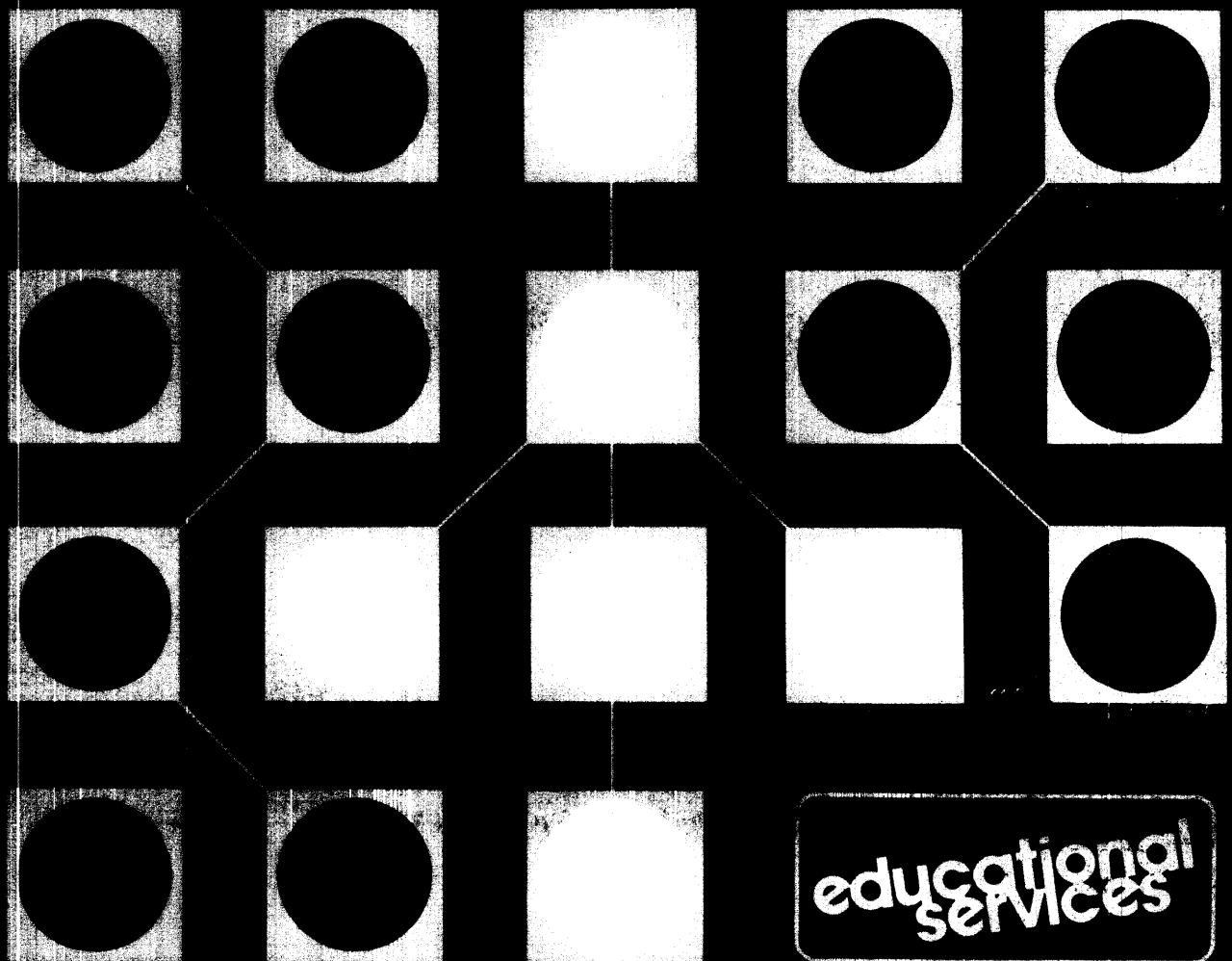


Data General

# AOS, AOS/V/S USER

self-study course

Licensed Materials



educational  
services

053-000032-00

## NOTICE

DATA GENERAL CORPORATION (DGC) HAS PREPARED THIS DOCUMENT FOR USE BY DGC PERSONNEL, LICENSEES, AND CUSTOMERS. THE INFORMATION CONTAINED HEREIN IS THE PROPERTY OF DGC AND SHALL NOT BE REPRODUCED IN WHOLE OR IN PART WITHOUT DGC PRIOR WRITTEN APPROVAL.

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

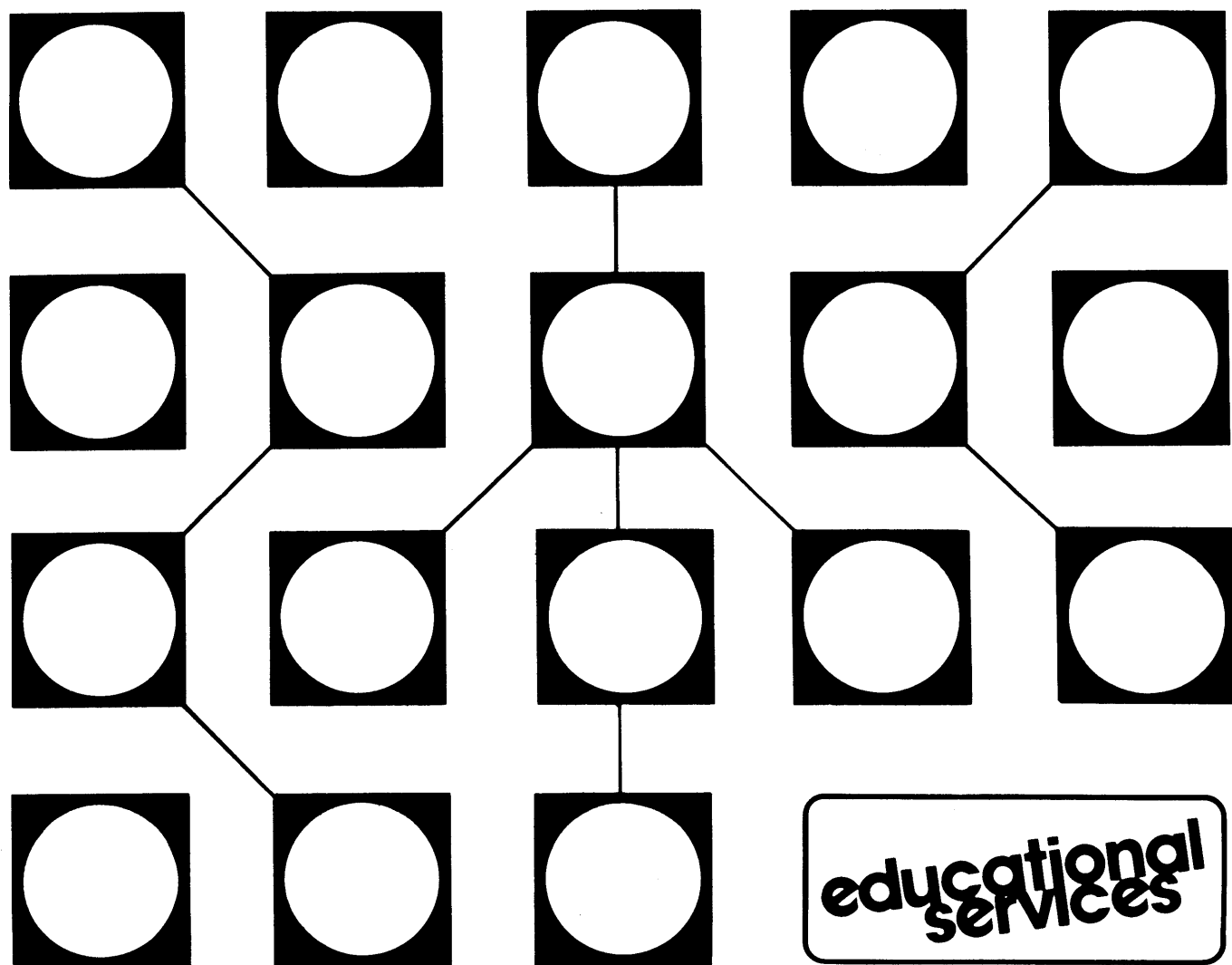
THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

CEO, DASHER, DATAPREP, ECLIPSE, ENTERPRISE, INFOS, MANAP, microNOVA, NOVA, PROXI, SUPERNOVA, ECLIPSE MV/4000, ECLIPSE MV/6000, and ECLIPSE MV/8000 are U.S. registered trademarks of Data General Corporation. AZ-TEXT, COMPUCALC, DG/L, DESKTOP GENERATION, ECLIPSE MV/10000, GW/4000, GDC/1000, GENAP, PRESENT, REV-UP, SWAT, TRENDVIEW, DEFINE, SLATE, microECLIPSE, BusiPEN, BusiGEN, BusiTEXT, and XODIAC are U.S. trademarks of Data General Corporation.

Copyright © Data General Corporation, 1982, 1984  
All Rights Reserved

# AOS, AOS/V/S USER

## self-study course





# Table of Contents

## Student Orientation

Module 1

Module 2

Module 3

Module 4

Module 5

Module 6

Module 7

Module 8

Module 9

Module 10

Module 11

Appendix A

Appendix B

Addendum

AOS, AOS/VS Principles

Gaining Access to the System

CLI Commands

Directories and Pathnames

File Security

Queues

The CLI Environment

The Process Tree

Advanced CLI Concepts

The SPEED Editor

Program Development

Module Tests Answers

AOS, AOS/VS Reference Manuals

The SED Editor



# Student Orientation

## Course Description

This course teaches you to use your AOS or AOS/VS system. You will learn to develop and execute programs. You will also learn to build and maintain files on the system.

## Prerequisites

This course is designed for students who are familiar with data processing concepts and who have some programming experience. Students can become familiar with data processing concepts by attending the Introduction to Small Computers lecture course, or by taking the *Introduction to Small Computers* Self-Study Course. Programming experience can be gained by attending the Introduction to Assembly Language lecture course. Students do not need a familiarity with Data General computers.

## Course Goals

Upon successful completion of this course, you should be able to:

1. Describe the function of AOS or AOS/VS.
2. Describe the steps involved in signing on to an AOS or AOS/VS system.
3. Given a command, write an instruction in the proper format.
4. Describe the function of a directory.
5. Describe the operation of file security.
6. Describe the operation of queues in AOS or AOS/VS.
7. Write the commands needed to change the CLI environment.
8. Write the commands needed to create a subordinate process.

9. Write commands to edit with the SPEED editor.
10. Develop and execute a program using AOS or AOS/VS.
11. Write commands using macros and pseudo-macros.

## Resources

To complete this course, you will need:

- *AOS, AOS/VS User Student Guide.*
- Audiotapes for 11 modules.
- Audiotape playback unit.
- Optional resource: Access to a functioning AOS or AOS/VS system.

### Related Publications

Appendix B of this *Student Guide* lists the Data General reference manuals that contain additional information about AOS and AOS/VS.

## Course Organization

This course is completely self-contained and is arranged in a modular, self-paced format. You can progress through the course at your own pace and in your own setting. The course contains 11 modules, each of which covers a specific topic or theme. Most modules consist of two parts:

1. An audiocassette tape.
2. A section of text, figures, and exercises contained in this *Student Guide*.

When you are instructed, listen to the audiotape as you follow along in the *Student Guide*. The *Student Guide* highlights the audiotaped lectures.

At the beginning of each module of the *Student Guide*, there is a list of Module Objectives that tell you exactly what you should learn in the module. You can evaluate your performance of these Module Objectives by completing the exercises and Module Test contained in each module. Answers are provided in Appendix A of this *Student Guide* so that you can score your own test to see how well you do. If you successfully complete the material, then you may continue with the next module; otherwise, you should restudy the module material before proceeding.

In addition, most modules contain Lab Activities that allow you to practice the module's Module Objectives. If you have a computer running AOS or AOS/VS available for your own use, then it is recommended that you do the Lab Activities using the computer. However, this is not required; the Lab Activities are designed so that you can do them without a computer.

On the next page is a Course Map. It illustrates the order in which you should progress through the course. As you can see, the modules should be studied consecutively as they appear in the *Student Guide*.



# Course Map

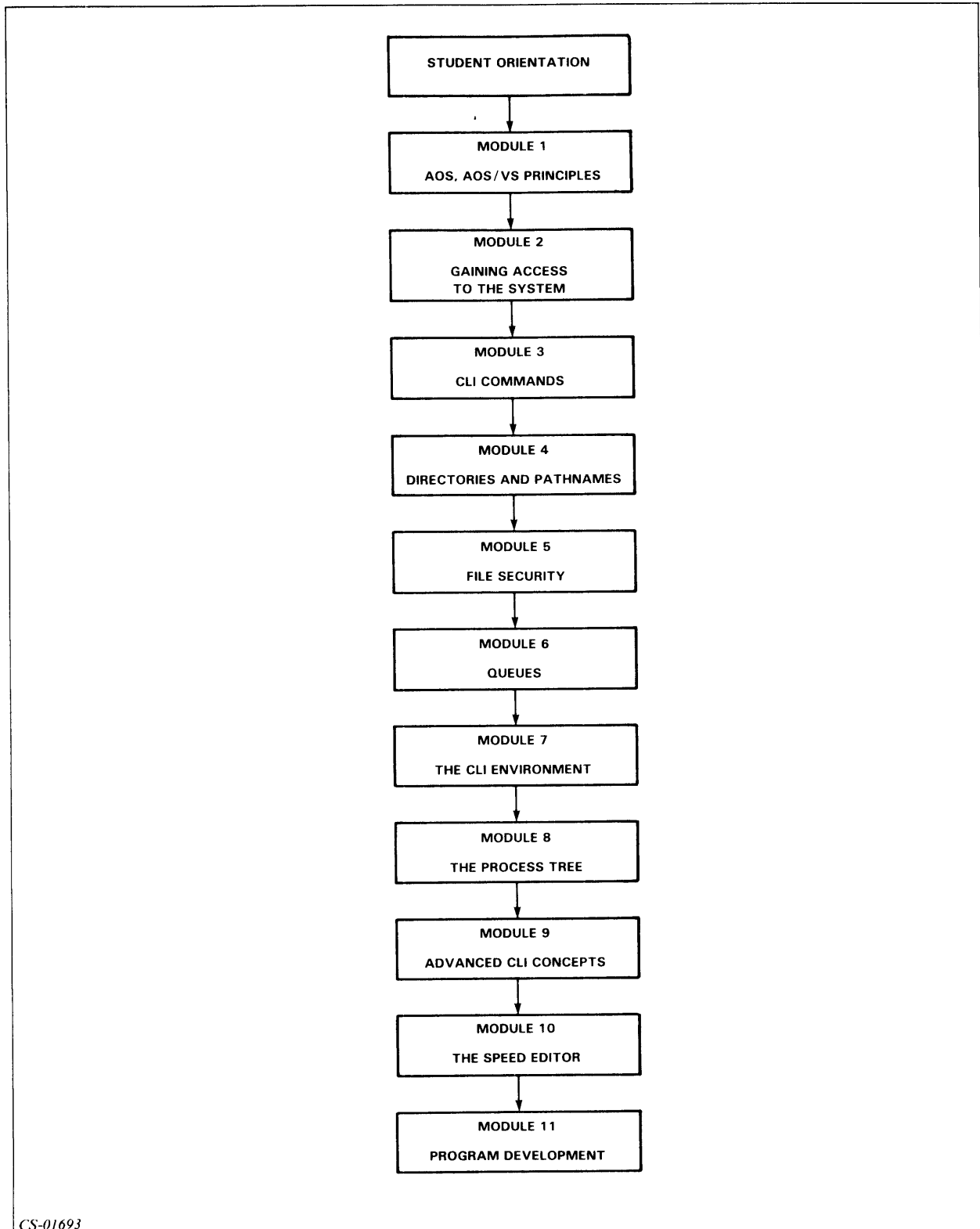


Figure 1 Course Map

## Course Duration

This is a *self-paced* instruction course. Work at your own pace; there is no specific completion time for this course. However, with diligent attention to the instructional materials, an average completion time is three to five working days.

## Typesetting Conventions

This course uses the following typesetting format to illustrate the various AOS and AOS/VS system procedures:

**THIS TYPEFACE TO SHOW YOUR ENTRY**

**THIS TYPEFACE TO SHOW SYSTEM RESPONSES**

This concludes the Student Orientation. Continue to Module 1.

# Module 1

## AOS, AOS/VS Principles

### Introduction

This module introduces you to the AOS and AOS/VS operating systems. It teaches the purpose and function of these operating systems. In addition, it introduces the Command Line Interpreter (CLI).

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Identify the function of the AOS and AOS/VS operating systems.
2. Define the Command Line Interpreter (CLI).

### Resources

To complete this module, you will need:

- Module 1 audiotape.
- Module 1 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

Module 1 discusses the following topics:

1. Purpose of AOS and AOS/VS
2. Function and features of AOS and AOS/VS
3. The CLI and its purpose

Now start the Module 1 audiotape. As you listen, follow along in Module 1 of your *Student Guide*.

## AOS and AOS/VS

Both AOS (Advanced Operating System) and AOS/VS (Advanced Operating System/Virtual Storage) are *general-purpose, disc-based* operating systems. These systems can be used as *time-sharing, batch, or real-time* control systems.

AOS and AOS/VS control and monitor processing on the computer system. They are multiprogramming systems, running more than one program at a time. Each program shares the resources of the computer with other users.

A user of the system's resources is called a *process*. AOS can have up to 64 processes running simultaneously; AOS/VS can have up to 255 simultaneous processes.

AOS and AOS/VS manage the resources of the computer system. The operating system:

- Controls input and output requests.
- Controls file processing.
- Acts as a program controller.

The resources managed by the operating system are the:

- *Processor*. Users share the Central Processing Unit (CPU).
- *Memory*. Memory is shared by swapping processes to disc.
- *Files of information*. The operating system supports four types of files.
- *Devices*. The operating system keeps track of device characteristics.

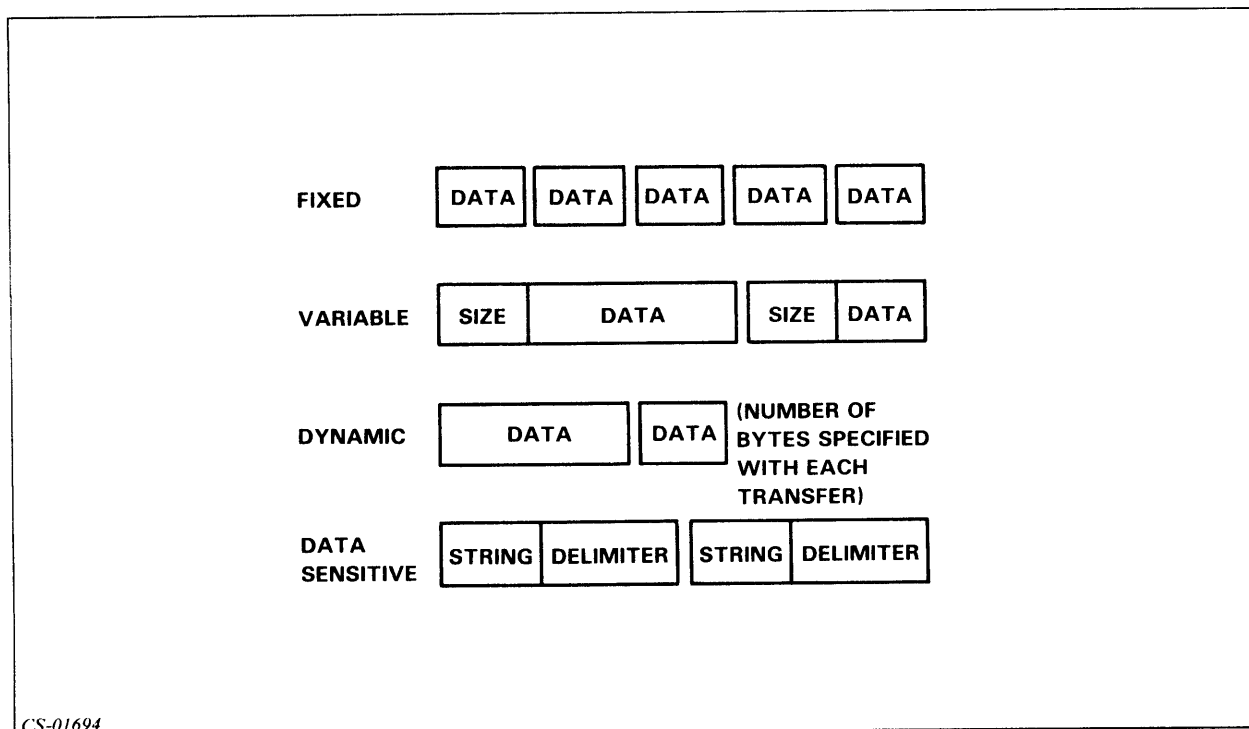


Figure 1.1 The Four Types of Data Records

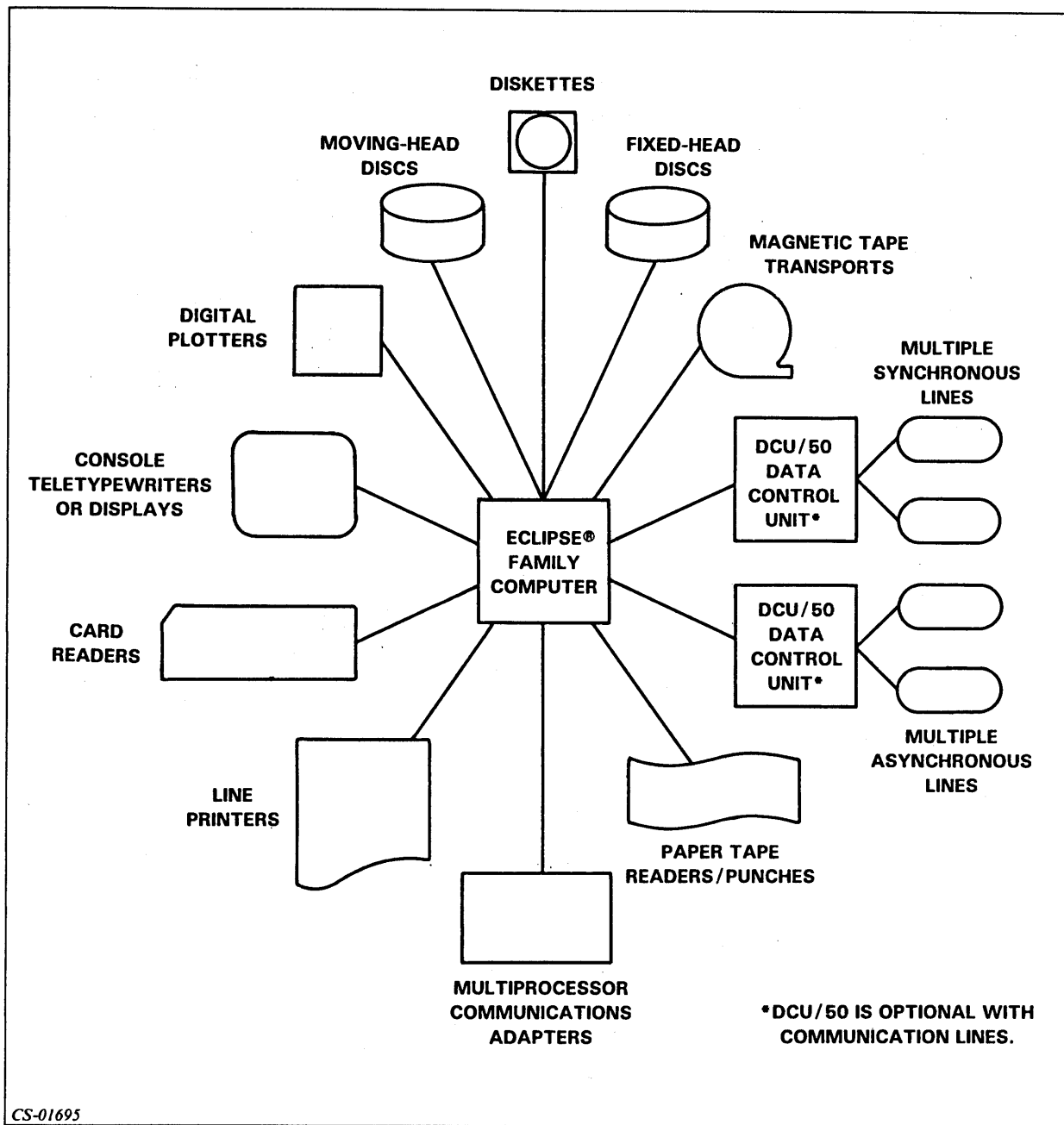


Figure 1.2 Types of Peripheral Devices

## **The Command Line Interpreter (CLI)**

The *Command Line Interpreter*, or *CLI*, is the user's primary interface with the operating system. Through the CLI, you can execute programs, control peripheral devices, and set and display system variables. There are over 90 commands available through the CLI.

To ensure system security, each user has distinct privileges that are established by the system manager. In addition, users can protect their own files from unauthorized use by determining who can have access to their files.

Now take the Module 1 Test on the next page.

## **Module 1 Test**

**Directions:** Answer the following questions by completing the sentence.

1. One function of the operating system is:
  - a. Managing the resources of the computer system.
  - b. Controlling the writing of computer programs.
  - c. Limiting the control of the computer user.
2. The Command Line Interpreter (CLI) is:
  - a. An operating system.
  - b. A user process.
  - c. An interface between the operating system and the user.
  - d. Dedicated to a specific user.
3. A process is:
  - a. Just another name for a program.
  - b. A set of procedures.
  - c. A user of system resources.
  - d. A Data General buzzword.

Now check your answers to the Module 1 Test in Appendix A. If you have answered all the questions correctly, continue to Module 2. Otherwise, go back and review the material in Module 1 and take the Module 1 Test again.

This concludes Module 1.



# Module 2

## Gaining Access to the System

### Introduction

This module introduces the procedure for gaining access to your AOS or AOS/VS system. It explains the procedures for logging on and methods for controlling your console. Finally, it teaches some basic CLI commands and the correct way to log off the system.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Identify the functions of console control characters.
2. State the procedures for logging on and off the system.
3. Write the CLI commands to display the date, time, and your process ID.
4. Identify the correct procedure for changing your password.

### Resources

To complete this module, you will need:

- Module 2 audiotape.
- Module 2 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

Module 2 discusses the following topics:

1. Logging on
  - a. Username
  - b. Password
2. Console control
  - a. Control characters
  - b. Control sequence
3. CLI commands
  - a. DATE
  - b. TIME
  - c. WHO
  - d. BYE

Now start the Module 2 audiotape. As you listen, follow along in Module 2 of your *Student Guide*.

## Logging On

To communicate with the system, you must first tell the system who you are. You are identified by a *user profile*. This profile is established by your system manager. Your user profile includes:

- Username
- Password
- Privileges
  - Access to disc space
  - Priority of execution
  - Ability to create additional processes
- Initial program
  - CLI
  - Basic Interpreter
  - A text editor

Figures 2.1 through 2.6 illustrate the steps of the log-on procedure. The position of the cursor is indicated by the white box on the screen.

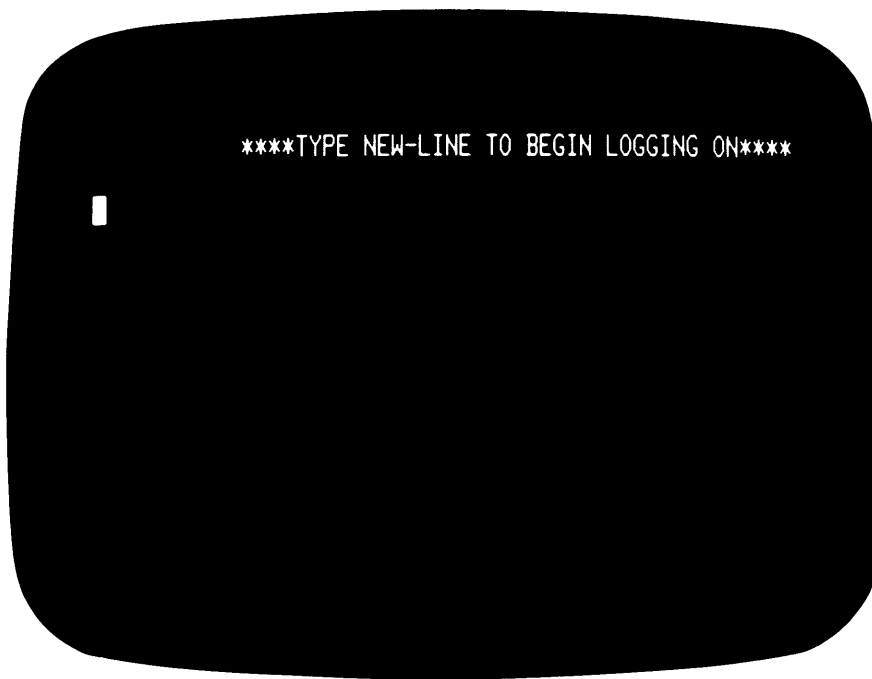


Figure 2.1 Initial Log-on Message

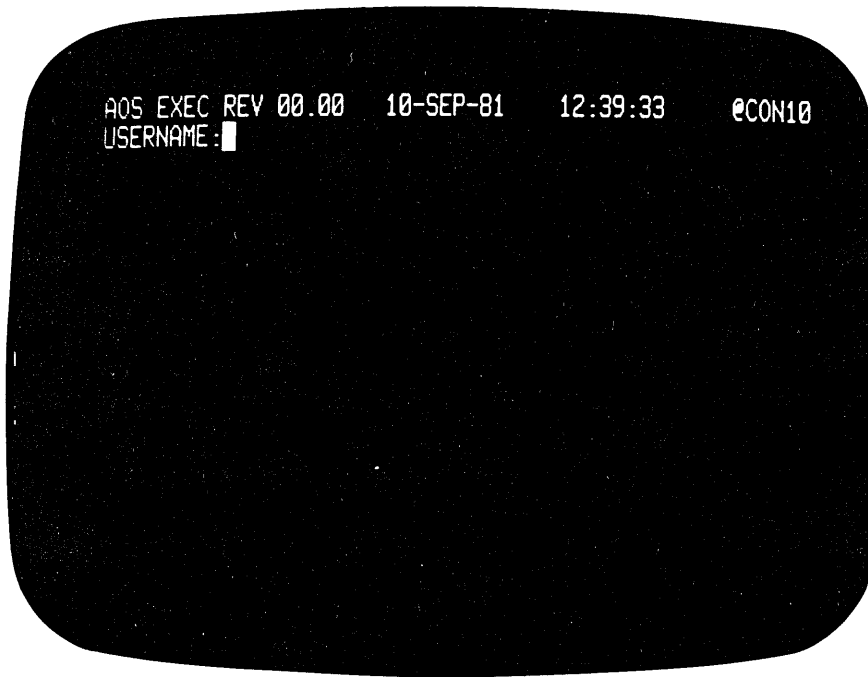


Figure 2.2 EXEC Requests Your Username.



Figure 2.3 EXEC Requests Your Password.

```
AOS EXEC REV 00 00 10-SEP-81 12:39:33 @CON10
USERNAME:MYNAME1
PASSWORD:
INVALID USERNAME-PASSWORD PAIR
USERNAME:MYNAME1
PASSWORD:
INVALID USERNAME-PASSWORD PAIR
USERNAME:MYNAME1
PASSWORD:
INVALID USERNAME-PASSWORD PAIR
USERNAME:MYNAME1
PASSWORD:
INVALID USERNAME-PASSWORD PAIR
TOO MANY ATTEMPTS, CONSOLE LOCKING FOR 10 SECONDS
█
```

Figure 2.4 Invalid Username-Password Pair

```
AOS EXEC REV 00.00 10-SEP-81 12:39:33 @CON10
USERNAME:MYNAME1
PASSWORD:
ENTER YOUR NEW PASSWORD:NEWWORD
--NEW PASSWORD IN EFFECT--
█
```

Figure 2.5 Enter CTRL-L at the End of Your Old Password.

Valid password characters are:

- A-Z
- 0-9
- \_ (underscore)
- . (period)
- ? (question mark)
- \$ (dollar sign)

Also, upper-case characters and lower-case characters are equivalent.

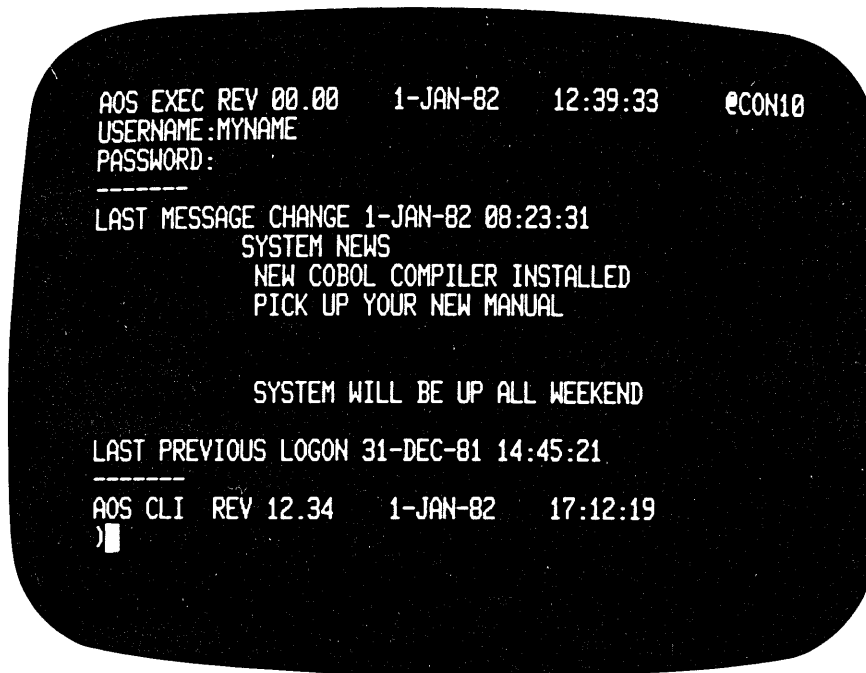


Figure 2.6 A Sample System Message

Review of the log-on procedure:

1. Press NEW LINE.
2. Enter your username; press NEW LINE.
3. Enter your password; press NEW LINE.

Now do Exercise 2-1 on the next page.

## Exercise 2-1

**Directions:** Answer the following questions by completing the sentences.

1. The steps in the log-on procedure are:
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_
2. The program that starts to run when you log on is the \_\_\_\_\_.
3. The name that identifies you to the system is your \_\_\_\_\_.
4. You have a(n) \_\_\_\_\_ to prevent unauthorized use of your log-on privileges.

Now check your answers on the next page.

