decsystem10
GETTING STARTED WITH TIMESHARING

digital equipment corporation · maynard, massachusetts

1st Printing June 1971
2nd Printing July 1972

CONTENTS

CONTENTS (Cont)

TABLES

FOREWORD

Getting Started With Timesharing is a simplified guide intended for the beginning timesharing user of the DECsystem-10. This document presents an overall view of the timesharing use of the System, but does not describe every command available to the user. DECsystem-10 OPERATING SYSTEM COMMANDS (DEC-10-MRDC-D) is the complete reference document for the command repertoire, and it should be referred to for any additional information.

Programs are typed directly into the computer by means of the terminal. By typing in programs, you establish communication with other programs already resident in the computer. The first resident program you communicate with is the monitor, the most important program in the computer. The monitor is the master program that plays an important role in the efficient operation of the computer. Just as the terminal is your link with the computer, the monitor is your link with the programs within the computer.

The monitor has many functions to perform, like keeping a record of what each user is doing and deciding what user should be serviced next and for how long. The one function of the monitor that is of greatest concern at this point is that the monitor retrieves any resident programs that you need. This retrieval happens only if the monitor "understands" what is expected of it. The commands to the monitor which are explained in the following sections are sufficient for the terminal to be the device by which information is inputted into the system and by which the system outputs its results.

See section 9.0 for a discussion on How to Live With the Terminal.

1.0 GETTING ON THE SYSTEM

In order to gain access to the timesharing system, you must say hello to the system by "logging in". The first move is to make contact with the computer facility by whatever means the facility has established (e.g., acoustic coupler, telephone, or dataphone). Next, notice the plastic knob (the power switch) on the lower right-hand side of the terminal. This knob has three positions: ON, OFF, and LOCAL (turning clockwise). When the knob is in the LOCAL position, the terminal is like a typewriter; it is not communicating with the system at all. The knob must be turned to the ON position in order to establish communication with the computer. When the terminal is turned ON, type a ↑C (depress the CTRL key and type C). This action establishes communication with the monitor. The monitor

---

We wish to express appreciation to Stanford University for the use of their Stanford A-1 Project User's Manual, Chapter 1, SAILON No. 54, as a guide in writing the material in this section.

signifies its readiness to accept commands by responding with a period (.). All the commands discussed in this document can only be typed to the monitor. They are operative when the monitor has typed a period, signifying that it is waiting for a command.

The first program the monitor should call in for you is the LOGIN program. This is accomplished by typing LOGIN followed by a carriage-return (depress the RETURN key). All commands to the monitor should be terminated with a carriage-return. When the monitor "sees" a carriage-return, it knows that a command has been typed and it begins to execute the command.

> In the text, underscoring is used to designate terminal output.
> A carriage-return is designated by a) .

By typing LOGIN, you cause the monitor to read the LOGIN program from the disk into core memory and it is this program that is now in control of your terminal. Before the LOGIN program is called in, the monitor assigns you a job number for system bookkeeping purposes. The system responds with an information message similar to the following.

```
JOB 17    5S0218A    TTY34
#
```

In the first line, the system has assigned your job number (17) and has given both the name of the monitor and its version number and the number of your terminal line. The version number changes whenever a change, or patch, is incorporated into the monitor. In the second line, the number sign (#), which is typed out by the LOGIN program, signifies that it wants your identification.

The standard identification code is in the form of project numbers and programmer numbers, but individual installations may have different codes. The numbers, or whatever code each installation uses, are assigned to each user by the installation. The LOGIN program waits for you to type in your project number and your programmer number, separated by a comma and terminated with a carriage-return, following the number sign.

```
JOB 17    5S0218A    TTY34
#27,400)
```

An alternate method of typing in your project number and programmer number is to type your identification on the same line as the LOGIN command and to follow it with a carriage return. The system responds with the information message, and the LOGIN program does not type out the number sign. For example,

```
.LOGIN 27,400)
JOB 17    5S0218A    TTY34
```

The LOGIN program needs one more item to complete its analysis of your identification. This it requests in the next line by asking for your password.

```
JOB 17    5S0218A    TTY34
#27,400 )
PASSWORD:          )
```

Type in your password, which is also assigned by the installation, followed by a carriage-return. To maintain password security, the LOGIN program does not print the password.

If the identification typed in matches the identification stored in the accounting file in the monitor, the LOGIN program signifies its acceptance by responding with the time, date, day of the week, the message of the day (if any), and a period.

```
.LOGIN )
JOB 17    5S0218A    TTY34
#27,400 )
PASSWORD:          )
1050    4-MAY-71    WED
TYPE SYS:SCHED FOR SYSTEM
SCHEDULE
.
```

```
.LOGIN 27,400 )
JOB 17    5S0218A    TTY34
PASSWORD:          )
1050    4-MAY-71    WED
TYPE SYS:SCHED FOR SYSTEM
SCHEDULE
.
```

This typeout indicates that the LOGIN program has exited and returned control to the monitor. You have successfully logged in and may now have the monitor call in other programs for you. If the identification typed in does not match the identification in the accounting file, the monitor types out the error message

```
?INVALID ENTRY-TRY AGAIN
#
```

If this error message occurs, type in the correct project-programmer numbers and password.

2.0 FILES

When you want to run a program, first type in the program and decide on a name for it. The program is stored on the disk with the specified name. Then translate the program by calling in a translator and giving it the name of the program you wish to translate.

A program, or data, is stored on the disk in files. If a program is being typed in to a text editor (for example, TECO), the editor is busy accepting the characters being typed in and generating a disk file for them. Then, when the program is to be translated, the translator reads this file just created and generates a relocatable binary file. Since you may have many files and the other users on the computer may have files, there must be a method for keeping all of these files separate. This is accomplished by

giving each user a unique area on the disk. This area is identified by your project and programmer numbers. For example, if your project and programmer numbers are 27,400, you have a disk area by that name. Each file you create goes to your disk area and must be uniquely named.

Files are named with a certain convention, the same as a person is named. The first name, the filename, is the actual name of the file, and the last name, the filename extension, indicates what group the file is associated with. The filename and the filename extension are separated by a period.

