPLE -

```
@DIRECTORY (OF FILES) ALL:TEXT.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 (ABOUT) LOGICAL-NAMES (OF) JOB command.

- EXAMPLE -

```
@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 (ABOUT) LOGICAL-NAMES (OF) SYSTEM command.

- EXAMPLE -

```
@INFORMATION (ABOUT) LOGICAL-NAMES (OF) SYSTEM
SYS: => PS:<SUBSYS>,PS:<NEW>
```

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>.

- EXAMPLE -

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

- EXAMPLE -

```
@COPY (FILE) CHECK.TST.1 (TO) ALL:CHECK.TST.1 !NEW FILE!
```

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 the logical name only, and press RETURN. The following example shows how to remove the logical name TEST::

- EXAMPLE -

```
@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:

- EXAMPLE -

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

   PS:<TEST-SPECS>
 ACCU.DAT.4
 ARCHIVE.DOC.1
 CHECK.TXT.7
 SAMPLE.MEM.2

TOTAL OF 4 FILES
@
```

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. (Refer to Chapter 5 for information on EDIT.)

- EXAMPLE -

```
@DEFINE (LOGICAL NAME) PP: (AS) R4-PROJECT-PLAN.RNO
@EDIT PP:
Edit: R4-PROJECT-PLAN.RNO.2
*
```

4.5.1 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:

- EXAMPLE -

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

- EXAMPLE -

```
@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:.

CHAPTER 5

CREATING AND EDITING FILES

This chapter describes:

- Selecting an editor (Section 5.1)
- Using EDIT to create a file (Section 5.2)
- Entering the contents of a file (Section 5.3)
- Saving the file (Section 5.4)
- Using the EDIT command (Section 5.5)
- Recalling arguments to the CREATE and EDIT commands (Section 5.6).
- Using line numbers (Section 5.7)
- Printing lines in a file (Section 5.8)
- Changing lines in a file (Section 5.9)
- Deleting lines in a file (Section 5.10)
- Inserting lines in a file (Section 5.11)
- Moving lines in a file (Section 5.12)
- Editing files in another directory (Section 5.13)

The TOPS-20 commands and programs mentioned in this chapter are:

CREATE	EDIT
DEFINE	TV
DIRECTORY	

5.1 SELECTING AN EDITOR

The TOPS-20 Operating System allows you to create or change files by using a system editor program. There are two editors available for your use, EDIT and TV.

If you want to always use the same editor, you can define that editor as a logical name and place the command in your LOGIN.CMD file. If you do not specify an editor, the system uses the default specified by the system manager.

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 substitute characters, or retype 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.

The EDIT program description in this chapter provides enough information for you to create files and do your editing work. For a more complete description of EDIT, refer to 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. If you need to use TV on a hard-copy terminal, refer to the TV Editor Manual.

If you want to use TV, define it as the logical name EDITOR.

- EXAMPLE -

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

When you use the logical name feature to define TV as your editor, you can use the TOPS-20 CREATE and EDIT commands described in this chapter to create and edit your file with TV. When TV begins execution, it clears the video screen, then prints a pointer (/\) at the top left side of the screen, and an asterisk (*) prompt at the bottom left side of the screen. You can now give commands to TV. For a complete description of the TV program, refer to the TV Editor Manual.

5.2 USING EDIT TO CREATE A FILE

To create a new file, give the TOPS-20 command CREATE. The CREATE command allows you to create a file by using the EDIT program. On the same line, type the file specification to label the new file. You can use any file specification as long as the filename contains 39 characters or less, and the file type contains 39 characters or less. However, if the file you create must be translated using a language compiler, then the filename can contain no more than six characters, and the file type no more than three characters.

You can use any combination of letters and digits in a filename or file type, but if you are creating a program or data for a program in the file, use one of the standard file types listed in Table 5-1.

Table 5-1
Some Standard File Types

File Type	Contents of File
.ALG	A program written in the ALGOL language
.CBL	A program written in the COBOL language
.DAT	Data to be read by a FORTRAN program
.FOR	A program written in the FORTRAN language
.MAC	A program written in the MACRO language

Therefore, if you are creating a program in the FORTRAN language, use the file type .FOR in the file specification. (Refer to Appendix A for a complete list of standard file types.)

To create a file, type CREATE and press ESC. The system prints (FILE). Type the filename and the file type, and press RETURN. The following example shows how to create a new file with the filename TEST and the file type :FOR:

- EXAMPLE -

```
@CREATE (FILE) TEST.FOR
Input: TEST.FOR.1
00100
```

After you type the filename and file type and press RETURN, the system creates the file and prints INPUT: followed by the filename, file type and a generation number. You do not have to type a generation number when you create a new file, because the system automatically assigns a generation number of 1. EDIT prints the number 00100 on the next line. You can now start to enter the contents of the file. For easy reference, EDIT numbers each line in the file as you enter its contents.

If you create a new file, and you already have a file with the same filename and file type, the system prints a generation number greater than 1. In order to keep this existing file, you must end the creation of the new file. If you do not, the existing file will be deleted when you exit from EDIT.

To end the creation of the new file, press ESC (even if you have not typed anything); EDIT prints the EDIT prompt,*. Type EQ and press RETURN; the system prints @. The EQ command instructs EDIT to End the session and Quit, and the file you started to create is not saved.

- EXAMPLE -

```
@CREATE (FILE) TEST.FOR
Input: TEST.FOR.2
00100 $
*EQ
@
```

Choose a different filename and give the CREATE command again.

- EXAMPLE -

```
@CREATE (FILE) NUMBER.FOR
Input: NUMBER.FOR.1
00100
```

You can now enter the program or data into the file.

5.3 ENTERING THE CONTENTS OF A FILE

To enter the contents of a file, wait for the EDIT program to print a line number; then type a line and press RETURN. EDIT prints the next line number; type the next line and press RETURN. Continue entering the contents of the file in this manner until you complete your work.

If you make typing mistakes, use DELETE to delete a single character; CTRL/U to delete the entire line; CTRL/W to delete the previous word; and CTRL/R to reprint the line. (Refer to Section 2.7 for more information on correcting input errors.)

When you finish entering lines, press ESC; the system prints the EDIT prompt, *. You can now do one of the following:

- End the EDIT program and save the file by typing the EDIT command E and pressing RETURN.
- Continue to work in the file by giving EDIT commands.

The following example shows how to create a file and enter a short FORTRAN program in the file. To do the following example, use tabs to space the text:

- EXAMPLE -

```
@CREATE (FILE) NUMBER.FOR
Input: NUMBER.FOR.1
00100            TYPE 101
00200    101     FORMAT (' TYPE A NUMBER.')
00300            ACCEPT 102,X
00400    102     FORMAT (F)
00500            TYPE 103,X
00600    103     FORMAT (' YOU TYPED ',F)
00700            END
00800    $
*
```

Type CREATE and the new file specification NUMBER.FOR; press RETURN. The system prints INPUT: followed by the new filename and file type, and assigns a generation number of 1. EDIT then prints the line number 00100. To type the first line, press TAB and type the word TYPE, type a space, type the number 101 and press RETURN. EDIT prints the line number 00200. Type the number 101, press TAB, type the word FORMAT followed by a space, type (' TYPE A NUMBER. '), and press RETURN.

Continue entering the FORTRAN program in this manner until the file is complete.

5.4 SAVING THE FILE

You can save the file by using one of the following EDIT commands.

1. E - saves the file including the line numbers, ends the EDIT program, and returns to TOPS-20 command level

2. EU - saves the file but removes the line numbers, ends the EDIT program, and returns to TOPS-20 command level. (EDIT creates new line numbers if you edit the file later.)

3. B - saves the file and continues at EDIT command level.

5.4.1 Using the E Command

To save the file and end the EDIT program:

1. Press ESC. This echoes as a $sign and tells the EDIT program to stop inserting lines. The system prints the EDIT prompt, * on the next line.

2. Type E. This notifies EDIT to end the session and save the file. Press RETURN. The system types a blank line, prints the file specification in brackets, and prints @ on the next line.

The following example shows how to end EDIT and save the new file NUMBER.FOR:

- EXAMPLE -

```
@CREATE (FILE) NUMBER.FOR
Input: NUMBER.FOR.1
00100           TYPE 101
00200   101     FORMAT (' TYPE A NUMBER.')
00300           ACCEPT 102,X
00400   102     FORMAT (F)
00500           TYPE 103,X
00600   103     FORMAT (' YOU TYPED ',F)
00700           END
00800   $
*E

[NUMBER.FOR.1]
@
```

If you want to save the file without line numbers and end the EDIT program, type EU and press RETURN. The system inserts a blank line, prints the file specification in brackets, and returns you to TOPS-20 command level.

You can save the file without ending EDIT by using the B command. Type B and press RETURN. The system prints the name of the file and leaves you at EDIT command level. The B command creates a backup file with a file type beginning with the letter Q.

Every time you change an existing file and give an E or B command to save the file, EDIT creates a backup file. The backup file is a copy of the file before you enter the changes. EDIT creates the file type

for backup files by replacing the first letter of the file type of the edited file with the letter Q. The backup copy of the file NUMBER.FOR would be NUMBER.QOR.

The following example illustrates how EDIT creates backup files when you edit a file:

- EXAMPLE -

```
@CREATE (FILE) TEST.FIL
Input: TEST.FIL.1
00100  ONE HUNDRED
00200  $
*E

[TEST.FIL.1]

@
```

After you create a file, only one copy of the file TEST.FIL.1 exists. EDIT does not create backup files when you use the CREATE command. Give the DIRECTORY command to see that only one file exists.

- EXAMPLE -

```
@DIRECTORY (OF FILES) TEST

   PS:<SARTINI>
 TEST.FIL.1
@
```

Now, edit the file by inserting a line and saving the file.

- EXAMPLE -

```
@EDIT (FILE) TEST.FIL
Edit: TEST.FIL.1
*I200
00200  TWO HUNDRED
00300  $
*E

[TEST.FIL.2]
@
```

The original file TEST.FIL.1 becomes the backup file TEST.QIL.1 and your current file is TEST.FIL.2.

- EXAMPLE -

```
@DIRECTORY (OF FILES) TEST

   PS:<SARTINI>
 TEST.FIL.2
   .QIL.1

 Total of 2 files
@
```

Edit the file again by adding a third line and save the file.

- EXAMPLE -

```
@EDIT (FILE) TEST.FIL
Edit: TEST.FIL.2
*I300
00300  THREE HUNDRED
00400  $
*E

[TEST.FIL.3]
@
```

The file TEST.FIL.2 becomes TEST.QIL.2 and the current file is TEST.FIL.3.