## **Exercise 2-1**

### **Answers**

1. a. Press NEW LINE.  
b. Enter your username; press NEW LINE.  
c. Enter your password; press NEW LINE.
2. Initial program
3. Username
4. Password

If you answered all the questions correctly, continue with Module 2 by restarting the Module 2 audiotape. Otherwise, review the material and do this exercise again before you continue.



## Console Control

### Control Characters

The delete key, or DEL, removes characters that precede the cursor. (This key may also be labeled RUBOUT.)

To enter control characters, hold down the CTRL key and press the appropriate character. Control characters work only on video terminals, not on printer terminals.

Control Character	Result
CTRL-L	Clears the display screen.
CTRL-S	Stops the scrolling display of information.
CTRL-Q	Restarts the scrolling display of information.
CTRL-O	Cancel the display of information.
CTRL-A	Recalls previous command line.
CTRL-E	Inserts characters into command line.
CTRL-U	Deletes command line.

Table 2.A Control Characters

The CTRL-A character recalls the last CLI command line, allowing you to modify only those characters that you wish to change.

#### Example

```
QPRINT FILE1
      (CTRL-A)
QPRINT FILE2A
      (CTRL-A)
QPRINT FILE2B
```

### Control Sequences

Control sequences consist of two control characters that are entered as a pair, one after the other.

Control Sequence	Result
CTRL-C, CTRL-A	Cancel current CLI command.
CTRL-C, CTRL-B	Cancel current process and returns to creating process.

Table 2.B Control Sequences

Use control characters and control sequences as you do the Lab Activities in this course.



## Exercise 2-2

**Directions:** Name the control character that performs the stated function.

1. Stops display upon the terminal. \_\_\_\_\_
2. Clears the display. \_\_\_\_\_
3. Repeats the previous CLI command. \_\_\_\_\_
4. Restarts the display of information. \_\_\_\_\_
5. Erases the current command line. \_\_\_\_\_
6. Cancels the display of information. \_\_\_\_\_
7. Inserts a character. \_\_\_\_\_

**Directions:** Name the control sequence or key that performs the stated function.

8. Cancels only the effects of the current CLI command. \_\_\_\_\_
9. Cancels the current process. \_\_\_\_\_
10. Removes the character that precedes the cursor. \_\_\_\_\_

Now check your answers on the following page.

## **Exercise 2-2**

### **Answers**

1. CTRL-S
2. CTRL-L
3. CTRL-A
4. CTRL-Q
5. CTRL-U
6. CTRL-O
7. CTRL-E
8. CTRL-C, CTRL-A
9. CTRL-C, CTRL-B
10. The delete key (DEL or RUBOUT)

If you answered all the questions correctly, continue with Module 2 by restarting the Module 2 audiotape. Otherwise, review the material and do this exercise again before continuing.

## CLI Commands

Command	Result
DATE	Returns the system date (DAY:MONTH:YEAR).
TIME	Returns the system time (HOURS:MINUTES:SECONDS).
WHO	Returns the process identification.
BYE	Logs you off the system correctly.

Table 2.C Four Basic CLI Commands



```
)DATE
12-DEC-81
)TIME
12:13:45
)WHO
PID 27  USER1  CON22  :CLI.PR
)
```

Figure 2.7 The DATE, TIME, and WHO Commands



Figure 2.8 The BYE Command for Logging Off

## Lab Activity 2-1

Complete this Lab Activity if you do not have access to a computer system, or if you want some practice before logging on to a working system.

**Directions:** After viewing the screen, correctly complete the sentence.

1. When you see the screen in Figure 2.9, you \_\_\_\_\_.

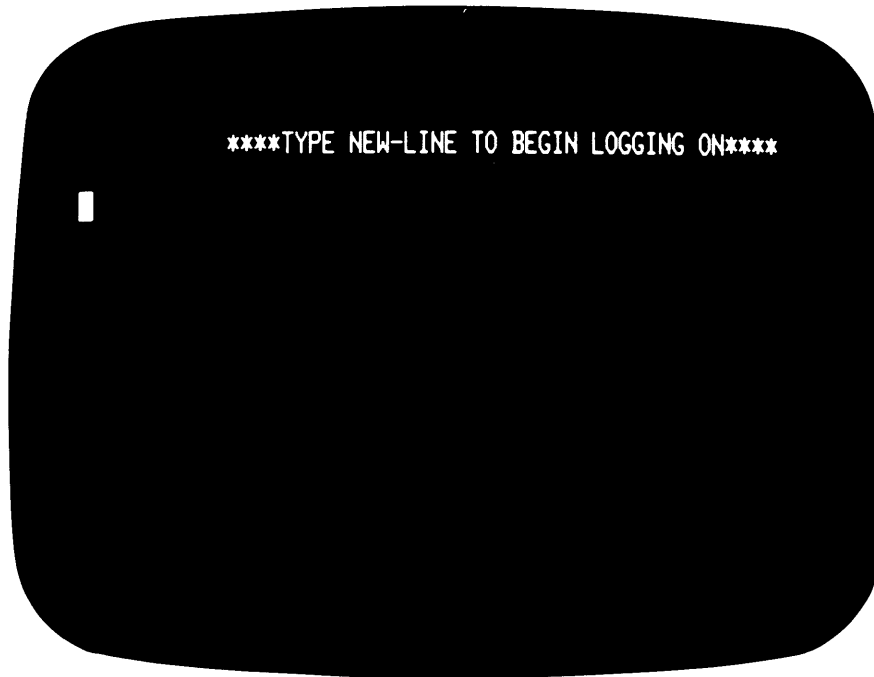


Figure 2.9

2. When you see the screen in Figure 2.10, you \_\_\_\_\_.



Figure 2.10

3. When you see the screen in Figure 2.11, you \_\_\_\_\_.

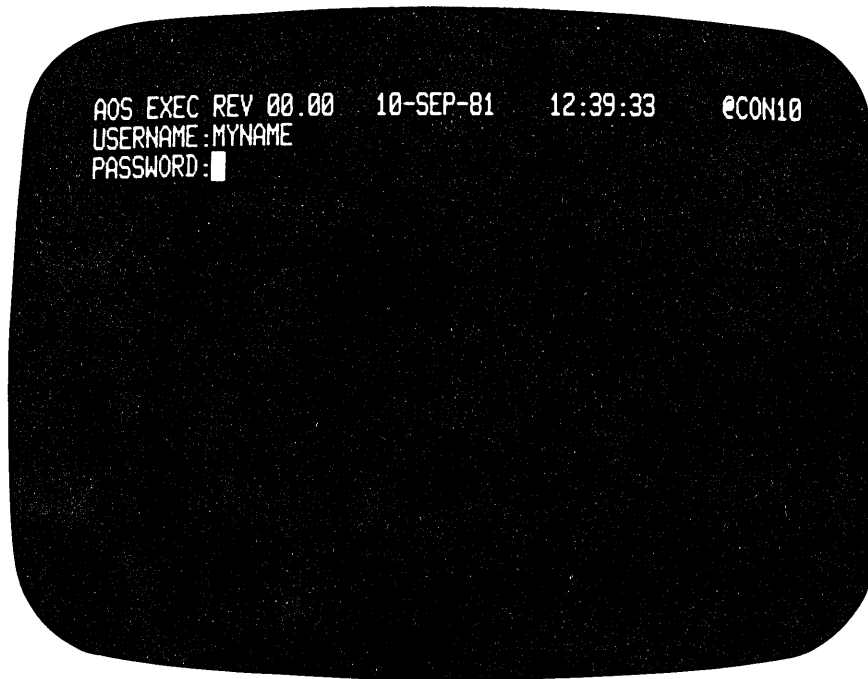


Figure 2.11

4. The screen in Figure 2.12 is displayed if you \_\_\_\_\_.

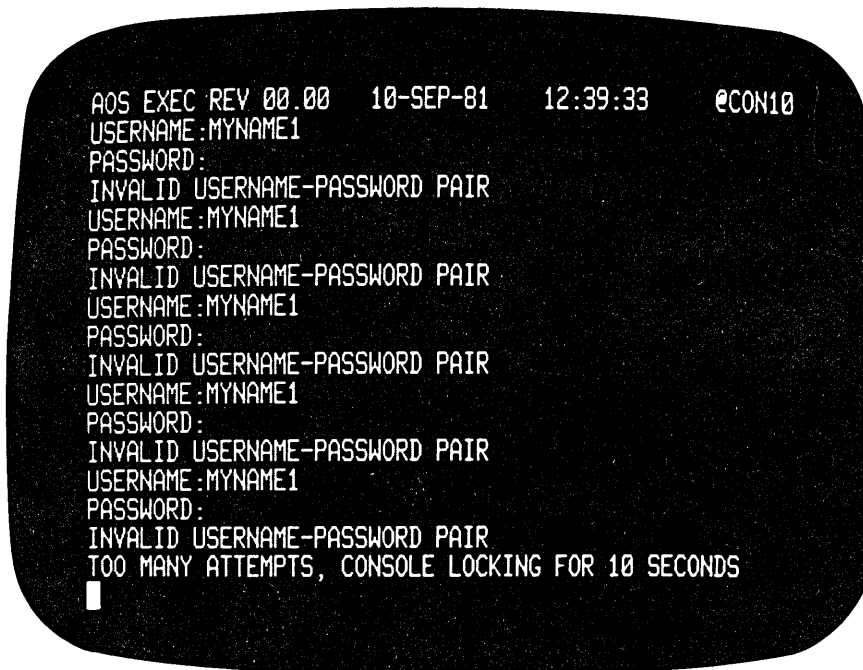


Figure 2.12



5. You would enter CTRL-L at the end of the old password (as shown in Figure 2.13) if you wanted to \_\_\_\_\_.

```
AOS EXEC REV 00.00 10-SEP-81 12:39:33 @CON10
USERNAME:MYNAME1
PASSWORD:
ENTER YOUR NEW PASSWORD:NEWORD
--NEW PASSWORD IN EFFECT--
█
```

Figure 2.13

6. Does the message in Figure 2.14 appear on every screen? \_\_\_\_\_

```
AOS EXEC REV 00.00 1-JAN-82 12:39:33 @CON10
USERNAME:MYNAME
PASSWORD:
-----
LAST MESSAGE CHANGE 1-JAN-82 08:23:31
SYSTEM NEWS
NEW COBOL COMPILER INSTALLED
PICK UP YOUR NEW MANUAL

SYSTEM WILL BE UP ALL WEEKEND

LAST PREVIOUS LOGON 31-DEC-81 14:45:21
-----
AOS CLI REV 12.34 1-JAN-82 17:12:19
)█
```

Figure 2.14

7. Name the CLI command to:

- a. Determine today's date. \_\_\_\_\_.
- b. Determine your process ID. \_\_\_\_\_.
- c. Determine the time of day. \_\_\_\_\_.
- d. Terminate your terminal session. \_\_\_\_\_.

Now check your answers on the next page.

## Lab Activity 2-1

### Answers

1. Press NEW LINE.
2. Enter your username.
3. Enter your password (you will not see it displayed).
4. Incorrectly enter your username or password.
5. Change your password.
6. No. This is a local message generated by the operator.
7. a. DATE  
b. WHO  
c. TIME  
d. BYE

If you answered all the questions correctly, continue to Lab Activity 2-2. Otherwise, review the material and do Lab Activity 2-1 again before you continue.



## Lab Activity 2-2

**Directions:** To receive the maximum benefit from this activity, you should complete it at a functioning AOS or AOS/VS system. If one is not available, write the appropriate responses. When you finish, check your answers on the following page. Before beginning this Lab Activity, obtain a username and profile from your system manager. Be sure that your initial program is the CLI. As you do this and subsequent Lab Activities, you may wish to use some of the control characters listed in Table 2.A.

1. Find a terminal displaying the appropriate message to begin logging on, or turn on a terminal.
2. Press the correct key to begin logging on.
3. Enter your username and password.
4. Determine if you have successfully logged on. (You should see the CLI prompt.)
5. Display today's date.
6. Determine your process ID.
7. Determine the time of day. ( Do not use your watch!)
8. Log off the system.

Now check your answers on the next page.

## Lab Activity 2-2

### Answers

1. The terminal displays this message if it is on:  
**\*\* TYPE NEW LINE TO BEGIN LOGGING ON \*\***
2. Press **NEW LINE** or the return key to begin logging on.
3. Enter the username and password assigned by your system manager. Press the new-line key after entering each one. For example:  
**USERNAME: JONES**  
**PASSWORD:** (Here enter your password. It does not show on the screen.)
4. If you have successfully logged on, you will see a message that tells you the AOS CLI revision, date, and time. For example:  
**AOS CLI REV 12.34 1-JAN-82 10:11:12**
5. Enter the CLI command **DATE**. A sample response is:  
**1-JAN-82**
6. Enter the CLI command **WHO**. A sample response is:  
**PID 27 USER1 CON22 :CLI.PR**
7. Enter the CLI command **TIME**. A sample response is:  
**10:11:12**
8. Enter the CLI command **BYE**. A sample response is:  
**AOS CLI TERMINATING 1-JAN-82 10:11:12**

If you successfully completed Lab Activity 2-2, go on to the Module 2 Test. If you had difficulty, review the material and do this Lab Activity again before you continue.

## Module 2 Test

1. Select the correct procedure to change your password.
  - a. Enter username, followed by NEW LINE.  
Enter old password, followed by CTRL-G.  
Enter new password, followed by NEW LINE.
  - b. Enter username, followed by NEW LINE.  
Enter old password, followed by CTRL-L.  
Enter new password, followed by NEW LINE.
  - c. Enter username, followed by NEW LINE.  
Enter new password, followed by CTRL-L.
  - d. Enter username, followed by NEW LINE.  
Enter old password, followed by NEW LINE.  
Enter new password, followed by CTRL-L.

**Directions:** Match the control characters with the appropriate result.

- |                 |   |
|-----------------|---|
| 2. _____ CTRL-L | a. Restarts the display of information.           |
| 3. _____ CTRL-E | b. Erases the current command line.               |
| 4. _____ CTRL-S | c. Clears the display.                            |
| 5. _____ CTRL-Q | d. Stops the display of additional information.   |
| 6. _____ CTRL-U | e. Cancels the display of additional information. |
| 7. _____ CTRL-A | f. Inserts characters.                            |
| 8. _____ CTRL-O | g. Repeats previous command.                      |

**Directions:** Complete the following sentences:

9. The control sequence to cancel a CLI command is:
  - a. CTRL-C, CTRL-B
  - b. CTRL-C, CTRL-F
  - c. CTRL-C, CTRL-A
10. The control sequence to cancel a process and return to the creating process is:
  - a. CTRL-C, CTRL-B
  - b. CTRL-C, CTRL-F
  - c. CTRL-C, CTRL-A

11. The CLI command that displays the system date is \_\_\_\_\_.
12. The CLI command that displays the system time is \_\_\_\_\_.
13. The CLI command that displays your process ID is \_\_\_\_\_.
14. The CLI command that logs you off the system is \_\_\_\_\_.
15. The steps of the log-on procedure are:
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_

Now check your answers to the Module 2 Test in Appendix A. If you have answered all the questions correctly, continue to Module 3. Otherwise, go back and review the material in Module 2 and take the Module 2 Test again.

This concludes Module 2.



# Module 3

## CLI Commands

### Introduction

This module introduces the CLI command format. In Module 2, you learned to use simple CLI commands. In this module you will learn to use some more complex commands and see the general structure of CLI commands.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Identify the elements of a CLI command line.
2. Determine if a CLI command line is correctly coded.
3. Determine if a filename is correct or incorrect.
4. Use coding aids to combine several command lines into one.
5. Write the following CLI commands:

HELP  
CREATE  
TYPE  
DELETE  
COPY  
RENAME

### Resources

To complete this module, you will need:

- Module 3 audiotape.
- Module 3 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

Module 3 discusses the following topics:

1. Coding a CLI command line
  - a. Switches
  - b. Arguments
2. Files and filenames
3. CLI commands
  - a. HELP
  - b. CREATE
4. More CLI commands
  - a. COPY
  - b. DELETE
  - c. RENAME
  - d. TYPE
5. Coding aids
  - a. Multiple-command line
  - b. Multiple-line commands
  - c. Parentheses
  - d. Brackets

Now start the Module 3 audiotape. As you listen, follow along in Module 3 of your *Student Guide*.

## Coding a CLI Command Line

The format of a CLI command line is:

**COMMAND[/SWITCH1.../SWITCHX],[ARGUMENT1...,ARGUMENTX]**

The *command* identifies the operation that you want to perform.

The *switch* modifies the function of the command.

The *argument* is the filename, pathname, or device that the command affects.

Commands and switches can be abbreviated. Use any number of characters that uniquely identifies the command or switch.

For the TIME command:

T	Not unique
TI	Acceptable
TIM	Acceptable
TIME	Acceptable

For the TYPE command:

T	Not unique
TY	Acceptable
TYP	Acceptable
TYPE	Acceptable

The rules for command abbreviations also apply to switches.

Possible variations of the /SORT switch:

```
/S
/SO
/SOR
/SORT
```

If you use a command abbreviation that is too short, you receive the message **COMMAND ABBREVIATION NOT UNIQUE**. If you use a switch abbreviation that is too short, you receive the message **ARGUMENT IS NOT A UNIQUE ABBREVIATION**.

A *switch* is a slash (/), followed by a word, number, or expression. Switches modify the default action of a CLI command. The order of switches is not important.

Command	Switch	Effect
TIME/L	/L Simple switch	Sends output to the current listfile. (1)
TIME/L=FILENAME	/L=FILENAME Keyword switch	Sends output to the file specified by filename. (2)
TIME	None	Sends output to your console. (3)

Table 3.A The TIME Command

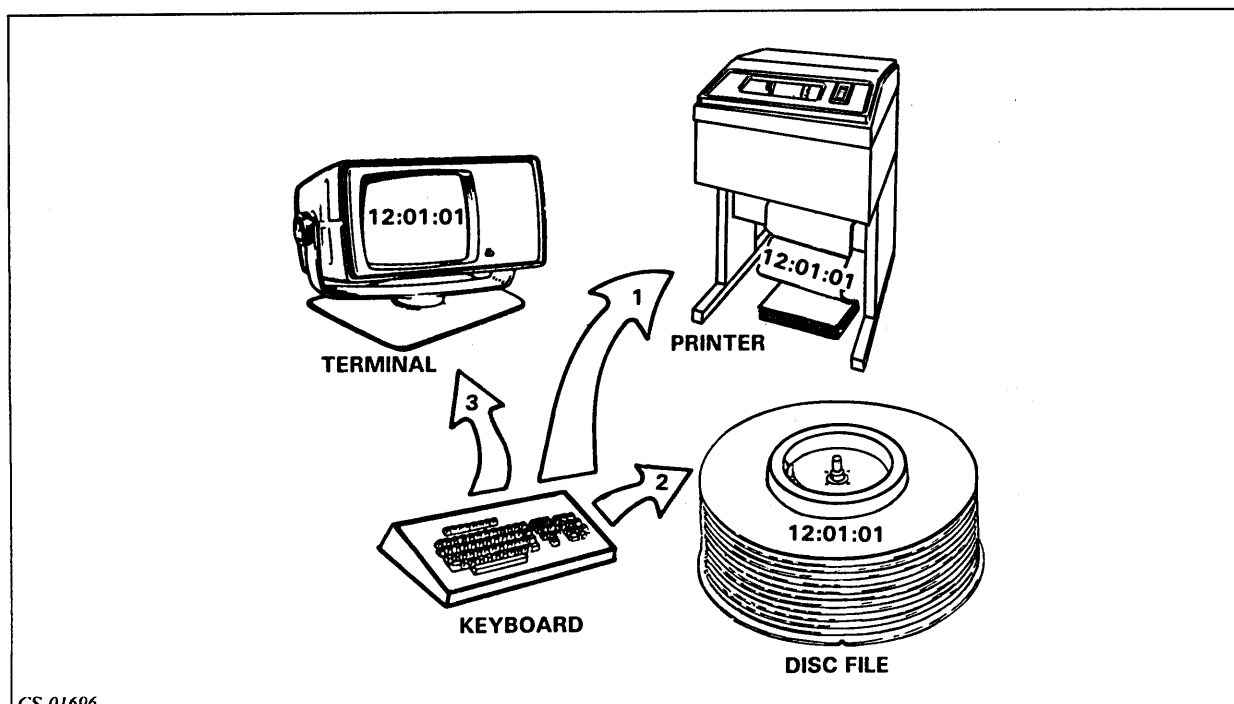


Figure 3.1 TIME Going to the Printer, Disc File, and Console

## Command Switches

There are five common switches that can be used with most CLI commands. These are:

- /1=
- /2=
- /L
- /L=PATHNAME
- /Q

The switches /1= and /2= pertain to *error handling*. Invalid commands cause errors (exceptional conditions). There are two classes of errors. Class 1 errors are severe; class 2 errors are less serious.

The system can take four possible actions in response to an error. These actions are described in Table 3.B.

Setting	Result
IGNORE	No effect. The exception is ignored and processing continues.
WARNING (Default class 2)	A warning message is displayed and processing continues.
ERROR (Default class 1)	Execution of the current command stops. A message is displayed and CLI prompts for a new command.
ABORT	Return to creating process. If you are in CLI, you will be logged off the system.

Table 3.B Four Error Actions

- /L Writes the CLI output of the current command to the listfile, which is usually the line printer.
- /L= Writes the CLI output of the current command to the file specified by the pathname.
- /Q Compresses the output of the CLI command by reducing spaces and tabs to single spaces by turning SQUEEZE on.

**Example: Output without SQUEEZE**

DIRECTORY :UDD:FLUFF:MORT

__FLUFF.TMP	TXT	11-FEB-82	13:39:58	0
__FLUFF.TUBE	TXT	11-FEB-82	13:40:02	6
MORT.FR	UDF	20-MAR-81	11:15:50	1096
TEMP	TXT	11-FEB-82	13:41:06	0
MOUT	TXT	10-FEB-82	16:24:06	3579
TEMP.BU	TXT	11-FEB-82	13:39:46	195
MORTIN	UDF	10-FEB-82	16:26:10	18
MORT.PR	PRG	25-SEP-80	11:11:50	65536
PRCMORT	UDF	10-FEB-82	16:21:38	61

**Example: Same output with SQUEEZE**

DIRECTORY :UDD:FLUFF:MORT

```

__FLUFF.TMP TXT 11-FEB-82 13:39:58 0
__FLUFF.TUBE TXT 11-FEB-82 13:40:02 6
MORT.FR UDF 20-MAR-81 11:15:50 1096
TEMP TXT 11-FEB-82 13:41:06 0
MOUT TXT 10-FEB-82 16:24:06 3579
TEMP.BU TXT 11-FEB-82 13:39:46 195
MORTIN UDF 10-FEB-82 16:26:10 18
MORT.PR PRG 25-SEP-80 11:11:50 65536
PRCMORT UDF 10-FEB-82 16:21:38 61

```

**Delimiters**

A command and each of its arguments must be separated by a delimiter. Delimiters are:

- Space or spaces
- Tabs
- Commas

**Example**

These commands are equivalent:

TYPE FILE1 (single space)

TYPE,FILE1 (comma)

TYPE FILE1 (tab)

## Terminators

A correctly coded CLI command line must be terminated. Valid terminators are:

- NEW LINE
- Carriage return
- Form-feed (CTRL-L)
- End of file (CTRL-D)

### Example

These commands are equivalent:

```
TIME (NEW LINE)
TIME (carriage return)
TIME (form-feed)
TIME (end of file)
```

Now do Exercise 3-1 on the next page.





## Exercise 3-1

**Directions:** In items 1 through 4, match the terms with the elements of the command line.

TYPE/L,FILE1  
↑ ↑ ↑ ↑  
1 2 3 4

1. \_\_\_\_ Delimiter
2. \_\_\_\_ Command
3. \_\_\_\_ Argument
4. \_\_\_\_ Switch

**Directions:** Indicate by writing a C for correct and an I for incorrect which of the following command lines are coded correctly.

5. \_\_\_\_ TYPE /L FILE1
6. \_\_\_\_ TYPE/L,FILE1
7. \_\_\_\_ TYPE/L FILE1
8. \_\_\_\_ TYPE/L FILE1
9. \_\_\_\_ TYPE L,FILE1

Now check your answers on the next page.

## **Exercise 3-1**

### **Answers**

1. 3

2. 1

3. 4

4. 2

5. I

6. C

7. C

8. C

9. I

If you answered all the questions correctly, continue with Module 3 by restarting the Module 3 audiotape. Otherwise, review the material and do this exercise again before you continue.

## Files and Filenames

### Filenames

- Maximum of 31 characters
- Valid characters are:
  - A through Z (upper-case and lower-case)
  - 0 through 9
  - ? (question mark)
  - . (period)
  - \_ (underscore)
  - \$ (dollar sign)

Invalid	Reason	Valid
FILE 1	Invalid space	FILE.1
FILE#2	# not allowed	FILE_2
FILE/NUMBER/THREE	/ not allowed	FILE_NUMBER_THREE
WHAT*IS*THIS?	* not allowed	WHAT_IS_THIS?
MONEY\$_STUFF!	! not allowed	MONEY\$_STUFF

Table 3.C Examples of Valid and Invalid Filenames

Now do Exercise 3-2 on the next page.



## Exercise 3-2

**Directions:** Mark the following filenames V for valid or I for invalid.

1. \_\_\_\_ more\_information
2. \_\_\_\_ here\_is\_my\_favorite\_file\_in\_the\_whole\_world
3. \_\_\_\_ THIS.IS.A.good.file
4. \_\_\_\_ DATA#FILE
5. \_\_\_\_ important\_stuff
6. \_\_\_\_ I\_don't\_know?
7. \_\_\_\_ PAYROLL!
8. \_\_\_\_ 9to5

Now check your answers on the next page.

## **Exercise 3-2**

### **Answers**

1. V
2. I (This filename contains more than 31 characters, and therefore is too long.)
3. V
4. I (# is an invalid character.)
5. V
6. I (Single quote is an invalid character.)
7. I (! is an invalid character.)
8. V

If you answered all of the questions correctly, continue with Module 3 by restarting the Module 3 audiotape. Otherwise, review the material and do this exercise again before you continue.

# CLI Commands

## HELP Command

The HELP command provides you with information about CLI commands.

```
)HELP
TOPICS ARE:

*1_SWITCH      *2_SWITCH      *AFTER_SWITCH  *CLI_INPUT     *COMMANDS
*CONDITIONALS *CONTROL_CHARS *CURSOR_CONTROL *DBMS          *DISPLAY
*ENROLL_DOC    *ENVIRONMENT   *EXCEPTIONS    *EXEC
*FILENAMES
*GENERIC_FILES *HISTO         *I_SWITCH      *LINK          *LINKS
*LOGSCALL      *L_SWITCH     *MACROS        *MAIL          *M_SWITCH
*NEWLINE       *OVTRAC       *PATCH        *PATHNAMES    *PED
*PSEUDO-MACROS *P_SWITCH     *QUEUES        *Q_SWITCH     *REPORT
*SED           *SNAP         *SWITCHES      *TEMPLATES    *TOOLS
*TOPICS        *XRO

FOR MORE HELP ABOUT ANY ITEM ABOVE, TYPE 'HELP *ITEM'
)
```

Figure 3.2 Result of the HELP Command

Using the /V switch invokes the verbose option, which provides more detailed information.

### Example

Enter **HELP/V,HELP**

```
HELP      PROVIDE HELPFUL INFORMATION
FORMAT   HELP <ITEM> ...
COMMAND SWITCHES  /1= /2= /L(=) /Q
                V      PROVIDE MORE INFORMATION IF AVAILABLE

IF YOU TYPE HELP WITH NO ARGUMENTS, YOU WILL BE SHOWN A LIST OF TOPICS
ABOUT WHICH YOU MAY ASK FOR HELP. EACH TOPIC BEGINS WITH AN ASTERISK.
YOU MAY GIVE ONE OR MORE OF THESE TOPICS AS ARGUMENTS TO A HELP COMMAND.
YOU MUST INCLUDE THE ASTERISK. ALL AOS SYSTEMS PROVIDE THE TOPICS
*COMMANDS AND *PSEUDO-MACROS. YOUR SYSTEM WILL PROBABLY HAVE OTHER
TOPICS
AS WELL

HELP *COMMANDS
        WILL GIVE YOU A LIST OF ALL CLI COMMANDS. YOU CAN FIND OUT THE
        ARGUMENT REQUIREMENTS OF A COMMAND AND ITS COMMAND SWITCHES BY TYPING
        THE COMMAND NAME AS AN ARGUMENT TO A HELP COMMAND. IF THERE IS
        ADDITIONAL
        INFORMATION AVAILABLE ON YOUR SYSTEM FOR THAT COMMAND, YOU WILL BE
        REMINDED
        THAT YOU CAN RE-ISSUE THAT HELP COMMAND WITH THE /V COMMAND SWITCH TO SEE
        THE ADDITIONAL INFORMATION.
```

Figure 3.3 Result of the HELP/V, HELP Command

**HELP Command Summary:**

- Provides information about more than the CLI.
- Contains topics of information for system users.
- Some information is provided by Data General.
- Some information is provided by users of the system.

**CREATE Command**

The CREATE command creates a data file.

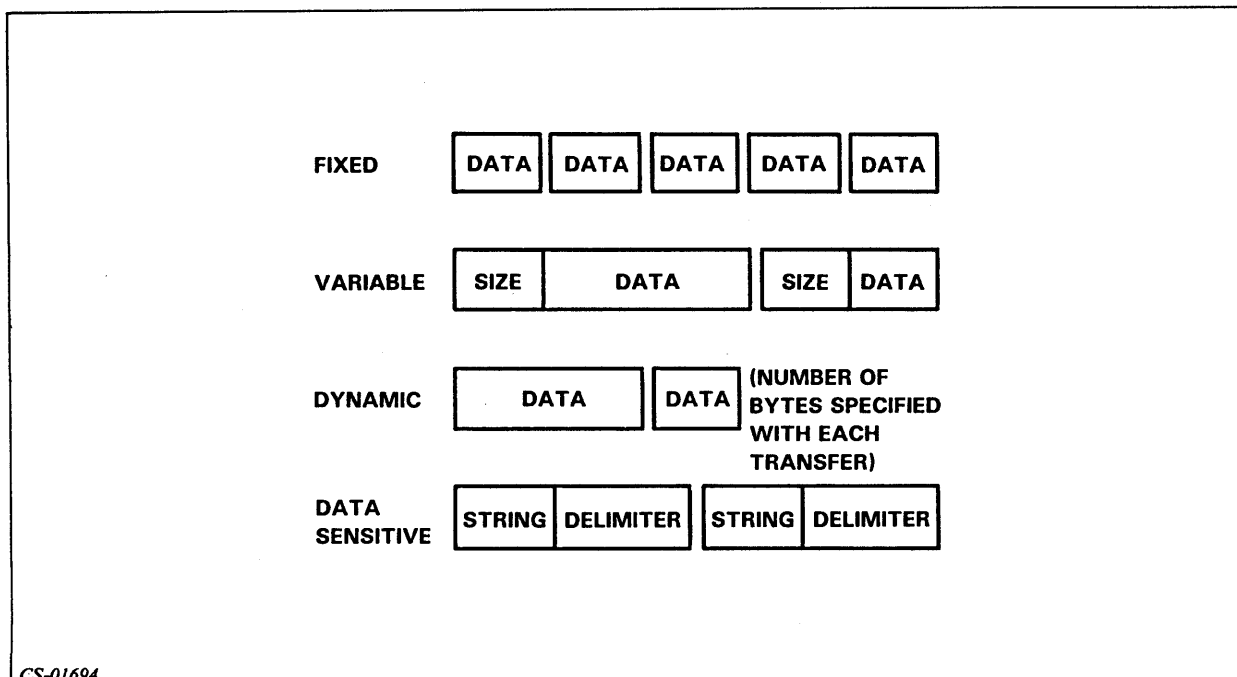
```

)HELP/V CREATE
CREATE CREATE A FILE

FORMAT: CREATE PATHNAME <RESOLUTION-PATHNAME>
WHERE RESOLUTION-PATHNAME IS THE PATHNAME
TO WHICH THE LINK WILL RESOLVE
(IF /LINK SWITCH IS USED)

COMMAND SWITCHES: /1= /2= /L(=) /Q
/DATASENSITIVE RECORD TYPE OF CREATED FILE
/DIRECTORY CREATE A DIRECTORY OR CONTROL POINT DIRECTORY
/DYNAMIC RECORD TYPE OF THE CREATED FILE
/ELEMENTSIZE=N NUMBER OF DISK BLOCKS FILE WILL GROW BY
/FIXED=N RECORD TYPE OF THE CREATED FILE
/HASHFRAMESIZE=N SIZE OF HASH FRAME (DEFAULT 7)
(DIRECTORIES ONLY)
/I CONTENTS OF FILE TAKEN FROM @INPUT
/INDEXLEVELS=N MAXIMUM NUMBER OF INDEX LEVELS (DEFAULT=3)
/LINK CREATE A LINK
/M FILE TAKEN FROM REMAINDER OF MACRO FILE
/MAXSIZE=N CREATE A CONTROL POINT DIRECTORY OF THE SPECIFIED
MAXIMUM SIZE (IN DISK BLOCKS) -- VALID ONLY
WITH THE /DIRECTORY SWITCH
/TYPE=N CREATE A FILE OF THE SPECIFIED TYPE (63 < N < 256)
    
```

Figure 3.4 HELP Output for the CREATE Command



CS-01694

Figure 3.5 Four Types of Data Records



## CREATE Command

### Example 1

```
CREATE/DATASENSITIVE,FIRST_FILE
```

### Example 2

```
CREATE/VARIABLE,VAR_FILE
```

### Example 3

```
CREATE/DYNAMIC,DYN_FILE
```

### Example 4

```
CREATE/FIXED=10,FIXED_FILE
```

### Example 5

```
CREATE,FILE1
```

### Example 6

```
CREATE/I,NEWFILE  
) THIS IS THE TEXT.  
) )
```

Now do Exercise 3-3 on the next page.



## Exercise 3-3

**Directions:** Write the CLI command that will:

1. Supply you with information about the DATE command.  
\_\_\_\_\_
2. Create a file called DYNAM\_FILE using the dynamic record format.  
\_\_\_\_\_
3. Create a file CALLED FIX\_FILE using the fixed record format. Records should be 15 characters in length. \_\_\_\_\_
4. Create a text file that allows you to enter data from your console. Call the file DATA\_FILE. \_\_\_\_\_
5. Supply you with a list of CLI commands. \_\_\_\_\_

Check your answers on the next page.

## **Exercise 3-3**

### **Answers**

1. **HELP DATE**
2. **CREATE/DYNAMIC,DYNAM\_FILE**
3. **CREATE/FIXED=15,FIX\_FILE**
4. **CREATE/I,DATA\_FILE**
5. **HELP \*COMMANDS**

If you answered all the questions correctly, continue with Module 3 by restarting the Module 3 audiotape. Otherwise, review the material and do this exercise again before you continue.