Filenames are from one to six letters or digits. All letters or digits after the sixth are ignored. The filename extension is from one to three letters or digits. It is generally used to indicate the type of information in the file. The following are examples of standard filename extensions.

.TMP	Temporary file
.MAC	Source file in MACRO language
.F4	Source file in FORTRAN IV language
.BAS	Source file in BASIC language
.ALG	Source file in ALGOL language
.CBL	Source file in COBOL language
.REL	Relocatable binary file
.SAV	A saved core image

Since files are identified by the complete name and the project and programmer numbers, two users may use the same filename as long as they have different project and programmer numbers; the files would be distinct and separate. The following are examples of filenames with filename extensions.

MAIN.F4	A FORTRAN file named MAIN
SAMPLE.BAS	A BASIC file named SAMPLE
TEST1.TMP	A temporary file named TEST1
NAME.REL	A relocatable binary file named NAME

3.0 CREATING FILES

The two commands mentioned in this section use two editors to create a new disk file. One of the editors is LINED, a line-oriented editor, and the other is TECO, the Text Editor and Corrector (refer to the LINED and TECO documents in the DECsystem-10 Software Notebooks). Each command requires a filename as its argument and should have a filename extension. A new file may be created with either of these commands, depending on the editor desired.

3.1 The CREATE Command

The CREATE command is used only to create a new disk file. When this command is executed, the monitor calls in LINED to initialize a disk file with the specified name and to accept input from the terminal. At this point, begin to type in your program, line by line. LINED types a line number at the beginning of each line so that later a reference to a given line may be made in order to make corrections. Below is a sample program using the commands discussed so far.

↑C	Establish communication with the monitor. Type C while depressing the CTRL key.
.LOGIN 27,400)	Begin the login procedure and type in your identification.
JOB 17 5S0218A TTY34	The job number assigned, followed by the monitor name and version and the terminal line number. The LOGIN program requests identification (project number and programmer number) if it was not typed on the same line as the LOGIN command.
PASSWORD:)	The LOGIN program requests password. Type it in; it is not printed.
1050 4-MAY-71 WED TYPE SYS:SCHED FOR SYSTEM SCHEDULE .	If identification matches identification stored in the system, the monitor responds with the time, date, day of the week, message of the day, and a period.
CREATE MAIN.F4)	A new file on the disk is to be created and called MAIN.F4. The extension .F4 is used because the program is to be a FORTRAN source file. LINED is called in to create the file.
*	Response from LINED signifying it is ready to accept commands.
I)	A command to LINED to insert line numbers starting with 10 and incrementing by 10 (refer to the LINED document).
00010 TYPE 53)	Type in your FORTRAN PROGRAM.
00020 53 FORMAT (' THIS IS MY PROGRAM'))	
00030 END)	
00040 $	The ($) (altmode) is a command to LINED to end the insert. On the terminal this key is labeled ALT, ESC, or PREFIX.
*	Response from LINED signifying it is ready to accept another command.
E)	A command to LINED to end the creation of the file.

*_	Response from LINED indicating readiness to accept a command.
↑C	Return to the monitor.
._	The monitor now has control of the program.

The three LINED commands (I, (Ⓢ), E) shown in the examples are fully discussed in the LINED document.

3.2 The MAKE Command

This command can also be used to open a new disk file for creation. It differs from the CREATE command in that TECO is used instead of LINED. (TECO is discussed in the DECsystem-10 Software Notebooks). Otherwise, the CREATE and MAKE commands operate in the same manner.

```
._MAKE    FILEA.F4 )
*_I (Text input)      $$
EX$$
EXIT

._
```

The altmode ($) and the EX command are commands to TECO and are explained in the TECO document.

4.0 EDITING FILES

After creating a text file, you may wish to modify, or edit, it. The following two commands cause an existing file to be opened for changes. One command (EDIT) calls in LINED, and the other (TECO) calls in TECO. In general, the editor used to create the file should be used for editing. Each command requires, as its argument, the same filename and filename extension used to create the file.

4.1 The EDIT Command

The EDIT command causes LINED to be called in and, as the name implies, signifies that you wish to edit the specified file. LINED responds with an asterisk and waits for input. The file specified must be an already existing sequence-numbered file on the disk. For example, in Paragraph 3.1, the file MAIN.F4 was created. If the command

```
._EDIT MAIM.F4 )
```

is given to edit the file, the computer responds with an error message (assuming that there was no file named MAIM.F4). The command

```
.EDIT MAIN.F4
```

causes the right file to be opened for editing.

4.2 The TECO Command

The TECO command is similar to the EDIT command except that it causes the TECO program to open an already existing file on the disk for editing purposes. The command sequence

```
.TECO    FILEA.F4
*(editing)      $$
*EX$$
```

causes TECO to open FILEA.F4 for editing and close the file upon completion, creating a backup file out of the original file. Whenever one of the commands used to create or edit a file is executed, this command with its arguments (filename and filename extension) is "remembered" in a temporary file on the disk. Because of this, the file last edited may be recalled for the next edit without having the filename specified again. For example, if the command

```
.CREATE PROG1.MAC
```

is executed, then you may type the command

```
.EDIT
```

instead of

```
.EDIT  PROG1.MAC
```

assuming that no other CREATE, TECO, MAKE, or EDIT command that changed the filename was used in-between. As mentioned before, if a command tries to edit a file that has not been created, an error message is given.

5.0 MANIPULATING FILES

You may have many files saved on your disk area. (For discussion on how to save a file on your disk area, refer to Paragraph 14.1.) The list of your files, along with lists of other users' files, is kept on the disk in what are called user directories. Suppose you cannot remember if you have created and saved a particular file. The next command helps in just that type of situation.

5.1 The DIRECT Command

The DIRECT command requests from the monitor a listing of the directory of your disk area. The monitor responds by typing on the terminal the names of your files, the length of each file in the number of DECsystem-10 disk blocks written (a block is 128_{10} words), and the date on which each file was created. The protection associated with each file is also output. This protection is a code that indicates which users are allowed to access your files. It is automatically assigned when you create the file. Refer to DECsystem-10 Monitor Calls (DEC-10-MRRB-D) for an explanation of file protection.