```
@DIRECTORY (OF FILES) TEST

   PS:<SARTINI>
 TEST.FIL.3
   .QIL.2

 Total of 2 files
@
```

Because the system default is set to keep only one generation of the file and one generation of the backup file, the system deleted the file TEST.QIL.1 from the directory.

5.5 USING THE EDIT COMMAND

To change an existing file, give the TOPS-20 command EDIT. The EDIT command allows you to edit a file using the EDIT program. On the same line, type the file specification of the file you want to change. The system prints EDIT: followed by the file specification and generation number. EDIT prints the EDIT prompt, *, on the next line indicating that you are at EDIT command level. You can now enter EDIT commands. (Refer to the EDIT Reference Manual for a complete description of the EDIT commands.)

The following example shows how to edit a file:

- EXAMPLE -

```
@EDIT (FILE) NUMBER.FOR
Edit: NUMBER.FOR
*
```

If the file whose name you type does not exist, the system prints a message and creates a file using the file specification you typed.

- EXAMPLE -

```
@EDIT (FILE) TEST.FOR

%File not found, Creating New file
Input: TEST.FOR
00100
```

Usually the changes you make are incorporated in the current copy of the file. However, if you want to leave the current copy untouched and save the changes in another file, specify an output file specification after the guidewords (OUTPUT AS). Give the EDIT command, followed by the filename NUMBER.FOR and press ESC. The system prints the generation number and the guidewords (OUTPUT AS); then type the output filename TEST.FOR.

- EXAMPLE -

```
@EDIT (FILE) NUMBER.FOR.1 (OUTPUT AS) TEST.FOR
Edit: NUMBER.FOR.1
*I450
00450 REPEAT STEP 4
00550 $
*E

[TEST.FOR.1]
@
```

5.5.1 Using Switches with EDIT

You can use switches with the EDIT command to specify options to the command, or to override system default options.

For example, if you want to automatically save your file after you insert a specified number of new lines, you can give the following command with the file TEST.FOR:

- EXAMPLE -

```
@EDIT (FILE)/ISAVE:10 TEST.FOR
```

In the above example, you specified the number 10 after the /ISAVE switch. This means that after every 10 lines you insert, EDIT saves the lines and prints the following message

```
[Doing auto-save, please wait]
```

followed by the filename. You can then continue to insert lines, and EDIT will continue to save these lines after every 10 insertions.

If you want to always use specific switches when you edit a file, you can create a SWITCH.INI file and include the switch(es). Every time you give the EDIT command, EDIT reads the SWITCH.INI file and uses the switches specified in that file.

The following example shows how to create a SWITCH.INI file and include an /ISAVE switch with EDIT:

- EXAMPLE -

```
@CREATE (FILE) SWITCH.INI
Input: SWITCH.INI.1
00100 EDIT/ISAVE:10
00200 $
*E

[SWITCH.INI.1]
@
```

Now, instead of typing the command

- EXAMPLE -

```
@EDIT (FILE) /ISAVE:10 TEST.FOR
```

you can type the following command, and the /ISAVE switch will automatically be included in the command:

- EXAMPLE -

```
@EDIT (FILE) TEST.FOR
```

If the switches occupy more than one line, use a hyphen at the end of the first line and continue on the next line. (Refer to the EDIT Reference Manual for a complete description of EDIT command options.)

NOTE

Create your SWITCH.INI file in your logged-in directory. Refer to Section 6.1 for a description of logged-in directories.

5.6 RECALLING ARGUMENTS TO CREATE AND EDIT COMMANDS

The system remembers arguments typed to a successful CREATE or EDIT command. When you give a subsequent EDIT or CREATE command without giving any arguments, the system uses the arguments from your last CREATE or EDIT command. Therefore, you can create a new file, exit from the editor, do some other work on the system, then return to edit that file without retyping the arguments.

The following example shows how to create a file and then edit that file. The EDIT command uses the arguments typed in the CREATE command.

- EXAMPLE-

```
@CREATE (FILE) FILE.TXT
Input: FILE.TXT.1
00100   THIS FILE CONTAINS
00200   TWO LINES.
00300   $
*E

[FILE.TXT.1]
*I300
00300   NOW, THERE ARE THREE.
00400   $
*E

[FILE.TXT.2]
@
```

When you want to edit this same file later (during the same terminal session), do not include a generation number in the file specification. If you include a generation number, the system uses that exact generation and not the highest generation for that file. For example, if you give the command CREATE (FILE) TEST.FOR.3 and save the file by giving an E or EU command, your most recent file is TEST.FOR.4. If you give the EDIT command without any arguments, the system tries to edit the file TEST.FOR.3 and not the most recent file TEST.FOR.4. If TEST.FOR.3 exists, the system uses that file; if it does not exist (as in the following example), the system allows you to create it.

- EXAMPLE -

```
@EDIT (FILE) TEST.FOR.3
Edit: TEST.FOR.3
*I500
00500   THIS IS A TEST FILE.
00600   $
*E

[TEST.FOR.4]
@EDIT

%File not found, Creating New file
Input: TEST.FOR.3
00100   $
*EQ

@
```

To correct this situation give the EDIT or CREATE command with a file specification, but do not include a generation number.

- EXAMPLE -

```
@EDIT (FILE) TEST.FOR
Edit:  TEST.FOR.3
*
```

5.7 USING LINE NUMBERS

You can locate and reference lines in your file by using line numbers.

When you save the file with an E command, EDIT saves the line numbers along with the file. When you give the EU command, EDIT removes the line numbers before saving the file.

Line numbers consist of five digits, followed by a tab. The EDIT line number default is 00100, through 99900, with increments of 100 for each line. This default allows you to insert lines between existing lines in the file. When specifying line numbers you do not need to type the leading zeroes. For example, you can type 500 instead of 00500.

You can specify a group or range of lines in a file by typing the line number of the first line in the group or range, followed by a colon and the number of the last line in the group. For example, to specify lines 600, 700, 800, and 900, you can type 600:900.

5.8 PRINTING LINES IN A FILE

You can print one line, a group of lines, or an entire file. Table 5-2 lists the various forms of the print command you can use to print lines.

Table 5-2
EDIT Print Commands

*P	Prints the current line and the next 15 lines.
*P500	Prints line 500.
*P500:700	Prints lines 500 through 700.
*P^	Prints the first line in a file.
P	Prints the last line in a file.
*P.	Prints the current line.
P^:	Prints the entire file.
*<ESC>	Prints the previous line (using ESC).
*<LF>	Prints the next line (using LF).

To print lines in a file, give the P command. When you type the P command without any arguments, the system prints the lines from your present location through the next 15 lines (if they exist), and prints the EDIT prompt, *. Type P again; the system reprints the last line and prints the next 15 lines in the file. In the following example the file contains only 7 lines so the P command prints the entire file.

- EXAMPLE -

```
*P
00100          TYPE 101
00200  101     FORMAT ('TYPE A NUMBER.')
00300          ACCEPT 102,X
00400  102     FORMAT (F)
00500          TYPE 103,X
00600  103     FORMAT ('YOU TYPED',F)
00700          END
*
```

To print one line, type P, followed by the number of the line.

- EXAMPLE -

```
*P500
00500          TYPE 103,X
*
```

To print the entire file, type P, an up-arrow, a colon, and an asterisk.

- EXAMPLE -

```
*P^:*
00100           TYPE 101
00200   101     FORMAT ('TYPE A NUMBER.')
00300           ACCEPT 102,X
00400   102     FORMAT (F)
00500           TYPE 103,X
00600   103     FORMAT ('YOU TYPED',F)
00700           END
*
```

5.9 CHANGING LINES IN A FILE

You can change lines in a file by either completely replacing them, or by substituting words or characters within the line.

To delete a line and replace it with another line, give the R command, followed by the number of the line you want to replace. EDIT prints the line number and you type the new line. On the next line, EDIT prints a message confirming the line(s) you deleted.

The following example shows how to print line 200 and then replace it with a similar line containing the word PLEASE and a comma:

- EXAMPLE -

```
*P200
00200   101     FORMAT ('TYPE A NUMBER.')
*R200
00200   101     FORMAT ('PLEASE, TYPE A NUMBER.')
1 LINES (00200/1) DELETED
*
```

To substitute a word or characters in a line, give the S command, followed by the word or characters you want to remove; type ESC; type the word or characters you want to insert; type ESC; and type the line number where you want to make the substitution.

The following example shows how to substitute an exclamation mark for the comma after PLEASE in line 200.

- EXAMPLE -

```
*S,$!$200
00200   101     FORMAT ('PLEASE! TYPE A NUMBER.')
*
```

NOTE

If you give an S command and press RETURN, but forget to press an ESC in the command line, the system prints S* on the next line. Type CTRL/U. After EDIT prints ABORTED...., give the command again. (Refer to the EDIT Reference Manual for a description of the S* feature.)

If you do not know the line number containing the word you want to change, give the F command, followed by the word you want to change and press ESC. EDIT searches the file until it finds the word and prints the line containing the word.

To do the following example, give the F command to find PLEASE, then give the S command to substitute an exclamation mark for the comma after PLEASE:

- EXAMPLE -

```
*FPLEASE,$
00200  101     FORMAT ('PLEASE, TYPE A NUMBER.')
*SPLEASE,$PLEASE!$200
00200  101     FORMAT ('PLEASE! TYPE A NUMBER.')
*
```

5.10 DELETING LINES IN A FILE

To delete a line or lines, type D, followed by the number or range of numbers of the lines you want to delete. EDIT prints a message confirming the lines you deleted.

- EXAMPLE -

```
*D100
1 LINES (00100/1) DELETED
*
```

NOTE

If you delete some lines by mistake, you can recover these lines by ending EDIT using the EQ command. However, by using the EQ command you also lose any changes made to the file since you gave the last CREATE or EDIT command.

5.11 INSERTING LINES IN A FILE

You can insert lines anywhere in your file: in the beginning, between existing lines, or at the end of the file.

To insert a new line, type I, followed by the line number you want to insert. For example to insert a line between the existing lines of 500 and 600, type I550. EDIT inserts the new line between the two existing lines.

- EXAMPLE -

```
*I550
00550  NEW LINE
*
```

To insert a series of new lines between existing lines, type I, followed by the number of the line preceding the location where you

want to insert the new lines, an exclamation mark (!), and the total number of lines in the series you are inserting.

To do the following example, give the I command and insert three new lines after line 600 but before line 700.

- EXAMPLE -

```
*I600!3
00620          Y=2*X
00640          TYPE 104,X,Y
00660  104     FORMAT ('TWICE ',F,' IS',F)
00680  $
*
```

When you insert a series of new lines between existing lines, EDIT always selects an appropriate line number increment to accommodate the number of lines you want to insert.

To insert a line at the beginning of a file, type I, followed by a line number that is smaller than the number of the first line in the file. To insert a line number at the end of a file, type I, followed by a line number that is higher than the number of the last line in the file. You can also type I*. In this case, after you press RETURN, EDIT continues the line numbers as if you were creating a new file.

5.12 MOVING LINES IN A FILE

You can move lines in a file by copying them or by transferring them. When you copy lines, EDIT moves copies of the lines in one location in the file to another. When you transfer lines, EDIT copies them to the destination and deletes them from the original location in the file.

To copy lines from one location in a file to another, type C, followed by the line number where you want to send the copied lines; type a comma; type the line number where you want the copying to begin; type a colon; and type the line number where you want the copying to end.

The example below shows how to create a data file containing two repetitive lines, and copy the lines into the file instead of retyping them. To copy lines 100 and 200 after line 300, type C350,100:200. Line number 350 is the destination of the copied lines 100 and 200. EDIT prints INC1=00020, indicating the increment it uses to accommodate the copied lines. Print the file to see the copied lines.

- EXAMPLE -