## More CLI Commands

### COPY Command

The COPY command creates an additional copy of one or more files.

Format: **COPY,DESTINATION\_FILE,SOURCE\_FILE**

These switches are used when the destination file already exists:

- /A Appends the contents of the source file to the destination file.
- /D Deletes the old contents of the destination file and replaces them with the contents of the source file.

#### Example 1

**COPY,FILEA,FILEB**

Copy FILEB into FILEA. Create FILEA. Before this command was issued, FILEA did not exist, and FILEB contained data B. After the COPY command is issued, both FILEA and FILEB contain data B.

#### Example 2

**COPY/A,FILEA,FILEB**

Append FILEB to FILEA. Before this command was issued, FILEA contained data A, and FILEB contained data B. After the COPY command is issued, FILEA contains both data A and data B, and FILEB contains its original data B.

#### Example 3

**COPY/D,FILEA,FILEB**

Replace the contents of FILEA with FILEB. Before this command was issued, FILEA contained data A, and FILEB contained data B. After the COPY command is issued, FILEA contains data B and FILEB also contains data B. Data A, the original contents of FILEA, has been deleted as a result of the /D switch.

## DELETE Command

The DELETE command deletes one or more files.

Format: **DELETE,FILENAME**

These switches help prevent you from accidentally deleting files:

- /C Confirm before deleting files.
- /V Verify after deleting files.

### Example 1

```
DELETE/C,FILE_ONE  
FILE_ONE? Y
```

### Example 2

```
DELETE/C/V,FILEA,FILEB  
FILEA? N  
FILE NOT DELETED  
FILEB? Y  
FILEB DELETED
```

You always should use the /C and /V switches to protect your files from accidental deletion.

Now do Exercise 3-4 on the next page.

## Exercise 3-4

**Directions:** Write the CLI command that will:

1. Append the contents of FILEX to the contents of FILEY. \_\_\_\_\_
2. Replace the contents of FILEX with the contents of FILEZ. \_\_\_\_\_
3. Delete FILE1 and FILE2, ask for confirmation before deletion, and verify the results.  
\_\_\_\_\_
4. Create FILEV and put the contents of FILEK into FILEV *in one command*.  
\_\_\_\_\_

Check your answers on the next page.

## **Exercise 3-4**

### **Answers**

1. **COPY/A FILEY,FILEX**
2. **COPY/D FILEX,FILEZ**
3. **DELETE/C/V FILE1,FILE2**
4. **COPY FILEV,FILEK**

If you answered all questions correctly, continue with Module 3 by restarting the Module 3 audiotape. Otherwise, review the material and do this exercise again before you continue.



## More CLI Commands (Continued)

### RENAME Command

The RENAME command changes the name of a file.

Format: **RENAME,OLD\_NAME,NEW\_FILE**

#### Example

**RENAME,OLD\_DATA,NEW\_DATA**

The file originally called OLD\_DATA is now called NEW\_DATA. There is no file named OLD\_DATA after execution of this command.

### TYPE Command

The TYPE command displays a file at your console.

Format: **TYPE,FILENAME**

#### Example

**CREATE/I,TESTFILE**

**) TEST LINE 1**

**) TEST LINE 2**

**)**

**TYPE,TESTFILE**

**TEST LINE 1**

**TEST LINE 2**

Now do Exercise 3-5 on the next page.



## Exercise 3-5

**Directions:** Write the CLI command that will:

1. Change the name of a file called JUNK to GOOD\_STUFF. \_\_\_\_\_
2. Display the contents of a file called GOOD\_STUFF. \_\_\_\_\_

Check your answers on the next page.

## **Exercise 3-5**

### **Answers**

1. **RENAME,JUNK,GOOD\_STUFF**

2. **TYPE,GOOD\_STUFF**

If you answered the questions correctly, continue with Module 3 by restarting the Module 3 audiotape. Otherwise, review the material and do the exercise again before you continue.

## Coding Aids

### Multiple-Command Line

A multiple-command line allows you to enter more than one command on a line, by using a semicolon to separate the commands.

**Example**

```
TYPE,FILE_ONE;RENAME,FILE_ONE,NEW_FILE;TYPE,NEW_FILE
```

### Multiple-Line Command

A multiple-line command is one command that requires more than one line. Use an ampersand (&) to continue to the next line.

**Example 1**

```
TYPE,FILE_ONE,FILE_TWO,SPECIAL_FILE_DATA,FILE_THREE &  
&) FILE_FOUR
```

**Example 2**

```
TYPE,FILE.F,FILE.V &  
&) FILE.X &  
&) FILE.B
```

### WRITE Command

The WRITE command displays its arguments on your terminal.

**Example**

```
WRITE This is an aardvark.  
This is an aardvark.
```

### Parentheses

Parentheses allow command repetition. Commands are executed as if each argument were entered on a separate line.

**Example 1**

```
TYPE (A B C) expands to:  
TYPE A  
TYPE B  
TYPE C
```

**Example 2**

```
WRITE A(B C)D expands to:  
WRITE A B D  
WRITE A C D
```

At your terminal, you see:

```
A B D  
A C D
```

**Example 3**

(TYPE DELETE) FILE\_ONE expands to:

```
TYPE FILE_ONE
DELETE FILE_ONE
```

**Example 4**

WRITE (A B C)(X Y) expands to:

```
WRITE AX
WRITE BY
WRITE C
```

At your terminal, you see:

```
AX
BY
C
```

## Angle Brackets

Angle brackets allow argument expansion. In the same command line, you can have one command with several arguments. The argument expands across the line.

**Example 1**

TYPE FILE<1 2 3> expands to:

```
TYPE FILE1 FILE2 FILE3
```

**Example 2**

WRITE <A B C> <1 2> expands to:

```
WRITE A1 A2 B1 B2 C1 C2
```

At your terminal, you see:

```
A1 A2 B1 B2 C1 C2
```

Now do Exercise 3-6 on the next page.

## Exercise 3-6

**Directions:** Answer the following questions:

1. Correctly write the following CLI commands on a single line:  
**TYPE FILE1**  
**RENAME FILE3 OLD\_FILE** \_\_\_\_\_
2. Name the character used to extend a command line to a second line. \_\_\_\_\_
3. Write a single CLI command using parentheses and/or angle brackets to accomplish the following tasks.
  - a. **TYPE FILE1 FILE2 FILE3** \_\_\_\_\_
  - b. **TYPE FILE1 FILE2 TEST1 TEST2** \_\_\_\_\_
  - c. **TYPE FILE12** \_\_\_\_\_  
**TYPE FILE22**  
**TYPE FILE32**

Check your answers on the next page.

## Exercise 3-6

### Answers

1. `TYPE,FILE1;RENAME,FILE3,OLD_FILE`
2. Ampersand (&)
3. a. `TYPE,FILE<1,2,3>`  
b. `TYPE,<FILE,TEST><1,2>`  
c. `TYPE,FILE(1,2,3)2`

If you answered all the questions correctly, continue to Lab Activity 3-1. Otherwise, review the material and do this exercise again before you continue.



## Lab Activity 3-1

**Directions:** Write the commands that will do each of the following. You can write the answers before entering the command at your terminal. Check your answers on the following page if you are unsure of your response.

1. Write a command to provide information about the CREATE command. (Use the option that provides the most information.)
2. Write commands to provide information about the other commands covered in this module (COPY, DELETE, RENAME, and TYPE).
3. Write the command to create a file named MODULE\_3\_PART\_1. When you create the file, include the following text:  
**THIS IS THE FIRST PART  
OF THE INFORMATION TO BE  
ENTERED INTO A FILE IN  
THE AOS, AOS/VIS USER COURSE**
4. Write the command to create another file. Call it MODULE\_3\_PART\_2. Include this text in the file:  
**THIS IS THE SECOND PART  
OF THE TEXT.  
THIS IS THE END OF THE TEXT.**
5. Write a single command line to display the contents of MODULE\_3\_PART\_1 and MODULE\_3\_PART\_2.
6. Build a file called PART\_1\_DUPLICATE that contains the exact information that MODULE\_3\_PART\_1 contains. (Do not enter the data again.)
7. Use the TYPE command to verify the success of Step 6.
8. Add the contents of MODULE\_3\_PART\_2 to the file created in Step 6.
9. Since PART\_1\_DUPLICATE now contains the contents of both files, change its name to MODULE\_3\_WHOLE\_FILE.
10. Type the contents of MODULE\_3\_WHOLE\_FILE to see if it is all there.
11. Delete MODULE\_3\_PART\_1 and MODULE\_3\_PART\_2. Be sure to use the confirm and verify switches.

Now check your answers on the next page.

## Lab Activity 3-1

### Answers

1. **HELP/V CREATE**
2. **HELP/V COPY**  
**HELP/V DELETE**  
**HELP/V RENAME**  
**HELP/V TYPE**
3. **CREATE/I MODULE\_3\_PART\_1**  
**THIS IS THE FIRST PART**  
**OF THE INFORMATION TO BE**  
**ENTERED INTO A FILE IN**  
**THE AOS, AOS/VS USER COURSE**
4. **CREATE/I MODULE\_3\_PART\_2**  
**THIS IS THE SECOND PART**  
**OF THE TEXT.**  
**THIS IS THE END OF THE TEXT.**
5. **TYPE MODULE\_3\_PART\_<1,2> or TYPE MODULE\_3\_PART(1,2)**
6. **COPY PART\_1\_DUPLICATE,MODULE\_3\_PART\_1**
7. Use the TYPE command verify the success of Step 6.  
**TYPE PART\_1\_DUPLICATE**
8. **COPY/A PART\_1\_DUPLICATE,MODULE\_3\_PART\_2**
9. **RENAME PART\_1\_DUPLICATE,MODULE\_3\_WHOLE**
10. **TYPE MODULE\_3\_WHOLE**
11. **DELETE/V/C MODULE\_3\_PART\_1**  
**DELETE/V/C MODULE\_3\_PART\_2**

If you answered all the questions correctly, continue to the Module 3 Test. Otherwise, review Module 3 and do the Lab Activity again before you continue.

## Module 3 Test

1. Identify the elements of the following command line:

```

      CREATE/DYNAMIC,NEWFILE
      ^      ^      ^      ^
      A      B      C      D
  
```

- A. \_\_\_\_\_ 1. CLI prompt  
 B. \_\_\_\_\_ 2. Argument  
 C. \_\_\_\_\_ 3. Delimiter  
 D. \_\_\_\_\_ 4. Command  
 5. Switch

**Directions:** Which of the following command lines are correctly coded? Mark C for correct and I for incorrect.

2. \_\_\_\_ **HELP/V HELP**  
 3. \_\_\_\_ **RENAME,FILE1 FILE3**  
 4. \_\_\_\_ **TYPE FILE4**  
 5. \_\_\_\_ **DELETE/C/V FILE9**  
 6. \_\_\_\_ **WRITE X**

**Directions:** Which of the following are valid filenames? Mark a V for valid and an I for invalid.

7. \_\_\_\_ **FILE-ONE**  
 8. \_\_\_\_ **MASTER\_FILE**  
 9. \_\_\_\_ **BACK\_UP\_OF\_TAPE\_FILE\_CREATED\_LAST\_FRIDAY**  
 10. \_\_\_\_ **BACK\_UP&SAVE**  
 11. \_\_\_\_ **new\_file**  
 12. \_\_\_\_ **FILE1**  
 13. \_\_\_\_ **FILE?**  
 14. \_\_\_\_ **DOLLARS\_AMOUNT**  
 15. \_\_\_\_ **1ST-DETAIL**  
 16. \_\_\_\_ **MY\_DOCUMENT**

**Directions:** Write the CLI command that will:

17. Delete a file named XYZ. \_\_\_\_\_
18. Display the contents of a file called NEWS\_FILE. \_\_\_\_\_
19. Change the name of a file called GOOD\_STUFF to OLD\_JUNK.  
\_\_\_\_\_
20. Create a file and allow data to be entered from the terminal. Call the file  
TERMINAL\_DATA. \_\_\_\_\_
21. Copy the contents from a file called PAYROLL into a file called PERSONNEL.  
PERSONNEL does not yet exist. \_\_\_\_\_
22. Display at your terminal information about the DELETE command.  
\_\_\_\_\_

**Directions:** Expand the commands as shown in the example:

**Example**

DELETE FILE(A B) Expands to:

DELETE FILEA  
DELETE FILEB

23. COPY/A,MASTER\_FILE,FILE<1 2><A B> \_\_\_\_\_
24. (TYPE DELETE),USELESS\_FILE \_\_\_\_\_
25. TYPE,<FILE TEST><1 2> \_\_\_\_\_
26. DELETE FILE<1 2 3>.OLD \_\_\_\_\_
27. TYPE,(FILE TEST)(1 2) (Note: Compare this to Item 25.)  
\_\_\_\_\_

Now check your answers to the Module 3 Test in Appendix A. If you have answered all the questions correctly, continue to Module 4. Otherwise, go back and review the material in Module 3 and take the Module 3 Test again.

This concludes Module 3.

# Module 4

## Directories and Pathnames

### Introduction

This module introduces the concept of the directory and the directory structure. It also teaches how to select a particular file from within the structure. Finally, directory creation is also discussed.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Identify the major components of the AOS or AOS/VS directory structure.
2. Write the pathname of a file given a directory structure.
3. Write CLI command lines using the following commands:  
CREATE  
FILESTATUS  
DIRECTORY  
SEARCHLIST  
PATHNAME
4. Use templates to access multiple files with one command.

### Resources

To complete this module, you will need:

- Module 4 audiotape.
- Module 4 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

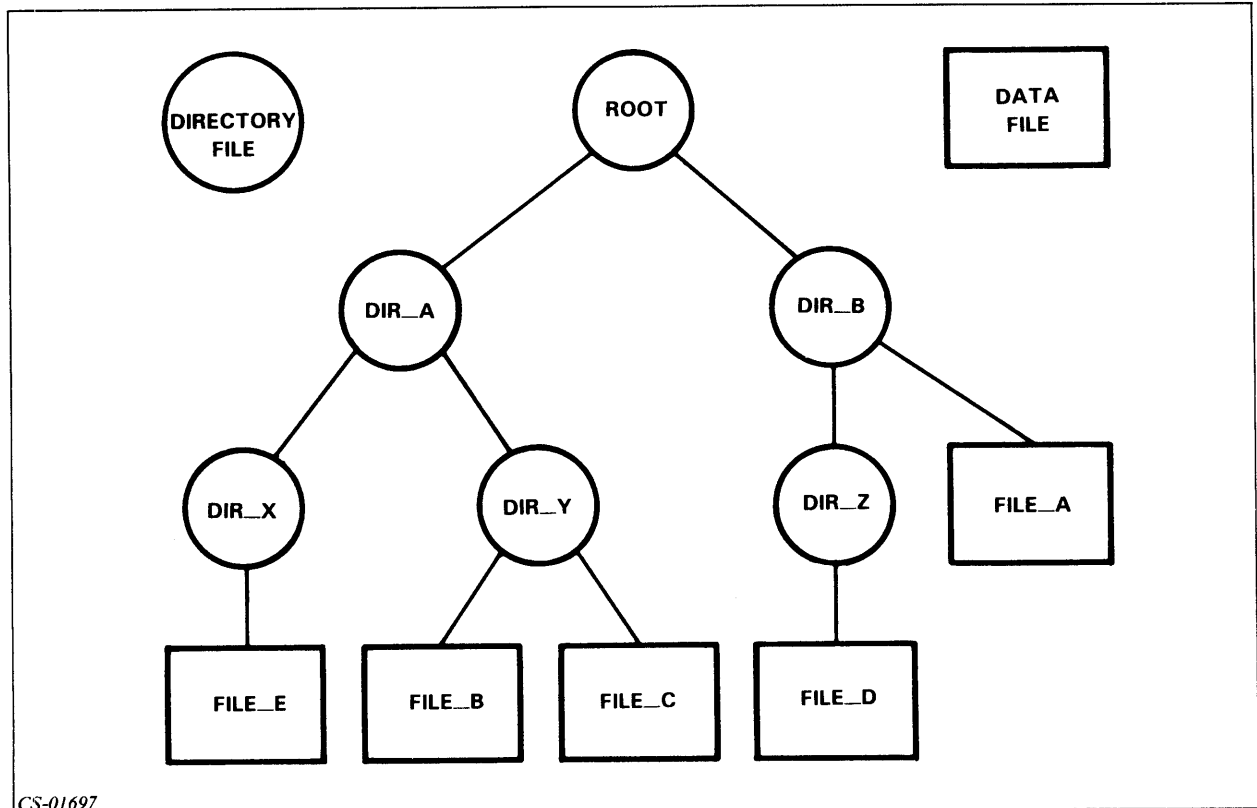
Module 4 discusses the following topics:

1. Directories
  - a. Definition
  - b. Working directory
  - c. AOS or AOS/VS directory structure
  - d. Directory creation
  - e. DIRECTORY command
2. Pathnames
  - a. Definition
  - b. Prefixes
3. Searchlists and templates
  - a. Definition of searchlist
  - b. File templates
4. FILESTATUS command

Now start the Module 4 audiotape. As you listen, follow along in Module 4 of your *Student Guide*.

## Directories

A *directory* is a *file* containing a catalog of bookkeeping information, and pointers to files and other directories. There are two types of directories. A *Control Point Directory*, or *CPD*, has fixed maximum disc space. A *directory* has no space limit.



CS-01697

Figure 4.1 Sample Directory Structure

*Inferior* or *subordinate* directories are lower in the directory structure. *Superior* directories are higher in the directory structure.

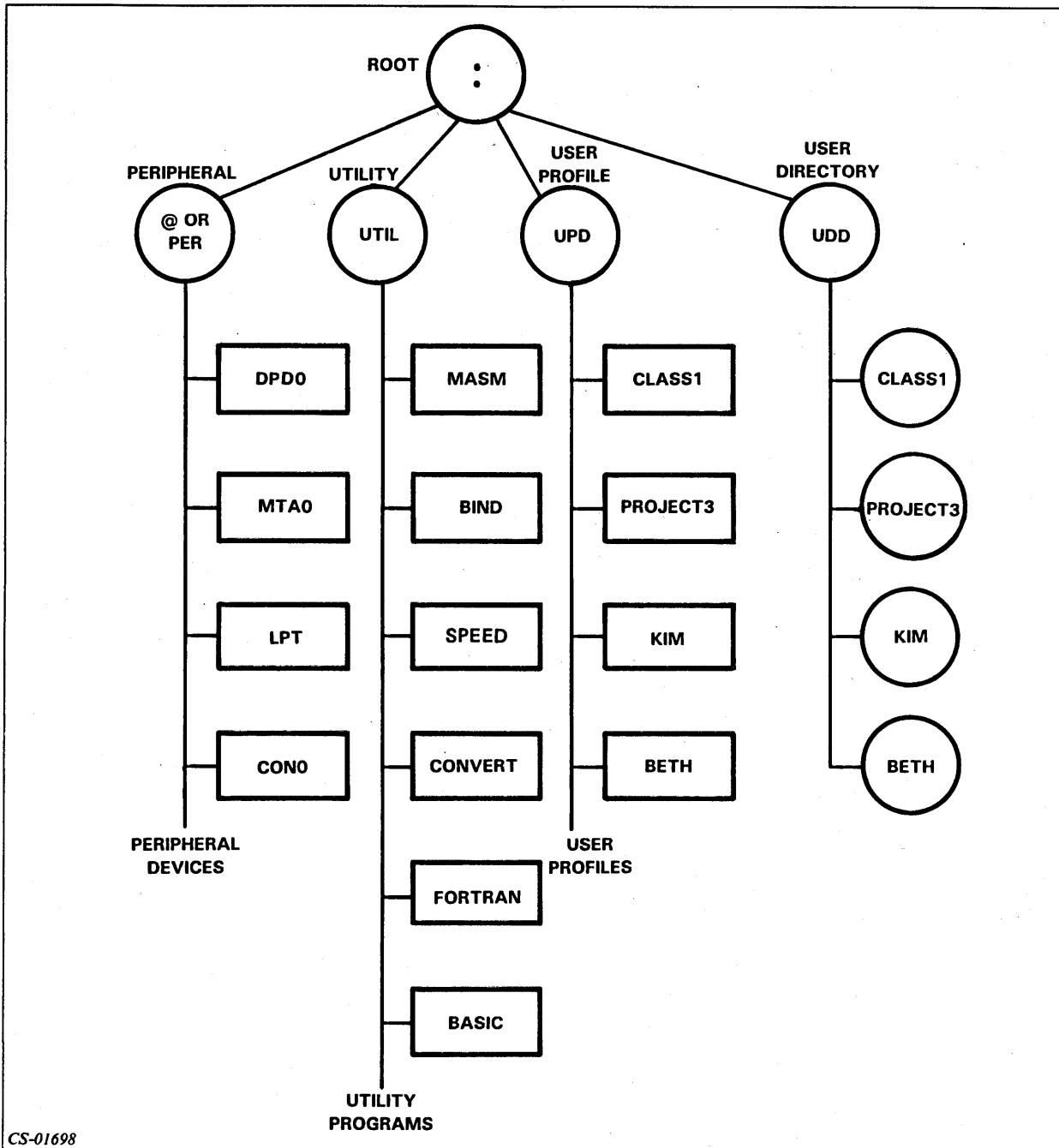


Figure 4.2 Typical Directory Structure



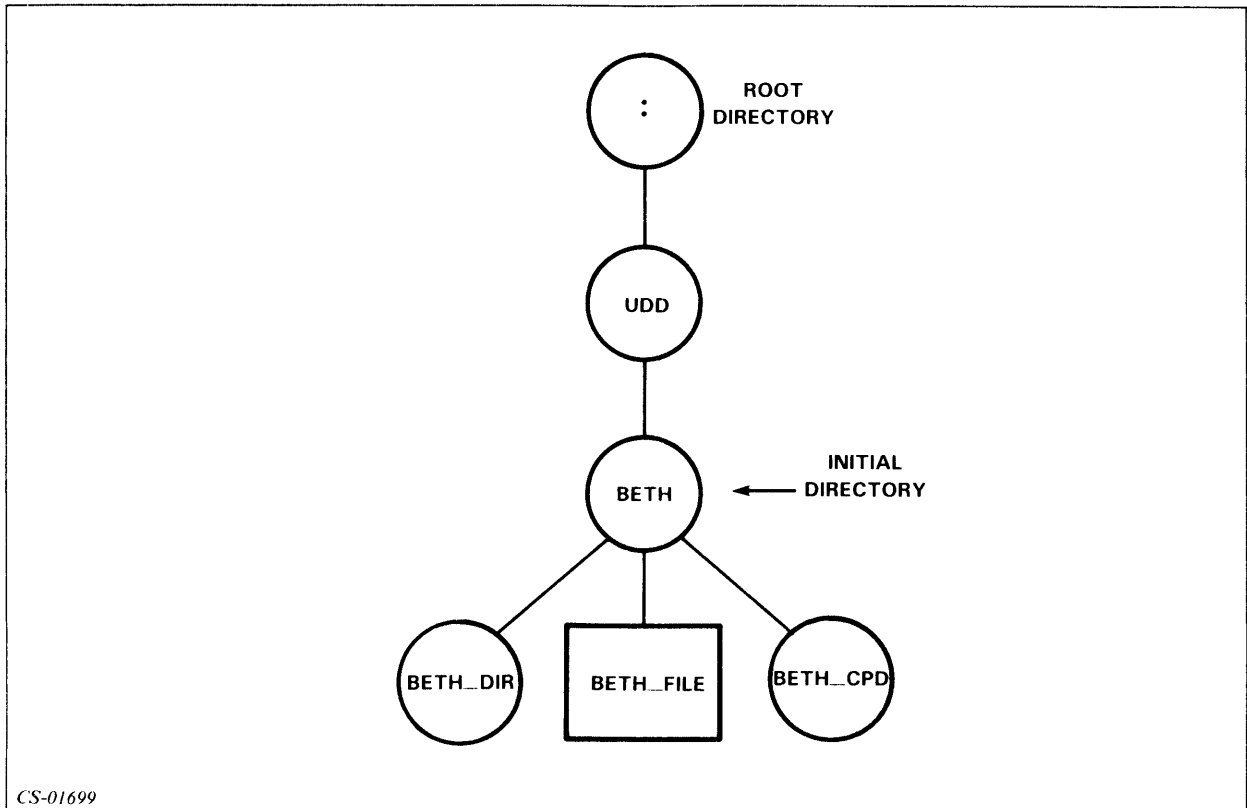


Figure 4.3 Partial Directory Structure

Use the **DIRECTORY** switch on the **CREATE** command to build a directory. Also, use the **MAXSIZE** switch to create a Control Point Directory (1 block = 512 characters).

**Example 1**

```
CREATE/DIRECTORY,BETH_DIR
```

**Example 2**

```
CREATE/DIRECTORY/MAXSIZE=100,BETH_DIR
```

Now do Exercise 4-1 on the next page.



### Exercise 4-1

1. Use the following labels to fill in the blanks in the directory structure shown in Figure 4.4.

:  
 UPD  
 UDD  
 User directory  
 User file  
 UTIL  
 @ or PER

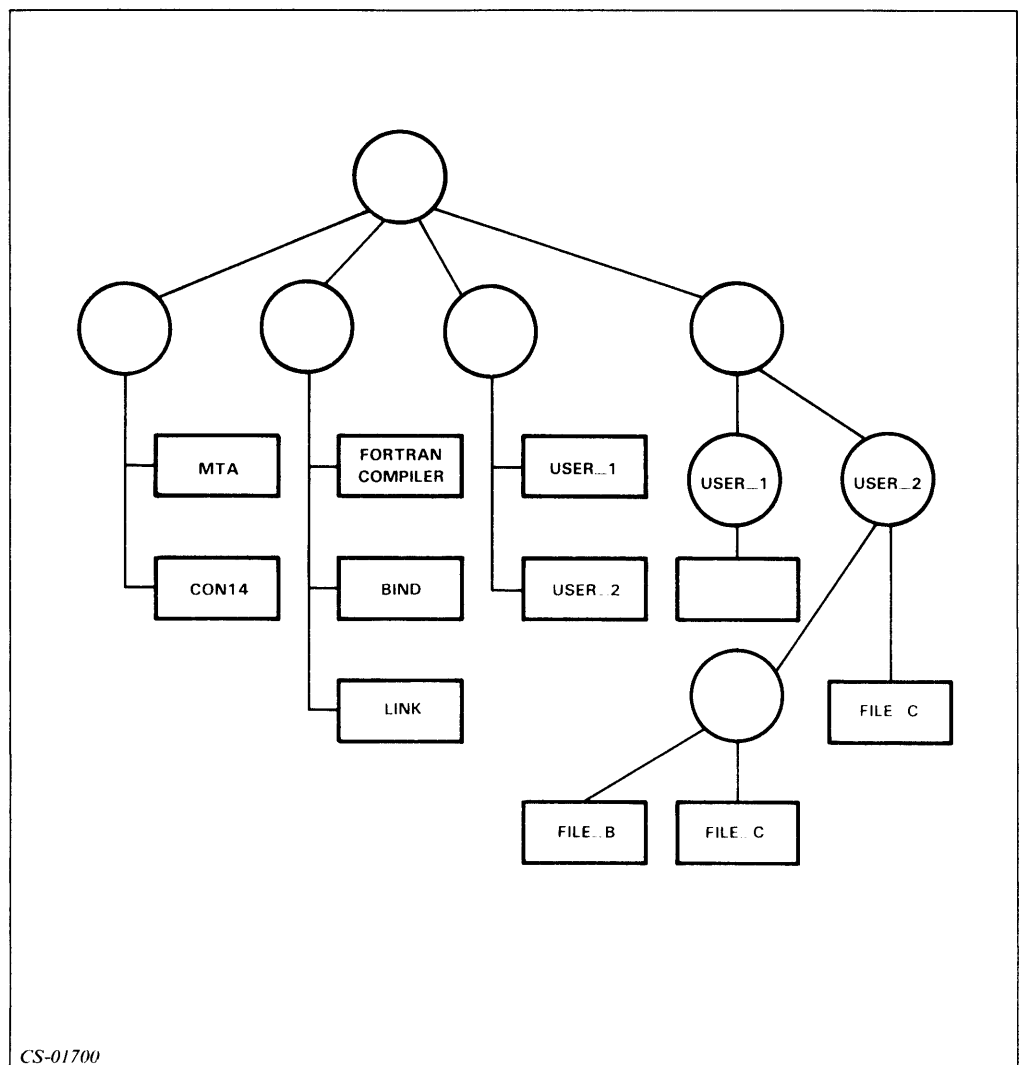


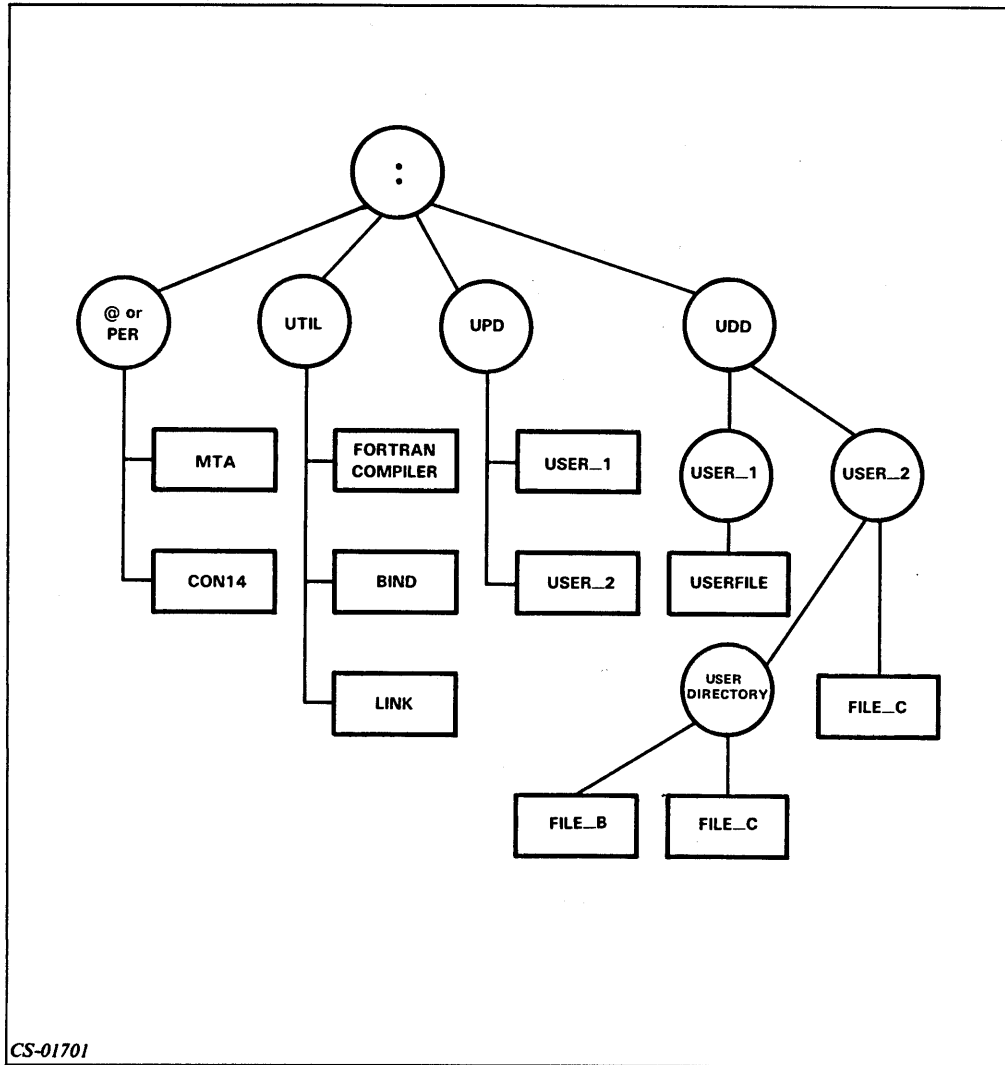
Figure 4.4

2. Write the command to add the directory file named REPORTDIR to UTIL.
- 

Check your answers on the next page.

## Exercise 4-1 Answers

1.



CS-01701

Figure 4.5

### 2. CREATE/DIR,REPORTDIR

If you answered all the questions correctly, continue with Module 4 by restarting the Module 4 audiotape. Otherwise, review the material and do this exercise again before you continue.

## Pathnames

A pathname is a route through the directory structure used to locate a particular file.