Names of files not explicitly created by you may show up in the directory. These files were created as intermediate files for storage by programs you may have used. For example, in translating a file, the translator generates a file with the same filename but with a filename extension of .REL. This file contains the relocatable binary translation of the source file. You may also notice filenames with the filename extension of .TMP. This extension signifies a temporary file created and used by different system programs.

5.2 The TYPE Command

By listing your directory on the terminal, you know the names of the files on your disk area. But what if you have forgotten the information contained in a particular file? The TYPE command causes the contents of source files specified in your command string to be typed on your terminal. For example, the command

```
.TYPE  MAIN.F4 )
```

causes the file MAIN.F4 to be typed on the terminal. Multiple files separated by commas may be specified in one command string, and only source files, not binary files, may be listed.

This command allows the "asterisk construction" to be used. This means that the filename or the filename extension may be replaced with an asterisk to mean any filename or filename extension. For example, the command

```
.TYPE  FILEB.* )
```

causes all files named FILEB, regardless of filename extensions, to be typed. The command

```
.TYPE  *.MAC )
```

causes all files with the filename extension of .MAC to be typed. The command

```
.TYPE *.* )
```

causes all files to be typed.

5.3 The DELETE Command

Having finished with a file, you may erase it from your disk area with the DELETE command. Multiple files may be deleted in one command string by separating the files with commas. For example,

```
.DELETE LINEAR )
```

and

```
.DELETE CHANGE.F4, SINE.REL )
```

are both legal commands. The asterisk convention discussed in section 5.2 may also be used with the DELETE command.

5.4 The RENAME Command

The names of one or more files on your disk area may be changed with the RENAME command. The old filename on the right and the new filename on the left are separated by an equal (=) sign. In renaming more than one file, each pair of filenames (new=old) is separated by commas. For example, the command

```
.RENAME SALES.CBL=GROSS.CBL,FILE2.F4=FILE1.F4 )
```

changes the name of file GROSS.CBL to SALES.CBL and file FILE1.F4 to FILE2.F4. The old filename no longer appears in your directory; instead the new filenames appear containing exactly the same data as in the old files. The asterisk convention may again be used. For example, the command

```
.RENAME  *.F4=* )
```

causes all files with no filename extension to have the extension .F4.

6.0 TRANSLATING, LOADING, EXECUTING, DEBUGGING PROGRAMS

As this point you know how to get on the system, how to create and edit a source file of a program, and how to list your source file on the terminal. The program has not been executed. This only happens after it has been translated into the binary machine language understandable to the computer and loaded into core memory. More often than not the program must be debugged.

6.1 The COMPILE Command

This command has as its argument one or more filenames separated by commas. It causes each command to be processed (translated) if necessary by the appropriate processor (translator). It is considered necessary to process a file if no .REL file of the source file exists, or if the .REL file was created

before the last time the source file was edited. If the .REL file is up-to-date, no translation is done. The appropriate processor is determined by examining the extension of the file. The following shows which processor is used for various extensions.

.MAC	MACRO assembler
.F4	FORTRAN IV compiler
.ALG	ALGOL compiler
.CBL	COBOL compiler
.REL	No processing is done
other than above, or null	"Standard processor"

The standard processor is used to translate programs with null or nonstandard extensions. The standard processor is FORTRAN at the beginning of the command string, but may be changed by use of various switches (refer to DECsystem-10 Operating System Commands). Although it is not necessary to indicate the extension of a file in the COMPILE command string, the standard processor can be disregarded if all source files are kept with the appropriate extension.

When the appropriate translator has translated the source file, there is a file on your disk area with the filename extension .REL and the same filename as the source file. This file is where the translator stores the results of its translation and is called the relocatable binary of the program. The program is now translated into binary machine language, but is still on the disk. Since the disk is used for storage and not for execution, a copy of the binary program must be loaded into core memory to form a core image. The core memory of the computer is used for execution; it is like a scratch pad. The COMPILE command does not generate a core image, but the following three commands do.

6.2 The LOAD Command

The LOAD command performs the same operations as the COMPILE command and in addition causes the LOADER to be run. The LOADER is a program that takes the specified REL files, links them together, and generates a core image. The LOAD command does not cause execution of the program.

6.3 The EXECUTE Command

This command performs the functions of the LOAD command and also begins execution of the loaded programs, if no translation or loading errors are detected. The compiled program is now in core memory and running, and what happens next depends on the program. More than likely, the program is not returning the correct answers, and you now enter the magic world of program debugging.

6.4 The DEBUG Command

This command prepares for the debugging of a program in addition to performing the functions of the COMPILE and LOAD commands. DDT, the Dynamic Debugging Technique program (refer to the DDT manual, DEC-10-CDDE-D), is loaded into core memory first, followed by the program. Upon completion of loading, DDT is started rather than the program. A command to DDT may then be issued to begin the program execution. This command should be used by the experienced programmer familiar with DDT. The above four commands have extended command forms discussed in DECsystem-10 Operating System Commands.