```
@CREATE (FILE) DATA.TST
Input: DATA.TST.1
00100  12,14,16,17,18,2,4,5,
00200  2,6,5,4
00300  3,6,3,
00400  7,1
00500  11,9,6,14,11,8,2,
00600  4,6,1
*C350,100:200
INC1=00020
```

```
*P^:*
00100  12,14,16,17,18,2,4,5,
00200  2,6,5,4,
00300  3,6,3,
00350  12,14,16,17,18,2,4,5,
00370  2,6,5,4,
00400  7,1
00500  11,9,6,14,11,8,2,
00600  4,6,1
*E

[DATA.TST.1]
@
```

To transfer lines from one location in a file to another, type T, followed by the line number where you want to send the transferred lines; type a comma; type the line number where you want to begin transferring; type a colon; and type the line number where you want to end transferring.

The following example shows how to transfer lines 500 and 600 to the beginning of the file, and print the file to see the transferred lines:

- EXAMPLE -

```
@EDIT (FILE) DATA.TST
Edit: DATA.TST.3
*P
00100  12,14,16,17,18,2,4,5,
00200  2,6,5,4,
00250  11,9,6,14,11,8,2,
00300  3,6,3,
00350  12,14,16,17,18,2,4,5,
00370  2,6,5,4,
00400  7,1
00500  11,9,6,14,11,8,2,
00600  4,6,1
00700  12,14,16,17,18,2,4,5,
*T10,500:600
INC1=00020
*P^:*
00010  11,9,6,14,11,8,2,
00030  4,6,1
00100  12,14,16,17,18,2,4,5,
00200  2,6,5,4,
00250  11,9,6,14,11,8,2,
00300  3,6,3,
00350  12,14,16,17,18,2,4,5,
00370  2,6,5,4,
00400  7,1
00700  12,14,16,17,18,2,4,5,
*
```

If EDIT prints INC1=ORDER after you give the T or C command, you know there were not enough lines in the file between the line you specified and the next line in the file to receive the transferred lines. The EDIT program completes the transfer in this case, but the line numbers are not in order. To renumber these lines give the N command. (Refer to the EDIT Reference Manual for a description of the N command.)

5.13 EDITING FILES IN ANOTHER DIRECTORY

If you edit a file in a directory you are not connected to, the edited output file usually appears in your connected directory. To put the edited file back in its original directory, you must specify the directory name (and structure name if the directory is on a different structure) after the guidewords (OUTPUT AS).

For example, if you are connected to directory <PORADA> and edit the file TEST.FIL.3 in directory <SARTINI> without specifying the directory where the file should be saved, the system places the edited file in the directory <PORADA> and names it TEST.FIL.1. (No file exists in directory <PORADA> with the filename TEST.FIL, so the system assigns the generation number of 1.)

- EXAMPLE -

```
@EDIT (FILE) <SARTINI>TEST.FIL.3 (OUTPUT AS)
Edit: <SARTINI>TEST.FIL.3
*I400
00400  FOUR HUNDRED
00500  $
*E

[TEST.FIL.1]
@
```

Now, specify the directory <SARTINI> after the guidewords (OUTPUT AS), and the system saves the file in directory <SARTINI> as TEST.FIL.4.

- EXAMPLE -

```
@EDIT (FILE)<SARTINI>TEST.FIL.3 (OUTPUT AS) <SARTINI>
Edit: <SARTINI>TEST.FIL.3
*I400
00400  FOUR HUNDRED
00500  $
*E

<SARTINI>[TEST.FIL.4]
@
```

NOTE

In order to create or edit files in other directories, you must have access rights to the directories, as well as the right to create files. Refer to Section 6.2 for information on directory and file protection.

CHAPTER 6

USING DISK FILES

This chapter describes:

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

The TOPS-20 commands mentioned in this chapter are:

ACCESS	DELETE	LOGIN	SET
APPEND	DIRECTORY	MODIFY	TYPE
ARCHIVE	DISMOUNT	MOUNT	UNDELETE
CANCEL	EDIT	PRINT	VDIRECTORY
CONNECT	EXPUNGE	RENAME	
COPY	INFORMATION	RETRIEVE	

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 though a file structure contains 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, called the public structure, always remains on line during system operation. This public structure, usually named PS:, contains a directory for every user of the system, and the necessary accounting information to allow the users to log in. (The examples in this manual refer to the public structure as

PS:.) When you log in, you are connected to your directory on the public structure. This directory is referred to as your logged-in directory and, in addition to the accounting information, contains some or all of your files.

You can have and/or 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 remains on line during system operation, other structures must 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.")

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@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 6.4 and 6.5 for more information on connecting to directories and accessing files).

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

- EXAMPLE -

```
@INFORMATION (ABOUT) STRUCTURE (NAME) *
Status of structure PS:
Mount count:  4, open file count:  227, units in structure:  2
Public Domestic
Users who have MOUNTed PS:  SUSSMAN, TOMCZAK, LNEFF, DNEFF
Users ACCESSing PS:  OPERATOR, R.ACE, SAMBERG, COMBS, SYLOR, KONEN,
COHEN, ZIMA, JENNESS, BLOUNT, SUSSMAN, REILLY, CIRINO, BERKOWITZ,
GUNN, HUTCHINS, LEACHE, SKOGLUND, HOFFMAN, OSMAN, ADLEY, FRIES,
ELFSTROM, HURLEY, WEISS, HOVSEPIAN, EIBEN.INFO, MILLER, BROWN, WEBBER,
JABLONSKY, ENGEL, BENCE, HOLLAND, DRUEKE, HALL, DBELL, BELANGER,
TOMCZAK, LEFEBVRE, GRANT, SMILLER, LNEFF, MCELMOYLE, HARAMUNDANIS,
SCOHEN, MIERSWA, LEAPLINE, SARTINI, DNEFF, SCHMITT
Users CONNECTed to PS:  OPERATOR, R.ACE, SAMBERG, SYLOR, KONEN,
CIRINO, BERKOWITZ, OSMAN, ELFSTROM, HURLEY, WEISS, BROWN, JABLONSKY,
ENGEL, DRUEKE, SARTINI
```

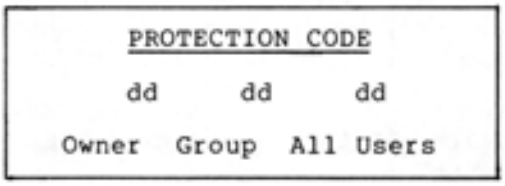
```
Status of structure LANG:
Mount count:  11, open file count:  5, units in structure:  1
Domestic
Users who have MOUNTed LANG:  OPERATOR, SYLOR,  COHEN,  REILLY,  GUNN,
FRIES, HOVSEPIAN, JABLONSKY, SMILLER, SCOHEN, DNEFF
Users ACCESSing LANG:  SYLOR, REILLY, GUNN, FRIES, SCOHEN
USERS CONNECTed to LANG:  COHEN, GUNN, FRIES, SCOHEN

Status of structure SNARK:
Mount count:  26, open file count:  0, units in structure:  1
Domestic
Users who have MOUNTed SNARK:  OPERATOR, R.ACE, KONEN, ZIMA,  JENNESS,
BLOUNT,  SUSSMAN,  REILLY,  BERKOWITZ,  GUNN,  LEACHE,  OSMAN,  ADLEY,
HURLEY, HOVSEPIAN,  MILLER,  JABLONSKY,  ENGEL,  BENCE,  HALL,  DBELL,
TOMCZAK, GRANT, DNEFF, SCHMITT
Users ACCESSing SNARK:  R.ACE, REILLY, OSMAN, HURLEY,  MILLER,  ENGEL,
HALL
Users CONNECTed to SNARK:  BLOUNT, REILLY, MILLER, GRANT

Status of structure MISC:
Mount count:  37, open file count:  8, units in structure:  1
Domestic
Users who have MOUNTed MISC:  OPERATOR, COMBS, ZIMA, JENNESS, SUSSMAN,
CIRINO,  BERKOWITZ,  GUNN, HUTCHINS, LEACHE, SKOGLUND, HOFFMAN, OSMAN,
ADLEY,  FRIES,  ELFSTROM,  HOVSEPIAN,  EIBEN.INFO,  BROWN,  WEBBER,
JABLONSKY, BENCE, HOLLAND, DRUEKE, DBELL, BELANGER, TOMCZAK, LEFEBVRE,
SMILLER, LNEFF, MCELMOYLE,  HARAMUNDANIS,  MIERSWA,  LEAPLINE,  DNEFF,
SCHMITT
Users  ACCESSing  MISC:   COMBS,  ZIMA,  JENNESS,  HUTCHINS,  HOFFMAN,
EIBEN.INFO,  HOLLAND,  DBELL,  TOMCZAK,  LEFEBVRE,  SMILLER,  LNEFF,
LEAPLINE, DNEFF, SCHMITT
Users CONNECTed to MISC:  COMBS,  ZIMA,  JENNESS,  SUSSMAN,  HUTCHINS,
LEACHE,  SKOGLUND, HOFFMAN, ADLEY, EIBEN.INFO, WEBBER, BENCE, HOLLAND,
DBELL,  BELANGER,  TOMCZAK,  LEFEBVRE,  SMILLER,  LNEFF,  MCELMOYLE,
HARAMUNDANIS, MIERSWA, LEAPLINE, DNEFF, SCHMITT

Status of structure REL4:
Mount count:  1, open file count:  0, units in structure:  1
Foreign
Users who have MOUNTed REL4:  HOVSEPIAN
No users are ACCESSing REL4:
Users CONNECTed to REL4:  HOVSEPIAN

Status of structure MEG6:
Mount count:  1, open file count:  0, units in structure:  1
Foreign
Users who have MOUNTed MEG6:  DRUEKE
No users are ACCESSing MEG6:
No users CONNECTed to MEG6:

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 rest of the users. Usually files are protected to prevent access from non-owners who are not group members (rest of users). 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 that directory. (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.



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	Connect to the directory without giving a password, undelete files, expunge the entire directory, change 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.

The system default directory protection number is 777700. If you want to either change your directory protection number, or find out what your directory protection number is, contact your system manager.

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 and delete the file.
10	Execute the file
04	Append to the file.
02	List the file specification using the DIRECTORY command.
00	List the file specification using the DIRECTORY command only if the file is specified explicitly and completely.

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

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.

- EXAMPLE -

```
@TYPE (FILE) <HOLLAND>TEST.TXT
?DIRECTORY ACCESS PRIVILEGES REQUIRED
@
```

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.

- EXAMPLE -

```
@TYPE (FILE) <HOLLAND>TEST.TXT
?READ PROTECT VIOLATION FOR FILE (HOLLAND)TEST.TXT
```

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

6.2.4 Printing a File Protection Number

To print the file protection number, use the VDIRECTORY command (or the DIRECTORY command with the PROTECTION subcommand).

- EXAMPLE -

```
@VDIRECTORY (OF FILES) TEST.FIL.*
   PS:<PORADA>
TEST.FIL.1;  P777700    1 110(7)    21-Aug-78 11:44:25
@
```

6.2.5 Changing a File Protection Number

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

- EXAMPLE -

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

6.3 USING TEMPORARY FILES

Use a temporary file when you do not need to keep a file permanently, such as a scratch file or a listing file. A temporary file has the attribute ;T appended to its file specification. The ;T instructs the system to delete and expunge the file when you log off the system.