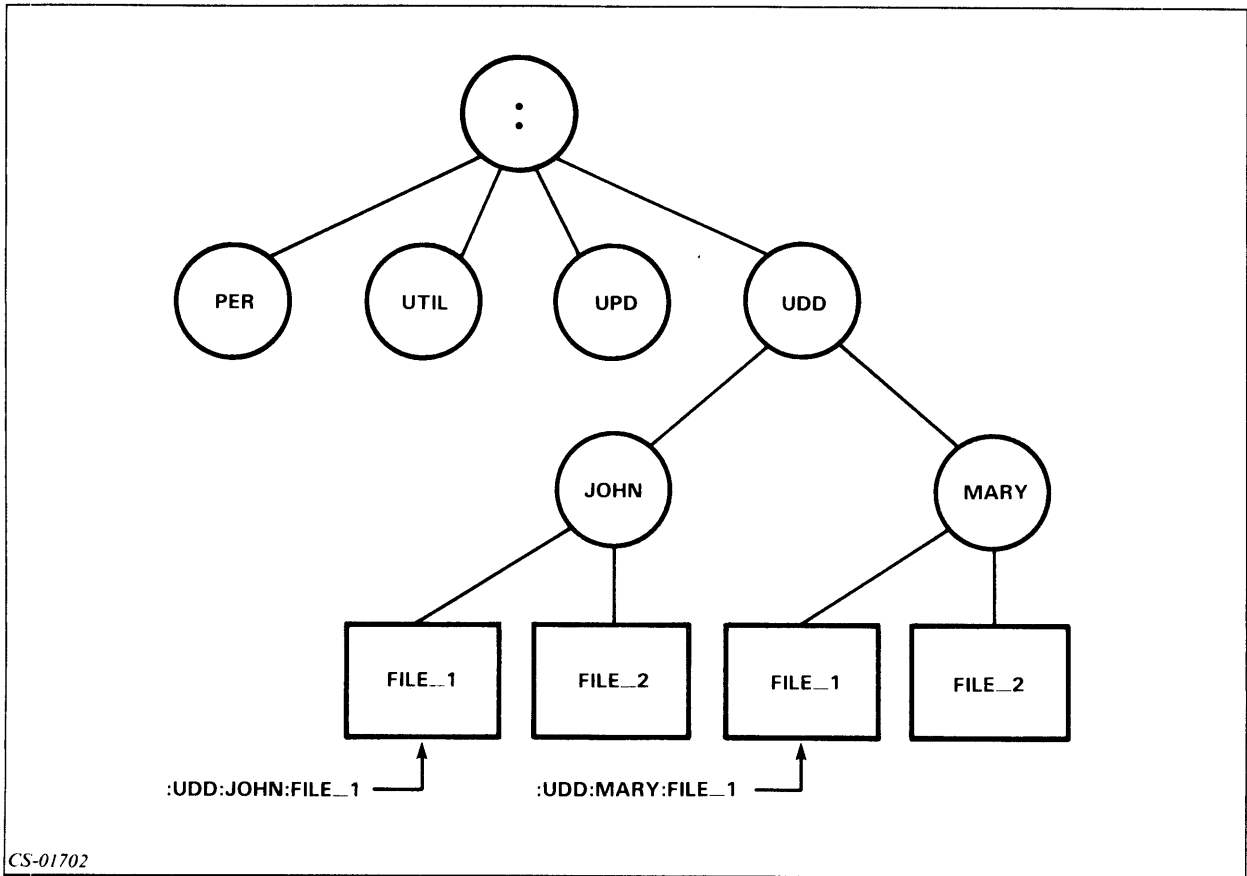
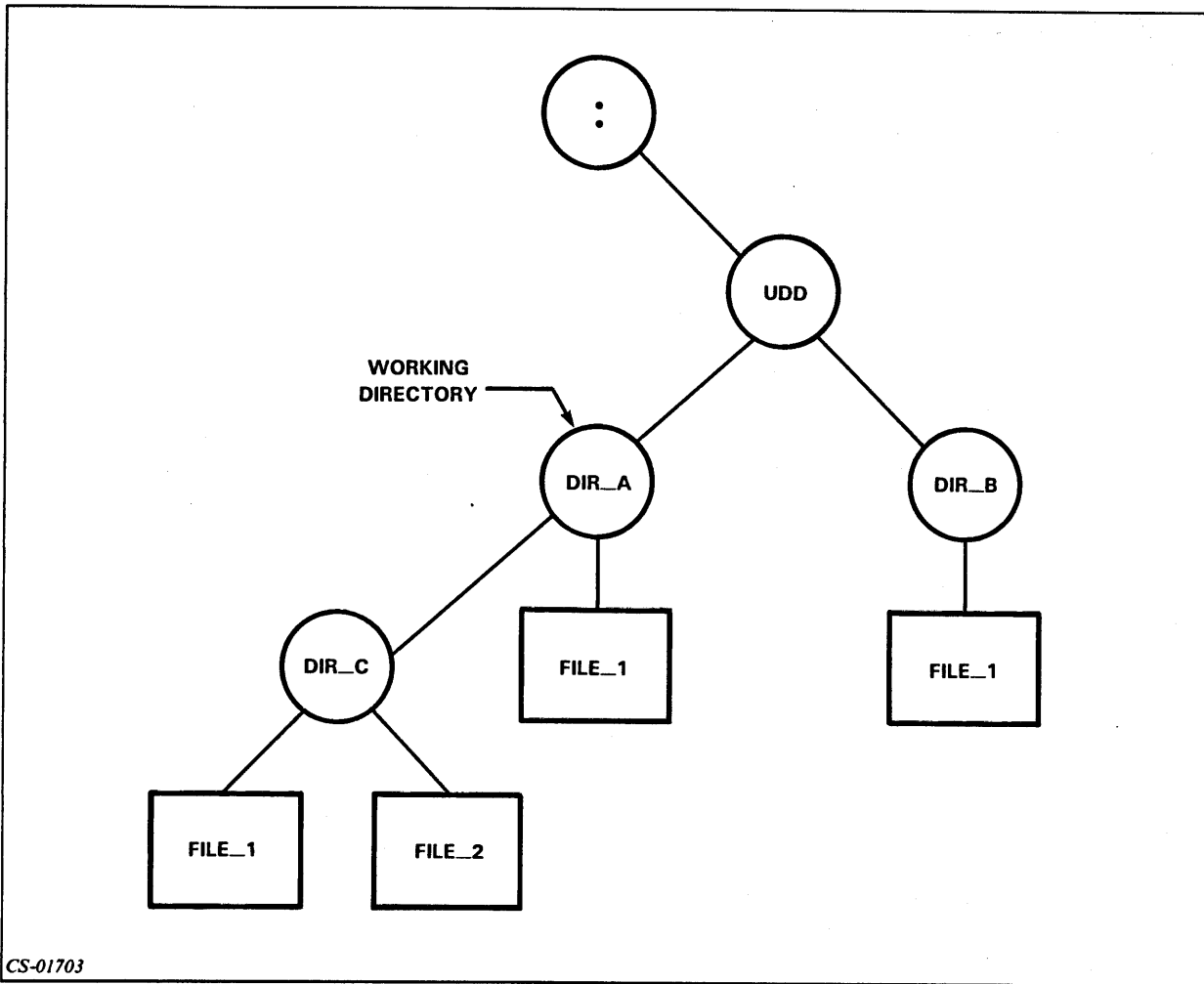


Figure 4.6 Directory Tree with Two Pathnames

The *working directory* is a reference point in the directory structure.

These CLI commands help you to move through the directory structure shown in Figure 4.7.

- |                        |   |
|------------------------|---|
| <b>DIRECTORY</b>       | Returns name of your working directory.     |
| <b>:UDD:DIR_A</b>      |   |
| <b>DIRECTORY DIR_C</b> | Makes DIR_C your current working directory. |
| <b>DIRECTORY/I</b>     | Returns to initial working directory.       |



CS-01703

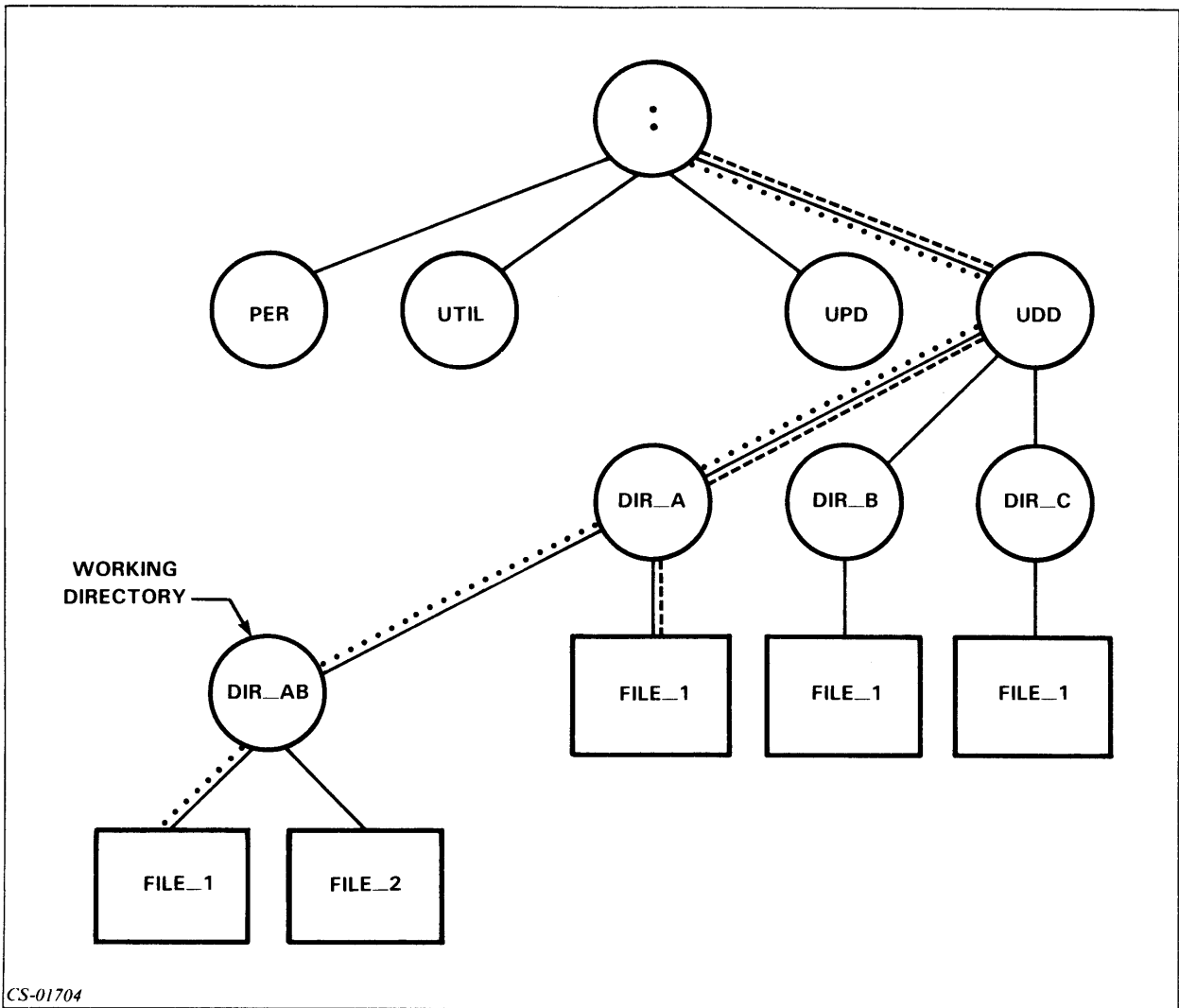
Figure 4.7 Working Directory

A *pathname* is a path through the directory structure to a particular file. A pathname consists of an optional prefix, and/or a series of filenames separated by colons.

In a pathname, each filename, except the last, must be a directory, and each must be inferior to the previous. A fully qualified pathname starts with a colon. Valid prefixes for pathnames are shown in Table 4.A.

Prefix	Result
:	Start with root directory. (Also used to separate files in a pathname.)
↑	Move up to superior directory.
=	Start with working directory.
@	Start with peripheral directory.

Table 4.A Pathname Prefixes



CS-01704

Figure 4.8 Hypothetical Directory Structure

**Example 1**

Assume DIR\_AB in Figure 4.8 is your working directory. To access FILE\_1, which is subordinate to DIR\_AB, enter the pathname **FILE\_1**, or **=FILE\_1**, or **:UDD:DIR\_A:DIR\_AB:FILE\_1**.

**Example 2**

To access FILE\_1, which is subordinate to DIR\_A, enter the pathname **]FILE\_1**, or **:UDD:DIR\_A:FILE\_1**.

**Example 3**

The PATHNAME command displays the fully qualified pathname. Assume DIR\_AB is your working directory.

```
PATHNAME FILE_1
:UDD:DIR_A:DIR_AB:FILE_1
```

Now do Exercise 4-2 on the next page.





## Exercise 4-2

Directions: Refer to Figure 4.9 to answer the following questions.

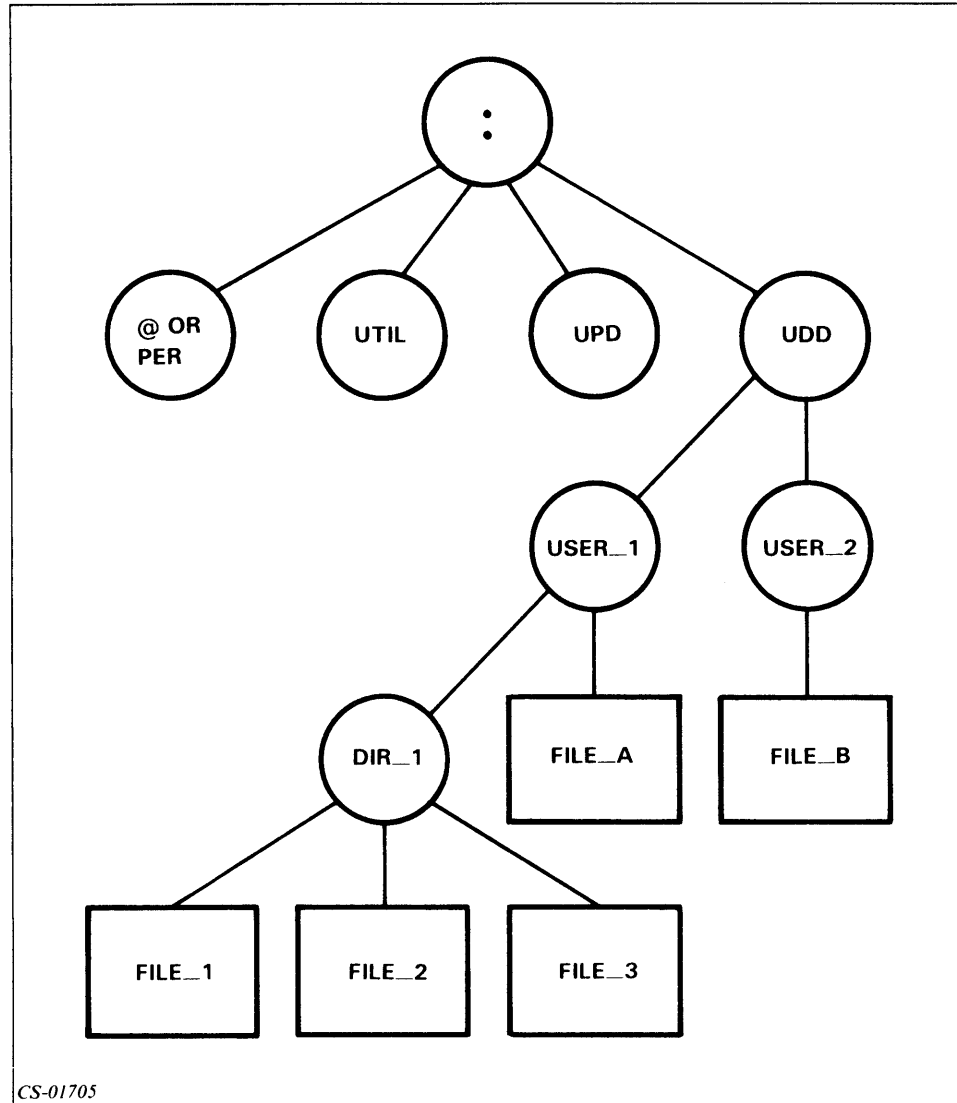


Figure 4.9

Assuming USER\_1 is your working directory in Figure 4.9, write the pathnames for each of the following files:

1. FILE\_1 \_\_\_\_\_
2. FILE\_A \_\_\_\_\_
3. FILE\_B \_\_\_\_\_

Assuming USER\_2 is your working directory in Figure 4.9, write the pathnames for each of the following files:

4. FILE\_2 \_\_\_\_\_

5. FILE\_A \_\_\_\_\_

6. FILE\_B \_\_\_\_\_

Check your answers on the next page.

## **Exercise 4-2**

### **Answers**

1. **DIR\_1:FILE\_1 or :UDD:USER\_1:DIR\_1:FILE\_1**
2. **FILE\_A or :UDD:USER\_1:FILE\_A**
3. **↑USER\_2:FILE\_B or :UDD:USER\_2:FILE\_B**
4. **↑USER\_1:DIR\_1:FILE\_2 or :UDD:USER\_1:DIR\_1:FILE\_2**
5. **↑USER\_1:FILE\_A or :UDD:USER\_1:FILE\_A**
6. **FILE\_B or :UDD:USER\_2:FILE\_B**

If you answered all of the questions correctly, continue with Module 4 by restarting the Module 4 audiotape. Otherwise, review the material and do this exercise again before you continue.

## Searchlists and Templates

### Searchlists

A *searchlist* is a list of directories to be searched in sequence when a file is not found in the working directory. The searchlist is only searched when the entry specified has no prefix.

#### Example 1

**SEARCHLIST**

:UDD, :UTIL

#### Example 2

**SEARCHLIST :UDD,:UTIL,:UDD:MYNAME:MYDIR**

**SEARCHLIST**

:UDD, :UTIL, :UDD:MYNAME:MYDIR

### Templates

Charac- ter	Matches
*	Matches any one character, except a period or a space.
-	Matches any string of characters, except those including a period.
+	Matches any series of characters, including those with a period.

Table 4.B Template Characters

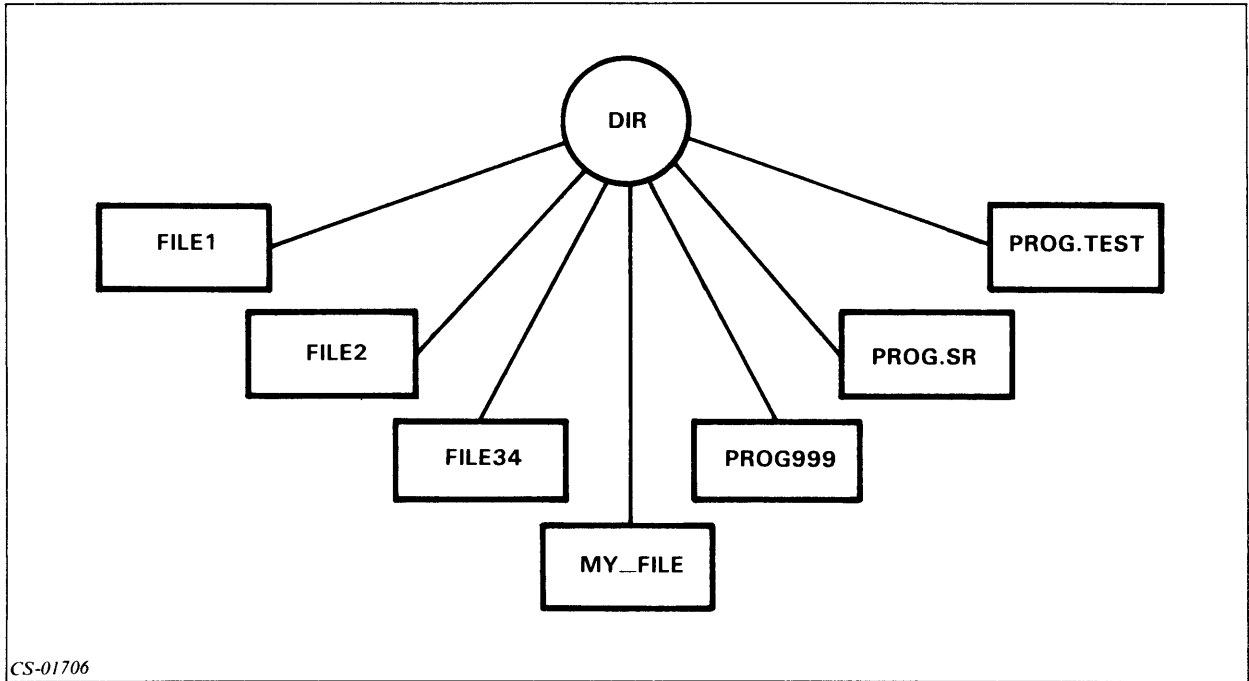


Figure 4.10 The Use of Templates

Template	Filenames Matched
FILE1	FILE1
FILE*	FILE1, FILE2
FILE-	FILE1, FILE2, FILE34
PROG.-	PROG.SR, PROG.TEST
PROG+	PROG.SR, PROG999, PROG.TEST
+	All filenames
-.-	PROG.SR, PROG.TEST

Table 4.C Templates

Now do Exercise 4-3 on the next page.



## Exercise 4-3

**Directions:** Use Figure 4.11 to answer the following questions.

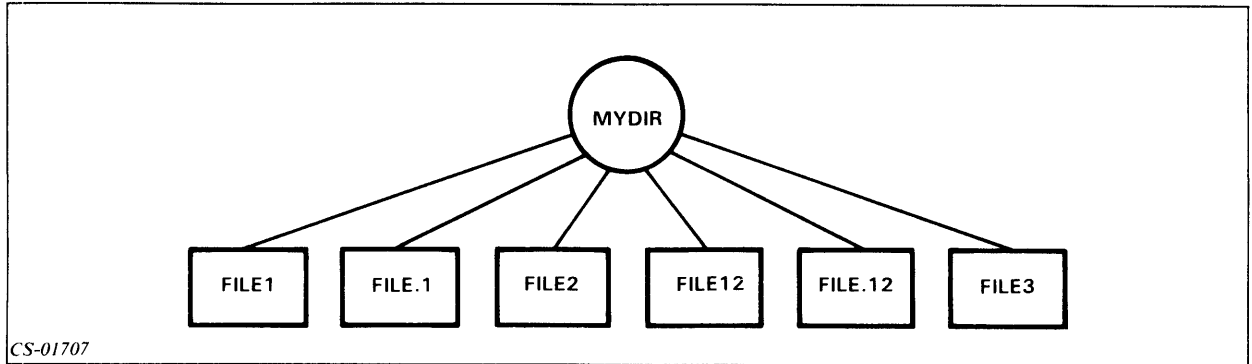


Figure 4.11

In Figure 4.11, which files are selected by the following templates? Note that MYDIR is the working directory.

1. FILE\* \_\_\_\_\_
2. FILE- \_\_\_\_\_
3. FILE+ \_\_\_\_\_
4. FILE1\* \_\_\_\_\_
5. FILE1+ \_\_\_\_\_
6. FILE1- \_\_\_\_\_

Check your answers on the following page.

## **Exercise 4-3**

### **Answers**

1. FILE1, FILE2, FILE3
2. FILE1, FILE2, FILE12, FILE3
3. All files.
4. FILE12
5. FILE1, FILE12
6. FILE1, FILE12

If you answered all the questions correctly, continue with Module 4 by restarting the Module 4 audiotape. Otherwise, review the material and do this exercise again before you continue.

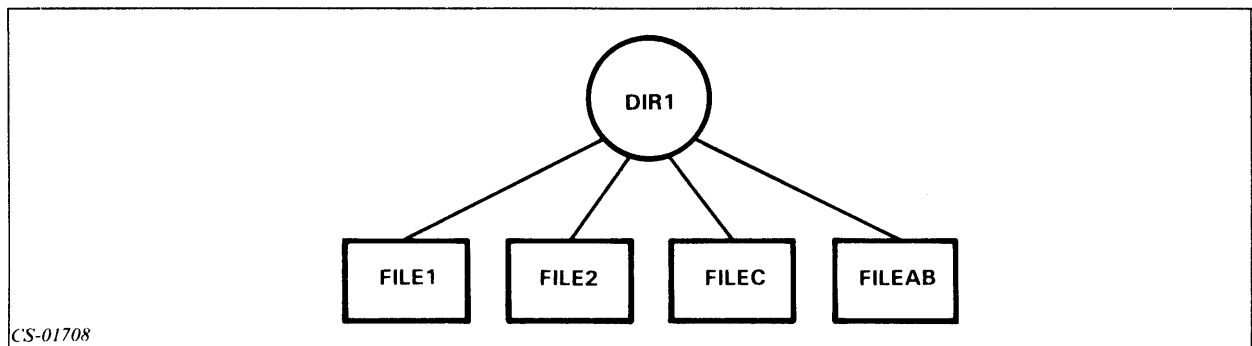


# Filestatus

The FILESTATUS command displays the names of the files in your working directory.

Switch	Result
/ASSORTMENT	Display type, date and time created, and length of file.
/DCR	Display date created.
/LENGTH	Display length of file.
/TCR	Display time and date created.
/TYPE	Display type of file.

Table 4.D Switches on the FILESTATUS Command



CS-01708

Figure 4.12 Directory Structure of DIR\_ONE

**Example 1**  
**FILESTATUS**

```

FILE1
FILE2
FILEC
FILEAB
  
```

**Example 2**  
**FILESTATUS/ASSORTMENT**

```

FILE1  UDF    25-SEP-79    10:54:50    1090
FILE2  UDF    25-OCT-80    10:50:50     80
FILEC  PRG    10-NOV-80    09:20:15     70
FILEAB UDF    11-OCT-80    10:20:19    100
  
```

**Example 3**  
**FILESTATUS/TYPE FILE\***

```

FILE1  UDF
FILE2  UDF
FILEC  PRG
  
```

Now do Exercise 4-4 on the next page.



### Exercise 4-4

**Directions:** Given the directory structure in Figure 4.13, which file(s) will be selected by the following commands?

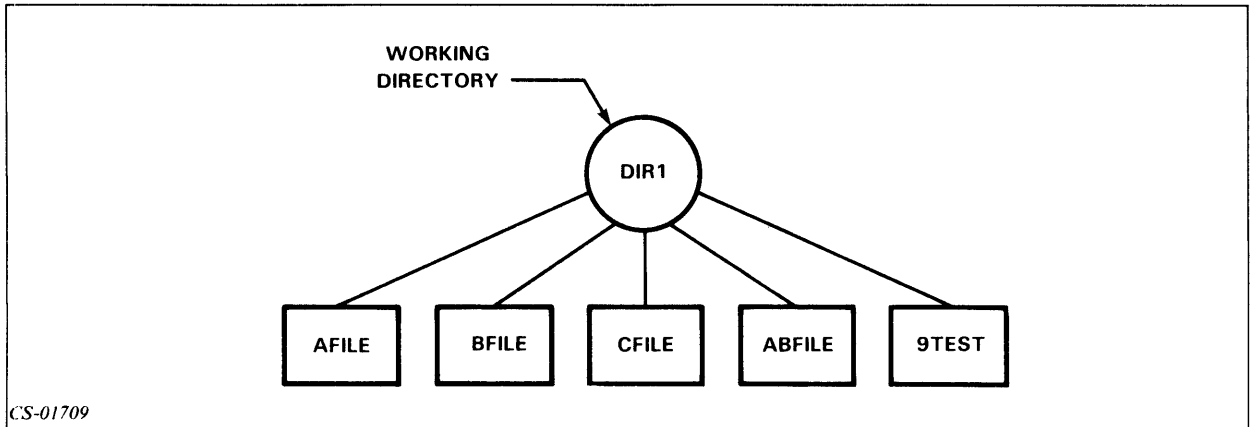


Figure 4.13

1. FILESTATUS \_\_\_\_\_
2. FILESTATUS -FILE \_\_\_\_\_
3. FILESTATUS \*FILE \_\_\_\_\_
4. FILESTATUS \*TEST \_\_\_\_\_

Check your answers on the next page.

## **Exercise 4-4**

### **Answers**

1. AFILE BFILE CFILE ABFILE 9TEST
2. AFILE BFILE CFILE ABFILE
3. AFILE BFILE CFILE
4. 9TEST

If you answered all of the questions correctly, continue to Lab Activity 4-1 and the Module 4 Test. Otherwise, review the material and do the exercise again before you continue.

## Lab Activity 4-1

**Directions:** Enter the command or commands necessary to accomplish each of the following tasks. There may be more than one way to perform each operation. If you have any problems, remember to use the HELP command for assistance. If that does not provide enough assistance, turn to the solution on the following page.

1. Determine what your working directory is.
2. Display the name and some information about all files that are inferior to the working directory. (These should be the files created in the Module 3 Lab Activity.)
3. Create a directory inferior to your working directory. Call it MYDIR and allow an unlimited amount of space for it.
4. Make MYDIR your working directory. Verify the success of this operation by repeating Step 1.
5. Determine if MYDIR has any inferior files. (There should be none.)
6. Change the working directory back to what it was when you signed on to the system.
7. Create a Control Point Directory inferior to your working directory. Allow for 10 blocks of space in the CPD and all associated files. Call it MYCPD.
8. Repeat Step 2. Be sure to notice the file type associated with MYDIR and MYCPD. (Use the assortment switch.)
9. Determine your current searchlist.
10. Determine the pathname of MYCPD.
11. Add MYCPD to your searchlist. (You saw the pathname for MYCPD in Step 10.)
12. Determine the pathname of the SPEED editor (SPEED.PR). Note the directory in which it is located.
13. Move from your current working directory to the directory immediately superior to it.
14. Display the new working directory. (Notice your current location in Figure 4.2.)
15. Log off.

Check your answers on the next page.

## Lab Activity 4-1

### Answers

Listed below are solutions to each question. Each solution listed is not the only one, but only one of the methods that work. If your answer is not the same, it still may be correct.

1. **DIRECTORY**
2. **FILESTATUS/ASSORTMENT**
3. **CREATE/DIRECTORY MYDIR**
4. **DIRECTORY MYDIR  
DIRECTORY**
5. **FILESTATUS/ASSORTMENT**
6. **DIRECTORY/I**
7. **CREATE/DIRECTORY/MAXSIZE=10 MYCPD**
8. **FILESTATUS/ASSORTMENT**
9. **SEARCHLIST**
10. **PATHNAME MYDIR**
11. **SEARCHLIST (include your searchlist from Step 9), (include the pathname from Step 10.)**
12. **PATHNAME SPEED.PR**
13. **DIRECTORY ↑**
14. **DIRECTORY**
15. **BYE**

If you completed Lab Activity 4-1 successfully, go on to the Module 4 Test. If not, review this module before continuing to the Test.

# Module 4 Test

Directions: Fill in the blanks numbered 1 through 5 in the directory structure in Figure 4.14.

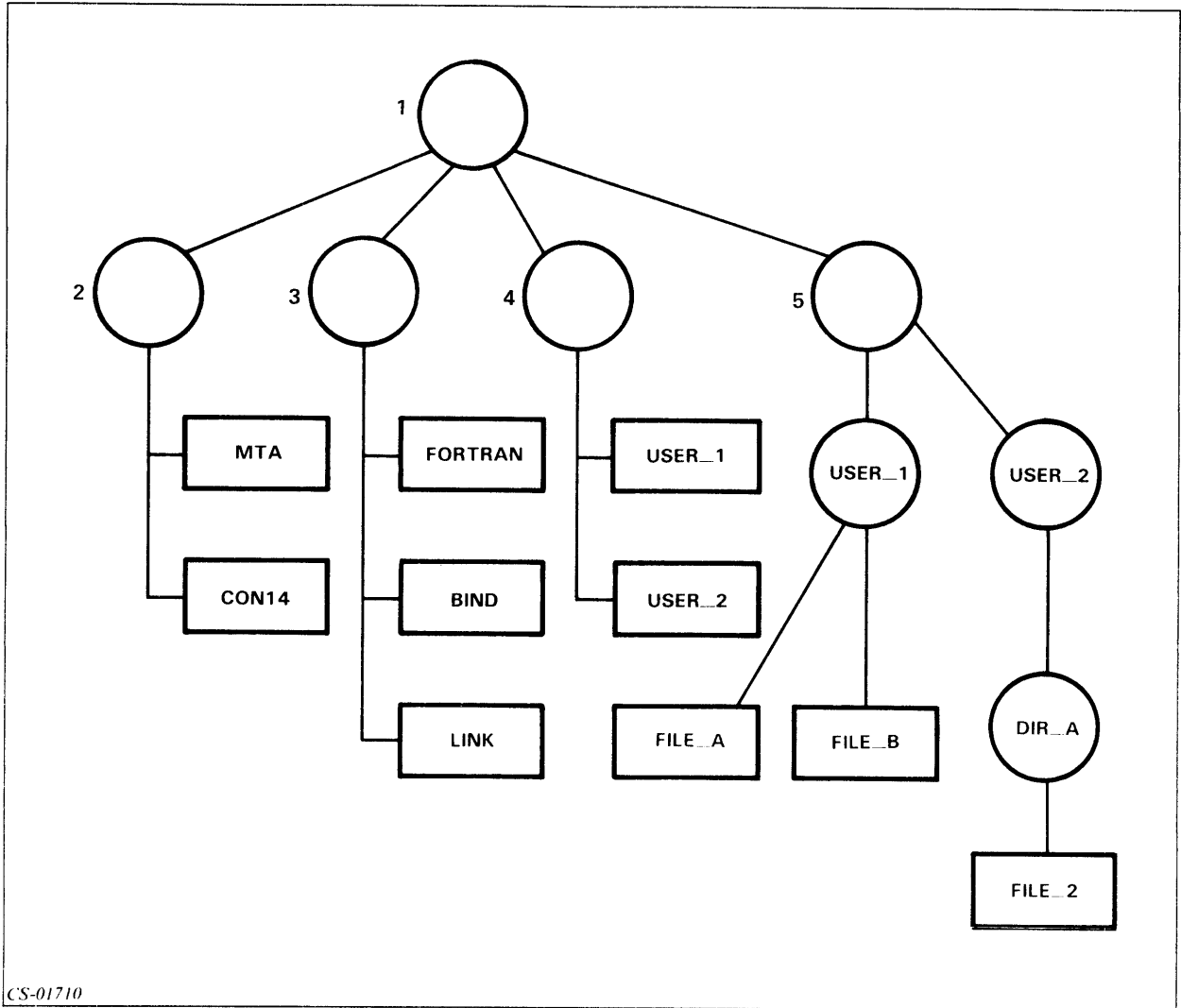
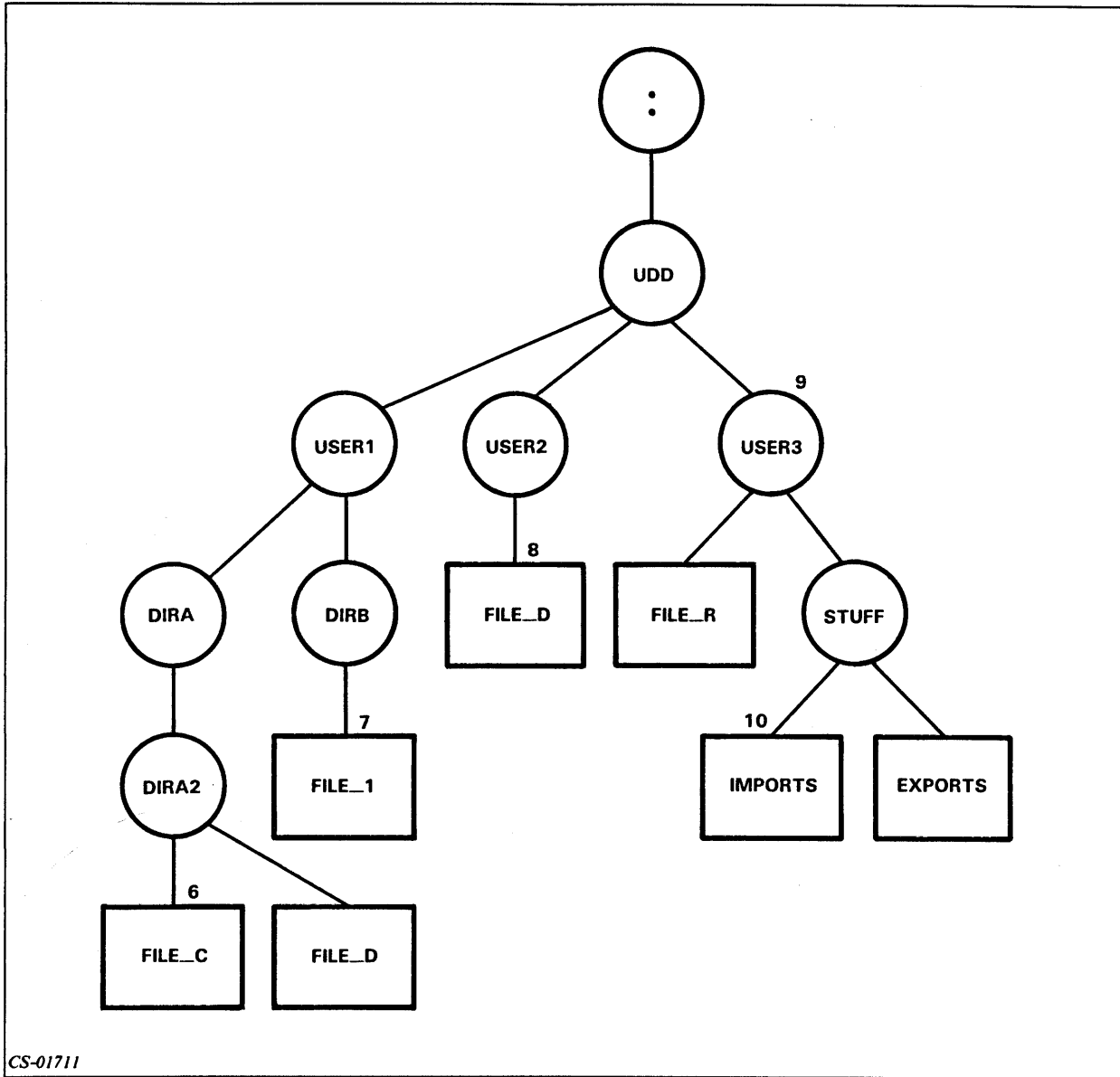


Figure 4.14

Directions: Given this partial directory structure in Figure 4.15, write the fully qualified pathname for the files labeled 6 through 10.