The following is an example showing the compilation and execution of a FORTRAN main program and subroutine. The login procedure is not shown.

```
.CREATE MAIN.F4 )                          CREATE a disk file.
*I )                                       Command to LINED to begin inserting on
                                           line 10, incrementing by 10.
00010      TYPE 69 )                       Statements of the FORTRAN main program.
00020 69   FORMAT (' THIS IS THE MAIN PROGRAM') )
00030      CALL SUB1 )
00040      END )
00050      $                               Altmode ends the insert.
*E )                                       LINED command to end the edit.
*↑C                                        Return to the monitor.
.CREATE PROG.F4 )                          Create a disk file for the subroutine.
*I )                                       Begin inserting at line 10 incrementing by
                                           10.
00010        SUBROUTINE SUBR )             Statements of the FORTRAN Subroutine.
00020        TYPE 105 )
00030  105   FORMAT (' THIS IS THE SUBROUTINE') )
00040        RETURN )
00050        $                             Altmode ends the insert.
*E )                                       LINED command to end the edit.
*↑C                                        Return to monitor.
.EXECUTE MAIN.F4,PROG.F4 )                 Request execution of the programs created.
FORTRAN:    MAIN.F4                        FORTRAN reports its progress.
FORTRAN:    PROG.F4
LOADING
```

```
000001 UNDEFINED GLOBALS
SUB1     000152                    There is no subroutine named SUB1.
?
LOADER   3K   CORE                 This includes the space for the loader.
?EXECUTION DELETED                 No execution was done.
EXIT

.EDIT )                            Ask to edit PROG.F4, filename need not be
                                   mentioned since it was the last file named.
*P10,20 )                          Type lines 00010 and 00020 on the terminal.
00010    SUBROUTINE SUBR
00020    TYPE 105
*I10 )                             Insert a new line 10.
00010'   SUBROUTINE SUB1 )
00020'   $                         Terminate the insert.
*E )                               End the edit.
*↑C
.EXECUTE MAIN.F4,PROG.F4 )         Request execution.
FORTRAN:   PROG.F4                 Only the subroutine is recompiled since
                                   only it has been edited.
LOADING                            Both MAIN and PROG are loaded.
LOADER 3K CORE
EXECUTION
THIS IS THE MAIN PROGRAM           Execution begins.
THIS IS THE SUBROUTINE
CPU TIME:  0.03 SEC. ELAPSED TIME:  0.13 SEC.
NO EXECUTION ERRORS DETECTED
EXIT                               Execution ends.
.
```

7.0 GETTING INFORMATION FROM THE SYSTEM

There are several monitor commands that are used to obtain information from the system. Three commands useful at this point are discussed in this section, and additional commands are discussed in Paragraph 16.0.

7.1 The PJOB Command

If you have forgotten the job number assigned to you at LOGIN time, you may use the PJOB command to obtain it. The system responds to this command by typing out your assigned job number. For example,

```
.PJOB )
17
```

7.2 The DAYTIME Command

This command gives the date followed by the time of day. The time is presented in the following format:

hh:mm:ss

where hh represents the hours, mm represents the minutes, and ss represents the seconds. For example,

```
.DAYTIME )
17-MAY-71    14:37:35
```

7.3 The TIME Command

The TIME command produces three lines of typeout. The first line is the total running time since the last TIME command was typed. The second line is the total running time since you logged in. The third line is used for accounting purposes. The time is presented in the following format:

hh:mm.ss

where hh represents the hours, mm the minutes, and ss the seconds to the nearest hundreth. For example,

```
.TIME )
52.45
02:29.95
KILO-CORE-SEC=57
```

In the first two lines, you are told that you have been running 52.45 seconds since the last time you typed the TIME command, and a total of 2 minutes and 29.95 seconds since you logged in. The third line of typeout is used by your installation for accounting and is the integrated product of running time and core size. Refer to DECsystem-10 Operating System Commands.

8.0 LEAVING THE SYSTEM

Now that you know how to log into the system and create and run a program, you might be wondering how you leave the system. You have to tell the system you are leaving, and you do this by the KJOB command.

8.1 The KJOB Command

The KJOB command is your way of saying goodbye to the system. Many things happen when you type the command. The job number assigned to you is released and your terminal is now free for another user. An automatic TIME command is performed. In addition, if you have any files on your disk area, the monitor responds with

CONFIRM:

and you have several options available to you. By typing H and a carriage return after the CONFIRM: message, the monitor lists the options available. For example, the following typeout occurs by responding to the CONFIRM: message with H and a carriage return.

```
IN RESPONSE TO CONFIRM:,TYPE ONE OF: BDFHIKLPQSUWX
B TO PERFORM ALGORITHM TO GET BELOW LOGGED OUT QUOTA
D TO DELETE ALL FILES
(ASKS  ARE YOU SURE?, TYPE Y OR CR)
F TO TRY TO LOGOUT FAST BY LEAVING ALL FILES ON DSK
H TO TYPE THIS TEXT
I TO INDIVIDUALLY DETERMINE WHAT TO DO WITH ALL FILES
   AFTER EACH FILE NAME IS TYPED OUT, TYPE ONE OF: EKPQS
  E TO SKIP TO NEXT FILE STRUCTURE AND SAVE THIS FILE IF
   BELOW LOGGED OUT QUOTA ON THIS FILE STRUCTURE
  K TO DELETE THE FILE
  P TO PRESERVE THE FILE
  Q TO REPORT IF STILL OVER LOGGED OUT QUOTA, THEN REPEAT FILE
  S TO SAVE THE FILE WITH PRESENT PROTECTION
K TO DELETE ALL UNPRESERVED FILES
L TO LIST ALL FILES
P TO PRESERVE ALL EXCEPT TEMP FILES
Q TO REPORT IF OVER LOGGED OUT QUOTA
S TO SAVE ALL EXCEPT TEMP FILES
U SAME AS I BUT AUTOMATICALLY PRESERVE FILES ALREADY PRESERVED
W TO LIST FILES WHEN DELETED
X TO SUPPRESS LISTING FILES WHEN DELETED

IF A LETTER IS FOLLOWED BY A SPACE AND A LIST OF FILE STRUCTURES
  ONLY THOSE SPECIFIED WILL BE AFFECTED BY THE COMMAND.  ALSO
  CONFIRM WILL BE TYPED AGAIN.

NOTE:  FILE SIZE IS NO. OF BLOCKS ALLOCATED WHICH MAY BE LARGER THAN THE
  NO. OF BLOCKS WRITTEN (DIRECTORY COMMAND).

A FILE IS PRESERVED IF ITS ACCESS CODE IS GE 100

CONFIRM:
```

You may now use the options available. If K was used as the option, the following is a sample of what is output to your terminal.

```
JOB 33, USER [27,560] LOGGED OFF TTY34   1317 20-MAY-71
DELETED ALL 2 FILES (3. DISK BLOCKS)
RUNTIME 0 MIN, 00.29 SEC
```

Remember that the CONFIRM message is typed only if there are files on your disk area. If there are no files on your disk area, the typeout would look like the following:

```
.KJOB )
JOB 17, USER [27,3201] LOGGED OFF TTY17   1317 20-MAY-71
RUNTIME 0 MIN, 00.29 SEC
```

9.0 HOW TO LIVE WITH THE TERMINAL

On the terminal, there is a special key marked CTRL called the Control Key. If this key is held down and a character key is depressed, the terminal types what is known as a control character rather than the character printed on the key. In this way, more characters can be used than there are keys on the keyboard. Most of the control characters do not print on the terminal, but cause special functions to occur, as described in the following sections.