To create a temporary file, add the temporary file descriptor (;T) at the end of the file specification.

- EXAMPLE -

```
@COPY (FROM) TTY:  (TO) T.FIL;T
THIS IS ONE-LINE FILE.
   ^Z
@DIRECTORY (OF FILES) T.FIL

   PS:<BOSACK>
T.FIL.100020;T
@
```

Note that the system assigns a generation number of 100000 plus the job number to the temporary file. Because this total is unique for every job, different jobs connected to the same directory can have temporary files with the same name and different contents.

The system deletes and expunges all your temporary files when you log out. The system recognizes the temporary files because they have a generation number of 100000 plus your job number.

6.4 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:

- EXAMPLE -

```
@LOGIN (USER) MCELMOYLE (PASSWORD) (ACCOUNT) 341
 Job 25 on TTY26 31-Aug-79 14:40
@
```

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.

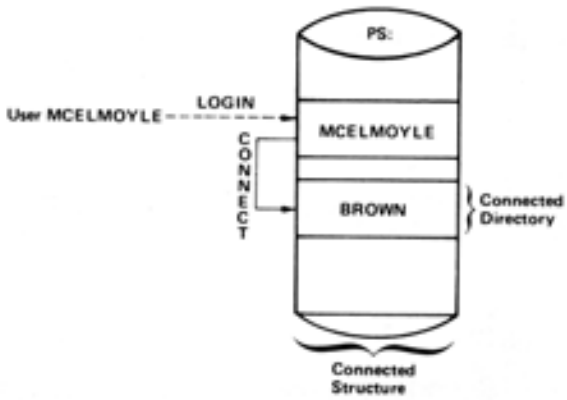
In addition, you have owner rights for that directory, just as if you logged in to it. The owner rights for a directory are valid as long as you are connected to that directory; the rights terminate when you connect to another directory. You always retain the owner rights to the files in your logged-in 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. The system prints PASSWORD: on the following line, and you type the password 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 PS:. When you omit a directory name and/or structure name in a file specification, the system assumes your logged-in directory <MCELMOYLE> on the structure PS:. After you log in, connect to the directory <BROWN> on PS:. Now, if you omit the directory name and/or structure name in a file specification, the system assumes your connected directory <BROWN> on PS:.

- EXAMPLE -

```
@LOGIN (USER) MCELMOYLE (PASSWORD) (ACCOUNT) 341
 Job 25 on TTY26 31-Aug-79 14:56
@CONNECT (TO DIRECTORY) <BROWN>
PASSWORD:
@
```



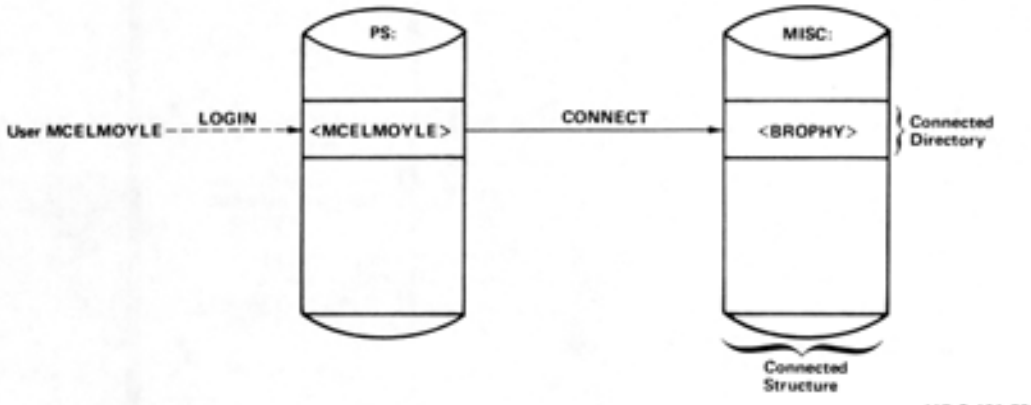
When you give the CONNECT command for a directory that is located on a different structure, your default structure also changes. The system assumes both the connected structure and the connected directory when you omit them in a file specification.

The example below illustrates the effects of logging in on PS: and then connecting to a directory on another structure named MISC:. When

you (user MCELMOYLE) log in, you are connected to your directory on PS:. After you log in, connect to the directory <BROPHY> on the structure MISC:.

- EXAMPLE -

```
@LOGIN (USER) MCELMOYLE (PASSWORD) _(ACCOUNT) 341
 Job 28 on TTY26 31-Aug-79 12:02
@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 (ABOUT) JOB-STATUS command.

- EXAMPLE -

```
@INFORMATION (ABOUT) JOB-STATUS
 Job 28, User MCELMOYLE, MISC:<BROPHY>, Account 341, TTY26
@
```

6.5 ACCESSING DIRECTORIES

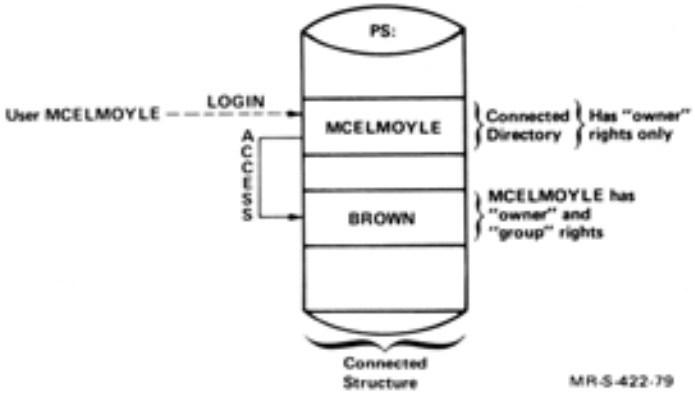
To access another directory and remain connected to your present directory, give the ACCESS command. To complete the ACCESS command, you must provide the password for the directory you want to access.

When you access a directory, you are actually working in your connected directory but you also have owner and group rights to another directory. This means that you can use the files in the directory you have given the ACCESS command for by specifying that directory in the file specification. Therefore, unless you specify otherwise, the latest version of any file you update in either directory appears in your connected directory. If you want the latest version of the file to appear in the directory you accessed, you must specify the directory name in the file specification. If the directory you access is located on an on-line structure other than your connected directory, you must specify the structure name in any file specification.

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

- EXAMPLE -

```
@LOGIN (USER) MCELMOYLE (PASSWORD) (ACCOUNT) 341
 Job 32 on TTY26 31-Aug-79 10:08
@ACCESS (TO DIRECTORY) <BROWN>
PASSWORD:
@
```

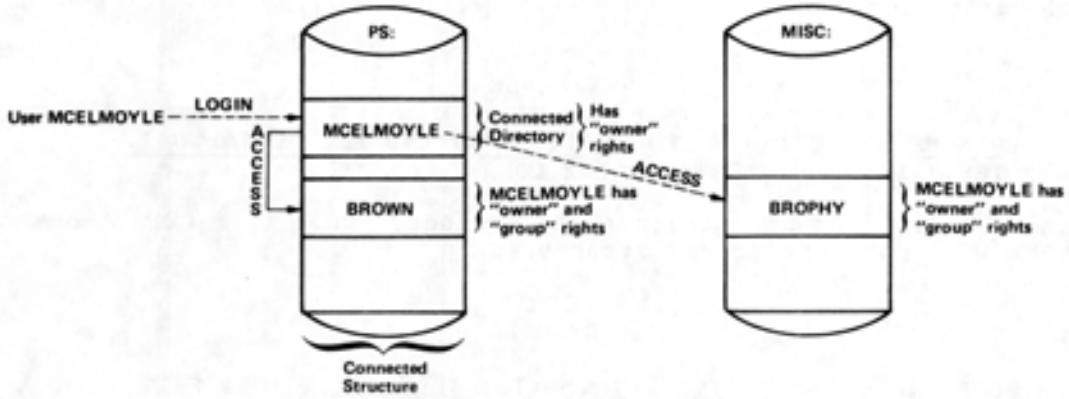


You can give the ACCESS command to more than one directory during a job session. You can access a directory on one structure and then access a directory on a different structure. If each directory you access is located on a different structure, the owner and group rights for these directories remain in effect throughout your entire job session (from LOGIN to LOGOUT) or until a structure is dismounted. If you access two or more directories on the same structure, only the rights for the last directory you accessed remain in effect (until you log out or the structure is dismounted). You always retain your owner rights to your logged-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 logged-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:

- EXAMPLE -

```
@LOGIN (USER) MCELMOYLE (PASSWORD) (ACCOUNT) 341
 Job 32 TTY26 31-Aug-79 10:08
@ACCESS (TO DIRECTORY) <BROWN>
PASSWORD:
@ACCESS (TO DIRECTORY) MISC:<BROPHY>
PASSWORD:
```



6.6 COPYING FILES

You can copy a file to a new file and keep the original file, or you can rename the file to a new file and lose the original file.

To copy one or more files 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. To do the following example, copy the existing file TEST1.DAT to the destination file 2TEST.DAT.

- EXAMPLE -

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

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

- EXAMPLE -

```
@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 4.1.5 for more information on symbolic generation numbers.)

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.

- EXAMPLE -

```
@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.

- EXAMPLE -

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

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:

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@APPEND (SOURCE FILE) *.FOR.0 (TO) ATEST.FOR.1 !New file!
 ACCOUN.FOR.2 [OK]
 ACCTST.FOR.1 [OK]
 CHECK.FOR.4 [OK]
 CKACCT.FOR.1 [OK]
 NEWACCT.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.

- EXAMPLE -

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

NOTE

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

6.8 PRINTING FILES

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

- EXAMPLE -

```
@PRINT (FILES) UPDATE.CBL
[Job UPDATE Queued, Request-ID 57, Limit 27]
@
```

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

- EXAMPLE -

```
@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.

- EXAMPLE -

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

See Command Ref Man

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.

- EXAMPLE -

```
@PRINT (FILES) LARGE.DAT, MYTEST.DAT, TEST1.DAT /AFTER:15-OCT-79
[Job LARGE Queued, Request-ID 58, Limit 27]
@INFORMATION (ABOUT) OUTPUT-REQUESTS

Printer Queue:
Job Name  Req#  Limit            User
--------  ----  -----  ------------------------
* BOX       53    270  LYONS                      On Unit:0
   Started at 14:29:29, printed 165 of 270 pages
  MIDAS      34     27  REILLY                     /Forms:NARROW
  LARGE      58     27  SARTINI                    /After:15-Oct-79 00:00
There are 3 Jobs in the Queue (1 in Progress)
@
```

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.

- EXAMPLE -

```
@PRINT (FILES) LARGE.DAT/COPIES:6, MYTEST.DAT, TEST1.DAT
[Job LARGE Queued, Request-ID 59, Limit 27
@
```

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.

- EXAMPLE -

```
@PRINT (FILES)/COPIES:6,LARGE.DAT,MYTEST.DAT,TEST1.DAT
 [Job LARGE Queued, Request-ID 60, Limit 30
@
```

6.8.1 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 slash (/) and a question mark (?).