CS-01711

Figure 4.15

- 6. \_\_\_\_\_
- 7. \_\_\_\_\_
- 8. \_\_\_\_\_
- 9. \_\_\_\_\_
- 10. \_\_\_\_\_



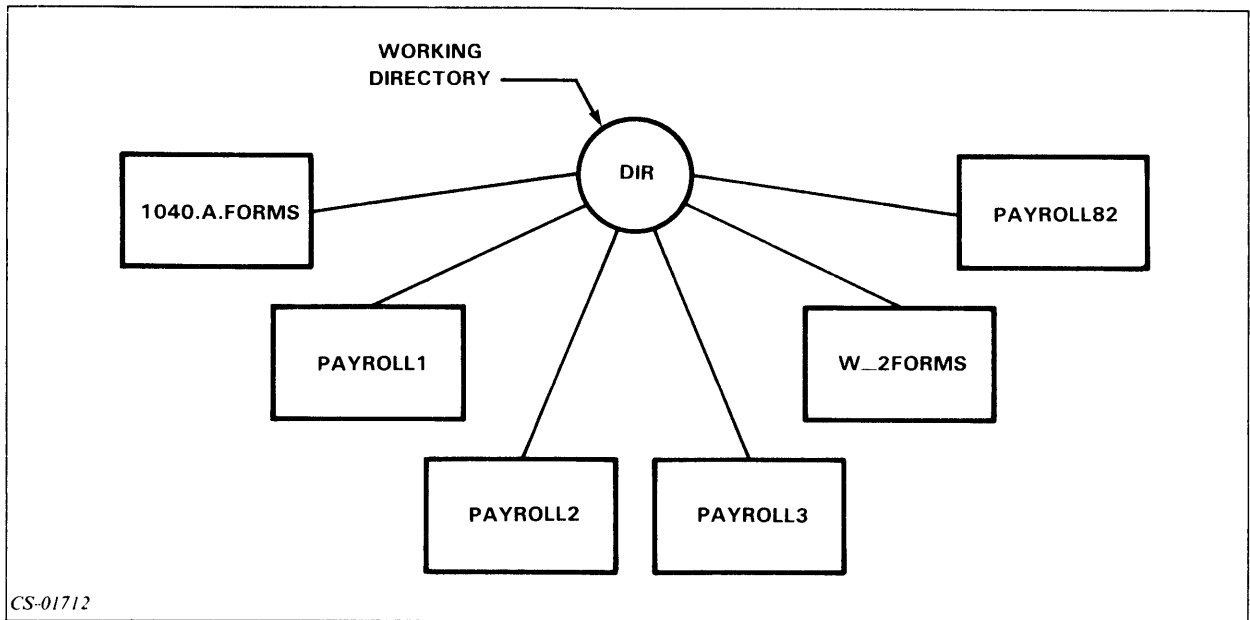


Figure 4.16

**Directions:** Write the template to make the following selections from the directory structure in Figure 4.16.

11. All of the files with PAYROLL in the name. \_\_\_\_\_
12. PAYROLL1, PAYROLL2, PAYROLL3.  
\_\_\_\_\_
13. All of the files with FORMS in the name. \_\_\_\_\_
14. All of the files except those with a period (.). \_\_\_\_\_
15. All of the files with a 2 in the name. \_\_\_\_\_

**Directions:** Write the CLI command to:

16. Create a directory with the name SUPER\_DIR. \_\_\_\_\_
17. Create a Control Point Directory with 10 blocks of space called LITTLE\_C.P.D.  
\_\_\_\_\_
18. List all of the files in your working directory. \_\_\_\_\_
19. Display your working directory. \_\_\_\_\_
20. Change your working directory to :UTIL. \_\_\_\_\_

21. Display your searchlist. \_\_\_\_\_
22. Change your searchlist to include only :UTIL. \_\_\_\_\_
23. Determine the pathname of FORTRAN.PR. \_\_\_\_\_

Now check your answers to the Module 4 Test in Appendix A. If you have answered all the questions correctly, continue to Module 5. Otherwise, go back and review the material in Module 4 and take the Module 4 Test again.

This concludes Module 4.

# Module 5

## File Security

### Introduction

This module discusses the need for and the implementation of file security. It explains various types of protection against unauthorized use of files and directories.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Identify the access permitted by each of the access types for directory and nondirectory files.
2. Change the Access Control List (ACL) for a file after determining the ACL of the file.
3. Change the default Access Control List (DEFACL) for a file after determining the DEFACL of the file.
4. Describe the superuser privilege.

### Resources

To complete this module, you will need:

- Module 5 audiotape.
- Module 5 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

Module 5 discusses the following topics:

1. File security
  - a. Need for security
  - b. Solution to the need
    - Access Control List
    - Types of access
    - The ACL command
    - The DEFACL command
2. Superuser privilege

Now start the Module 5 audiotape. As you listen, follow along in Module 5 of your *Student Guide*.

## File Security

File security protects your files and allows access only to authorized users. The *Access Control List (ACL)* is a list of usernames and the types of access that these users are permitted. There are five types of access:

- Owner
- Write
- Append
- Read
- Execute

Access	Abbreviation	Nondirectory File	Directory File
Execute	E	User can execute the file.	User can use the directory in a path-name.
Read	R	User can read (examine) data in the file.	User can examine the list of files.
Append	A	N/A	User can insert new files in directory.
Write	W	User can modify the contents of the file.	User can insert and delete files and change ACLs of files in the directory.
Owner	O	User can change file's ACL or delete files.	User can change directory's ACL or delete the directory.

Table 5.A Five Types of File Access

## ACL

The ACL command displays or resets the Access Control List for a file. The access given is that indicated by the first match in the ACL list.

To display an ACL:

**Example 1**  
**ACL MYFILE**  
 JOHN OWARE

Indicates that John has all access privileges. All other users have no access.

**Example 2**  
**ACL OTHERFILE**  
 JOHN E  
 SUE1 RE  
 SUE\* WRE

Indicates that John has only execute access, SUE1 has read and execute accesses, and SUE followed by any character except a 1 or a period has write, read, and execute access. All other users have no privileges.

**Example 3**  
**ACL OTHERFILE**  
JOHN E  
SUE\* WRE  
SUE1 RE

Indicates that SUE1 has write, read, and execute access. All other accesses are the same as in Example 2.

To reset an ACL:

**Example 4**  
**ACL MYFILE,JOE,RE**

Change the ACL for MYFILE to allow only JOE to have read and execute access. All others have no access.

**Example 5**  
**ACL OTHERFILE,JOHN,OWARE,SUE,WARE,+,RE**

Change the ACL for OTHERFILE to allow JOHN all five types of access. SUE has write, append, read, and execute access, but not owner access. All other users have read and execute access only.

## Default ACL

The *default ACL* is the ACL assigned at the time of creation of a file. The DEFACL command sets or displays the default ACL of a file. The new default ACL remains in effect only for the current session.

**Example 1**  
To determine the current default ACL:

```
DEFACL  
GARY OWARE
```

**Example 2**  
To change the default ACL:

```
DEFACL JOHN,OWARE,SUE,RE,+,E
```

## Superuser

The *superuser privilege* is assigned by the system manager. This privilege:

- Allows all accesses to all files.
- Allows access to any file.
- Causes the prompt character to change from ) to \*).

### Example

```
SUPERUSER
```

```
OFF
```

```
SUPERUSER ON
```

```
*)SUPERUSER
```

```
ON
```

```
*)SUPERUSER OFF
```

Now do Exercise 5-1 on the next page.





## Exercise 5-1

**Directions:** Fill in the blanks in the table below:

Access	Abbreviation	Nondirectory File	Directory File
1. _____	E	User can execute the file.	_____
2. _____	R	_____	User can examine list of files.
3. _____	A	N/A	_____
4. _____	W	_____	User can insert and delete files and change ACLs of files in the directory.
5. _____	O	User can change file's ACL or delete files.	_____

Table 5.B Exercise: ACLs

**Directions:** Write the command that:

6. Changes the access to the file OURFILE to permit all users to read and execute the file. \_\_\_\_\_
7. Makes all files that will be created have access by John only and John has all accesses. \_\_\_\_\_
8. Determines the setting of the superuser privilege. \_\_\_\_\_

Now check your answers on the next page.

## **Exercise 5-1**

### **Answers**

1. EXECUTE. User can use the directory in a pathname.
2. READ. User can read (examine) data in the file.
3. APPEND. User can insert new files in the directory.
4. WRITE. User can modify the contents of the file.
5. OWNER. User can change the directory's ACL or delete the directory.
6. ACL,OUTFILE,+RE
7. DEFACL,JOHN,OWARE
8. SUPERUSER

If you answered all of the questions correctly, go on to Lab Activity 5-1 and the Module 5 Test. Otherwise, review the material in Module 5 and do the exercise again before you continue.

## Lab Activity 5-1

**Directions:** After logging on to an AOS or AOS/VS system, try the following activities. If you have any problems, refer to the answers on the following pages.

1. Determine your default Access Control List.
2. Create a file called D\_FILE\_1.
3. Change your default Access Control List to give yourself owner, write, read, and execute access to any files that you create. Verify the success of this action.
4. Create a file called D\_FILE\_2.
5. Examine the Access Control Lists of D\_FILE\_1 and D\_FILE\_2. Note the differences.
6. Return your default Access Control List to what it was in Step 1. Verify that you have been successful.
7. Determine the ACL of MYDIR. (You created this file in the Module 4 Lab Activity.) Is it the same as your default ACL?
8. Change the Access Control List of MYDIR to allow yourself all privileges except execute access, and user John execute access. Verify the result.
9. Try to make MYDIR your working directory. What happened? Why?
10. Now change the ACL of MYDIR to give yourself all privileges and make it the working directory.

Check your answers on the next page.

## Lab Activity 5-1 Answers

1. **DEFACL**
2. **CREATE D\_FILE\_1**
3. **DEFACL MYID,OWARE,+E**  
**DEFACL**
4. **CREATE D\_FILE\_2**
5. **ACL D\_FILE\_1**  
**ACL D\_FILE\_2**

The ACL for D\_FILE\_1 should be your old default ACL. D\_FILE\_2 should have the new default ACL.

6. **DEFACL** (Include the results returned in Step 1.)
7. **ACL MYDIR** (It should be the same as your original default ACL.)
8. **ACL MYDIR,MYID,OWAR,JOHN,E**  
**ACL MYDIR**
9. **DIR MYDIR** (You should have received a message, indicating that you did not have the proper access to perform this action.)
10. **ACL MYDIR,MYID,OWARE**  
**DIR MYDIR**

If you answered all of the questions correctly, continue to the Module 5 Test. Otherwise, review the material in Module 5 and do Lab Activity 5-1 again before you continue.

## Module 5 Test

**Directions:** Answer the following question:

1. Describe the superuser privilege.

\_\_\_\_\_

**Directions:** Write the command to:

2. Determine the ACL of FILE\_1. \_\_\_\_\_
3. Change the ACL of FILE\_1 to allow John all accesses and Mary only read access.  
\_\_\_\_\_
4. Determine your current default ACL. \_\_\_\_\_
5. Change your default ACL to allow all access to all users. \_\_\_\_\_
6. Turn on the superuser privilege. \_\_\_\_\_
7. Turn off the superuser privilege. \_\_\_\_\_

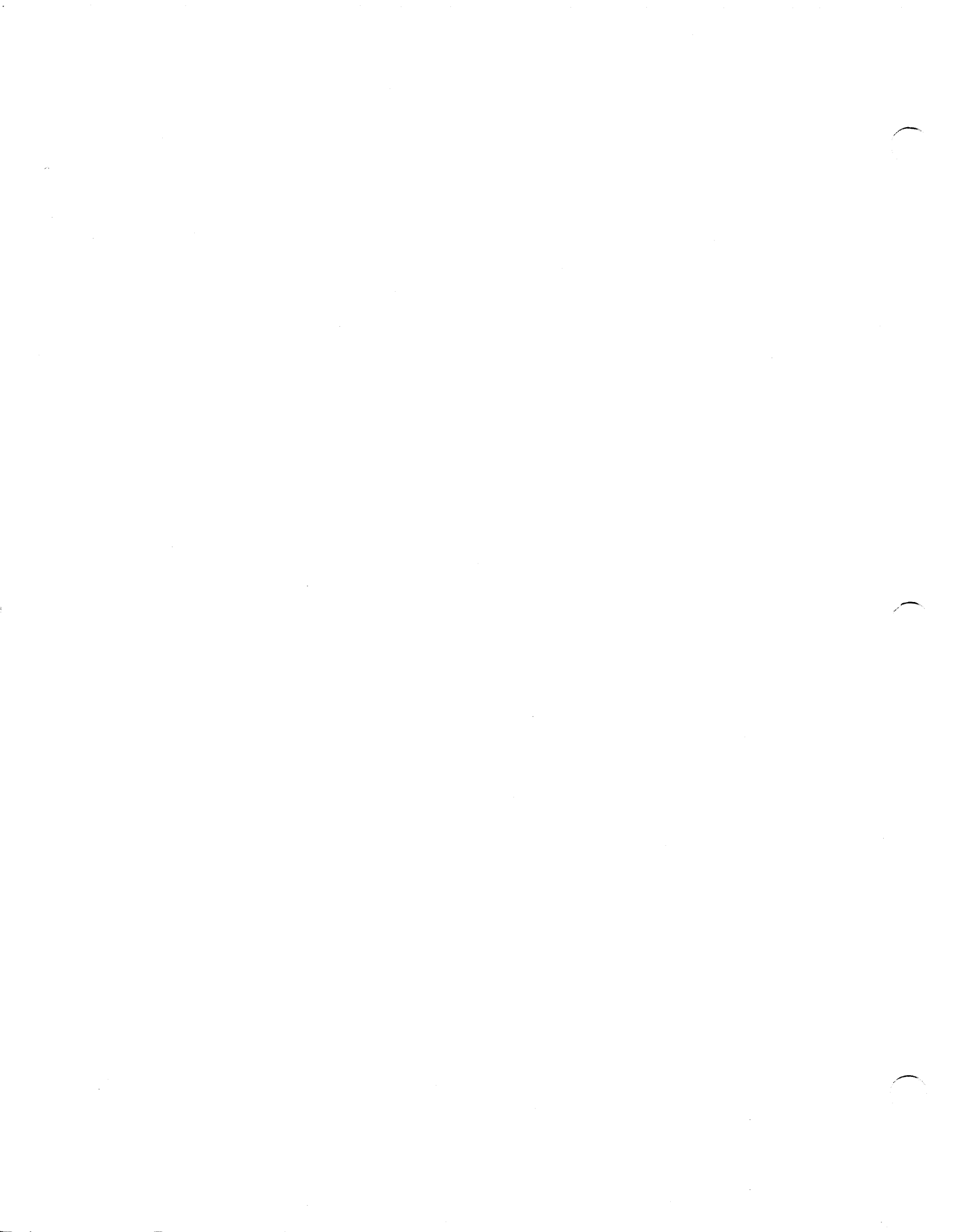
**Directions:** Complete the following table by filling in the blanks with a description of the access permitted for each access type.

Access	Abbreviation	Nondirectory File	Directory File
Execute	E	_____	_____
Read	R	_____	_____
Append	A	_____	_____
Write	W	_____	_____
Owner	O	_____	_____

Table 5.C Module 5 Test: ACLs

Now check your answers to the Module 5 Test in Appendix A. If you answered all the questions correctly, go on to Module 6. Otherwise, review the material in Module 5 and take the test again before continuing.

This concludes Module 5.



# Module 6

## Queues

### Introduction

This module introduces the concept of a queue. It teaches you to use queues to print and process your data.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Describe the operations of a queue.
2. Describe the batch processing operations.
3. Write CLI command lines using the following commands:

QPRINT  
QBATCH  
QCANCEL  
QUNHOLD  
QHOLD  
QPLOT  
QPUNCH

### Resources

To complete this module, you will need:

- Module 6 audiotape.
- Module 6 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

Module 6 discusses the following topics:

1. Queue concepts
2. Batch concepts
3. Using batch and queues
4. Queue manipulation
5. Printing

Now start the Module 6 audiotape. As you listen, follow along in Module 6 of your *Student Guide*.



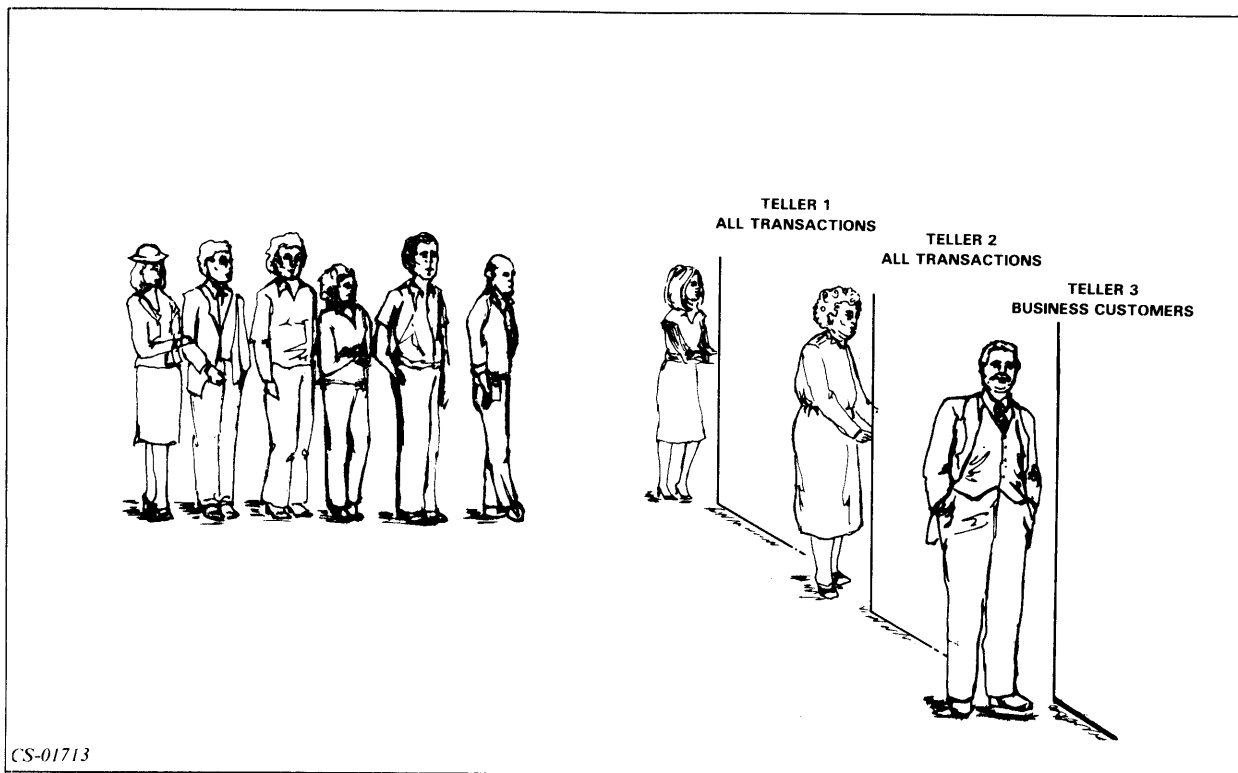


Figure 6.1 A Queue in a Bank

A *queue* is an ordered list of elements. AOS and AOS/VS have queues for:

- Batch processing
- Printing
- Plotting
- Punching (AOS only)

### QDISPLAY Command

The QDISPLAY command displays the queue status.

Format: **QDISPLAY**

Switches:

- /QUEUE= Display only a specific queue (BATCH\_INPUT,BATCH,OUTPUT...).
- /TYPE= Display only a specific type of queue (BATCH,PRINT,PUNCH,PLOT).

**Example 1**  
**QDISPLAY**

Display all queues.

```
BATCH__INPUT      BATCH      OPEN
BATCH__OUTPUT     PRINT      OPEN
* 362 D           NANETTE :QUEUE:NANETTE.OUTPUT.362

BATCH__LIST       PRINT      OPEN

LPT                PRINT      OPEN
  367 N           GEORGE :UDD:GEORGE:AMOD6:QSIXBI
  368 N           GEORGE :UDD:GEORGE:AMOD6:QSIX
  369 N           GEORGE :UDD:GEORGE:AMOD6:QSIX1
  370 N           GEORGE :UDD:GEORGE:AMOD6:QSIXI

LQP                PRINT      OPEN
* 366            MENARD :UDD:MENARD:OEM
```

**FLAGS EXPLANATION:**

D = /DELETE  
N = /NOTIFY  
\* = ACTIVE

**Example 2**  
**QDISPLAY/QUEUE=BATCH\_\_INPUT**

Display only the BATCH\_\_INPUT queue.

```
BATCH__INPUT      BATCH      OPEN
* 360 DN          CAROL :UDD:CAROL:TRAINING__CREDIT__SYSTEM:2020.CLI.003.JOB
  362            NANETTE :LD1:REGISTRATION__SYSTEM:MACROS:GENERATE__REPORTS
```

**FLAGS EXPLANATION:**

D = /DELETE  
N = /NOTIFY  
\* = ACTIVE

**Example 3**  
**QDISPLAY/TYPE=PRINT**

Display all PRINT queues.

BATCH__OUTPUT	PRINT	OPEN
BATCH__LIST	PRINT	OPEN
LPT	PRINT	OPEN
ID__LPT	PRINT	OPEN
LQP	PRINT	OPEN
* 97 N	GARYS :UDD:GARYS:N3:1ST:M1:SCR3.LP	
98 N	GARYS :UDD:GARYS:N3:INFO:CONTENTS	

FLAGS EXPLANATION:

N = /NOTIFY

\* = ACTIVE

## QBATCH Command

The QBATCH command creates a batch job and places it in the batch queue. Your job executes independently of your terminal. You don't need to be logged on for your job to run.

Format: **QBATCH ARGUMENT**

Switches:

- /HOLD Do not execute the job until released by owner.
- /NOTIFY Inform the user when job is complete.
- /I Create job from subsequent lines from terminal.
- /AFTER= Do not execute job until time specified.
- /JOBNAME= Assign the name that follows the = to the batch job.

**Example 1**  
**QBATCH XEQ NET**

**Example 2**  
**QBATCH/NOTIFY XEQ NET**

**Example 3**  
**QBATCH/HOLD/JOBNAME=PAYROLL XEQ COBOL FILE1**

Compiles FILE1 when released.

```
Example 4
QBATCH/I
)XEQ PROG1
)DIR DIR4
)XEQ PROG4
))
```

Execute PROG1 from current directory, then change to DIR4 and execute PROG4. While this is happening, you can be doing something else.

## **QHOLD Command**

The QHOLD command holds a queue entry by putting the job in a waiting status.

Format: QHOLD JOBNAME or SEQ NUMBER

```
Example 1
QHOLD PAYROLL
```

Holds entry called PAYROLL. (You must have specified /JOBNAME switch when the entry was created.)

```
Example 2
QHOLD 36
```

Holds queue entry number 36.

## **QUNHOLD Command**

The QUNHOLD command releases a held queue entry and puts the job back into active status.

Format: QUNHOLD JOBNAME or SEQ NUMBER

```
Example 1
QUNHOLD PAYROLL
```

Releases an entry called PAYROLL. (You must have specified /JOBNAME switch.)

```
Example 2
QUNHOLD 36
```

Releases queue entry number 36.

## QCANCEL Command

The QCANCEL command deletes a queue entry.

Format: QCANCEL SEQ NUMBER or JOBNAME

### Example 1

**QCANCEL JOB1**

Cancels the entry named JOB1.

### Example 2

**QCANCEL 36**

Cancels the entry with sequence number 36.

## QPRINT Command

The QPRINT command places an entry in the printer queue.

Format: QPRINT PATHNAME

Switch	Result
/NOTIFY	Informs the user when printing is finished.
/COPIES= <i>n</i>	Creates <i>n</i> copies of the output.
/TITLES	Includes titles on printed document.
/FORMS=	Uses special forms for printing.
/QUEUE=	Places in a specified queue.

Table 6.A QPRINT Command Switches

### Example 1

**QPRINT DATA1**

Prints file DATA1.

### Example 2

**QPRINT/NOTIFY/TITLES DATA1**

This notify message is displayed when the file printing is complete:

FROM PID3: (EXEC) @LPB COMPLETED :UDD:USER1:DATA1

The title looks like this:

UDD:USER1:DATA1                      10-MARCH-82    14:35:21    PAGE 1

### Example 3

**QPRINT/FORMS=FORM2 DATA1**

## QPLOT Command

The QPLOT command places an entry in the plotter queue.

Format: QPLOT PATHNAME

Switches:

- /COPIES=*n* Plot *n* copies of the file.
- /NOTIFY Inform user when plotting is finished.

Example

QPLOT/COPIES=3/NOTIFY DATA\_PLOT

## QPUNCH Command

The QPUNCH command places an entry in the punch queue (AOS only).

Format: QPUNCH PATHNAME

Switches:

- /COPIES=*n* Punch *n* copies of the file.
- /NOTIFY Inform user when punching is complete.

Example

QPUNCH/COPIES=2/NOTIFY DATA\_PUNCH

Now do Exercise 6-1 on the next page.

## Exercise 6-1

**Directions:** Match the following commands with their results.

- |                   |   |
|-------------------|---|
| 1. _____ QPRINT   | a. Prevent an entry from being processed until later. |
| 2. _____ QPLOT    | b. Delete an entry from a queue.                      |
| 3. _____ QPUNCH   | c. Submit an entry for CPU processing.                |
| 4. _____ QDISPLAY | d. Place an entry in the printer queue.               |
| 5. _____ QBATCH   | e. List the queues and their contents.                |
| 6. _____ QCANCEL  | f. Place an entry in the paper-tape punch queue.      |
| 7. _____ QHOLD    | g. Place an entry into the plotter queue.             |

**Directions:** Mark the following statements true or false.

- \_\_\_\_\_ The QPRINT command actually causes the file to be printed.
- \_\_\_\_\_ The QDISPLAY command can be used to print a file on the line printer.
- \_\_\_\_\_ You can use the /FORMS switch to have your file printed on special paper.
- \_\_\_\_\_ You can check the contents of the queues by using the QDISPLAY command.
- \_\_\_\_\_ The QPUNCH command is used to have output punched on a paper-tape punch.
- \_\_\_\_\_ If a queue entry is placed on hold, it must be released by the QRELEASE command.
- \_\_\_\_\_ An entry can be put on hold either by the QHOLD command or by the /HOLD switch.
- \_\_\_\_\_ To cancel an entry in the batch queue, use the QUNBATCH command.

Check your answers on the following page.

## **Exercise 6-1**

### **Answers**

1. d.
2. g.
3. f.
4. e.
5. c.
6. b.
7. a.
8. F
9. F
10. T
11. T
12. T
13. F
14. T
15. F

If you answered all of the questions correctly, continue to Lab Activity 6-1 and the Module 6 Test. Otherwise, review the material and do this exercise again before you continue.



## Lab Activity 6-1

**Directions:** Enter the commands to perform the following functions. If you have any problems, you can check the answers on the following page.

1. Create a batch entry that will execute a program named `JOB_ONE` and put the job on hold when you create it.
2. Determine the sequence number of the job created in Step 1 and release it.
3. Display all of the queues.
4. Create a job that will execute the following series of steps:
  - Change the working directory to `:UDD:MYDIR:SPECIAL`.
  - Execute `MYJOB`.
  - Return to the initial directory.
  - Execute `MYJOB_ONES`.

(Note: This job should be put on hold when created.)
5. Find the sequence number of the job in Step 4.
6. Cancel this job.
7. Print out the file `MODULE_3_WHOLE_FILE` (from the Module 3 Lab Activity). Include the title at the top of each page. Have the system notify you when the file is printed.
8. Create a job that will issue a series of `HELP` commands to inform you about some of the commands that you learned in this module. Name this job `HELPER`. Be sure to examine the printed output of this job.

Now check your answers on the next page.

## Lab Activity 6-1 Answers

1. QBATCH/HOLD XEQ JOB\_ONE
2. QDISPLAY  
QUNHOLD 3
3. QDISPLAY
4. QBATCH/HOLD/I  
)DIRECTORY :UDD:MYDIR:SPECIAL  
)XEQ MYJOB  
)DIRECTORY/I  
)XEQ MYJOB\_ONE
5. QDISPLAY/TYPE=BATCH
6. QCANCEL 99
7. QPRINT/NOTIFY/TITLES MODULE\_3\_WHOLE\_FILE
8. QBATCH/I/JOBNAME=HELPER  
)HELP/V QBATCH  
)HELP/V QCANCEL  
)HELP/V QPRINT  
)HELP/V QHOLD  
)HELP/V QUNHOLD  
))

If you successfully completed Lab Activity 6-1, continue to the Module 6 Test. Otherwise, review the material and do the Lab Activity again before you continue.

## Module 6 Test

**Directions:** Complete the following sentences.

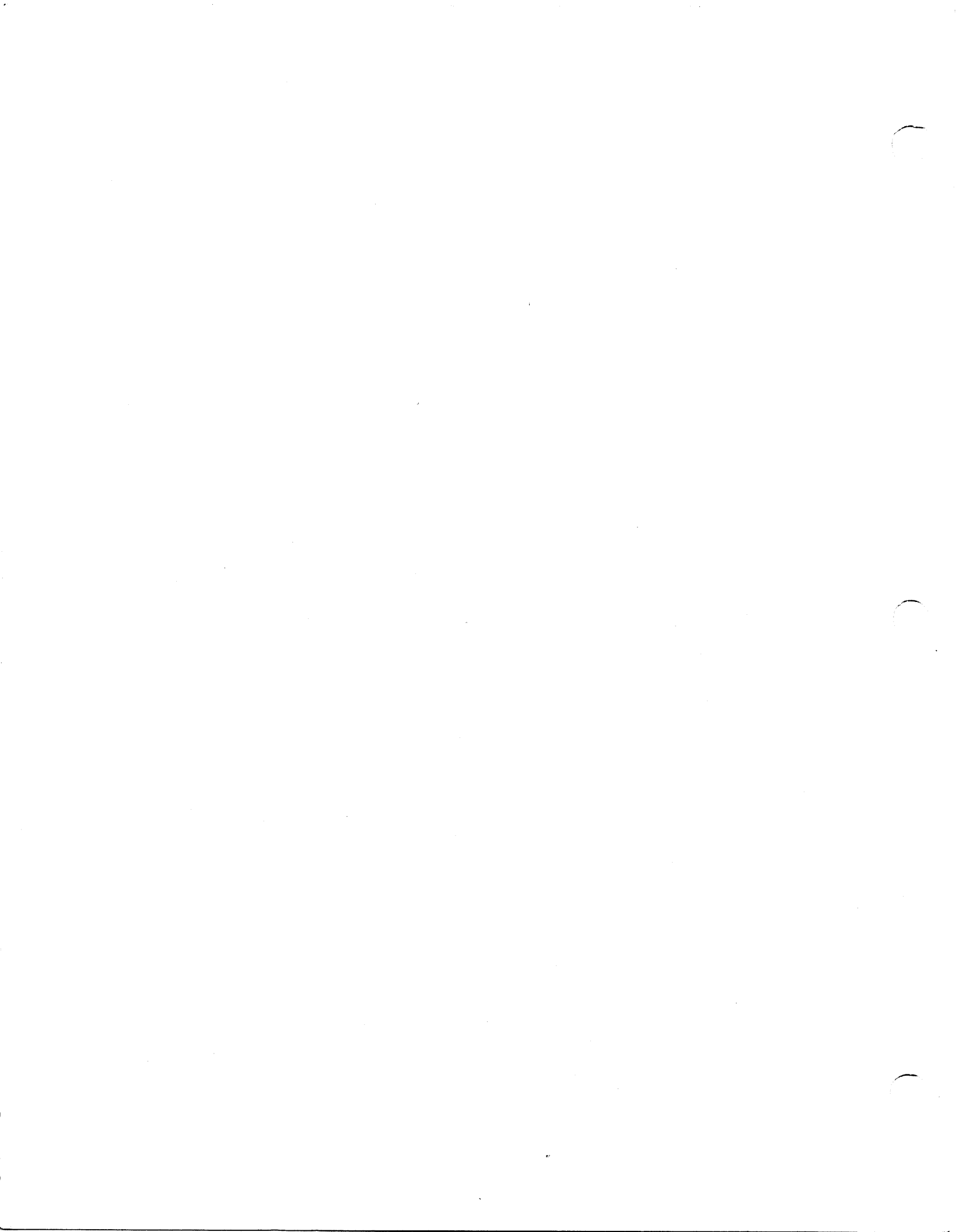
1. When you submit a job to the batch queue:
  - a. Your terminal is locked until the job terminates.
  - b. You are free to continue working at your terminal.
  - c. Your CLI process is blocked.
  - d. Your CLI process is automatically swapped.
2. You can cancel a queue entry by using either the \_\_\_\_\_ number or the \_\_\_\_\_ name.

**Directions:** Write the CLI command to do the following:

3. Place a file named PRINT\_1 in the print queue and notify you when finished.  
\_\_\_\_\_
4. Place a file named PLOT\_FILE in the plotter queue and notify you when finished plotting five copies. \_\_\_\_\_
5. Display the contents of the plotter queue only. \_\_\_\_\_
6. Cancel entry 453. \_\_\_\_\_
7. Hold entry 29. \_\_\_\_\_
8. Release entry 90. \_\_\_\_\_
9. Place a file named INDEPENDENT\_1 in the batch queue.  
\_\_\_\_\_

Now check your answers to the Module 6 Test in Appendix A. If you answered all the questions correctly, go on to Module 7. Otherwise, review the material in Module 6 and take the Test again before continuing.