There are several other special keys that are recognized by the system. The system constantly monitors the typed characters and, most of the time, sends the characters to the program being executed. The important characters not passed to the program are also explained in the following sections. (Refer to DECsystem-10 Monitor Calls for more explanations of special characters.)

9.1 Control - C

Control - C (↑C) interrupts the program that is currently running and takes you back to the monitor. The monitor responds to a control - C by typing a period on your terminal, and you may then type another monitor command. For example, suppose you are running a program in BASIC, and you now decide you want to leave BASIC and run a program in AID. When BASIC requests input from your terminal by typing an asterisk, type control - C to terminate BASIC and return to the monitor. You may now issue a command to the monitor to initialize AID (.R AID). If the program is not requesting input from your terminal (i.e., the program is in the middle of execution) when you type control - C, the program is not stopped immediately. In this case, type control - C twice in a row to stop the execution of the program and return control to the monitor. If you wish to continue at the same place that the program was interrupted, type the monitor command CONTINUE. As an example, suppose you want the computer to add a million numbers and to print the square root of the sum. Since you are charged by the amount of processing time your program uses, you want to make sure your program does

not take an unreasonable amount of processing time to run. Therefore, after the computer has begun execution of your program, type control - C twice to interrupt your program. You are now communicating with the monitor and may issue the monitor command TIME to find out how long your program has been running. If you wish to continue your program, type CONTINUE and the computer begins where it was interrupted.

9.2 The RETURN Key

This key causes two operations to be performed: (1) a carriage-return and (2) an automatic line-feed. This means that the typing element returns to the beginning of the line (carriage-return) and that the paper is advanced one line (line-feed). Commands to the monitor are terminated by depressing this key.

9.3 The RUBOUT Key

The RUBOUT key permits correction of typing errors. Depressing this key once causes the last character typed to be deleted. Depressing the key n times causes the last n characters typed to be deleted. RUBOUT does not delete characters beyond the previous carriage-return, line-feed, or altmode. Nor does RUBOUT function if the program has already processed the characters you wish to delete.

The monitor types the deleted characters, delimited by backslashes. For example, if you were typing CREATE and go as far as CRAT, you can correct the error by typing two RUBOUTS and then the correct letters. The typeout would be

CRAT\TA\EATE

Notice that you typed only two RUBOUTS, but \TA\ was printed. This shows the deleted characters, but in reverse order. (Note that when using TECO, deleted characters are not enclosed in backslashes.)

9.4 Control - U

Control - U (↑U) is used if you have completely mistyped the current line and wish to start over again. Once you type a carriage-return, the command is read by the computer, and line-editing features can no longer be used on that line. Control - U causes the deletion of the entire line, back to the last carriage-return, line-feed, or altmode. The system responds with a carriage-return, line-feed so you may start again.

9.5 The ALTMODE Key

The ALTMODE key, which is labeled ALTMODE, ESC, or PREFIX, is used as a command terminator for several programs, including TECO and LINED. Since the ALTMODE is a nonprinting character, the terminal prints an ALTMODE as a dollar sign ($).

9.6 Control - O

Control - O (↑O) tells the computer to suppress terminal output. For example, if you issue a command to type out 100 lines of text and then decide that you do not want the typeout, type control - O to stop the output. Another command may then be typed as if the typeout had terminated normally.

10.0 PERIPHERAL DEVICES

The system controls many peripheral devices, such as terminals, magnetic tape drives, DECtape drives, card readers and punches, line printers, papertape readers and punches, and disks. The monitor is responsible both for allocating these peripheral devices, as well as other system resources (e.g., core memory), and for maintaining a pool of such available resources from which you can draw.

Each device controlled by the system has a physical name associated with it. The physical name is unique. It consists of three letters and zero to three numerals specifying a unit number. The following table lists the physical names associated with various peripheral devices.

Table 1
Peripheral Devices

Device	Physical Name
Terminal	TTY0, TTY1, ..., TTY77
Console TTY	CTY
Paper Tape Reader	PTR
Paper Tape Punch	PTP
Plotter	PLT
Line Printer	LPT
Card Reader	CDR
Card Punch	CDP
DECtape	DTA0, DTA1, ..., DTA7
Magnetic Tape	MTA0, MTA1, ..., MTA7
Disk	DSK
Display	DIS

You may also give each device a logical device name. The logical device name is an alias, and the device can be referred to either by this alias or by the physical name. The logical name consists of one to six alphanumeric characters of your choice. The reason for logical device names is that in writing a program you may use arbitrarily selected device names (logical device names) that can be assigned to the most convenient physical devices at runtime. However, care should be exercised in assigning logical device names because these names have priority over physical device names. For example, if a DECtape is assigned the logical name DSK, then all of your programs attempting to use the disk via the physical name DSK end up using the DECtape instead. It is wise not to give any device the logical name DSK because certain monitor commands (such as the COMPILE commands) make extensive use of special features that the disk has but other devices do not have. The following examples show the use of logical and physical device names.

.ASSIGN DTA ABC)	Assign a DECtape the logical name ABC.
.ASSIGN MTA1 XYZ)	Assign magnetic tape drive #1 the logical name XYZ.
.ASSIGN PTR FOO)	Assign the papertape reader the logical name FOO.

In order to use most peripheral devices, you must assign the desired device to your job. You may assign a device either by a program or from the console. The first kind of assignment occurs when you write a program that uses a particular device. When the program begins using the device, it is assigned to you on a temporary basis and released from you when your program has finished with it. The second kind of assignment occurs when you explicitly assign the device by means of the ASSIGN or MOUNT monitor command. This is a permanent assignment that is in effect until the device is released by a DEASSIGN, DISMOUNT, or FINISH monitor command or by your logging off the system.