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

- EXAMPLE -

```
@MODIFY (REQUEST TYPE) PRINT (ID) LARGE /AFTER:25-OCT-79
[1 Job Modified]
@
```

After you give the command, the system prints a message informing you that the job was modified. In this case (using the /AFTER: switch) you can give the INFORMATION (ABOUT) OUTPUT-REQUESTS to see that the switch was added to your job.

6.8.2 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.

- EXAMPLE -

```
@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. Give the CANCEL command followed by PRINT and an *.

The following example shows how to cancel all your PRINT requests:

- EXAMPLE -

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

6.8.3 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.

- EXAMPLE -

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

To avoid having to type the SET DEFAULT PRINT command 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 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 or LOGIN.CMD:

- EXAMPLE -

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

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

To see which defaults you set for the PRINT command, give the INFORMATION (ABOUT) DEFAULTS (FOR) PRINT command.

- EXAMPLE -

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

6.9 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 when you do one of the following:

- Give the EXPUNGE command.
- You (or another user connected to your directory) log off the system.
- The operator gives the EXPUNGE command.

The EXPUNGE command erases all files marked for deletion since you logged in to the system or gave the last EXPUNGE command. Deleting and erasing files are separate operations. Therefore, once you delete a file, it does not immediately disappear, and if you change your mind, you can restore the file or files using the UNDELETE command.

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

- EXAMPLE -

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

To see that TEST.FIL has been deleted from your directory, give the following command.

- EXAMPLE -

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

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

- EXAMPLE -

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

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

To restore TEST.FIL, give the UNDELETE command.

- EXAMPLE-

```
@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.

- EXAMPLE -

```
@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.

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.

6.10 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 (ABOUT) DISK-USAGE command.

- EXAMPLE -

```
@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 a message ?DISK 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, 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 PAGES
```

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. The operator runs this program as often as required to bring users' areas 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 logged-in directory. In the MIGRATION.ORDER file, you can list the files you want moved first, such as temporary files, or files with the .LST file type.

- EXAMPLE -

```
@CREATE  (FILE) MIGRATION.ORDER
Input:   MIGRATION.ORDER.1
00100    *.TMP,*.LST
0200     $
*E

[MIGRATION.ORDER.1]
@
```

The SET FILE RESIST (MIGRATION OF FILES) command[1] also gives you some control over involuntary file removal. It delays migration of the specified files for as long as possible.

- EXAMPLE -

```
@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[1] :

- EXAMPLE -

```
@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.

[1] This command/subcommand is not available with TOPS-20 monitors prior to Version 5.

- EXAMPLE -

```
@DIRECTORY (OF FILES)

  PS:<SARTINI>
 2TEST.DAT.3
 ADDTWO.FOR.3
 BCHECK.TST.1
 CCHECK.TST.1
 CHECK.TST.7
 COMMANDS.TV.1
 DCHECK.TST.1
 LARGE.DAT.1
 LOGIN.CMD.2
 MAIL.TXT.1
 NEWACCT.LST.1;OFFLINE
 OUTLINE.LST.24;OFFLINE
 OVERVIEW.LST.10;OFFLINE
 SQUARE.B20.1
 TEST.FIL.2

 Total of 15 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. On the following line, the system prints the filename and [OK], indicating that it received the request.

- EXAMPLE -

```
@RETRIEVE (FILES) MYTEST.DAT.1
 MYTEST.DAT.1 [OK]
@
```

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

- EXAMPLE -

```
@INFORMATION (ABOUT) RETRIEVAL-REQUESTS

Retrieval Queue:
 Name     Req#    Tape 1   Tape 2   User
 ------   ----    ------   ------   --------------------
 ADVENT      6      5845     5641   ENGEL
 CHESS       7      5845     5641   ENGEL
 OTHELL      8      5845     5641   ENGEL
 NVTHAK      2      5845     5641   CRUGNOLA
 NVTHAK      5      5845     5641   CRUGNOLA
 NVTHAK     11      5845     5641   CRUGNOLA
 MYTEST     68      5854     5852   SARTINI
 OTHELL      9      5641     8459   ENGEL
 There are 8 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.

- EXAMPLE -

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

6.11 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.

6.11.1 Archiving Files

To mark a file for archiving, give the ARCHIVE command, followed by the name of the file you want archived. On the following line, the system prints the name of the file, and in brackets, the word "Requested".

- EXAMPLE -

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

6.11.2 Getting Information About Archive Status of Files

To see that the file is marked for archiving, give the INFORMATION (ABOUT) ARCHIVE-STATUS command, followed by the name of the file. The system prints the filename and a message that archiving has been requested.

- EXAMPLE -

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

You can also give the INFORMATION (ABOUT) 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.

- EXAMPLE -

```
@DIRECTORY,
@@ARCHIVE
@@

   PS:<SARTINI>

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

 Total of 2 files
@
```

When you mark a file for archiving, you cannot change the contents of the file nor delete the file. Because you marked the file CHECK.TXT.1 for archiving, CHECK.TXT.1 no longer appears in your directory unless you include the ARCHIVE subcommand in the DIRECTORY command. If you try to edit that file, the EDIT program finds that the file has been marked for archiving and cannot be opened for editing. EDIT than prints a message telling you there is no such file and creates a new file. The new file has a generation number one higher than that of the file you marked for archiving.

- EXAMPLE -

```
@EDIT CHECK.TXT

%No such filename, Creating New file
Input:  CHECK.TXT.2
00100
```

If you try to delete the file (in this case, CHECK.TXT.1) with the DELETE command or copy the file with the COPY command, the system prints a message telling you that there is no such file.

6.11.3 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 (ABOUT) ARCHIVE-STATUS command shows that archive is requested but not completed.

- EXAMPLE -

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

6.11.4 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.

- EXAMPLE -

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

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

- EXAMPLE -

```
@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.

- EXAMPLE -

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

6.11.5 Archiving Expired Files Automatically[1]

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 (OF DIRECTORY) command [1] enables this automatic archiving. This command is discussed at the end of the section.

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

- EXAMPLE -

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

PS:<TUCKER.USER>
Online expiration

ARCHIV.MEM.4        3-May-82
  .QNO.15           5-May-82
  .RNO.15          21-Nov-82
COMAND.CMD.5       21-Nov-82
MEMO.INI.1          8-Apr-82
USER.RNO.2          8-Apr-82

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 (OF DIRECTORY) command [1]:

- EXAMPLE -

```
@SET DIRECTORY ONLINE-EXPIRATION-DEFAULT (OF DIRECTORY) <TUCKER> (TO) 26-NOV-82
Password:
@
```

You can specify a time interval rather than a specific date[1] :

- EXAMPLE -

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

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

---

[1] This feature/command is not available with TOPS-20 monitors prior to Version 5.

You can also establish on-line expiration dates for individual files[1] :

- EXAMPLE -

```
@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 (FILES) command[1] :

- EXAMPLE -

```
@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[1] :

- EXAMPLE -

```
@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[1] :

- EXAMPLE -

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

This is the default setting for directories.

---

[1] This command/feature is not available with TOPS-20 monitors prior to Version 5.

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

- EXAMPLE -

```
@INFORMATION (ABOUT) DIRECTORY (DIRECTORY NAME) <TUCKER>
Name PS:<TUCKER>
Password - not available
Working disk storage page limit 195
Permanent disk storage page limit 215
OPERATOR
Archive online expired files
Number of directory 1036
Account default for LOGIN 341
Maximum subdirectories allowed 293
Last LOGIN 9-Jul-81 11:11:38
Online expiration default 26-Nov-82

@
```

The line "Archive online expired files" indicates that automatic archiving will take place. If the SET DIRECTORY NO ARCHIVE-ONLINE-EXPIRED-FILES command[1] is in effect, this line does not appear in the information display.

[1] This command is not available with TOPS-20 monitors prior to Version 5.

CHAPTER 7

USING MAGNETIC TAPE

This chapter describes:

- Using magnetic tape storage (Section 7.1)
- Using unlabeled tapes (Section 7.2)
- Using labeled tapes (Section 7.3)

TOPS-20 commands and programs mentioned in this chapter are:

ASSIGN	INFORMATION
BACKSPACE	MOUNT
CANCEL	REWIND
DEASSIGN	SET
DISMOUNT	SKIP
DUMPER	UNLOAD

7.1 USING MAGNETIC TAPE STORAGE

Magnetic tape provides off-line storage for data. You put data onto tape for storage using the DUMPER program or a program of your own. (For a complete description of the DUMPER program, refer to the TOPS-20 User Utilities Guide.) Prior to Release 4 of TOPS-20, all installations used unlabeled tapes. With Release 4, your installation now has the option of using labeled tapes.

An unlabeled tape is identified only by a gummed label on the outside of the tape reel.

A labeled 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 labeled and unlabeled tapes.

7.2 USING UNLABELED TAPES

Before you use an unlabeled tape, give the INFORMATION (ABOUT) 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 Unlabeled Tapes With Tape Allocation Enabled

If tape allocation is enabled on your system, you can mount an unlabeled 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.

- EXAMPLE -

```
@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 (ABOUT) MOUNT-REQUESTS command.

- EXAMPLE-

```
@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.

- EXAMPLE-

```
@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 (ABOUT) AVAILABLE-DEVICES command.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@ASSIGN (DEVICE) MTA2:
@
```

After assigning the drive to your job, you can run the PLEASE program and ask the operator to mount your tape.

- EXAMPLE -

```
@PLEASE OPERATOR PLEASE MOUNT TAPE TEST:
[Message sent at 10:15:01 Waiting for Operator Response]

11:20:40 From Operator Terminal 4:
=>Your Tape is mounted.
@
```

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 (ABOUT) TAPE-PARAMETERS command.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@SET TAPE DENSITY (TO) 800
@
```

These changed parameters remain in effect until you log off, or change the parameters. 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.

- EXAMPLE -

```
@SKIP (DEVICE) MTA2: 4 FILES
@
```

7.3 USING LABELED TAPES

The operator creates the labeled 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 identifer, also called a VOLID. The VOLID is a unique number assigned to the tape.

Once the operator creates the labeled 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.