This concludes Module 6.



# Module 7

## The CLI Environment

### Introduction

This module introduces you to the CLI environment. The CLI environment consists of several parameters that you can control. This module shows you these parameters and the commands necessary to display and change these parameters.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Write commands to change the CLI environment.
2. Write commands to display and change the components of the CLI environment.

### Resources

To complete this module, you will need:

- Module 7 audiotope.
- Module 7 of your *Student Guide*.
- Audiotope playback unit.

### Module Outline

Module 7 discusses the following topics:

1. The CLI environment
  - a. Overview
  - b. Levels of the CLI environment
  - c. PUSH and POP commands
2. CLI commands to change the environment
  - a. SUPERUSER
  - b. SUPERPROCESS

- c. SCREENEDIT
  - d. SQUEEZE
  - e. CLASS1
  - f. CLASS2
  - g. Variables
  - h. LISTFILE
  - i. DATAFILE
3. More environmental CLI commands
- a. DIRECTORY
  - b. SEARCHLIST
  - c. DEFACL
  - d. STRING
  - e. PROMPT
  - f. CHARACTERISTICS

Now start the Module 7 audiotape. As you listen, follow along in Module 7 of your *Student Guide*.

## The CLI Environment

Your CLI environment is composed of several *parameters*. These parameters are:

LEVEL	LISTFILE
SUPERUSER	DATAFILE
SUPERPROCESS	LOGFILE
SCREENEDIT	DIRECTORY
SQUEEZE	SEARCHLIST
CLASS1	DEFACL
CLASS2	STRING
TRACE	PROMPT
VARIABLES	CHARACTERISTICS

These parameters are initialized to *default values* when you log on. You can use the **CURRENT** command to list the settings.

```

CURRENT
LEVEL          0
SUPERUSER      ON
SUPERPROCESS   OFF
SCREENEDIT     ON
SQUEEZE        OFF
CLASS1         ERROR
CLASS2         WARNING
TRACE
VARIABLES      0      0      0      0      0
               0      0      0      0      0
LISTFILE       @LIST
DATAFILE       @DATA
LOGFILE
DIRECTORY      UDD,BUBBIE,MODULE7
SEARCHLIST     UTIL,UTIL:INFOS,UDD,BUBBIE
DEFACL         BUBBIE,OWARE+,RE
STRING         TIME
PROMPT         TIME,DIRECTORY
CHARACTERISTICS /605X/LPP=24/CPL=80
               /ON/ST/EB0/ULC/PM/WRP

/OFF/SFF/EPI/BBT/SPO/RAF/RAT/RAC/NAS/OTT/EOL/UCO/LT/FF/EB1/NRM/MOD/TO/TSP
/PBN/ESC/FKT/NNL

```

Figure 7.1 Sample Output of the **CURRENT** Command

### LEVEL Command

The *level* is your location within the CLI environment. Zero is the highest level.

The **LEVEL** command displays your current level. Each level contains all environment parameters.

#### Example

```

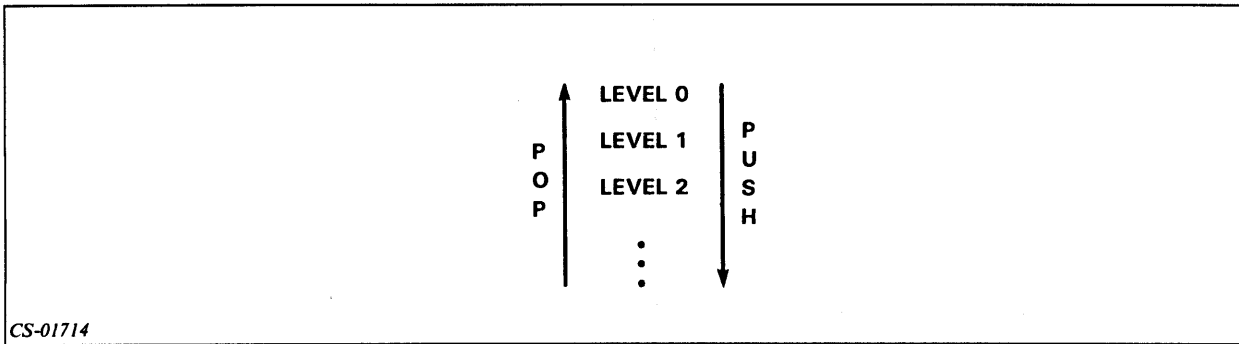
LEVEL
LEVEL 0

```

## PUSH and POP Commands

Use the PUSH command to move one level *down*.

Use the POP command to move one level *up*.



CS-01714

Figure 7.2 The PUSH and POP Commands

The PREVIOUS command is similar to the CURRENT command. It displays the same information as the CURRENT command, but for the *next higher* level of the environment.

### Example

If you are at level 2:

#### CURRENT

Displays level 2.

#### PREVIOUS

Displays level 1.

```

1 → )LEVEL
    LEVEL 0
2 → )DIRECTORY :UDD:USER1:SPECIAL
    )SEARCHLIST :UDD,:UTIL,:UDD:USER1:PROGRAMS
    )DEFACL USER1,OWARE,+RE
    )PROMPT TIME
    12:30:03
3 → )PUSH
    12:30:06
4 → )LEVEL
    LEVEL 1
    12:30:10
    )PROMPT/K
    )DIRECTORY/I
    )SEARCHLIST :UDD:USER1:NORMAL
5 → )DEFACL USER1,OWARE
    )CREATE FILE1
    )COPY FILE1,FILE2
    )QPRINT FILE1
6 → )POP
    12:32:13
7 → )
    
```

Figure 7.3 Changing Your Environment Level

Now do Exercise 7-1 on the next page.



## Exercise 7-1

**Directions:** Write the command to:

1. Move down one level. \_\_\_\_\_
2. Move up one level. \_\_\_\_\_
3. Display the current setting of the environment's variables. \_\_\_\_\_
4. Display your location in the environment. \_\_\_\_\_
5. Display the setting of the next higher level of your environment's variables.  
\_\_\_\_\_

Now check your answers on the next page.

## **Exercise 7-1**

### **Answers**

1. **PUSH**
2. **POP**
3. **CURRENT**
4. **LEVEL**
5. **PREVIOUS**

If you answered all of the questions correctly, continue with Module 7 by restarting the Module 7 audiotape. Otherwise, review the material and do this exercise again before you continue.

## CLI Commands to Change the Environment

### /P Switch

The /P switch sets a variable to setting in previous level of the environment.

#### Example

```
DIRECTORY :UTIL
PUSH
DIRECTORY :UDD:ME
.
.
.
DIRECTORY/P
DIRECTORY
:UTIL
```

### Superuser Privilege

```
SUPERUSER {ON }
           {OFF }
```

When SUPERUSER is on, you can access *any file*. Enter the SUPERUSER command without an argument to display SUPERUSER status.

#### Example 1

To display the status:

```
SUPERUSER
```

#### Example 2

To turn SUPERUSER on:

```
SUPERUSER ON
```

#### Example 3

To turn SUPERUSER off:

```
SUPERUSER OFF
```

### Superprocess Privilege

```
SUPERPROCESS {ON }
              {OFF }
```

When SUPERPROCESS is on, you can control *any process*. Enter the SUPERPROCESS command without an argument to display SUPERPROCESS status.

#### Example 1

To display the status:

```
SUPERPROCESS
```

#### Example 2

To turn SUPERPROCESS on:

```
SUPERPROCESS ON
```

**Example 3**

To turn SUPERPROCESS off:

**SUPERPROCESS OFF****SCREENEDIT****SCREENEDIT { ON }  
{ OFF }**

When SCREENEDIT is on, you can use the cursor control characters listed in Table 7.A.

Control Character	Effect on the Cursor
CTRL-A	Move to the end of the character string.
CTRL-B	Move to the end of the previous word.
CTRL-E	Enter/exit the insert character mode.
CTRL-F	Move to the beginning of the next word.
CTRL-H	Move to the beginning of the character string.
CTRL-I	Insert a tab.
CTRL-K	Erase everything to the right of the cursor.
CTRL-X	Move to the right one character. (The → key on the function keypad has the same effect.)
CTRL-Y	Move to the left one character. (The ← key on the function keypad has the same effect.)
RUBOUT	Delete the previous character.

Table 7.A SCREENEDIT Cursor Control Characters

**Example 1**

To display the status:

**SCREENEDIT  
ON****Example 2**

To turn SCREENEDIT on:

**SCREENEDIT ON****Example 3**

To turn SCREENEDIT off:

**SCREENEDIT OFF****SQUEEZE****SQUEEZE { ON }  
{ OFF }**

When SQUEEZE is on, output is compressed by removing spaces.

**Example 1**

To display the status:

**SQUEEZE OFF**

**Example 2**

To turn SQUEEZE on:

**SQUEEZE ON**

**Example 3**

To turn SQUEEZE off:

**SQUEEZE OFF**

**Class 1 and Class 2 Errors**

```
CLASS1      {ABORT}
             {ERROR}
             {WARNING}
             {IGNORE}
```

Sets or displays reaction level to class 1 errors (errors that affect the environment).

```
CLASS2      {ABORT}
             {ERROR}
             {WARNING}
             {IGNORE}
```

Sets or displays reaction level to class 2 errors (errors that do not affect the environment).

Assume XYZ does not exist in Examples 1 and 2.

**Example 1**

**DIRECTORY XYZ**

**ERROR: FILE DOES NOT EXIST, FILE XYZ**

**TYPE XYZ**

**WARNING: FILE DOES NOT EXIST, FILE XYZ**

**Example 2**

**CLASS1 IGNORE**

**CLASS2 ABORT**

**Variables**

There are 10 variables you can use to pass data between programs.

**CLI Commands**

**VAR0**

**VAR1**

.

.

.

.

**VAR9**

**Example 1**

**VAR7,25**

Sets variable 7 to 25.

**Example 2**

**VAR4**  
**37**

Displays value of variable 4.

**LISTFILE Command**

The LISTFILE command sets or displays the filename used for output in conjunction with the /L switch.

**Example 1**

**LISTFILE,HOLD\_DATA\_ONE**

To use the listfile HOLD\_DATA\_ONE:

**TYPE/L FILEA**  
**FILESTATUS/L**  
**QPRINT,HOLD\_DATA\_ONE**

The displayed output of the TYPE and FILESTATUS commands are stored in HOLD\_DATA\_ONE. It is printed with the QPRINT command.

**DATAFILE Command**

The DATAFILE command sets or displays the filename used for input.

**Example**

**DATAFILE,DATA\_IN\_ONE**

**LOGFILE Command**

The LOGFILE command sets or displays the filename used as the logfile.

**EXAMPLE**  
**LOGFILE,LOGGER**

The logfile can be used in conjunction with the TRACE command.

**TRACE Command**

The TRACE command keeps a record of the activity at your console. When TRACE is on, all CLI transactions at your terminal are recorded in the logfile.

**Example**

To turn TRACE on:  
**TRACE/ON/LOG**

To turn TRACE off:  
**TRACE/OFF/LOG**

Now do Exercise 7-2 on the next page.

## Exercise 7-2

**Directions:** Write the CLI command that will:

1. Turn on the superuser privilege. \_\_\_\_\_
2. Set the listfile to be a file called GARBAGE\_OUT. \_\_\_\_\_
3. Set the datafile to be a file called GARBAGE\_IN. \_\_\_\_\_
4. Turn off SCREENEDIT. \_\_\_\_\_
5. Set the action for severe errors to ABORT. \_\_\_\_\_
6. Turn SQUEEZE on. \_\_\_\_\_
7. Display variable 6. \_\_\_\_\_
8. Set the logfile to LOG7. \_\_\_\_\_
9. Turn on TRACE. \_\_\_\_\_

Now check your answers on the next page.

## **Exercise 7-2**

### **Answers**

1. SUPERUSER,ON
2. LISTFILE,GARBAGE\_OUT
3. DATAFILE,GARBAGE\_IN
4. SCREENEDIT,OFF
5. CLASS1,ABORT
6. SQUEEZE,ON
7. VAR6
8. LOGFILE,LOG7
9. TRACE/ON/LOG

If you answered all of the questions correctly, continue with Module 7 by restarting the Module 7 audiotape. Otherwise, review the material and do this exercise again before you continue.



## More Environmental CLI Commands

Command	Result
DIRECTORY	Sets or displays your working directory.
SEARCHLIST	Sets or displays your searchlist.
DEFACL	Sets or displays your default Access Control List.

Table 7.B Review of Three CLI Commands.

### STRING Command

The STRING command sets or displays the contents of the string (used to return information from programs).

**Example**

```
STRING THIS IS THE NEW STRING
STRING
THIS IS THE NEW STRING
```

### PROMPT Command

The PROMPT command sets or displays the prompt (information that appears before the right parenthesis).

**Example**

```
PROMPT TIME
08:30:01
PROMPT,TIME,DATE
08:31:05
10-MAR-82
PROMPT/K
```

### CHARACTERISTICS Command

The CHARACTERISTICS command sets or displays the characteristics of your terminal. (Use one of the first five switches to identify your console.)

Switch	Result
/HARDCOPY	Hard-copy terminals.
/40101	DGC Model 40101.
/6012	DGC Model 6012.
/605x	DGC Model 6052 or 6053.
/CRT4	Other CRTs.
/LPP= <i>n</i>	Lines-per-page, in decimal.
/CPL= <i>n</i>	Characters-per-line, in decimal.
/ON	Set the bit in the device characteristics words for each of the command switches that follow. This bit remains set until you issue a /OFF switch or a delimiter. (Note: This bit is automatically set unless you include the /OFF switch. Therefore, this switch is optional.)
/EBO	If you want echoing to occur on your console, you must set /EBO or /EB1. This causes the system to echo control characters such as ^A, and ^B, and to echo ESC as \$. For more information see ?GCHR in the <i>AOS Programmer's Manual</i> .
/EPI	Accept only even parity on input; if this switch is off, accept any parity on input.
/EOL	Do not output a NEW LINE if CPL length is exceeded on output.
/ESC	ESC character produces ^C^A interrupt.
/FF	Output a form-feed on open.
/LT	Output 60 (decimal) nulls on open and close.
/MOD	Device is on a modem interface.
/NAS	Set non-ANSII standard bit.
/NRM	Do not allow this console to receive SEND messages.
/OTT	On input, convert octal 175 and 176 to octal 33.
/PBN	(Packed Binary Notation. Card readers only.) Packed format on binary read, 4 columns are put in 3 words. If you don't include this switch, columns are right-justified in memory.
/PM	Page mode. On output, write LPP lines per page, then suspend output until you type CTRL-Q.
/RAC	(Rubout After Carriage return.) Send two rubouts after each NEW LINE and carriage return.
/RAF	Send 21 (decimal rubouts after each form-feed.)
/RAT	Send two rubouts after each tab (CTRL-I).
/SFF	Simulate form-feed.
/SPO	Output characters in even parity; if this switch is off, output characters as sent by program.
/ST	Simulate tab stop every eight columns.
/TO	Enable time-outs.
/TSP	(Card readers only.) Include trailing spaces; if this switch is off, trailing blanks are suppressed.
/UCO	On output, convert lower-case to upper-case.
/ULC	On input, accept both upper-case and lower-case; if this switch is off, lower-case input is converted to upper-case.
/WRP	Hardware generates NEW LINE on line-too-long (Wrap).

Table 7.C Setting and Displaying the Characteristics of Your Terminal

**Example 1**

To turn on page mode:

**CHARACTERISTICS/ON/PM**

**Example 2**

To turn off page mode:

**CHARACTERISTICS/OFF/PM**

**Example 3**

To set characters-per-line to 132:

**CHARACTERISTICS/CPL=132**

**Example 4**

To convert output to upper-case:

**CHARACTERISTICS/ON/UCO**

This command only changes those characteristics listed in the CLI command. All those not specified remain at their previous setting.

Now do Exercise 7-3 on the next page.



## Exercise 7-3

**Directions:** Write the command to:

1. Set your prompt to date and directory. \_\_\_\_\_
2. Set the string variable to "PROGRAM 7 FAILED TO EXECUTE." \_\_\_\_\_
3. Change your terminal's characteristics to turn off the switch to accept lower-case input. \_\_\_\_\_
4. Set lines-per-page to 60, using the CHARACTERISTICS command.  
\_\_\_\_\_

Now check your answers on the next page.

## **Exercise 7-3**

### **Answers**

1. **PROMPT,DATE,DIRECTORY**
2. **STRING PROGRAM 7 FAILED TO EXECUTE**
3. **CHARACTERISTICS/OFF/ULC**
4. **CHARACTERISTICS/LPP=60**

If you answered all the questions correctly, continue with Module 7 by doing Lab Activity 7-1. Otherwise, review the material and do this exercise again before you continue.

## Lab Activity 7-1

**Directions:** Log onto your system. Then enter the command or commands to:

1. Display the current level and the present setting of all of the parameters associated with this level.
2. Try to move up one level. Note what happens.
3. Change your prompt to display the current level.
4. Move down one level.
5. Change your prompt to display the date, time, and directory.
6. Display the characteristics of your terminal.
7. Display the environment of level zero.
8. Change your prompt to what it was at level zero.
9. Set your listfile to a file called `HOLD_OUTPUT`.
10. Use the appropriate switch to cause the `FILESTATUS` command to list your files in the listfile.
11. Display the contents of the listfile.
12. Change the prompt to display your working directory.
13. Place the value 25 into variable 7.
14. Try to delete a file called `DOES_NOT_EXIST`. (You should see a warning message.)
15. Set the reaction level for class 2 errors to `IGNORE`.
16. Repeat Step 14.
17. Log off.

Now check your answers on the next page.

## Lab Activity 7-1

### Answers

1. CURRENT
2. POP (The screen displays ERROR: CAN'T POP FROM LEVEL ZERO.)
3. PROMPT LEVEL
4. PUSH
5. PROMPT DATE TIME DIRECTORY
6. CHARACTERISTICS
7. PREVIOUS
8. PROMPT/P
9. LISTFILE,HOLD\_OUTPUT
10. FILESTATUS/L
11. TYPE,HOLD\_OUTPUT
12. PROMPT,DIRECTORY
13. VAR7,25
14. DELETE/C/V,DOES\_NOT\_EXIST
15. CLASS2,IGNORE
16. DELETE/C/V,DOES\_NOT\_EXIST (You should not see any warning message.)
17. BYE

If you successfully completed Lab Activity 7-1, go on to the Module 7 Test. If you had any difficulty with the lab, review the material and do the Lab Activity again before continuing.



## Module 7 Test

**Directions:** In each of the following questions, select the best answers.

1. Select the command that moves you down one level.
  - a. **POP**
  - b. **DOWN**
  - c. **PUSH**
  - d. **LEVEL +1**
2. Select the command that moves you up one level.
  - a. **POP**
  - b. **UP**
  - c. **PUSH**
  - d. **LEVEL -1**
3. Select the command that displays only the present level of the environment.
  - a. **POP**
  - b. **LEVEL**
  - c. **PREVIOUS**
  - d. **CURRENT**
4. Select the command that displays the present settings of the environment.
  - a. **POP**
  - b. **LEVEL**
  - c. **PREVIOUS**
  - d. **CURRENT**
5. Select the command that displays the settings of the next higher level of the environment.
  - a. **OLD**
  - b. **LEVEL**
  - c. **PREVIOUS**
  - d. **CURRENT**

6. Which of the following commands can you use to gain access to any file in the system?
  - a. SUPERPROCESS
  - b. SEARCHLIST
  - c. SUPERUSER
  - d. SQUEEZE
7. Which file can you use to hold output data?
  - a. SEARCHLIST
  - b. LISTFILE
  - c. DATAFILE
  - d. PROMPT
8. Which of the following commands can you use to change your working directory?
  - a. SEARCHLIST
  - b. DATAFILE
  - c. CHARACTERISTICS
  - d. DIRECTORY
9. Which of the following commands can you use to hold a message from a program?
  - a. STRING
  - b. PROMPT
  - c. DEFACL
  - d. SQUEEZE
10. Which of the following commands can you use to change the system's reaction to an error condition?
  - a. ERROR
  - b. SQUEEZE
  - c. CLASS2
  - d. LISTFILE

**Directions:** Write the command to:

11. Move down to a new environment level. \_\_\_\_\_
12. Move up to the previous level. \_\_\_\_\_
13. Display the current level. \_\_\_\_\_
14. Display the settings of the environment. \_\_\_\_\_
15. Set your prompt to display the time of day. \_\_\_\_\_
16. Set your listfile to be the file called TEMPORARY\_OUTPUT.  
\_\_\_\_\_
17. Display the setting of SUPERUSER. \_\_\_\_\_
18. Set SUPERPROCESS to on. \_\_\_\_\_
19. Turn SQUEEZE on. \_\_\_\_\_
20. Set the action for less severe errors to IGNORE. \_\_\_\_\_

Now check your answers to the Module 7 Test in Appendix A. If you answered all the questions correctly, go on to Module 8. Otherwise, review the material in Module 7 and take the Test again before continuing.

This concludes Module 7.



# Module 8

## The Process Tree

### Introduction

This module discusses the process tree, which is the relationship of one process to another. You will learn to create a process and to terminate a process. This module also discusses the processes that control your use of the system.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Identify a process and the components of a process.
2. List the types and states of a process.
3. Identify the major components of the process tree.
4. Write CLI command lines using the TREE, PROCESS, TERMINATE, SUPERPROCESS, and WHO commands.

### Resources

To complete this module, you will need:

- Module 8 audiotope.
- Module 8 of your *Student Guide*.
- Audiotope playback unit.

### Module Outline

Module 8 discusses the following topics:

1. Process concepts
  - a. Elements of a process
  - b. Process functions
  - c. Process tree

2. Creating processes with EXECUTE
3. Creating processes with PROCESS
4. Process control
  - a. TERMINATE command
  - b. SUPERPROCESS command

Now start the Module 8 audiotape. As you listen, follow along in Module 8 of your *Student Guide*.

## The Process Concept

A *process* is a set of system resources. A process is more than just a program, however. It is composed of a program plus:

- Unique ID
- Username
- Memory
- Priority
- Privileges
- State
- Type

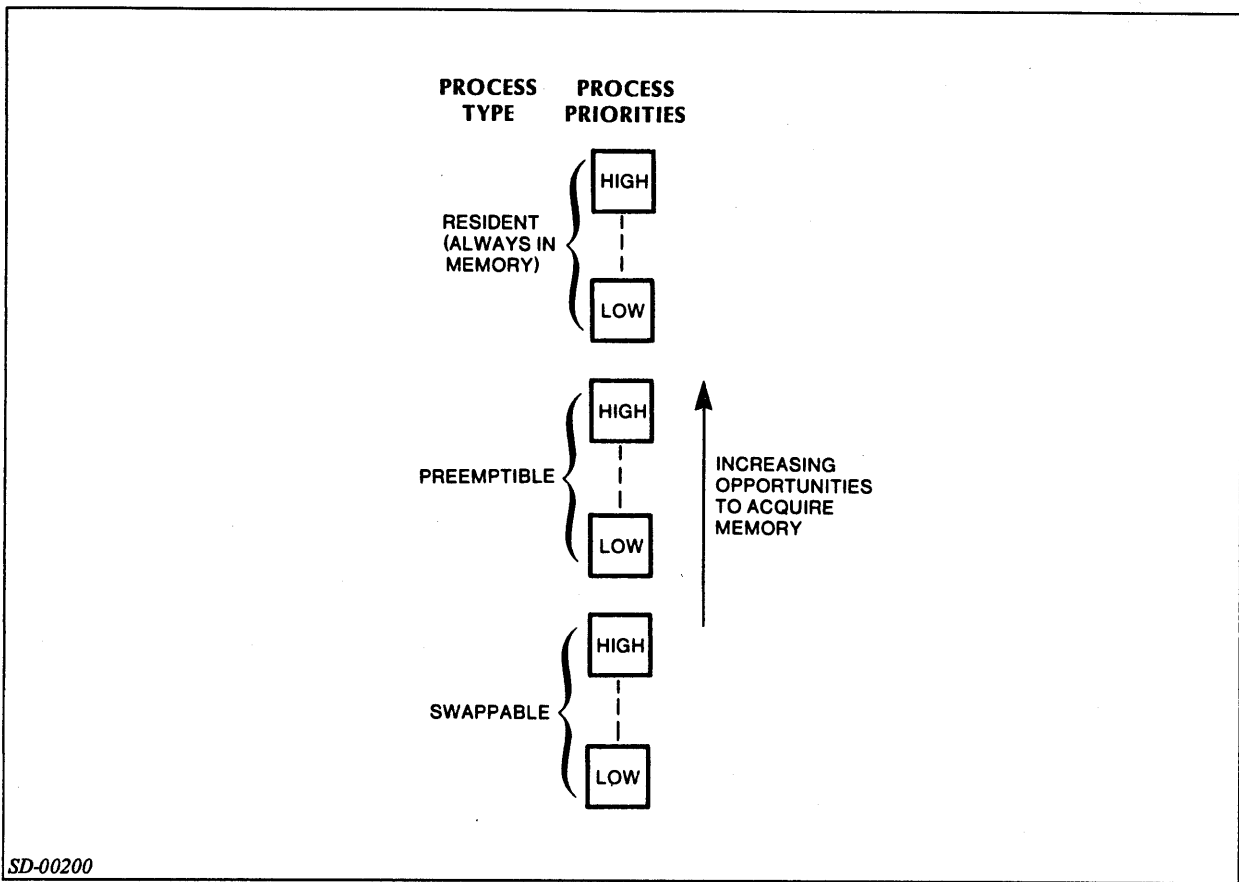
Component	Explanation
Program	Set of instructions supplied by the user or the system.
Unique ID	PID, or process ID (1-255).
Username	Your unique username, assigned by the system manager.
Memory	Active storage space.
Priority	1-3 or 1-255. Determines use of main storage.
Privileges	Create a new process. Change priority. Change type.
Type	The way a process uses memory. There are three types. Resident: Always in main memory. Preemptible: Can be moved to disc, but usually in memory. Swappable: Can be moved to disc to allow other processes to execute.
State	Eligible Ineligible Blocked

Table 8.A Components of a Process

## Process States

*State* refers to the ability of a process to access the central processor. A process may be in one of three states:

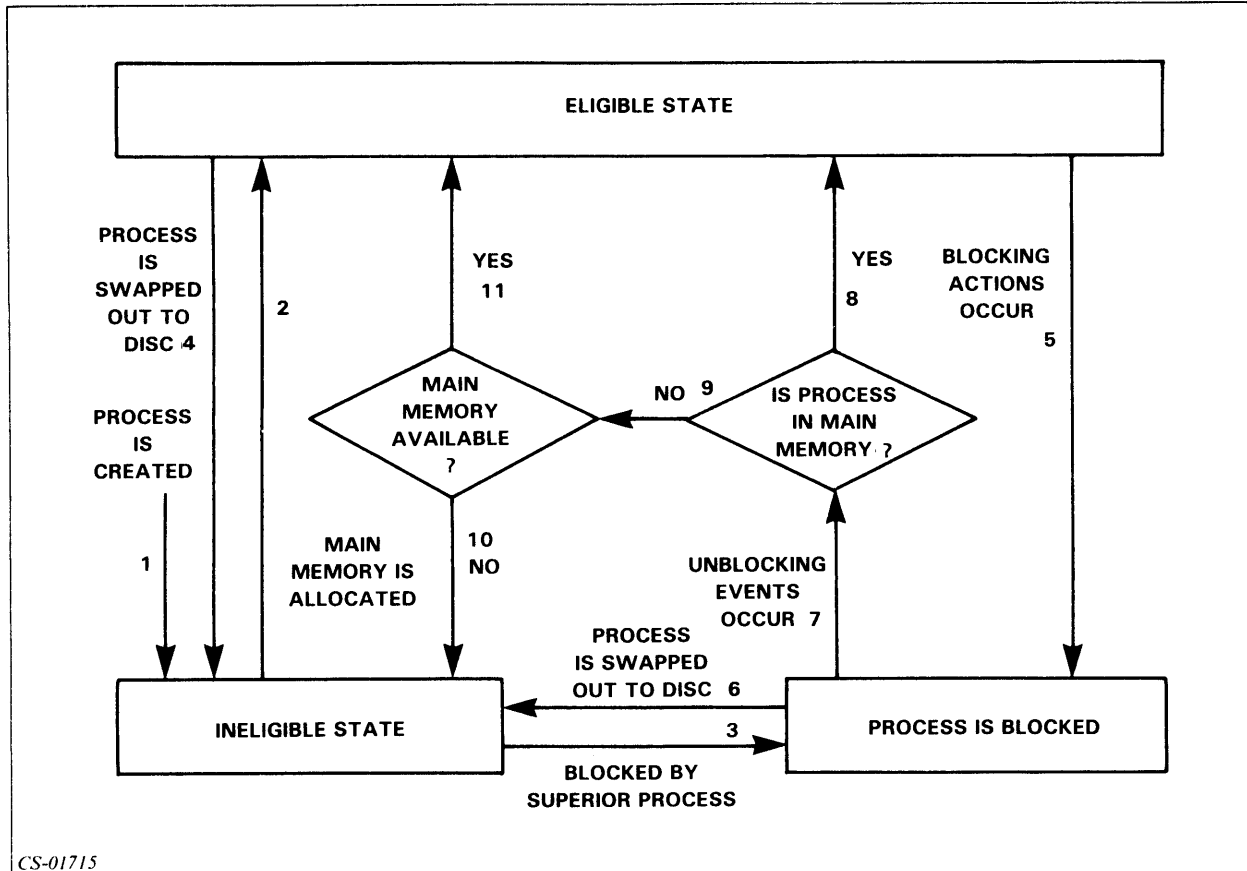
- An *eligible* process is allocated main memory. (To gain CPU control, a process must be allocated main memory.)
- An *ineligible* process is not allocated main memory. An ineligible process becomes eligible when memory is allocated.
- A *blocked* process is waiting for an external event to occur. A process can become blocked by creating another process.



SD-00200

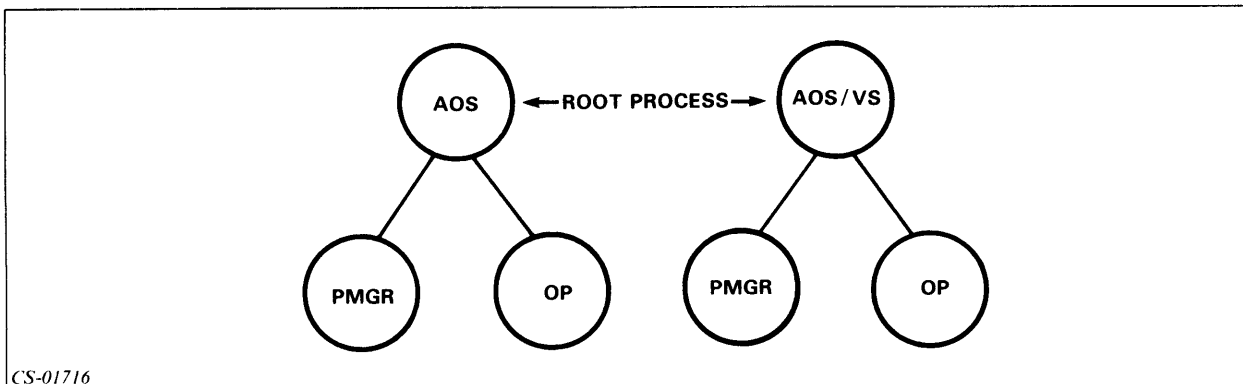
Figure 8.1 Process Types and Priorities





CS-01715

Figure 8.2 State Transitions of a Process



CS-01716

Figure 8.3 A Basic Process Tree

*PMGR* is the peripheral manager that manages input and output operations. *OP* is a highly privileged process that can create other processes.

The process hierarchy is similar to the data file and directory hierarchy.

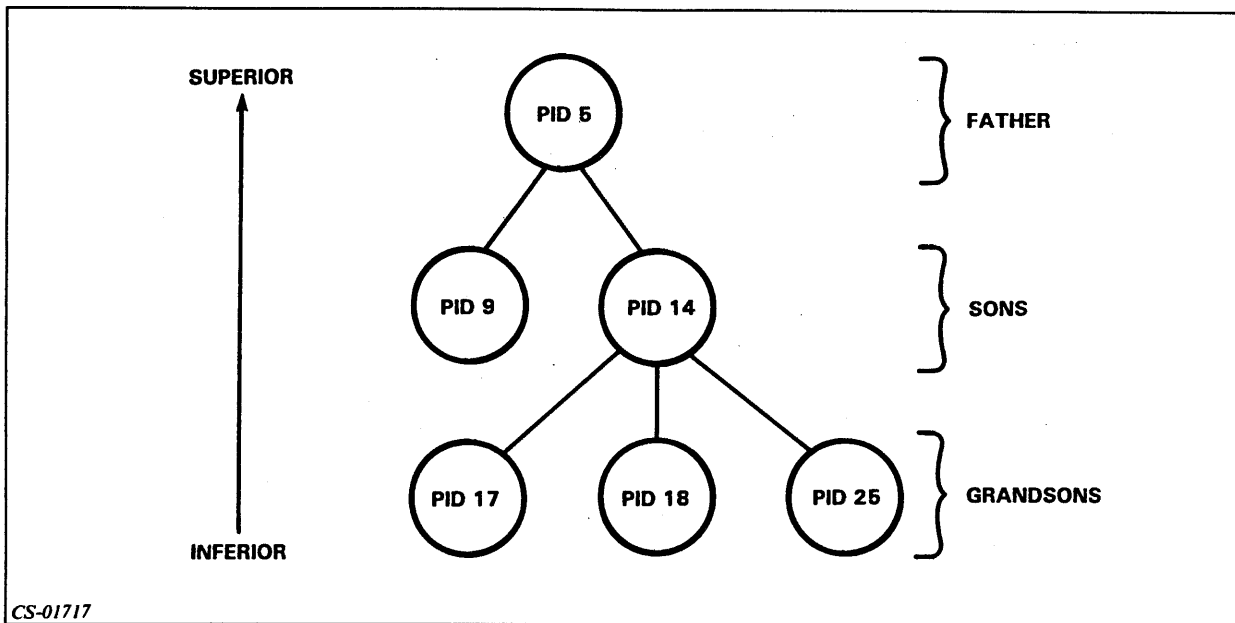


Figure 8.4 Process Tree

**Example 1****WHO 3**

PID: 3 OP EX :UTIL:EXEC.PR

**Example 2****TREE 14**

PID: 14, FATHER: 5 SONS: 17 18 25

Note: The TREE and WHO commands will not work for PID 0.