When you assign a device to your job, the monitor associates your job number with that device. This means that no other user may use the device while you are using it. The only exception is the disk, which is accessible by all users. When you assign the disk, you are allocated a fraction of the disk, not the entire unit. When you deassign a device or kill your job, the device is returned to the monitor's pool of available resources.

Under normal circumstances, the spooling mechanism built into the system is used to output to slow-speed devices. Spooling is the method by which output to these devices (usually the line printer, card punch, paper tape punch, and plotter) is placed on the disk first and then output to the device at a later time. This method of using a device saves you time because you do not have to wait for the device to be freed if it is being used by another user nor do you have to wait for your files to be output before you

can perform another operation. Once your files have been placed on the disk, you can do another task, such as running a program or leaving the system by killing your job. After you leave the system (KJOB), your files will be output whenever the device you requested to output them is available. The spooling of files to the line printer is described in Paragraph 13.0. Refer to the DECsystem-10 Operating System Commands manual for a discussion of spooling to other devices.

11.0 COMMANDS TO ALLOCATE SYSTEM RESOURCES

11.1 The ASSIGN Command

The ASSIGN command is used to assign a peripheral device on a permanent basis for the duration of your job or until you explicitly deassign it. This command must have as an argument the legal physical device name (see Table 1) of the device you wish to assign. For example, if you want to assign a DECtape drive to your job, type

```
.ASSIGN DTA)
```

The monitor responds with the message

```
DTAn ASSIGNED
```

where n is the unit number of the DECtape drive assigned to your job. If all drives are in use, the monitor responds with

```
ASSIGNED TO JOBS N1, N2, ...
```

and you must wait until a drive becomes available. You may assign a specific DECtape drive as follows:

```
.ASSIGN DTA3)
```

The monitor responds with

```
DTA3 ASSIGNED
```

if the drive is available, or

```
ALREADY ASSIGNED TO JOB n
```

if job n is using DECtape drive #3.

The ASSIGN command may also have, as an optional argument, a logical device name following the physical device name. The logical device name may be used in place of the physical device name in all references to the device. For example, if you want to use DECtape drive #1 and have it named SAMPLE, type the command

```
.ASSIGN DTA1 SAMPLE )
```

If DECtape drive #1 is free, the monitor responds with

```
DTA1 ASSIGNED
```

and stores the logical name you typed. You may then refer to the DECtape by the name SAMPLE until you explicitly release the device, assign the name SAMPLE to another device, or kill your job.

Logical names can be very useful. Suppose you write a program that uses DECtape drive #5 and refers to it by its physical name (DTA5). When you run your program, you find that DECtape drive #3 is the only drive available. Instead of rewriting your program to use DECtape drive #3, type

```
.ASSIGN DTA3 DTA5 )
```

Thereafter, whenever your program refers to DTA5, it is actually referring to DTA3. Since logical device names are strictly your own, they are different from the logical names of other users. The following is an example using physical and logical device names.

.ASSIGN DTA NAME)	Assign a DECtape drive the logical name NAME.
DEVICE DTA4 ASSIGNED	DECtape drive #4 has been assigned.
.ASSIGN DTA LINE)	Find another DECtape drive; assign the logical name LINE.
ASSIGNED TO JOBS N_1, N_2, ...	All DECtape drives are in use.
.ASSIGN PTP,NAME) %LOGICAL NAME WAS IN USE, DEVICE PTP ASSIGNED	Reserve paper tape punch. Paper tape punch is assigned and NAME now refers to PTP.
.ASSIGN DTA3 LINE)	Request DECtape drive #3 and give it the logical name LINE.
ALREADY ASSIGNED TO JOB7	Another user (job 7) has DTA3.

11.2 The MOUNT Command

The MOUNT command is similar to the ASSIGN command in that it is used to assign a peripheral device to your job. However, unlike the ASSIGN command, it requests operator intervention. This is useful for users who cannot place their devices on the computer because they are too far away. These users are called remote users because they are connected to the computer via communications lines. For example, if you have DECtapes at the location of the computer (commonly called the central site) but are using the computer remotely, you can use the MOUNT command to assign a DECtape drive and to have the operator place the DECtape on the drive.

This command must have as an argument the legal physical device name (see Table 1) of the device you wish to assign and may have a logical device name. These arguments are the same as in the ASSIGN command. In addition, switches can be used to specify items to be considered by the operator. Only the following three switches are applicable in this manual; the remainder are described in DECsystem-10 Operating System Commands

/RONLY or /WLOCK	Specifies that the volume is read only and that it cannot be written on.

/VID:name	Specifies the name used to identify the volume (storage medium) to the operator. The name can be in one of two forms: 1) any string of 25 characters or less containing only letters, digits, periods, and hyphens or 2) any string of 25 characters or less enclosed in single quotes. The string cannot contain break characters or single quotes.
/WENABL	Specifies that the volume is enabled for writing. This condition is assumed if no switches appear in the MOUNT command string.

11.3 The DEASSIGN Command

The DEASSIGN command is used to release one or more devices currently associated with your job. This command may have as an argument a physical or logical device name. If an argument is given, the specified devices are released. If an argument is not specified, all devices assigned to your job are released. When devices are released, they are returned to the monitor's pool of available resources for use by other users. The DEASSIGN command does not affect any temporary assignments your job may have for devices.

11.4 The DISMOUNT Command

The DISMOUNT command is similar to the DEASSIGN command because it is used to return devices to the monitor. In addition, it notifies the operator to remove the volume (storage medium) from the device (i.e., DECtape from a DECtape drive, cards from a card reader, and so forth). This command takes a physical device name as an argument. The device must have been previously assigned with the ASSIGN or MOUNT command. The switch /REMOVE follows the device name in order to tell the operator to physically remove the volume from the device. For example,

```
.DISMOUNT DTA4:/REMOVE )
```

notifies the operator to deassign DTA4 and remove the tape from the drive.