- EXAMPLE -

```
@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 (ABOUT) VOLUMES (OF TAPE) command, followed by the tape setname 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:

- EXAMPLE -

```
@INFORMATION (ABOUT) VOLUMES (OF TAPE) ABCD:
Volumes of tape set BCD:  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.

tape setname and the /VOLIDS: switch in the MOUNT TAPE command.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@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 you that 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 (ABOUT) MOUNT-REQUESTS command. The system prints a list of mount requests in the queue, and indicates the status of the request, for example, waiting.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@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)

The TOPS-20 commands and programs mentioned in this chapter are:

BASIC	EDIT	PUSH
CONTINUE	EXECUTE	RUN
CREATE	FILCOM	TERMINAL
DIRECTORY	HELP	
DUMPER	POP	

8.1 RUNNING SYSTEM PROGRAMS

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:

- EXAMPLE -

```
@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:

```
destination file specification = source file specification(s)
/switch(es)
```

For example, the FILCOM (for FILe COMparison) program works as follows:

- EXAMPLE -

```
@FILCOM

*TRACK.SCM=TRACK1.FOR,TRACK2.FOR

%files are different

*^C
@
```

Type FILCOM and press RETURN; the system prints an asterisk. Type the following FILCOM command: TRACK.SCM=TRACK1.FOR,TRACK2.FOR. This command compares TRACK2.FOR and TRACK1.FOR, and places a list of the differences in the file TRACK.SCM. TRACK1.FOR and TRACK2.FOR are input files; TRACK.SCM is the output file. The .SCM file type stands for Source CoMparison.

You cannot use recognition on file specifications or switches when you run any of the programs listed in Table 4-2. (Refer to Section 4.1 for a description of the exact file specification you must type.)

Other system programs respond by printing a prompt that identifies the program, such as the prompt for the DUMPER program.

- EXAMPLE -

```
@DUMPER
DUMPER>
```

You can use recognition on commands and arguments to these programs.

8.3 GETTING INFORMATION ABOUT SYSTEM FEATURES

To get information about a system feature, use the HELP command. Type HELP, followed by a space and a question mark. The system prints a list of features for which it has helpful information.

- EXAMPLE -

```
@HELP ? one of the following:
 ACCT20    ACCTPR   APL      APLSF    BLIS10   CHECKD   CHKPNT
 COBDDT    COBOL    CREF     DAEMDB   DBINFO   DBMEND   DIRTST
 DLUSER    DUMPER   EDIT     FE       FILCOM   FORDDT   FORDML
 FORTRA    HELP     ISAM     LIBARY   LINK     LPTSPL   MACRO
 MAIL      MAKLIB   MAKRAM   MAKVFU   OPLEAS   PLEASE   PTYCON
 QUEUE     RDMAIL   RERUN    RSXFMT   RUNINP   RUNOFF   SCHEMA
 SORT      SYSERR   SYSJOB   TRANSL   ULIST    WATCH
  or '*'
  or confirm with carriage return
@HELP
```

To get help on a specific feature, type HELP, leave a space, type the name of the feature, and press RETURN. The system prints some helpful information.

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:

- EXAMPLE -

```
@RUN (PROGRAM) LESTSQ
```

You can use recognition in typing the file specification.

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-Producing a Simple Program.)

To run another user's program, give the file specification with the RUN command:

- EXAMPLE -

```
@RUN (PROGRAM) <HOLLAND>TEST
```

You must have read and/or execute access to the file and access to the directory. If you do not have file read access, the system prints:

- EXAMPLE -

```
@RUN (PROGRAM) <HOLLAND>TEST
?Read access required
@
```

If you do not have access to the directory, the system prints the message:

- EXAMPLE -

```
@RUN (PROGRAM) <HOLLAND>TEST
?Directory access privileges required
@
```

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

If you start a program (or command) and cannot stop it, type two CTRL/Cs. Only one CTRL/C echoes on the terminal. The program (or command) stops and returns you to command level.

- EXAMPLE -

```
@FILCOM

*TEST.SCM=TEST.1,TEST.2
^C
@
```

You can now give any command that does not change the contents of memory, for example, the TERMINAL command. 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.) You can also give the PUSH command, do some other work, give the POP command, and then CONTINUE the program.

- EXAMPLE -

```
@FILCOM

*TEST.SCM=TEST.1,TEST.2
^C
@TERMINAL (MODE IS) PAGE
@CONTINUE

%files are different

*
```

Some programs (namely 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.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@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.

8.5.3 Typing CTRL/T to Print the Run Status

To find out if a program is running, type CTRL/T. The system prints the current time, the state of the program, the amount of system time you used since logging in, and the load average for the system.

- EXAMPLE -

```
@RUN (PROGRAM) TEST
 09:36:35 TEST Running at 404157  Used 0:00:35.8 in 0:30:39, Load   4.04
```

Typing CTRL/T does not affect your program; it simply prints information about it. The information is in the form:

time name status Used CPU-time in logged-in-time, Load load average

The status message tells you the status of the program. Table 8-1 lists some of the common status messages.

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 Commands Reference Manual and to the TOPS-20 WATCH document for further information on load averages.

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.

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.

Table 8-2 (Cont.)
Unexpected Process Termination Messages

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 process (fork) 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 for example, 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 EDIT) to create the file and return to continue your original program. Before running another program to create the file, type two CTRL/Cs and give a PUSH command, which creates a new copy of memory and the TOPS-20 command, language. You can now run as many programs as you wish. 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 destroy the old program, 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.

- EXAMPLE -

```
@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 4(2441)
@CREATE (FILE) NUMBER.DAT       !Create the missing file
INPUT: NUMBER.DAT.1
00100    23461                  !Store required date in it.
00200    $
*EU                             !Save the file and unsequence
                                 the line numbers.

[NUMBER.DAT.1]
@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 cannot execute the command, it cancels the 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.

When you give a PUSH command, the contents of memory are saved in their exact state and cannot be changed until you give a POP command to return to that level.

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 command (Section 9.3)

The TOPS-20 commands and programs mentioned in this chapter are:

ALGOL	CPL	EXECUTE	MACRO
APL	CREATE	FILCOM	MAKLIB
BASIC	CREF	FORTRAN	RUN
COBOL	DEBUG	INFORMATION	SAVE
COMPILE	EDIT	LOAD	START

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. For convenient operation, the source program should have the standard file type recognized by the appropriate language compiler. Once a language compiler compiles a source program, it produces an object program with a .REL file type. To write the source program, choose one of the programming languages: ALGOL, COBOL, FORTRAN, or MACRO. 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.

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.

- EXAMPLE -

```
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.

- EXAMPLE -

```
@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:

- EXAMPLE -

```
@COMPILE (FROM) SMALL/LIST/COMPILE
FORTRAN: SMALL
MAIN.
@
```

To see the location of your program in the line printer output queue, give the INFORMATION (ABOUT) OUTPUT-REQUESTS command and press RETURN.

- EXAMPLE -

```
@INFORMATION (ABOUT) OUTPUT-REQUESTS

Line 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.

If you repeatedly edit your file, then give a LOAD-class command, you can save some typing by using the G command to leave EDIT. The G command saves your file and gives your last LOAD-class command. The following example shows how to edit the file to print three times a number, give the EXECUTE command, edit the file again, and then give the G command to reissue the previous LOAD-class command:

- EXAMPLE -

```
@EDIT (FILE) SMALL.FOR
Edit:  SMALL.FOR.1
*I800!3
00820            Z=3:X
00840            TYPE 104,X,Z
00860    104     FORMAT (' THREE TIMES ',F,' IS ',F)
00880    $
*E

[SMALL.FOR.2]
@EXECUTE (FROM) SMALL
FORTRAN:  SMALL
00820            Z=3:X
?FTNFWE LINE:00820 FOUND ':'  WHEN EXPECTING A STATEMENT END

?FTNFTL   MAIN.          1 FATAL ERRORS AND NO WARNINGS
LINK:   Loading
[LNKNSA No start address]

EXIT
@EDIT (FILE)
Edit:  SMALL.FOR.2
*R820
00820            Z=3*X
1 Lines (00820/1) deleted
*G

[SMALL.FOR.3]

FORTRAN:  SMALL
MAIN.
LINK:   Loading
[LNKXCT SMALL Execution]

TYPE A NUMBER: 5

TWO TIMES        5.0000000 IS      10.0000000
THREE TIMES        5.0000000 IS      15.0000000
STOP

END OF EXECUTION
CPU TIME: 0.08  ELAPSED TIME: 5.77
EXIT
@
```

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 core image file.

- EXAMPLE -

```
@LOAD (FROM) SMALL
LINK:   Loading

EXIT
@SAVE (ON FILE)
 SMALL.EXE.1 SAVED
@
```

To run the program, give a RUN command.

- EXAMPLE -

```
@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:

- EXAMPLE -

Create COMP.FOR:

```
@CREATE (FILE) COMP.FOR
Input: COMP.FOR.1
00100           TYPE 101
00200    101    FORMAT (' TYPE TWO NUMBERS: '$)
00300           ACCEPT 102,A,B
00400    102    FORMAT (2F)
00500           CALL ADDEM(A,B)
00600           CALL DIFFER(A,B)
00700           STOP
00800           END
00900    $
*E                              !Save it

[COMP.FOR.1]
@
```

Create ADDEM.FOR:

```
@CREATE (FILE) ADDEM.FOR
Input: ADDEM.FOR.1
00100           SUBROUTINE ADDEM(A,B)
00200           C = A + B
00300           TYPE 101,C
00400    101    FORMAT (' THE SUM IS: ',F)
00500           RETURN
00600           END
00700    $
*E

[ADDEM.FOR.1]
@
```

Create DIFFER.FOR:

```
@CREATE (FILE) DIFFER.FOR
Input: DIFFER.FOR.1
00100           SUBROUTINE DIFFER(A,B)
00200           C = ABS(A - B)
00300           TYPE 101,C
00400    101    FORMAT (' THE DIFFERENCE IS: ',F)
00500           RETURN
00600           END
00700    $
*E

[DIFFER.FOR.1]
@
```

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.

- EXAMPLE -

```
@EXECUTE (FROM) COMP,ADDEM,DIFFER
FORTRAN: COMP
MAIN.
FORTRAN: ADDEM
ADDEM
FORTRAN: DIFFER
DIFFER
LINK: LOADING
[LNKXCT COMP EXECUTION]
TYPE TWO NUMBERS: 34,56
THE SUM IS:       90.0000000
THE DIFFERENCE IS:      22.0000000
STOP

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 directly to the line printer. The following example produces a cross reference listing for the COMP, ADDEM, and DIFFER programs.

- EXAMPLE -

```
@EXECUTE (FROM) /CREF COMP,ADDEM,DIFFER  !Include /CREF
FORTRAN: COMP
MAIN.
FORTRAN: ADDEM
ADDEM
FORTRAN: DIFFER
DIFFER
LINK: Loading
[LNKXCT COMP EXECUTION]
TYPE TWO NUMBERS: 34,56
THE SUM IS:       90.0000000
THE DIFFERENCE IS:      22.0000000
STOP

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:

- EXAMPLE -

```
@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 give the CREF command.

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.

- EXAMPLE -

```
@CREATE (FILE) OPREAD.FOR
Input: OPREAD.FOR.1
00100    C      SUBROUTINE OPREAD(NAME)
00200           OPREAD - OPENS A FILE FOR READING
00300           DOUBLE PRECISION NAME
00400           OPEN(UNIT=21,ACCESS='SEQIN',FILE=NAME)
00500           RETURN
00600           END
00700    $
*E

[OPREAD.FOR.1]

@CREATE (FILE) OPWRIT.FOR
Input: OPWRIT.FOR.1
00100    C      SUBROUTINE OPWRIT(NAME)
00200           OPWRIT - OPENS A FILE FOR WRITING
00300           DOUBLE PRECISION NAME
00400           OPEN(UNIT=21,ACCESS='SEQOUT',FILE=NAME)
00500           RETURN
00600           END
00700    $
*E

[OPWRIT.FOR.1]

@CREATE (FILE) CLREAD.FOR
Input: CLREAD.FOR.1
00100    C      SUBROUTINE CLREAD(NAME)
00200           CLREAD - CLOSES A FILE OPENED FOR READING
00300           DOUBLE PRECISION NAME
00400           CLOSE(UNIT=21,FILE=NAME)
00500           RETURN
00600           END
00700    $
*E
[CLREAD.FOR.1]

@CREATE (FILE) CLWRIT.FOR
Input: CLWRIT.FOR.1
00100    C      SUBROUTINE CLWRIT(NAME)
00200           CLWRIT - CLOSES A FILE OPENED FOR WRITING
00300           DOUBLE PRECISION NAME
00400           CLOSE(UNIT=21,FILE=NAME)
00500           RETURN
00600           END
00700    $
*E

[CLWRIT.FOR.1]
@
```

9.2.4.2 **Compiling the Subroutines** - After entering the subroutines into files, compile them to produce four separate object files.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@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. (Chapter 5 discusses how to create and edit files.) 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

- EXAMPLE -

```
@MAKLIB
*MASTER=NEW/LIST
```

you can type the following command, and the /LIST switch is automatically included in the command:

- EXAMPLE -

```
@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.