Follow the steps below to construct the process hierarchy on your system.

**1. TREE 1**

PID: 1 FATHER: 0 SONS:

**2. WHO 1**

PID: 1 PMGR PMGR :PMGR.PR

**3. TREE 2**

PID: 2 FATHER: 0 SONS: 3 4 17

**4. WHO 2**

PID: 2 OP OP :CLI.PR

**5. WHO 3**

PID: 3 EXEC EXEC :UTIL:EXEC.PR

**6. WHO 4**

PID: 4 OP INFOS :INFOS.PR

**7. WHO 17**

PID: 17 OP FORMQ :UTIL:AZTEXT:FORMQ.PR

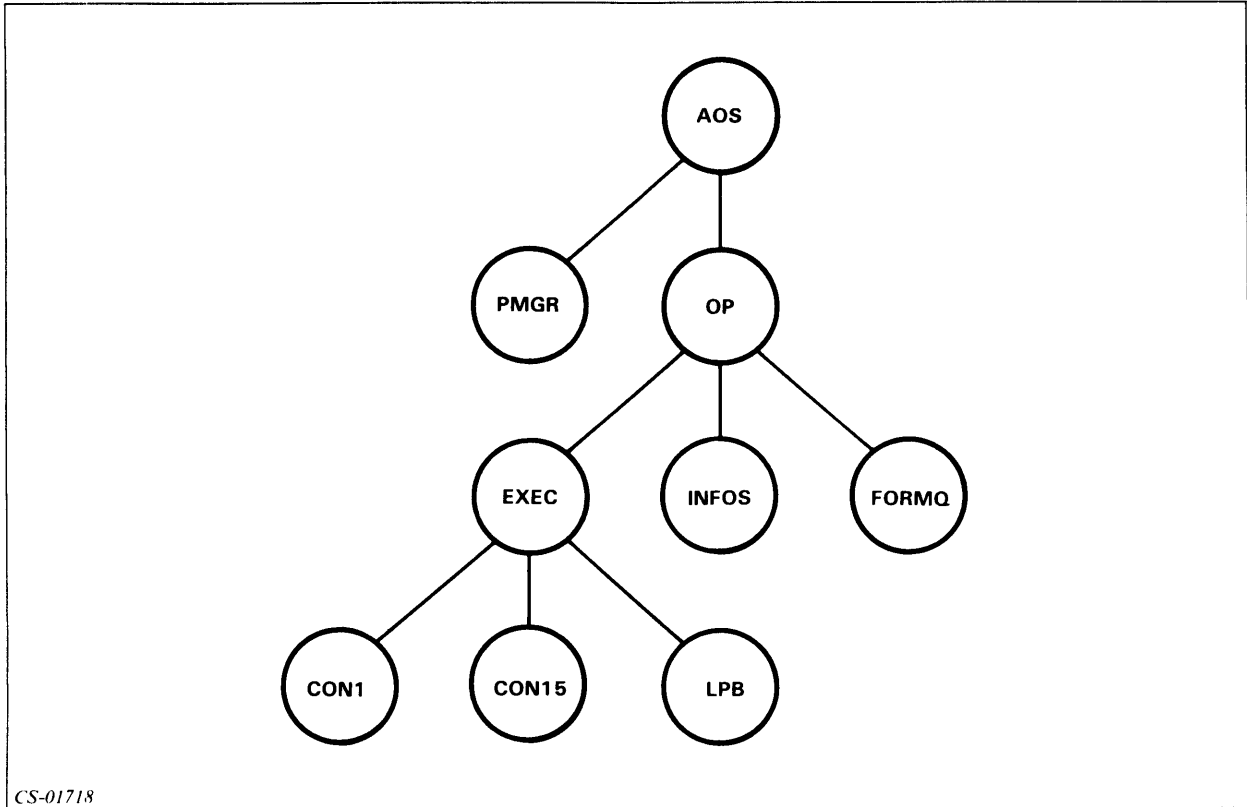
**8. TREE 3**

PID: 3 FATHER: 2 SONS: 5 6 9 10 11

9. WHO 5

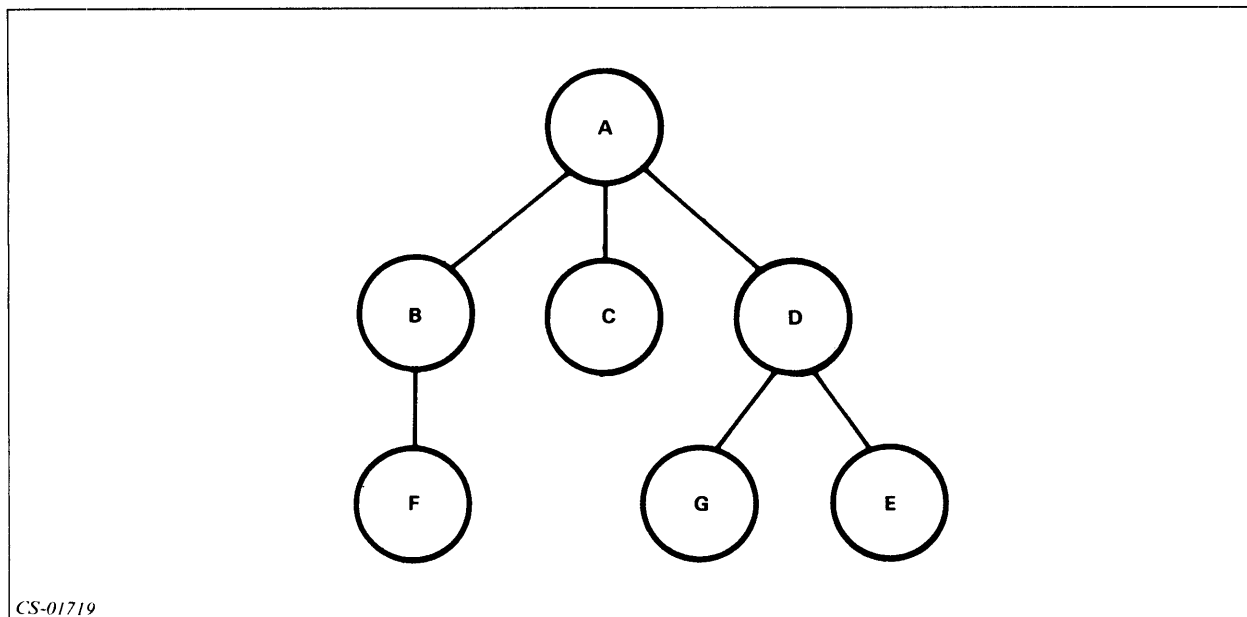
⋮

These steps result in the process hierarchy illustrated in Figure 8.5.



CS-01718

Figure 8.5 The Complete Process Hierarchy

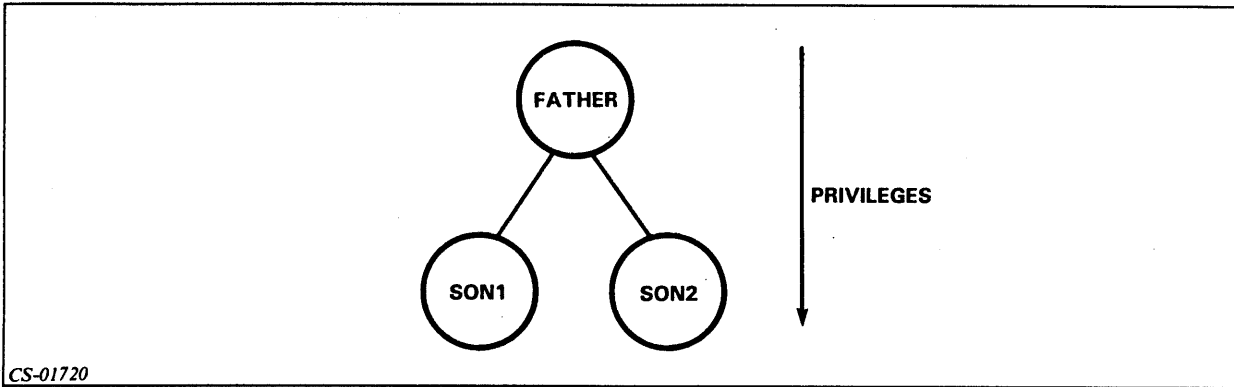


CS-01719

Figure 8.6 Creating Sons

Processes B, C, D, E, F, and G are all subordinate to A. Process F is subordinate to B. E and G are subordinate to D.

Any privilege that a father has can be passed on to a son. No father can pass a privilege that he lacks to a son.



CS-01720

Figure 8.7 Passing Privileges

Now do Exercise 8-1 on the next page.

## Exercise 8-1

Directions: Fill in the blanks in the process tree in Figure 8.8.

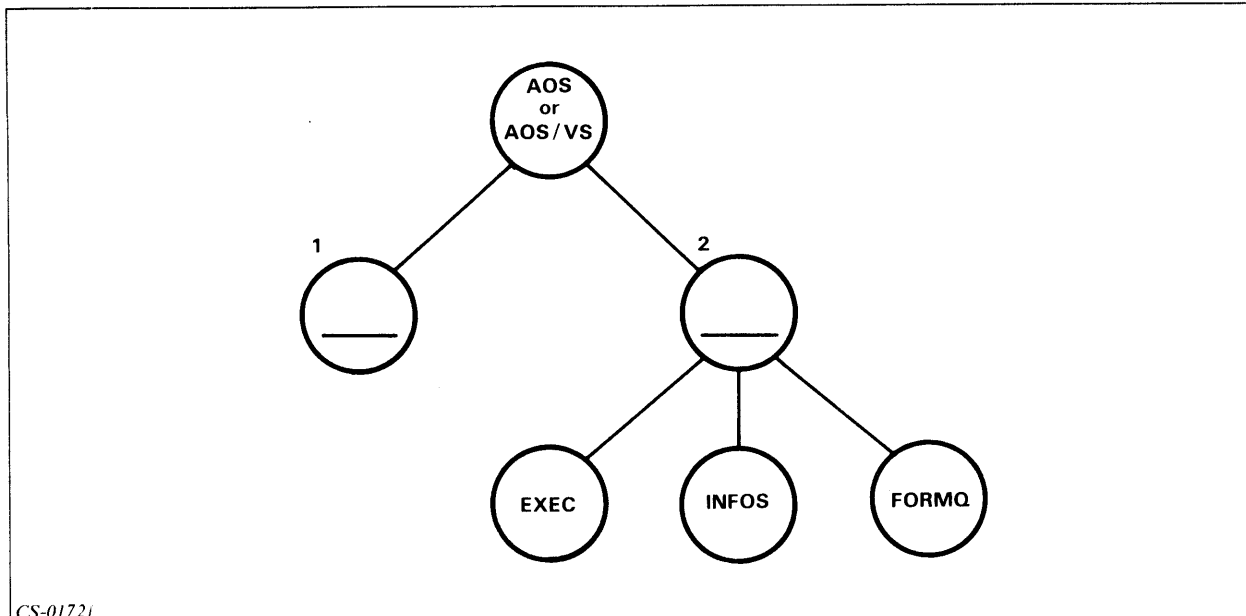


Figure 8.8

Directions: Using the process tree in Figure 8.9, answer the following questions.

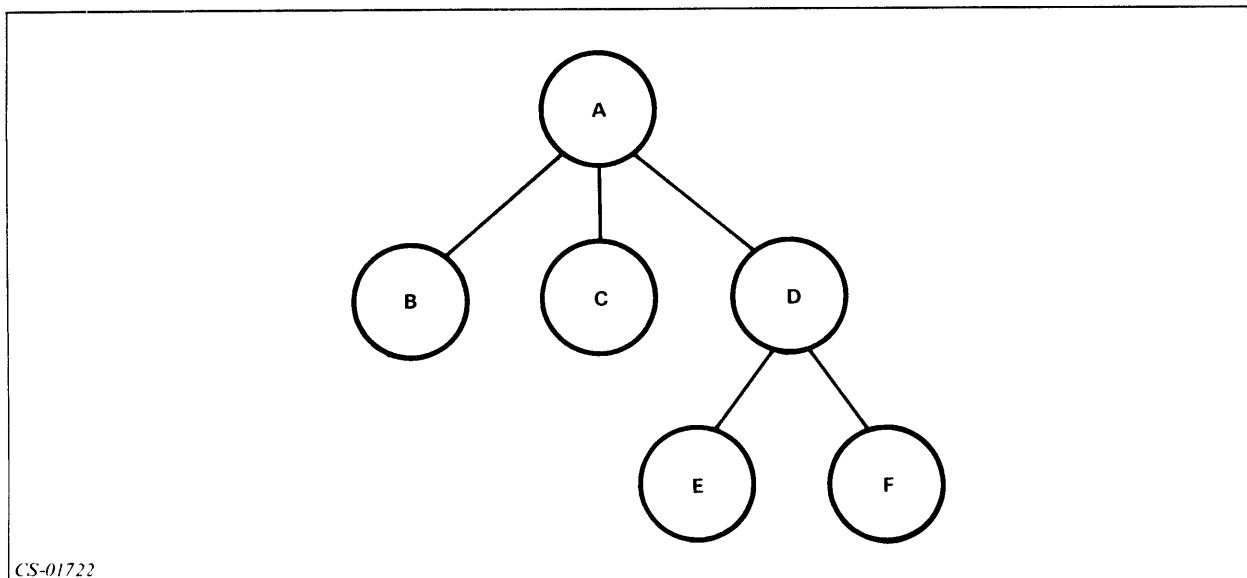


Figure 8.9

3. D is subordinate to \_\_\_\_\_.
4. D is superior to \_\_\_\_\_.
5. A is superior to \_\_\_\_\_.
6. The father of F is \_\_\_\_\_.
7. The sons of D are \_\_\_\_\_.

Now check your answers on the next page.

## **Exercise 8-1**

### **Answers**

1. PMGR
2. OP
3. A
4. E, F
5. B, C, D, E, F
6. D
7. E, F

If you answered all of the questions correctly, continue with Module 8 by restarting the Module 8 audiotape. Otherwise, review the material and do this exercise again before you continue.

## Creating Subordinate Processes

### EXECUTE Command

- The EXECUTE command creates a subordinate swappable process with the same priority and privileges as your process.
- The program comes from the file that you specify as the argument.
- The new process blocks your process.

#### Example 1 EXECUTE PROG1

The command invokes a program called PROG1. That is, it creates a subordinate process whose program is PROG1. Figure 8.10 illustrates a portion of the process tree before and after the command takes effect. PROG1 has the same priority and privileges as YOU.

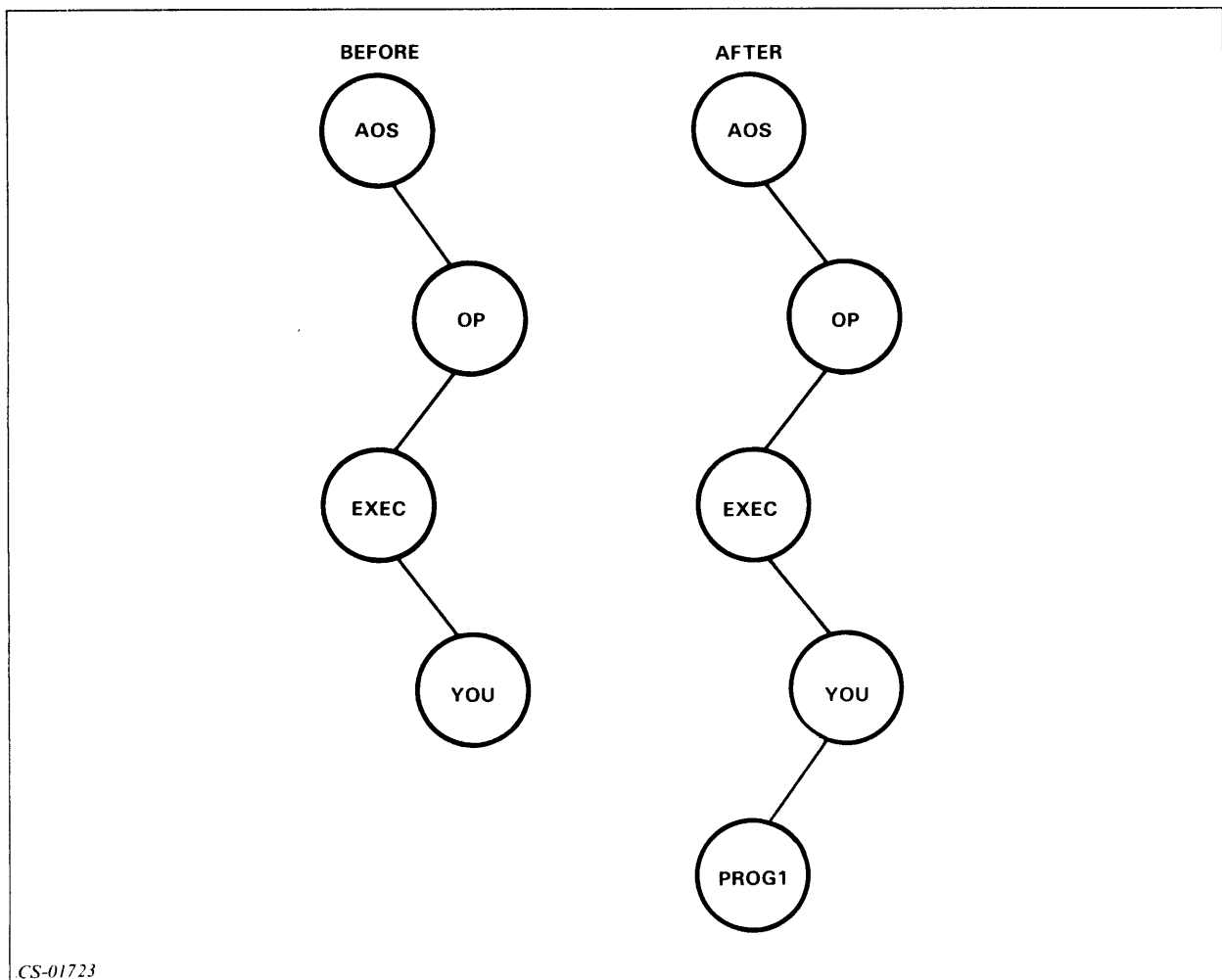


Figure 8.10 Before and After EXECUTE

Now do Exercise 8-2 on the next page.





## Exercise 8-2

**Directions:** Given the process tree in Figure 8.11, draw the process tree if process USER issued the following command:

**EXECUTE,USER\_PROG\_TEN**

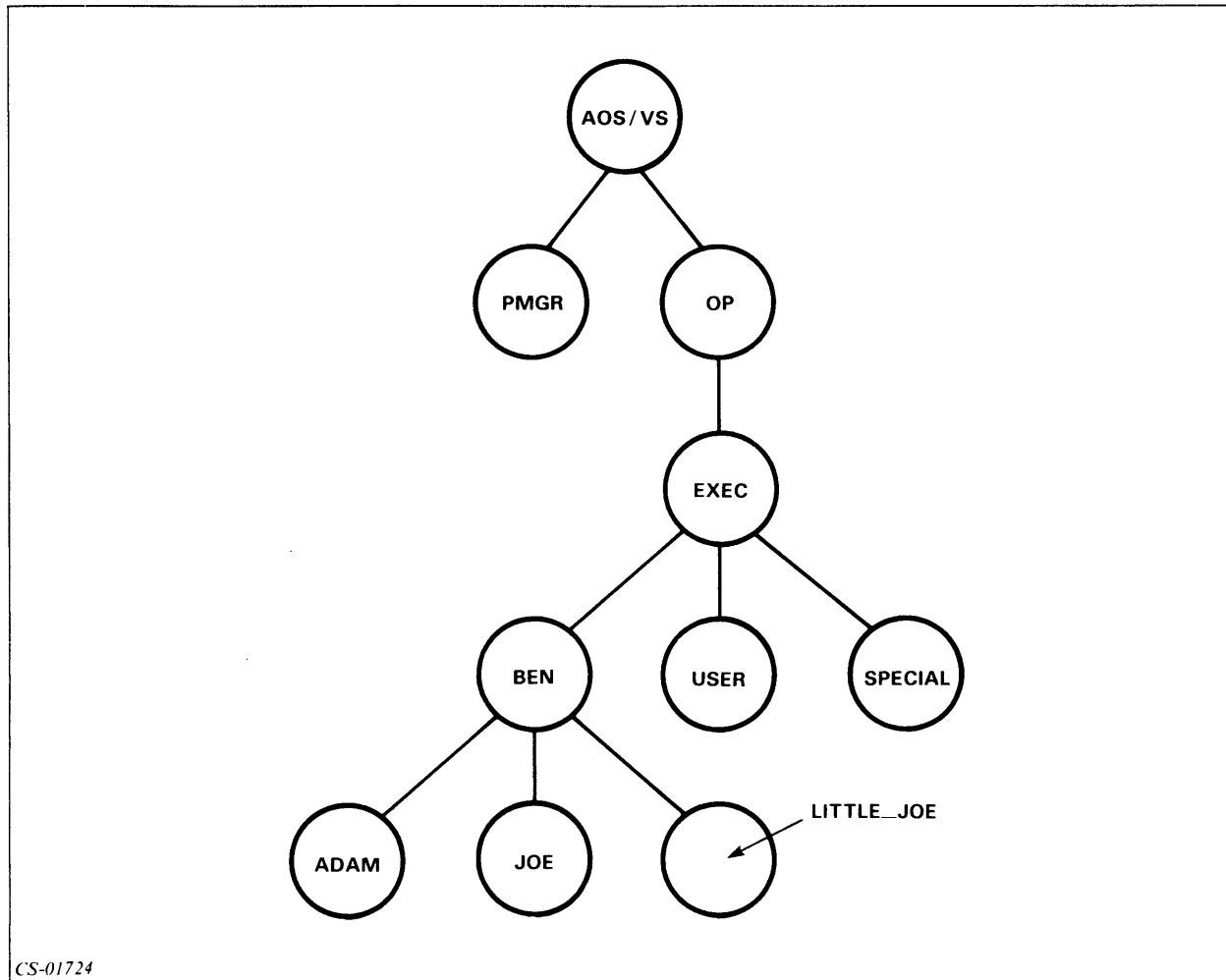
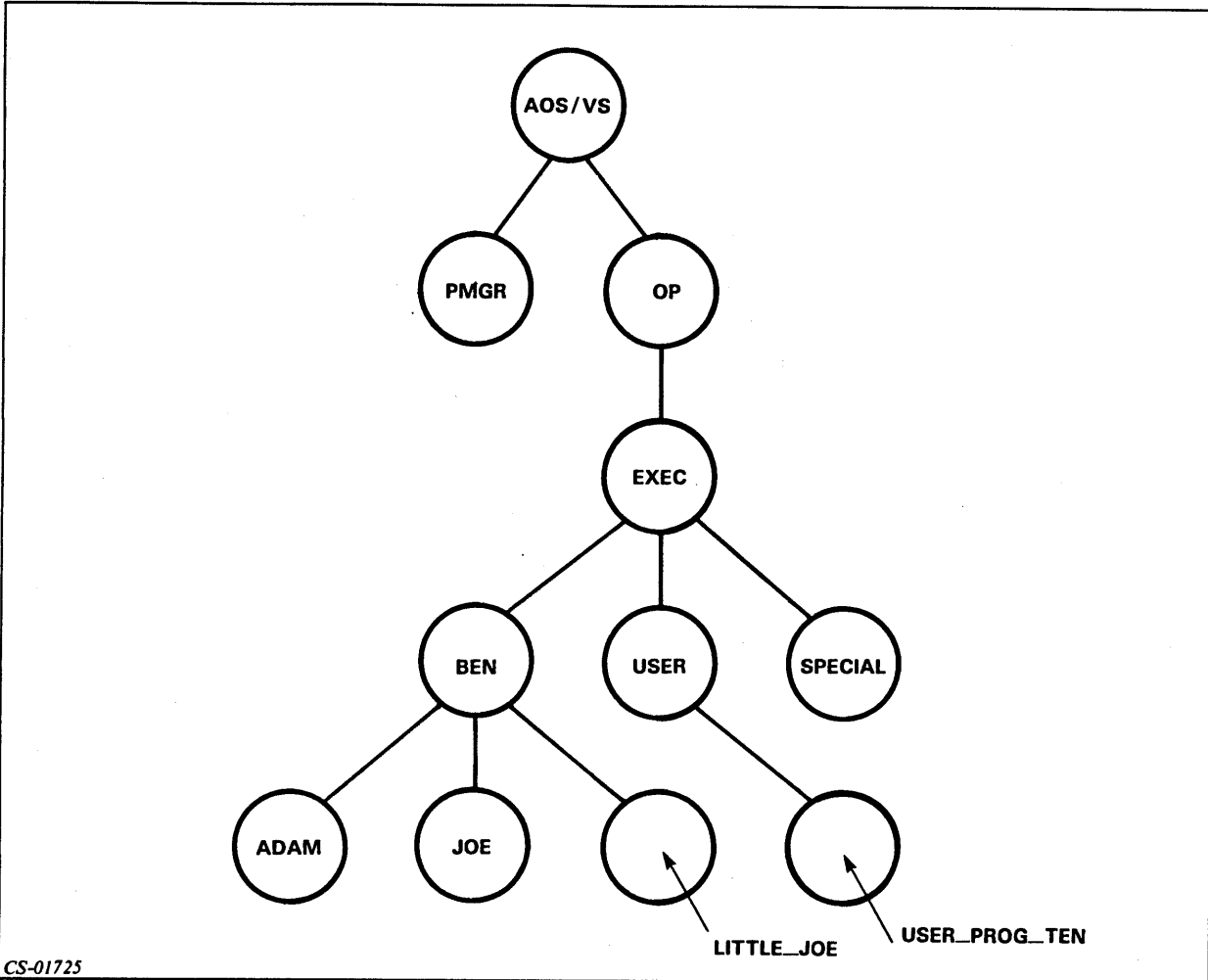


Figure 8.11 Process Tree Before EXECUTE Command

Now check your answer on the next page.

## Exercise 8-2 Answers



CS-01725

Figure 8.12 Process Tree After EXECUTE Command

If your answer is correct, continue with Module 8 by restarting the Module 8 audiotape. Otherwise, review the material and do this exercise again before you continue.

## PROCESS Command

The PROCESS command creates a son process. It differs from the EXECUTE command in two ways. Using the PROCESS command:

- You can determine privileges, priority, and type.
- Does not always block your process.

If you specify no switches, the new process has no privileges.

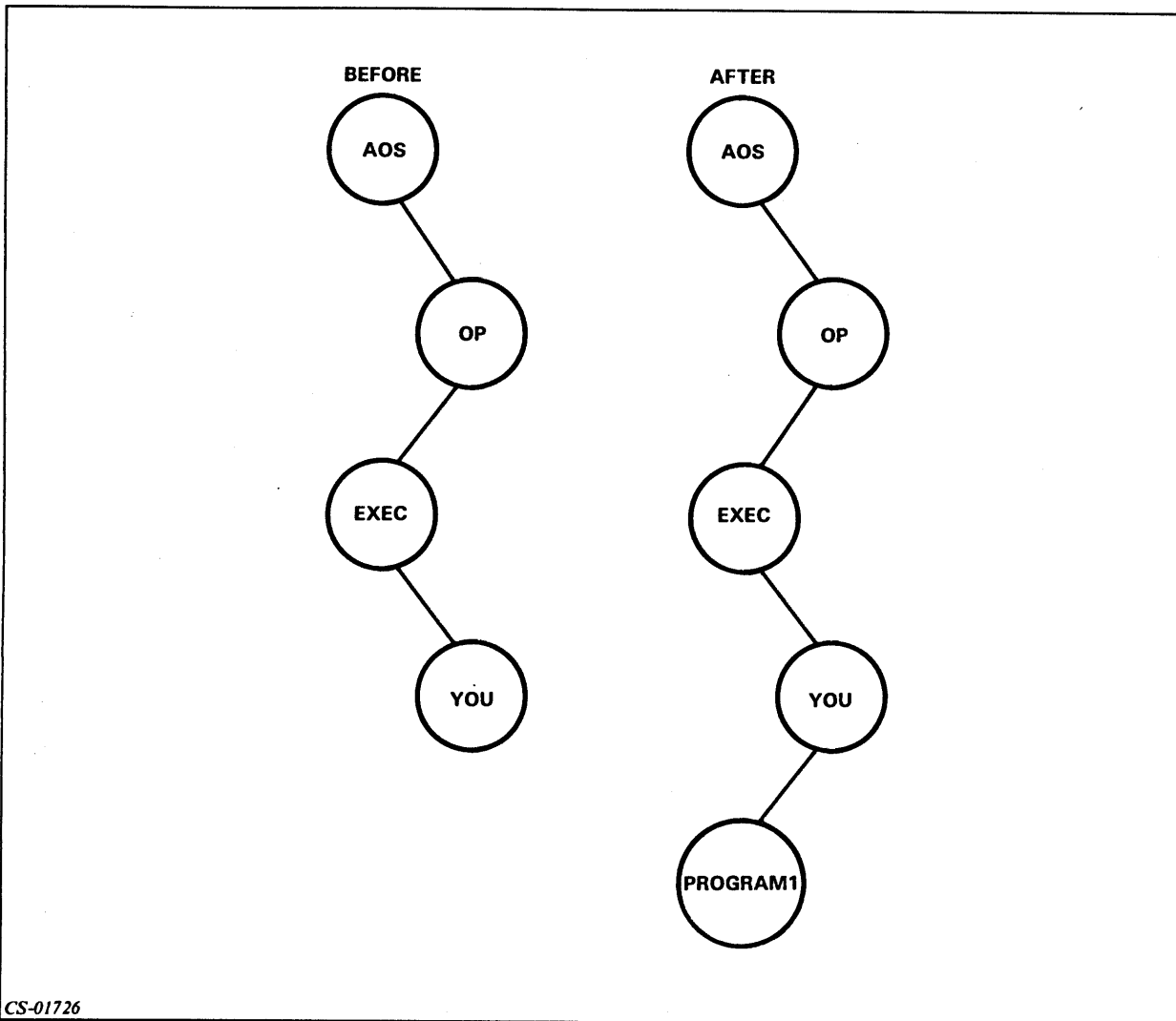
Switch	Result
Some Switches:	
/DEFAULT	Pass same privileges as creating process.
/BLOCK	The new process will block the old process, if you use /BLOCK, you must also use /IOC.
/PREEMPTIBLE	Make the type of the new process preemptible.
/RESIDENT	Make the type of the new process resident.
/PRIORITY= <i>n</i>	Set the priority of the new process to <i>n</i> .
Privilege Passing:	
/CHPRIORITY	Allow new process to change its priority.
/CHTYPE	Allow new process to change its type.
/SUPERUSER	Allow new process to use the superuser privilege.
/SUPERPROCESS	Allow new process to use the superprocess privilege.
Parameter Setting and Passing:	
/CONSOLE	Make the new process's console the same as the parent's.
/CONSOLE= <i>name1</i>	Make the new process's console the file referred to by <i>name1</i> .
/INPUT	Make the new process's input the same as the parent's.
/INPUT= <i>name2</i>	Make the new process's input the file referred to by <i>name2</i> .
/OUTPUT	Make the new process's output the same as the parent's.
/OUTPUT= <i>name3</i>	Make the new process's output the file referred to by <i>name3</i> .
/IOC	Make the new process's input, output and console the same as the parent's. (You must also use /BLOCK if you use /IOC).
/IOC= <i>name4</i>	Make the new process's input, output, and console the file referred to by <i>name4</i> .
/DATA	Make the new process's datafile the same as the parent's.
/DATA= <i>name5</i>	Make the new process's datafile the file referred to by <i>name5</i> .
/LIST	Make the new process's listfile the same as the parent's.
/LIST= <i>name6</i>	Make the new process's listfile the file referred to by <i>name6</i> .
/SONS	Make the new process's number of sons one less than the parent's number of sons.
/SONS= <i>n</i>	Make the new process's number of sons equal to <i>n</i> .
/DIRECTORY	Make the new process's directory the same as the parent's.
/DIRECTORY= <i>name7</i>	Make the new process's directory the file referred to by <i>name7</i> .

Table 8.B Switches on the PROCESS Command

If you try to pass a privilege that you do not have, you will see the following message:

**ERROR: CALLER NOT PRIVILEGED FOR THIS ACTION**

**Example 1**  
**PROCESS PROGRAM1**



CS-01726

Figure 8.13 Process Tree Before and After PROCESS Command

PROGRAM1 has only the privileges that you give it. Since you passed no privileges, PROCESS\_SON has no privileges.

**Example 2**

**PROCESS/IOC/BLOCK:CLI.PR**

Your process becomes blocked, and a new CLI is created using the same console for input and output operations. If you use the /IOC switch, you must also use /BLOCK.

**Example 3**

**PROCESS/RESIDENT/PRIORITY-1/SONS/SUPERUSER PAYROLL\_3**

Now do Exercise 8-3 on the next page.

## Exercise 8-3

**Directions:** Answer the following questions.

1. If you pass no privilege to a process that you create, it will have:
  - a. All privileges.
  - b. No privileges.
2. Can you pass a privilege that you do not have?
  - a. Yes
  - b. No
3. Write the command to create a process to run `PROG_1`, block your process, and use your terminal for input, output, and console files.  
\_\_\_\_\_
4. Write the command to create a process to run `PGM_10` that has the same privilege as your process. \_\_\_\_\_

Now check your answers on the next page.

## **Exercise 8-3**

### **Answers**

1. b.
2. b.
3. **PROCESS/IOC/BLOCK PROG\_1**
4. **PROCESS/DEFAULT PGM\_10**

If you answered all of the questions correctly, continue with Module 8 by restarting the Module 8 audiotape. Otherwise, review the material and do this exercise again before you continue.