11.5 The REASSIGN Command

The REASSIGN command allows you to give a device assigned to you to another user without having the device returned to the monitor's pool of available resources. Two arguments are required with this command: the name of the device being reassigned and the job number of the user who is receiving the device. For example, suppose you have finished with DECtape drive #6 and the person who is job 10 wants it. Type the command

```
.REASSIGN DTA6 10 )
```

This deassigns DECtape drive #6 from your job and assigns it to job 10, just as if you had typed

```
.DEASSIGN DTA6 )
```

and job 10 had typed

```
.ASSIGN DTA6 )
```

immediately thereafter. All devices except the job's terminal can be reassigned.

11.6 The FINISH Command

The FINISH command is used to prematurely terminate a program that is being executed while preserving as much output as possible. If this command is not used, part or all of the output file may be lost. The FINISH command may be followed by a physical or logical device name, in which case any input or output currently in progress in relation to that device is terminated. If no device is specified, input or output is terminated on all devices assigned to your job. The monitor responds to this command by terminating output, closing the file, and releasing the device for use by others.

This command could be used if you were generating an assembly listing of a program on your disk area and decided that you wanted only the first part of the listing, not the entire listing. Type

```
↑C
.FINISH DSK )
```

and the monitor completes the writing of your listing and releases the disk.

11.7 The CORE Command

The CORE command allows you to modify the amount of core assigned to your job. The command is followed by a decimal number representing the total number of 1K blocks (1024 word blocks) that you want the program to have from this point on. For example, if you want the program to have 8K blocks of core, type

```
.CORE 8 )
```

and the monitor gives the program 8K blocks, if available. If you request additional core and there is none available, the monitor responds with an error message. If the CORE command is followed by the decimal number 0, your program disappears from core because you are requesting 0K blocks of core. If the decimal number following the command is omitted, the monitor types out (1) the total number of 1K blocks you have, (2) the maximum you can request, and (3) the amount of core not assigned to any user.

12.0 COMMANDS TO MANIPULATE TERMINALS

12.1 The SEND Command

The SEND command allows you to send a line of text to another terminal in the system. The command is typed followed by the number of the terminal to which you are sending the message followed by the message and a carriage return. This message is printed on the receiving terminal and is preceded by

the number of your terminal. If the receiver of the message is busy, that is, his terminal is not communicating with the monitor, you receive the message BUSY and your message is not sent. If you are sending a message to an operator, the receiving terminal is never busy.

12.2 The DETACH Command

The DETACH command causes your terminal to be disconnected from your program and released to control another job. This means that, while your program is disconnected, you may log in again, receive a new job number, and do something else. The job that was disassociated from your terminal is said to be a detached job. This means that it is not under control of any user's console. If your detached job attempts to type something to the terminal, it is stopped, for there is no terminal attached to it.

12.3 The ATTACH Command

The ATTACH command allows you to attach a console to a detached job. You must specify the number of the job to which you wish to attach. If you are the owner of the detached job, your console is immediately detached from your current job and attached to your detached job. After this command is executed, the console is in communication with the monitor. If the job you just attached to happens to be running, type CONTINUE without affecting the status of the job.

If you are not the owner of the detached job, you must also specify the project-programmer number of the owner. The project-programmer number must be enclosed in square brackets (e.g., [27,400]) for this command to work. If the job whose job number you typed is already attached to a terminal, you cannot attach and the monitor responds with

```
?TTYn ALREADY ATTACHED
```

where n is the number of the terminal attached to the job. Observe that only one terminal can be attached to a job at any time.

13.0 COMMANDS TO REQUEST LINE PRINTER OUTPUT

In Paragraph 5.2, the TYPE command for listing source files on your terminal was discussed. In addition, there are three commands that may be used to list files on the line printer via the spooling mechanism.

13.1 The PRINT Command

The PRINT command is used to list disk files on the line printer via the spooling mechanism. This command takes a filename, or many filenames separated by commas, as an argument. Switches can also be used with the PRINT command. Although many switches are available, only a few pertinent ones are mentioned below. The remainder are discussed in DECsystem-10 Operating System Commands.

/COPIES:n	Specifies the number of copies that you want of the file. This number must be less than 64. If this switch is not given, one copy is produced.

/LIMIT:n	Specifies the maximum number of pages you want printed. If this switch is not given, the maximum number is 200 pages.
/SPACING:DOUBLE /SPACING:SINGLE /SPACING:TRIPLE	Specifies that the output will be double, single, or triple spaced. If the /SPACING switch is not given, the output is single-spaced.

All files remain in your disk area except for temporary files; these files are deleted after they are printed.

13.2 The CREF Command

The CREF command is used to list a certain type of file called a cross-reference file. This command is an invaluable aid in program debugging. If a COMPILE, LOAD EXECUTE, or DEBUG command string (refer to Paragraph 6.0) has a /CREF switch, the command string generates an expanded listing that includes (1) the original code as it appears in the file, (2) the octal values the code represents, (3) the relative locations into which the octal values go, (4) a list of all the symbols your program uses, and (5) the numbers of the lines on which each symbol appears. This is called a cross-reference listing. To print this listing file, you must call in a special cross-reference lister with the CREF command. All the cross-reference listing files you have generated since the last CREF command are printed on the line printer. The file containing the names of the cross-reference listing files is then deleted so that subsequent CREF commands will not list them again.

13.3 The DIRECT Command

When a DTAn: argument is specified with the DIRECT command, the directory of DECtape n is typed on the terminal. (Refer to Paragraph 5.1 for a discussion of the DIRECT command when no argument is specified.) For example, the command

```
.DIRECTORY DTA2:)
```

types the directory of DECtape drive #2 on the terminal.

Besides having optional device arguments, this command has several switch options. One switch option is /F. Including /F in the command string causes the short form of the directory to be listed on the terminal. The short form of the directory consists of the names of your files. (The long form of the directory also lists the creation dates and lengths of each file.) Another switch option is /L. Including /L in the command string causes the output of the directory to go to the line printer rather than to the terminal. For example, the command

```
.DIRECTORY /L )
```

lists your directory of your disk area on the line printer. The line printer is assigned to you on a temporary basis and is released when the output is finished.