- EXAMPLE -

```
*TTY:=DOFILE/LIST
        Listing of Modules
Produced by MAKLIB Version 2A(67) on 26-Sep-79 at 15:00:48

        ****************************

DSK:DOFILE.REL[4,164] Created on 26-Sep-79 at 15:00:00

OPREAD  400016  000007
OPWRIT  400016  000010
CLREAD  400016  000007
CLWRIT  400016  000010
*
```

To end MAKLIB, type a CTRL/C.

```
*^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.

- EXAMPLE -

```
@CREATE (FILE) WRITER.FOR
Input: WRITER.FOR.1
00100           DOUBLE PRECISION NAME,DAY
00200           CALL DATE(DAY)
00300           CALL OPWRIT('DATE.FIL')
00400           TYPE 101,DAY
00500     101   FORMAT (' UPDATING AS OF: ',A10)
00600           WRITE (21,102) DAY
00700     102   FORMAT (' => UPDATED ON: ', A10)
00800           CALL CLWRIT('DATE.FIL')
00900           STOP
01000           END
01100     $
*E

[WRITER.FOR.1]
@
```

After entering the main program, load it with the library file and start it. Remember to include the /LIBRARY switch.

- EXAMPLE -

```
@LOAD (FROM) WRITER,DOFILE/LIBRARY
FORTRAN: WRITER
MAIN.
LINK:   Loading

EXIT
@START
UPDATING AS OF:  26-Sep-79
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.

- EXAMPLE -

```
@EDIT (FILE) OPWRIT.FOR
EDIT: OPWRIT.FOR.1
*I450!2
00450          TYPE 101,NAME
00470    101   FORMAT (' [',A10,' OPENED]')
00490    $
*E

[OPWRIT.FOR.2]
@
```

After editing the file, compile a new object file.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@MAKLIB
*TTY:=DOFILE/LIST
        Listing of Modules
Produced by MAKLIB Version 2A(67) on 26-Sep-79 at 15:05:06

        ****************************

DSK:DOFILE.REL[4,164] Created on 26-Sep-79 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.

- EXAMPLE -

```
*TTY:=DOFILE/LIST
        Listing of Modules
Produced by MAKLIB Version 2A(67) on 26-Sep-79 at 15:10:10

        ****************************

DSK:DOFILE.REL[4,164] Created on 26-Sep-79 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.

- EXAMPLE -

```
@LOAD (FROM) WRITER,DOFILE/LIBRARY
LINK:   Loading

EXIT
@SAVE
 WRITER.EXE.1 SAVED
@START
[DATE.FIL   OPENED]
UPDATING AS OF:  26-Sep-79
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).

- EXAMPLE -

```
@LOAD (FROM) WRITER,DOFILE/LIBRARY
LINK:   Loading

EXIT
@
```

Give the SAVE command to save the program. To use the program later, give the RUN command.

- EXAMPLE -

```
@SAVE
 WRITER.EXE.1 Saved
```

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.

- EXAMPLE -

```
@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. The following example shows how to create a command file that will compile the four subroutines:

- EXAMPLE -

```
@CREATE (FILE) D.CMD
Input: D.CMD.1
00100   OPREAD,OPWRIT,CLREAD,CLWRIT
00200   ↑Z
*E

[D.CMD.1]
@
```

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.

- EXAMPLE -

```
@COMPILE (FROM) @D
FORTRAN: OPREAD
OPREAD
FORTRAN: OPWRIT
OPWRIT
FORTRAN: CLREAD
CLREAD
FORTRAN: CLWRIT
CLWRIT
@
```

The following example shows an indirect file you can use to create the program WRITER and to search the library:

- EXAMPLE -

```
@CREATE (FILE) W.CMD
Input: W.CMD.1
00100   WRITER.FOR,DOFILE.REL/LIBRARY
00200   ↑Z
*E

[W.CMD.1]
@LOAD (FROM) @W
FORTRAN: WRITER
MAIN.
LINK:   Loading

EXIT
@
```

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 file specification = source file specification1, source file specification2,/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.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@FILCOM                                  !Start FILCOM

*TTY:=WRITER.FOR,UPDATE.FOR              !Type the command
FILE 1) DSK:WRITER.FOR  CREATED: 1554 2-JAN-1979
FILE 2) DSK:UPDATE.FOR  CREATED: 1556 24-JAN-1979

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
```

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.

If you repeatedly edit your file and then give a LOAD-class command, you can save some typing by using the G command to leave EDIT. The G command saves the file you were editing and gives your last LOAD-class command. Refer to Section 9.1.3 for an example of using the G command.

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 file specifications listing 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).

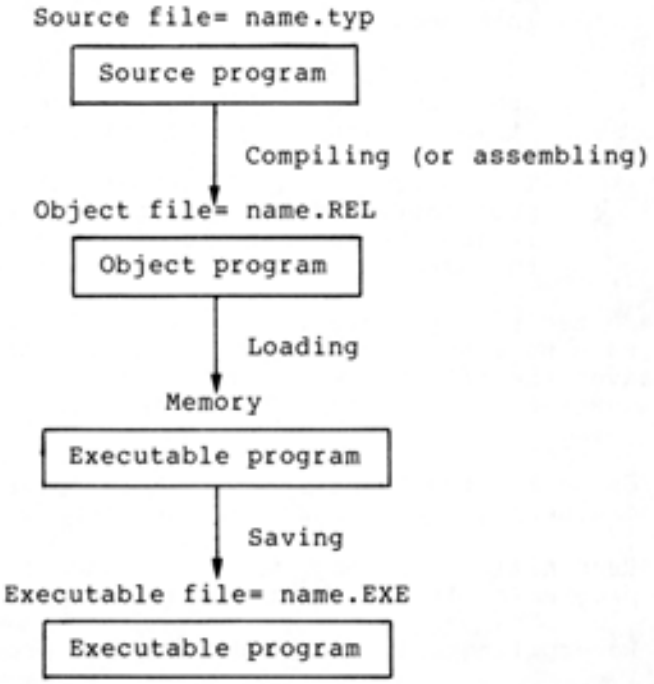


Figure 9-1 Source, Object, and Executable Programs

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:

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@EXECUTE (FROM) LAYOUT,FILLER,TTYOUT
```

The third program, GAMMA, has a POLAR subroutine that is included in the EXECUTE command.

- EXAMPLE -

```
@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 period at the end of the filename and file type. For example, you can give the command:

- EXAMPLE -

```
@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 specification matching the first file specification in Table 9-1. If there is no matching file specification, the system searches for a match of the second file specification. If there is none, the system continues the search in this manner until it finds a file specification matching one in Table 9-1, or until it reaches the end of Table 9-1. If there is no matching file, the system prints an error message.

Upon finding a matching file, the system (if necessary) compiles it using the language 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 in one of the file

specifications 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 Specifications

File Specification	Language Compiler
name.	FORTRAN (default)
name.FOR	FORTRAN
name.MAC	MACRO
name.CBL	COBOL
name.ALG	ALGOL
name.REL	Object program, do not compile

For example, if you type the file name PAYROL, the system looks for PAYROL., PAYROL.FOR, PAYROL.MAC, PAYROL.CBL, PAYROL.ALG, and finally PAYROL.REL. 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.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@COMPILE (FROM) ENABLE.MON/MACRO
MACRO: ENABLE
@
```

9.3.2.2 **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.

- EXAMPLE -

```
@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.

- EXAMPLE -

```
@LOAD (FROM) MIDDLE.OBJ/RELOCATABLE
LINK:   Loading

EXIT
@
```

9.3.2.3 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.

- EXAMPLE -

```
@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 the parameter definitions and some storage and the PROMPT file contains the main logic of the program.

- EXAMPLE -

```
@CREATE (FILE) DEFS.MAC          !Create the parameter file
Input: DEFS.MAC.1
00100           SEARCH MONSYM,MACSYM
00200   PRMTXT: ASCIZ/NEXT COMMAND>/
00300           T1==1
00400           T2==2
00500   $
*E

[DEFS.MAC.1]
@CREATE (FILE) PROMPT.MAC        !Create the main file
Input: PROMPT.MAC.1
00100           TITLE PROMPT
00200   PROMPT: HRROI T1,PRMTXT  !Get address of string
00300           PSOUT            !Print it
00400           HALTF            !Stop
00500           END PROMPT
00600   $
*E

[PROMPT.MAC.1]
@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:

- EXAMPLE -

@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 (FROM) /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:

- Submitting a Batch job (Section 10.1)
- Creating a control file (Section 10.1.1)
- Submitting a control file to Batch (Section 10.1.2)
- Checking a batch job (Section 10.1.3)
- Examining the output from a batch job (Section 10.1.4)
- Modifying a batch job (Section 10.2)
- Canceling a batch job (Section 10.3)

The TOPS-20 commands and system programs mentioned in this chapter are:

CANCEL	MAIL
CREATE	MODIFY
DIRECTORY	PRINT
FILCOM	SUBMIT
INFORMATION	TERMINAL

10.1 SUBMITTING 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 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 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 using recognition or 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:

- type only one @ before a subcommand.
- type a @ for an extra line for the RETURN that terminates the subcommand.

You can also create a BATCH.CMD file that is read by the system every time you submit a batch job. 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.

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, give the CREATE command, followed by a file name, and a .CTL file type. 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:

- EXAMPLE -

```
@CREATE (FILE) TEST.CTL
Input: TEST.CTL.1
00100  @FILCOM
00200  *SAMPLE.SCM=DATA.OLD,DATA.NEW
00300  @PRINT SAMPLE.SCM
00400  $
*E

[TEST.CTL.1]
@
```

NOTE

You can include the command to run the MAIL program in your batch control file. You can use the MAIL program to send a message informing you when the batch job is done. (Refer to the TOPS-20 User Utilities Guide for more information on the MAIL program.)

10.1.2 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.

- EXAMPLE -

```
@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:

- EXAMPLE -

```
@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]
@
```

To submit a simple batch job, it is not necessary to include switches. However, you can include switches with the SUBMIT command. To obtain a list of valid switches, type SUBMIT followed by a ?. The system prints a list of switches, and reprints SUBMIT. Type /AFTER: and a ?; the system prints the valid arguments to type following /AFTER: switch.