14.0 COMMANDS TO MANIPULATE CORE IMAGES

By using one of the following commands, you can load core image files (refer to Paragraph 6.1 for the definition of a core image file) from disk, DECtapes, and magnetic tapes into core and then later save the core images. These files can be retrieved and controlled from the user's console. Files on disk and DECtape are called by filename, and if you have any files on magnetic tape, you must position the tape to the beginning of the file.

14.1 The SAVE Command

The SAVE command causes your current core image to be saved on the specified device with the specified filename. This command must be followed by several arguments. First, you must tell the monitor the device on which you want to save the core image. A colon must follow the device name. Second, you must give a name to the core image file. If the filename extension is not specified, the monitor designates one. You may specify the amount of core in which you want your file saved by specifying a decimal number to represent the number of 1K blocks. For example, if you want to save your core image on DECtape drive #2, give it the name SALES, and allow 12K of core for storage, type

```
.SAVE DTA2:  SALES 12 )
```

A file called SALES is created and your core image is stored in it. If you list your DECtape directory, the length of the file is slightly over 12,000 words. After you use this command, you cannot continue executing the program. The program can be restarted only from the beginning.

14.2 The RUN Command

The RUN command allows you to run programs you previously saved on the disk, DECtape, or magnetic tape. This command reads the core image file from a storage device and starts its execution. You must specify the device containing the core image file and the name of that file. The file must have been saved previously with a SAVE command. If the file is not a saved program, the monitor responds with an error message. If the core image file you want to execute is on another user's disk area, you must specify his project-programmer number, enclosed in square brackets. Again, you may specify the amount of core to be assigned to the program if different from the minimum core needed to load the program or from the core argument of the SAVE command.

14.3 The R Command

The R command is a special form of the RUN command. This command runs programs that are part of the system, rather than programs that are your own. The R command is the usual way to run a system program that does not have a direct monitor command associated with it. For example, the only way to run BASIC and AID is by the commands

```
.R BASIC )
```

and

```
.R AID )
```

A device name or a project programmer number may not be specified for this command.

14.4 The GET Command

The GET command is the same as the RUN command except that it does not start the program; it merely generates a core image and exits. The monitor types

```
JOB SETUP
```

and is ready to accept another command.

15.0 COMMANDS TO START A PROGRAM

15.1 The START Command

The START command begins execution of the program at its starting address, the location specified within the file, and is valid only if you have a core image. This command allows you to specify another starting address by typing the octal address after the command. Normally, to start a program, type

```
.START )
```

but to start a program at the specified octal location 347, type

```
.START 347 )
```

A GET command followed by a START command is equivalent to a RUN command.

15.2 The HALT (↑C) Command

Typing ↑C stops your program and takes you back to the monitor. The program "remembers" at what point it was interrupted so that it may subsequently be continued. After typing ↑C, you may type any commands that do not affect the status of your program (e.g., PJOB, DAYTIME, RESOURCES) and still be able to continue the execution of the program with a CONTINUE command. However, continuing is impossible if you issue any command that runs a new program, such as a RUN or R command.

15.3 The CONTINUE Command

If you stop your program by a HALT (↑C) command, you may resume execution from the point at which it was interrupted by typing the CONTINUE command. You may continue the program only if you exit by typing control - C. If the program exited on an error condition of some sort, the monitor does not let you continue. It types

```
CAN'T CONTINUE
```

if you try. However, you may continue your program if it has halted and given the typeout

```
HALT AT USER n
```

16.0 ADDITIONAL COMMANDS TO GET INFORMATION FROM THE SYSTEM

16.1 The RESOURCES Command

The RESOURCES command types out a list of all the available devices (except terminals) on your terminal. For example,

```
.RESOURCES )
PTY1,CDR,PTR,MTA1,CDP,PLT
```

At the time of this command, there were six devices available.

16.2 The SYSTAT Command

The SYSTAT command produces a summary of the current status of the system and may be typed without logging in. Included in the summary is a list of the jobs currently logged in, along with their project-programmer numbers, program names being run, and runtime. The following typeout is a partial example of SYSTAT output. More information is contained in this program and can be obtained by running SYSTAT.

```
STATUS OF     5S0224D  SYSTEM #2 AT  1:34:02 P.M. ON 11-MAY-71

UPTIME 5:10:56, 24% NULL TIME = 19% IDLE + 5% LOST
22 JOBS IN USE OUT OF 37.  22 LOGGED IN, 1 DETACHED

JOB     WHO      LINE#     WHAT    SIZE(K)  STATE    RUN TIME

 1     [OPR]      P0      OMOUNT   2+4      SL SW        21
 2     [OPR]      P1      OMOUNT   2+4      SL SW        22
 3     [OPR]      P2      CDRSTK   2        SL SW      1:01
 4     [OPR]      P3      BATCON   4+4      SL SW        47
 5     [OPR]      P4      LPTSPL   3+4      CB SW      5:39
 6     [OPR]      P5      PTPSPL   2+3      SL SW        41
 7     [OPR]      P6      CHKPNT   2        SL SW         5
 8     [OPR]      P7      MSCOPE   1+SPY    SL         58:15 &
 9     [OPR]      P10     TYLOST   2+5      SL SW         3
10     10,16      23      DIRECT   1+3      ↑C SW      2:31
11     [OPR]      12      SYSDPY   3+SPY    RN        45:09
12     **,**      DET     DAEMON   7+SPY    SL SW         1
13     [OPR]      CTY     OPSER    1+2      SL SW        25
14     20,574      1      DIRECT   1+3      TI SW        13
15     40,65      21      TECO     2+3      TI SW        20
16     10,566      3      BATCON   0+4      CB SW      4:17
17     11,131     11      DIRECT   1+3      ↑C SW         4
18     10,77      20      MONLOD   12+2     RN SW      4:59
19     [OPR]       2      FAILSA   10       WS           16
20     10,63       0      FD5224   23       TI SW         3
21     [SELF]     26      SYSTAT   4+SPY    RN            4
22     10,34      24      KJOB     6+4      RN            3
& MEANS LOCKED IN CORE
PNN CORRESPONDS TO TTY42+NN
```