- EXAMPLE -

```
@SUBMIT (BATCH JOB)/? Switch, one of the following:
/ACCOUNT:                   /AFTER:          /ASSISTANCE:
/BATCH-LOG:                 /BEGIN:          /CARDS:
/CONNECTED-DIRECTORY:       /DELETE          /DEPENDENCY-COUNT:
/DESTINATION-NODE:          /FEET:           /JOBNAME:
/LOGDISPOSITION:            /LOGNAME:        /NOTIFY:
/OUTPUT:                    /PAGES:          /PRIORITY:
/PROCESSING-NODE:           /READER          /RESTARTABLE:
/TAG:                       /TIME:           /TPLOT:
/UNIQUE:                    /USER:
@SUBMIT (BATCH JOB) /
```

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:.

- EXAMPLE -

```
@SUBMIT (BATCH JOB)/AFTER:8-AUG-79 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:

- EXAMPLE -

```
@SUBMIT (BATCH JOB) TEST/AFTER:10-Aug-79 DATA
[Job TEST Queued, Request-ID 110, Limit 0:05:00]
[Job DATA Queued, Request-ID 111, Limit 0:05:00]
@
```

10.1.2.1 **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.

- EXAMPLE -

```
@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 give the /OUTPUT: switch with SUBMIT commands, place the following command in COMAND.CMD:

- EXAMPLE -

```
@SET DEFAULT (FOR) SUBMIT /OUTPUT:NOLOG
```

To see the defaults you have set for the SUBMIT command, give the INFORMATION (ABOUT) DEFAULTS (FOR) SUBMIT command.

- EXAMPLE -

```
@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.

To see the defaults you have set for the SUBMIT command, give the INFORMATION (ABOUT) DEFAULTS (FOR) SUBMIT command.

10.1.3 Checking a Batch Job

To check the progress of the batch job, give the INFORMATION (ABOUT) 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 (ABOUT) BATCH-REQUESTS command. To print the status of another user's job, use the /USER: switch, followed by the user's name.

- EXAMPLE -

```
@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
  SYSERR      7  00:05:00  BLOUNT                   /After: 6-Aug-79 23:59
  TEST      110  00:05:00  SARTINI                  /After:10-Aug-79 00:00
There are 7 Jobs in the Queue (2 in Progress)

@
```

10.1.4 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.

- EXAMPLE -

```
@DIRECTORY (OF FILES) TEST

   SNARK:<SARTINI>
 TEST.CTL.1
 .LOG.1

TOTAL OF 2 FILES
@
```

The following example contains the log file from the batch job, TEST.CTL.1.

- EXAMPLE -

```
14:25:09 BAJOB  BATCON version 103(3000) running TEST sequence 6000 in stream 1
14:25:09 BAFIL  Input from SNARK:<SARTINI>TEST.CTL.1
14:25:09 BAFIL  Output to  SNARK:<SARTINI>TEST.LOG
14:25:09 BASUM  Job parameters
                Time:00:05:00   Unique:YES   Restart:NO   Output:NOLOG

14:25:09 MONTR
14:25:09 MONTR   SYSTEM 2102 DEVELOPMENT SYSTEM, TOPS-20 Monitor 4(2621)
14:25:09 MONTR  @LOGIN SARTINI 341
14:25:13 MONTR   Job 30 on TTY225 25-Aug-79 14:25:13
14:25:14 MONTR   End of COMAND.CMD.1
14:25:14 MONTR  @
14:25:14 MONTR  [CONNECTED TO SNARK:<SARTINI>]
14:25:14 MONTR  @SET TIME-LIMIT 300
14:25:15 MONTR  @@FILCOM
14:25:18 USER
14:25:18 USER   **SAMPLE.DIF=DATA.OLD,DATA.NEW
14:25:19 USER
14:25:19 USER   %files are different
14:25:20 USER
14:25:20 USER   *^C
14:25:21 MONTR  @@PRINT SAMPLE.DIF
14:25:22 MONTR  [LPT:SAMPLE=/Seq:6001/Limit:52, 1 File]
14:25:22 MONTR  @^C
14:25:22 MONTR  @LOGOUT
14:25:23 MONTR  Killed Job 30, User SARTINI, Account 341, TTY 225,
14:25:23 MONTR    at 25-Aug-79 14:25:23, Used 0:0:1 in 0:0:10
```

The system begins each line in the log file with the exact 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). The remainder of the line contains system output and the lines in the control file.

The first command in the control file is FILCOM, preceded by an @. In the log file the system prints an @ before it prints the line from the control file @FILCOM. Therefore, you see an @@ before FILCOM when you read the log 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.

- EXAMPLE -

```
@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, 1979:

- EXAMPLE -

```
@MODIFY (REQUEST TYPE) BATCH (ID) TEST/AFTER:15-AUG-79
[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.

- EXAMPLE -

```
@CANCEL (REQUEST TYPE) BATCH (ID) TEST
[1 Job Canceled]
@
```

If you have several batch jobs running, you can cancel them all by using an asterisk. 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:

- EXAMPLE -

```
@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.

SYSTEM ACCESS COMMANDS

These commands allow you to gain and relinquish access to the system, to change your job's account, 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.

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	Starts the system editor, to make a new 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	Starts the system editor to change an existing 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.
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 the specified 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.

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.

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.
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.
EXAMINE	Allows you to examine an address in memory.
EXECUTE	Translates, loads, and begins execution of a program.
FORK	Makes the TOPS-20 language work for a particular address space.
GET	Loads an executable program from the specified file into memory, but does not start it.
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 Language and returns control to the previous copy of the Command Language.
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 with the proper file specification.
SET	Sets the value of various job parameters.
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.

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.

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 terminal links and advising links.
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.
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 the 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.

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. 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
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
B10	Source	Source file in BLISS
B11	Source	Source file in BLISS-11
BAK	Source	Backup file from TECO

Table B-1 (Cont.)
Standard File Types

File Type	Kind of File	Meaning
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
BLB	ASCII	Blurb file
BLI	Source	Source file in BLISS language
BOX	ASCII	Output of box program - picture for use in specifications and manuals
BUG	Object	Saved to show a program error
BWR	ASCII	Beware file listing warnings about a file or program
CAL	Object	CAL data and program files
CBL	Source	Source file in COBOL language
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

Table B-1 (Cont.)
Standard File Types

File Type	Kind of File	Meaning
CTL	ASCII	Batch control file
DAT	ASCII, Binary	Data (FORTRAN) file
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
LAP	ASCII	Output from the LISP compiler

Table B-1 (Cont.)
Standard File Types

File Type	Kind of File	Meaning
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
LSQ	ASCII	Queue listing created by QUEUE program
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	Memorandum file
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

Table B-1 (Cont.)
Standard File Types

File Type	Kind of File	Meaning
PAL	Source	Source file in PAL 10 (PDP-8 assembler)
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
RAM	ASCII	DECSYSTEM-20 microcode
REL	Object	Relocatable binary file
RIM	Object	RIM loader file
RNB	ASCII	RUNOFF input for producing a .BLB file
RNC	ASCII	RUNOFF input for producing a .CCO file
RND	ASCII	RUNOFF input for producing a .DOC file
RNE	ASCII	RUNOFF input for producing error message text
RNH	ASCII	RUNOFF input for producing a .HLP file
RNL	ASCII	RUNOFF input for a program logic manual
RNM	ASCII	RUNOFF input for producing a .MAN file
RNO	ASCII	Programming specifications in RUNOFF input
RNP	ASCII	RUNOFF input for producing a .OPR file
RNS	ASCII	RUNOFF input for a text file of standards

Table B-1 (Cont.)
Standard File Types

File Type	Kind of File	Meaning
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

Table B-1 (Cont.)
Standard File Types

File Type	Kind of File	Meaning
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
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

THE BAUD-RATE SWITCHES FOR VT50 AND VT52 TERMINALS

ENVIRONMENT	DESIRED BAUD RATE Trans.	Receiv.	S1-S2 SWITCH POSITIONS VT52	VT50(H)
OFF-LINE	9600	9600	1-G	1-G
	4800	4800	n.a.	1-F
	2400	2400	1-F	1-E
	1200	1200	1-E	1-D
	600	600	1-D	1-C
	110	110	1-B	1-B
FULL DUPLEX WITH LOCAL COPY	9600	9600	2-G	2-G
	4800	4800	7-A	2-F
	2400	2400	2-F	2-E
	1200	1200	2-E	2-D
	600	600	2-D	2-C
	300	300	4-A	n.a.
	150	150	5-A	n.a.
	110	110	2-B	2-B
FULL DUPLEX	9600	9600	3-G	3-G
	4800	4800	7-C	3-F
	2400	2400	3-F	3-E
	1200	1200	3-E	3-D
	600	600	3-D	3-C
	300	300	4-C	4-A
	150	150	5-C	5-A
	110	110	3-B	3-B
	75	75	6-C	6-A
	4800	9600	7-G	n.a.
	4800	2400	7-F	n.a.
	4800	1200	7-E	n.a.
	4800	600	7-D	n.a.
	4800	110	7-B	n.a.
	300	9600	4-G	4-G
	300	4800	n.a.	4-F
	300	2400	4-F	4-E
	300	1200	4-E	4-D
	300	600	4-D	4-C
	300	110	4-B	n.a.

THE BAUD-RATE SWITCHES FOR VT50 AND VT52 TERMINALS

ENVIRONMENT	DESIRED BAUD RATE Trans.	Receiv.	S1-S2 SWITCH POSITIONS VT52	VT50(H)
FULL DUPLEX (Cont.)	150	9600	5-G	5-G
	150	4800	n.a.	5-F
	150	2400	5-F	5-E
	150	1200	5-E	5-D
	150	600	5-D	5-C
	150	110	5-B	n.a.
	75	9600	6-G	6-G
	75	4800	n.a.	6-F
	75	2400	6-F	6-E
	75	1200	6-E	6-D
	75	600	6-D	6-C
	75	110	6-B	n.a.

n.a. - This combination of transmitting and receiving Baud rates is not available on this terminal.

The Positions of Switch S1

Switch 1 generally dictates transmitting speed. In the "match" positions, the transmitting speed will be the same as the receiving speed, which must be indicated by the setting of switch S2. Position 1 is the most counterclockwise position.

1 - Off-Line
2 - Full Duplex with Local Copy
3 - Full Duplex
4 - 300 Baud
5 - 150 Baud
6 - 75 Baud
7 - 4800 Baud (VT52 only)

The Positions of Switch S2

Switch S2 generally dictates receiving speed. In the "match" positions, the receiving speed will be the same as the transmitting speed, which must be indicated by the setting of Switch S1. Position A is the most counterclockwise position.

VT52	VT50(H)
A - Match (Bell 103) - Local Copy	A - Match (Bell 103)
B - 110 Baud	B - 110 Baud
C - Match (Bell 103)	C - 600 Baud
D - 600 Baud	d - 1200 Baud
E - 1200 Baud	E - 2400 Baud
F - 2400 Baud	F - 4800 Baud
G - 9600 Baud	G - 9600 Baud

THE BAUD-RATE SWITCHES FOR VT50 AND VT52 TERMINALS

If both switches are in a "match" position, the terminal will not function - the key-click will cease, and characters will cease to be transmitted and received. The following switch positions are illegal on the VT52:

1-A 1-C 2-A 2-C 3-A 3-C

The following switch positions will cause the VT50 and VT50H to fail:

1-A 2-A 3-A 7-A 7-B 7-C 7-E 7-F 7-G

Transmission and reception will resume if the switches are reset to a valid position.

INDEX