## Process Control

### TERMINATE Command

Format: TERMINATE PID

**Example 1**  
TERMINATE 14

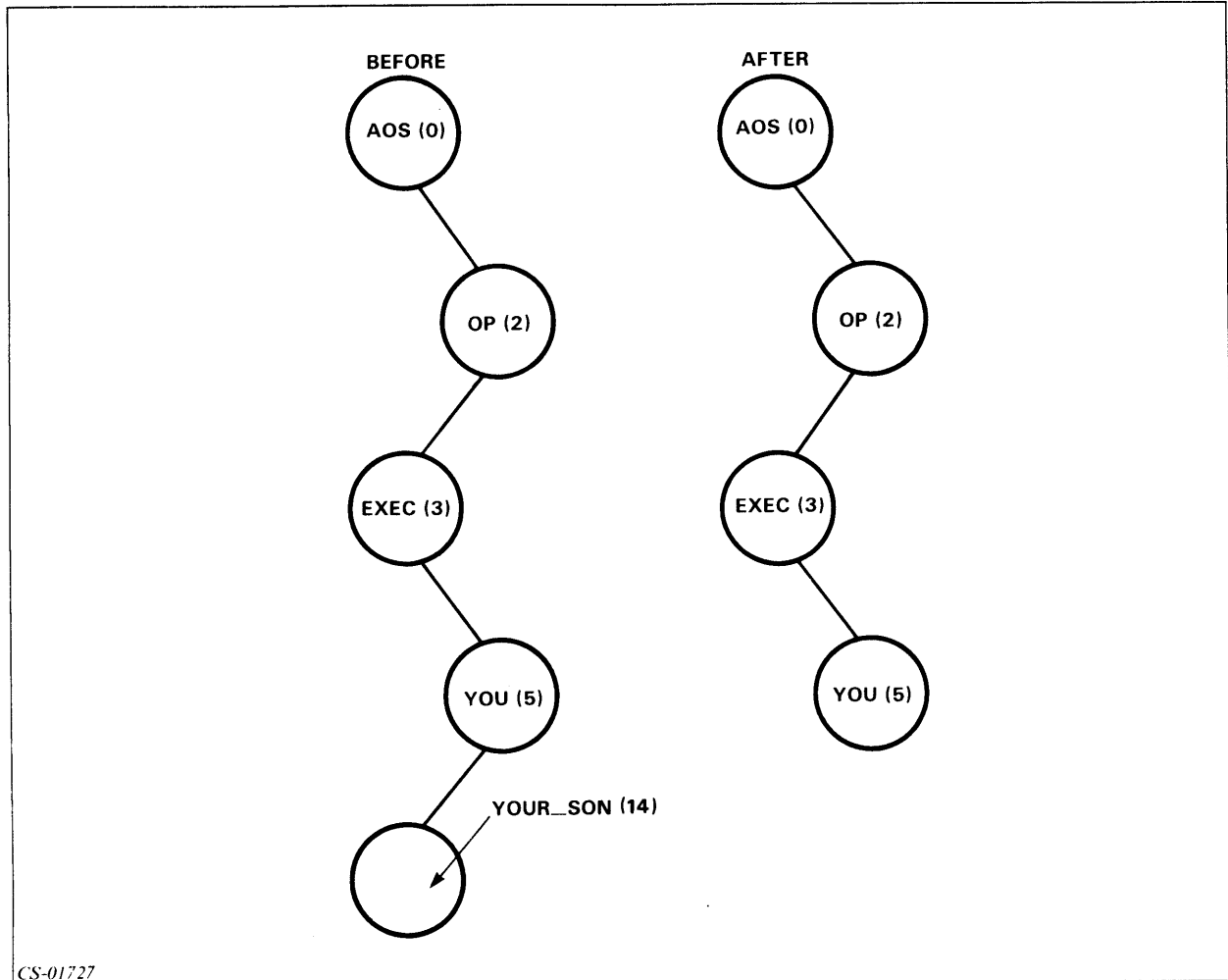
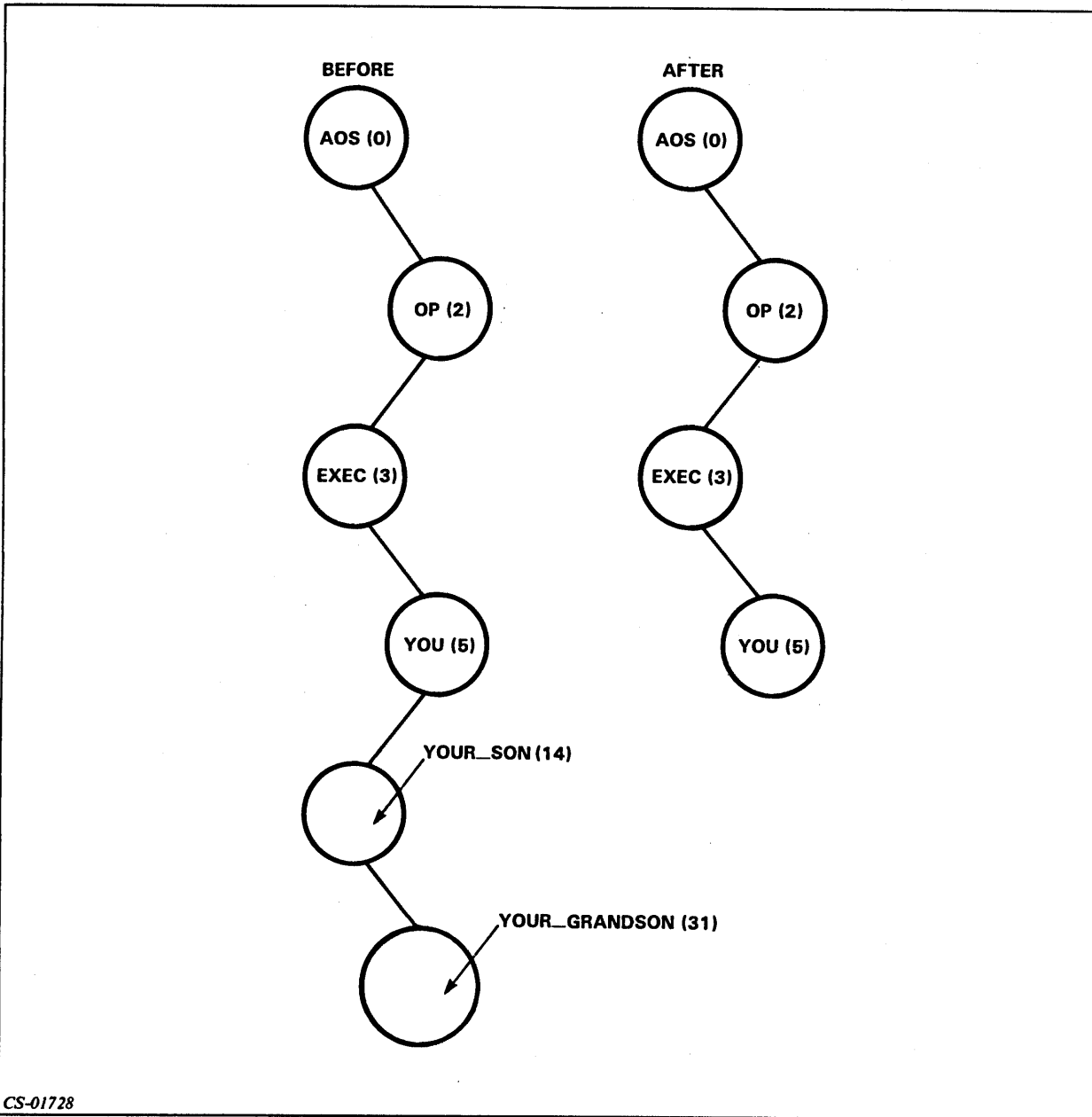


Figure 8.14 Process Tree Before and After TERMINATE Command

**Example 2**  
TERMINATE 14



CS-01728

Figure 8.15 Process Tree Before and After `TERMINATE` Command

PID 14 and PID 31 are both terminated by the `TERMINATE 14` command.

### SUPERPROCESS Command

The `SUPERPROCESS` command sets or displays the superprocess setting. When `SUPERPROCESS` is on, you can control any process.

Format: `SUPERPROCESS {ON}`  
`{OFF}`



Prompt	Superprocess	Superuser
)	OFF	OFF
*)	OFF	ON
+) )	ON	OFF
#) )	ON	ON

Table 8.C Superuser and Superprocess Prompts

**Example 1****1. SUPERPROCESS**

OFF

**2. SUPERPROCESS ON****3. +)SUPERPROCESS**

ON

**4. +)SUPERUSER**

OFF

**5. +)SUPERUSER ON****6. #)SUPERPROCESS OFF****7. \*)SUPERUSER OFF**

Now do Exercise 8-4 on the next page.



## Exercise 8-4

**Directions:** Using the process tree in Figure 8.16, write the command to:

1. Terminate process XYZ. \_\_\_\_\_
2. Terminate USER2 and ABC in one command. \_\_\_\_\_
3. Turn on the superprocess privilege. \_\_\_\_\_

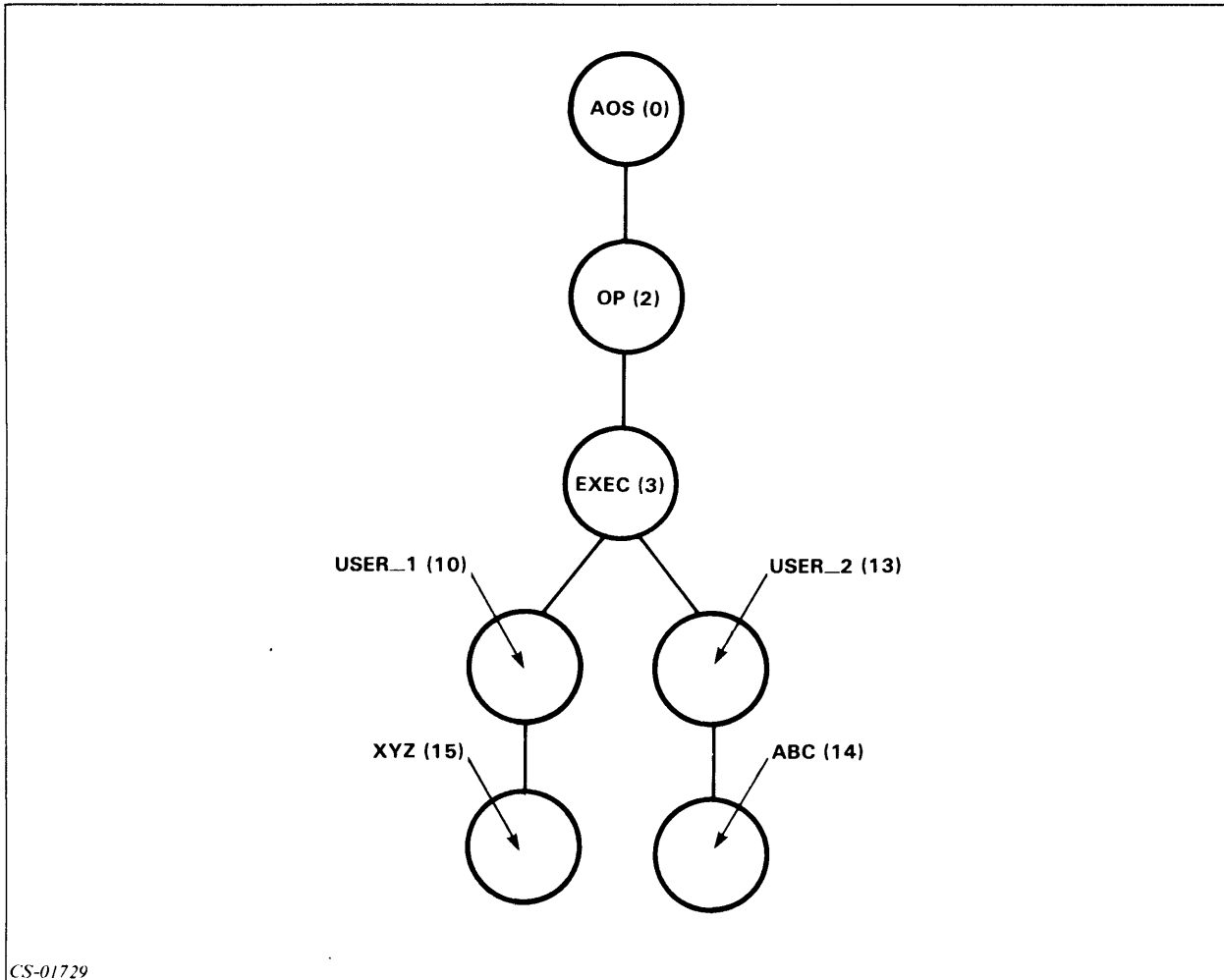


Figure 8.16

Now check your answers on the next page.

## **Exercise 8-4**

### **Answers**

- 1. TERMINATE 15**
- 2. TERMINATE 13**
- 3. SUPERPROCESS ON**

**If you answered all of the questions correctly, go on to Lab Activity 8-1. Otherwise, review the material and do the exercise again before continuing.**

## Lab Activity 8-1

1. Try to determine the process hierarchy on your system. Keep in mind that the hierarchy may be changing as you are doing this exercise.
  - a. Use the TREE command on Process 1
  - b. Use the WHO command to determine the name of each process listed
  - c. Repeat a and b for Process 2
  - d. Repeat c for the sons of Process 2.
  - e. Continue with the grandsons of Process 2.

Note: (Some of the processes in Steps d and e may terminate as you are working.)

2. Create a process. Make this process execute another copy of the CLI (:CLI.PR). Determine the process ID of the new process. Terminate the new process, but do not affect the creating process.
3. Determine the setting of the superprocess switch. Try to turn on superprocess. (You may not have the privilege.) If you turned it on, turn it off.

Now check your answers on the next page.

## Lab Activity 8-1

### Answers

1. The process hierarchy differs from system to system. Refer to the example before Figure 8.5 if you have difficulty.
2. **PROCESS/BLOCK/IOC :CLI.PR**  
**WHO**  
**TERMINATE nn**
3. **SUPERPROCESS**  
**SUPERPROCESS ON**  
**SUPERPROCESS OFF**

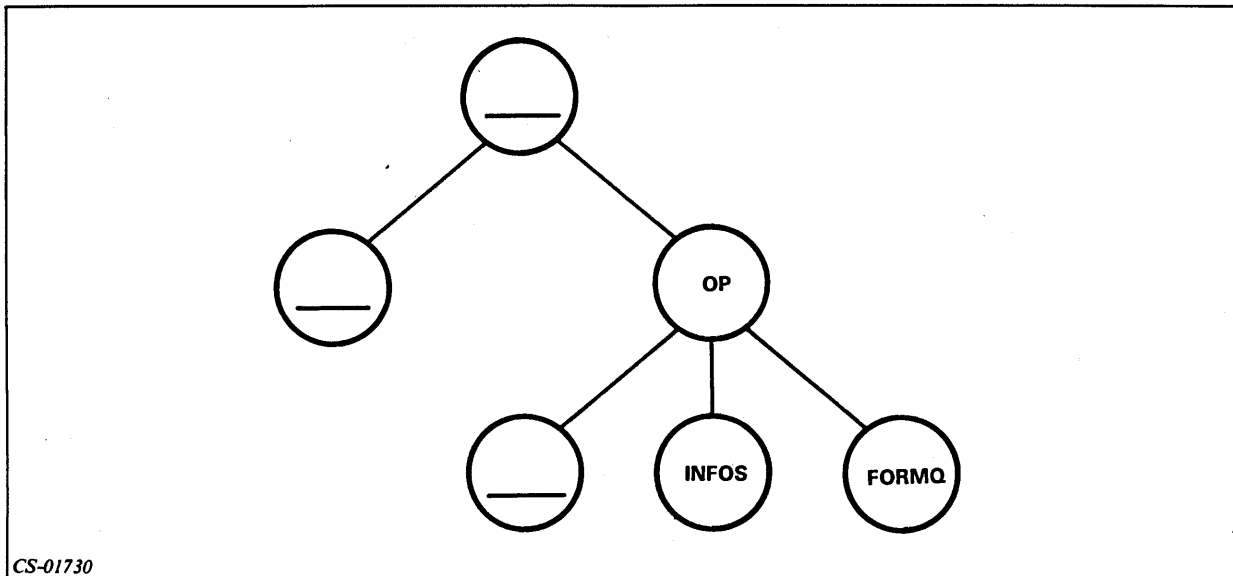
If you successfully completed this Lab Activity, continue to the Module 8 Test. If you had any difficulty, review the module and do the Lab Activity again before taking the Test.

## Module 8 Test

**Directions:** Answer the following questions by completing the sentence.

1. The three process types are:
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_
2. The three possible states that a process can attain are:
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_
3. The state in which all processes begin is:
  - a. Ineligible.
  - b. Eligible.
  - c. Blocked.
  - d. Resident.
4. A process consists of:
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
  - c. \_\_\_\_\_
  - d. \_\_\_\_\_
  - e. \_\_\_\_\_
  - f. \_\_\_\_\_
  - g. \_\_\_\_\_
  - h. \_\_\_\_\_

5. Identify the processes in the process tree shown in Figure 8.17.



CS-01730

Figure 8.17

6. Select the command that you can use to determine the father process of a process:
  - a. TREE
  - b. PARENT
  - c. PROCESS
  - d. SON
7. Select the command that you can use to determine the son of a process:
  - a. TREE
  - b. PARENT
  - c. PROCESS
  - d. SON
8. Select the command that cancels a subordinate process, but keeps your process intact:
  - a. BYE
  - b. TERMINATE
  - c. CANCEL
  - d. OFF



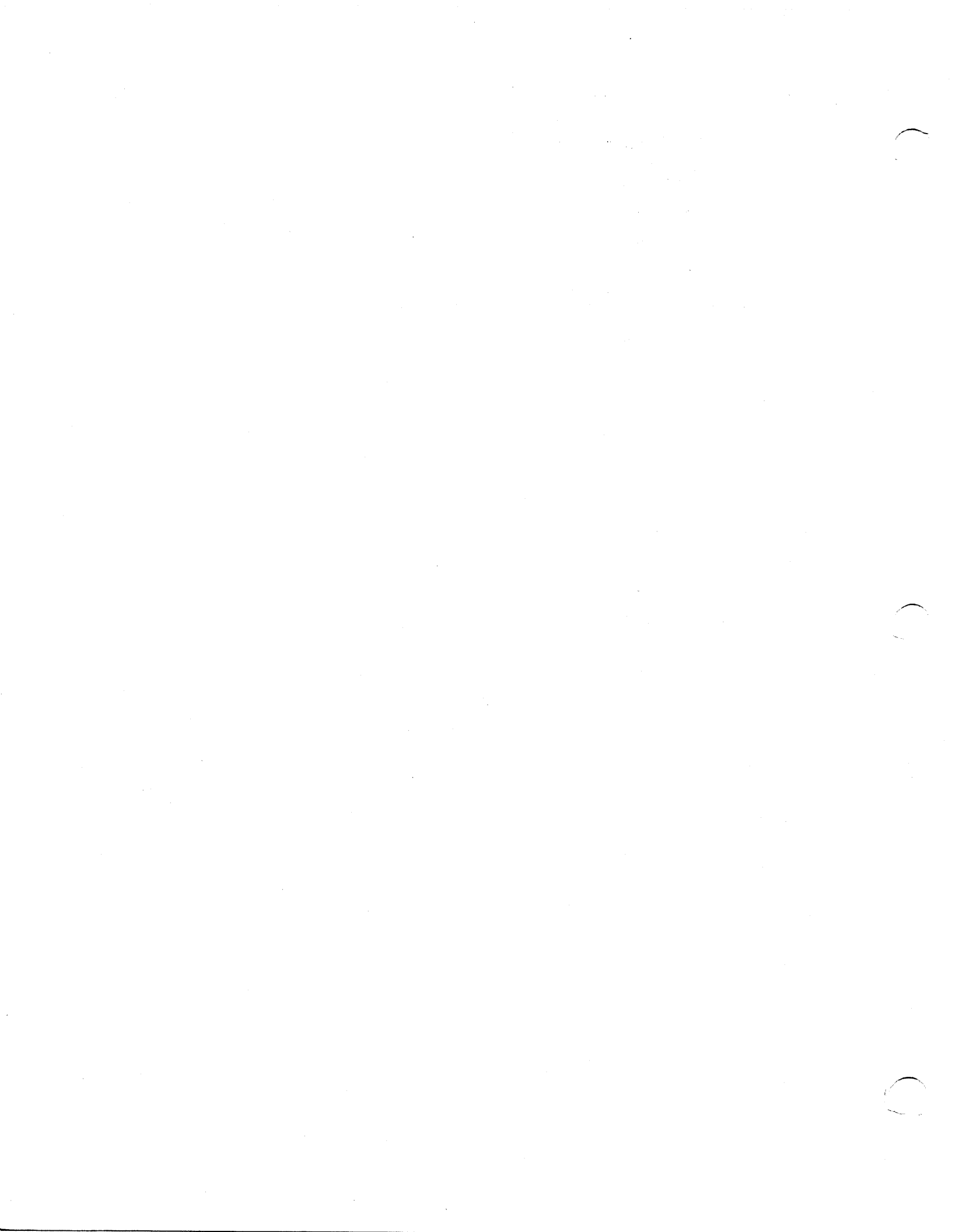
9. Select the command that creates a new subordinate process:
  - a. **START**
  - b. **CREATE**
  - c. **PROCESS**
  - d. **SON**
10. Select the command that allows you to control any process:
  - a. **SUPERUSER**
  - b. **BOSS**
  - c. **MASTER**
  - d. **SUPERPROCESS**

**Directions:** Write the command to:

11. Find the father and any sons of process 14. \_\_\_\_\_
12. Terminate process 17.  
\_\_\_\_\_
13. Turn on the superprocess privilege. \_\_\_\_\_
14. Determine the process name of process 13. \_\_\_\_\_
15. Create a process that blocks your process, uses your console for input and output, and runs a program called GOOD\_STUFF. \_\_\_\_\_
16. Create a process that has no privileges and executes a program called MODULE\_EIGHT\_CONCLUSION.  
\_\_\_\_\_

Now check your answers to the Module 8 Test in Appendix A. If you answered all the questions correctly, go on to Module 9. Otherwise, review the material in Module 8 and take the Test again before continuing.

This concludes Module 8.



# Module 9

## Advanced CLI Concepts

### Introduction

In this module we will discuss the use of macros and pseudo-macros. We will see how to use them to ease the performance of repetitive tasks. In addition, we will look at some commands that you can use in conjunction with magnetic tape processing.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Write a CLI macro.
2. Use pseudo-macros in command lines.
3. Use the CLI commands `LOAD` and `DUMP` to perform magnetic tape operations.

### Resources

To complete this module, you will need:

- Module 9 audiotape.
- Module 9 of your *Student Guide*.
- Audiotape playback unit.

### Module Outline

Module 9 discusses the following topics:

1. Tape processing
  - a. `DUMP` command
  - b. `LOAD` command

2. Macros
  - a. Creating macros
  - b. Using macros
  - c. Passing arguments
3. Pseudo-macros
  - a. Environmental
  - b. Conversion
  - c. Conditional

Now start the Module 9 audiotape. As you listen, follow along in Module 9 of your *Student Guide*.

## Tape Processing

Tape backup:

- Adds data security.
- Protects against disasters.
- Protects against mistaken deletions.
- Protects against hardware problems.

### DUMP Command

The DUMP command copies one or more files into a specified destination file.

Format: **DUMP,DESTINATION\_FILENAME,SOURCE\_PATHNAME**

Switches on the DUMP command:

- /V Verifies or lists the names of the files dumped.
- /NACL Eliminates ACLs when files are dumped.

#### Example 1

**DUMP @MTA0:0,SPECIAL+**

Dump all files that are named SPECIAL followed by any other characters onto the tape mounted on the tape drive referred to as MTA0.

#### Example 2

**DUMP/V,:UTIL:DUMP\_AREA\_8**

Dump all files in the working directory and in all subordinate directories into a disc file called :UTIL:DUMP\_AREA\_8. Whenever you do not specify a source file, all files in or subordinate to the working directory are dumped. Use extreme caution to ensure that the dump file is not in or subordinate to the working directory. The /V switch causes all files dumped to be listed at your console.

To back up your files without the help of a macro, use this procedure:

Monday

**DUMP,@MTA1:0,FILE\_1**

**DUMP,@MTA1:1,DATA\_FILE**

**DUMP,@MTA1:2,SPECIAL\_DATA**

**DUMP,@MTA1:3,PERSONNEL\_FILE**

**DUMP,@MTA1:4,PAYROLL\_FILE**

**DUMP,@MTA1:5,ACCOUNT\_DATA**

Tuesday

**DUMP,@MTA1:0,FILE\_1**

**DUMP,@MTA1:1,DATA\_FILE**

**DUMP,@MTA1:2,SPECIAL\_DATA**

**DUMP,@MTA1:3,PERSONNEL\_FILE**

**DUMP,@MTA1:4,PAYROLL\_FILE**

**DUMP,@MTA1:5,ACCOUNT\_DATA**

Wednesday

DUMP...  
DUMP...  
DUMP...  
DUMP...  
DUMP...  
DUMP...

## LOAD Command

The LOAD command copies one or more files from FILENAME to a specified source file.

Format: **LOAD,FILENAME,SOURCE\_FILENAME**

Switches on the LOAD command:

- **/DELETE** Deletes any existing file that has the same name as file on the dump file and replaces it with file from the dump file.
- **/N** Do not load files; only list file names and dates.
- **/V** Verify each loaded file.

### Example 1

**LOAD @MTA0:0 +.BU**

Load all files that end in the characters .BU from the first file on tape drive MTA0 into the working directory.

### Example 2

**LOAD/N @MTA1:1**

List the filenames and dates of all the files in the second file that is mounted on tape drive MTA1.

### Example 3

**LOAD/V/DELETE DUMP\_FILE\_A**

Load all of the files that are in the disc file DUMP\_FILE\_A into the working directory. If a file of the same name exists, delete the existing file and replace it with the one from the dump file. Provide a list of all files that are loaded.

## Macros

A *macro* is a file that contains a list of commands. To create a macro, you can use the CREATE command.

### Example 1

```
1. CREATE/I DUMPER.CLI
2. )DUMP,@MTA1:0,FILE_1
3. )DUMP,@MTA1:1,DATA_FILE
   )DUMP,@MTA1:2,SPECIAL_DATA
   )DUMP,@MTA1:3,PERSONNEL_FILE
   )DUMP,@MTA1:4,PAYROLL_FILE
   )DUMP,@MTA1:5,ACCOUNT_DATA
4. ))
```

To invoke this macro, only a single command is required. Note that it is not necessary to add the .CLI extension to the macro name.

### DUMPER

To back up your files after you have written this macro, all that you must do is:

```
Monday      DUMPER
Tuesday     DUMPER
Wednesday   DUMPER
Thursday    DUMPER
```

### Example 2

If you wanted to also have a file TRANSACTION\_01\_22\_82 (01\_22\_82 is the transaction date.) in your dump file, you would write DUMPER this way:

```
CREATE/I DUMPER.CLI
)DUMP,@MTA1:0,FILE_1
)DUMP,@MTA1:1,DATA_FILE
)DUMP,@MTA1:2,SPECIAL_DATA
)DUMP,@MTA1:3,PERSONNEL_FILE
)DUMP,@MTA1:4,PAYROLL_FILE
)DUMP,@MTA1:5,ACCOUNT_DATA
)DUMP,@MTA1:6,TRANSACTION_%1%    (Add this line to include transaction file.)
))
```

You would run DUMPER this way: **DUMPER 06\_01\_82**

## Argument Passing

- %1% First argument
- %2% Second argument
- %3% Third argument
- .
- .
- %n% nth argument
- %0% Macro name

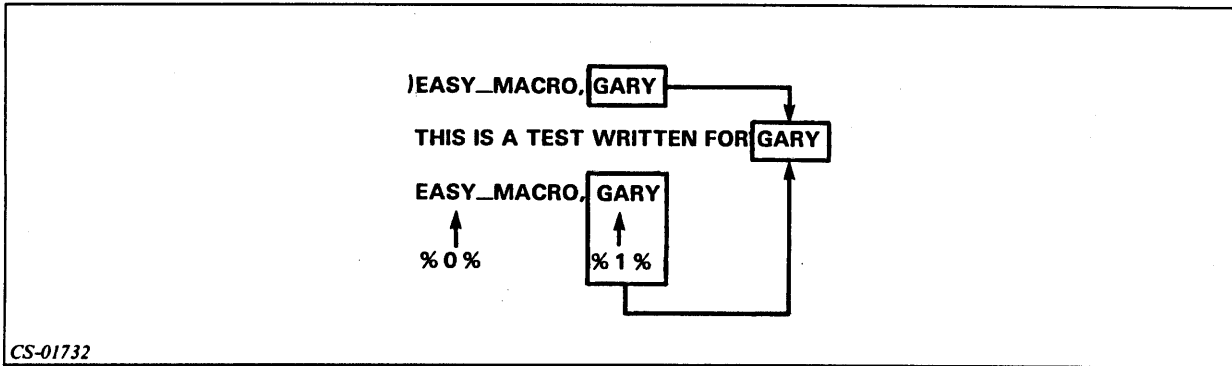
### Example 1

```
CREATE/I EASY_MACRO
)WRITE THIS IS A TEST WRITTEN FOR %1%.
)
```

To execute and see the results:

```
EASY_MACRO GARY
THIS IS A TEST WRITTEN FOR GARY.
```

```
EASY_MACRO,GARY
  %0%      %1%
```



CS-01732

Figure 9.1 Results of EASY\_MACRO

### Example 2

```
CREATE/I TEST_MACRO
```

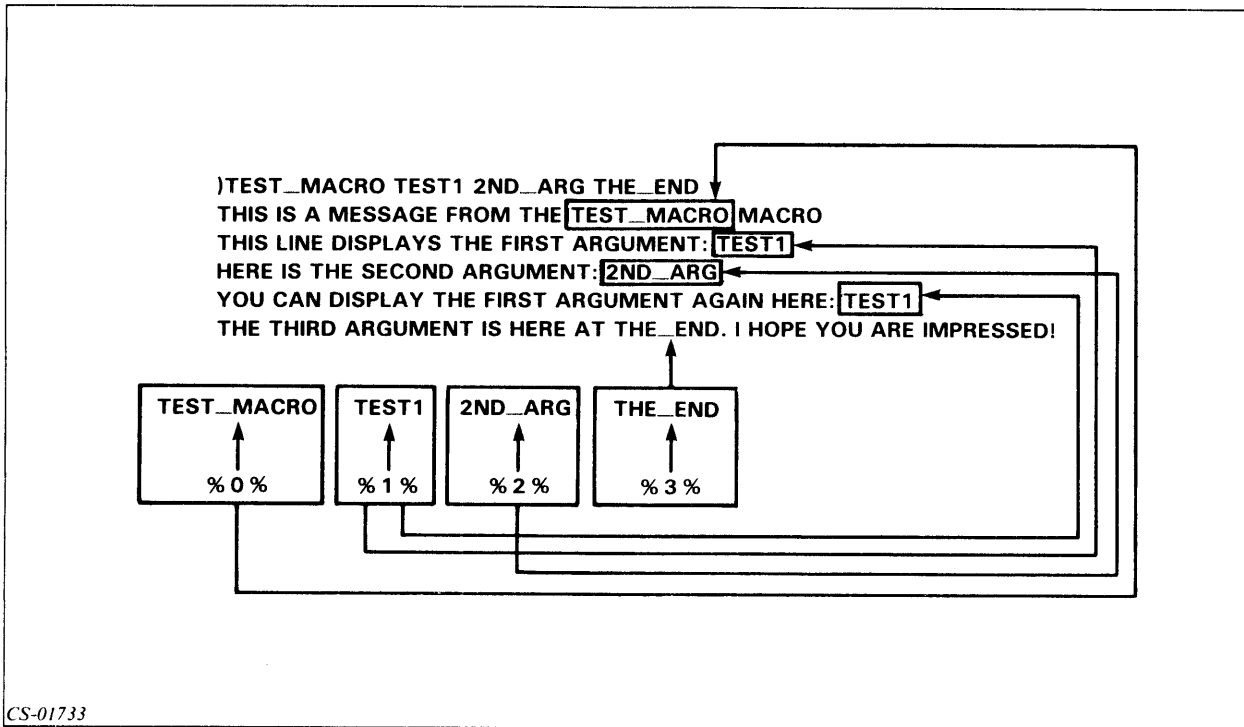
1. )WRITE THIS IS A MESSAGE FROM THE %0% MACRO
2. )WRITE THIS LINE DISPLAYS THE FIRST ARGUMENT: %1%
3. )WRITE HERE IS THE SECOND ARGUMENT: %2%
4. )WRITE YOU CAN DISPLAY THE FIRST ARGUMENT AGAIN HERE: %1%
5. )WRITE THE THIRD ARGUMENT IS HERE AT %3%. I HOPE YOU ARE IMPRESSED!



To execute and see the results:

```
TEST_MACRO TEST1 2ND_ARG THE_END
THIS IS A MESSAGE FROM THE TEST_MACRO MACRO
THIS LINE DISPLAYS THE FIRST ARGUMENT: TEST1
HERE IS THE SECOND ARGUMENT: 2ND_ARG
YOU CAN DISPLAY THE FIRST ARGUMENT AGAIN HERE: TEST1
THE THIRD ARGUMENT IS HERE AT THE_END. I HOPE YOU ARE IMPRESSED!

TEST_MACRO TEST1 2ND_ARG THE_END
    %0%      %1%      %2%      %3%
```



CS-01733

Figure 9.2 Results of TEST\_MACRO

Now do Exercise 9-1 on the next page.



## Exercise 9-1

**Directions:** Select the correct answer.

1. Select the command to copy all of the files in your working directory to a tape.
  - a. **LOAD @MTA0:0**
  - b. **COPY TAPE DIRECTORY**
  - c. **DUMP @MTA0:0**
  - d. **DUMP @MTA1:3,MYFILES**
2. Select the command that loads the contents of a tape into your working directory.
  - a. **LOAD @MTA0:0**
  - b. **COPY TAPE DIRECTORY**
  - c. **DUMP @MTA0:0**
  - d. **DUMP @MTA1:3,MYFILES**
3. The primary purpose of using a macro is:
  - a. To save storage space.
  - b. To save work for the computer user.
  - c. To reduce computer usage.
  - d. To create back-up tapes.
4. You can pass arguments to a macro.
  - a. True
  - b. False
5. The LOAD and DUMP commands only work with magnetic tape.
  - a. True
  - b. False

**Directions:** Construct macros for the following.

6. Write a macro called `SEE_FILES` that displays the names of all of the files in your working directory and writes a message `ALL FINISHED` after the display.

---

---

---

---

7. Write a macro called **SPECIAL** that accepts a name as an argument and returns **THIS IS ESPECIALLY FOR**, and then the name that was entered.

---

---

---

---

Now check your answers on the next page.

## Exercise 9-1

### Answers

1. c.
2. a.
3. b.
4. a.
5. b.
6. **CREATE/I SEE\_FILES.CLI**  
    **)FILESTATUS**  
    **)WRITE ALL FINISHED**  
    **)**
7. **CREATE/I SPECIAL.CLI**  
    **)THIS IS ESPECIALLY FOR %1%**  
    **)**

If you answered all of the questions correctly, continue with Module 9 by restarting the Module 9 audiotape. Otherwise, review the material and do this exercise again before you continue.

## Pseudo-macros

*Pseudo-macros* return values that you can use in your processing. They are always enclosed in square brackets and preceded by an exclamation point (i.e., [!pseudo-macro]). The three types of pseudo-macros are:

- Environmental settings (i.e., SEARCHLIST, ACL, TIME, DATE).
- Conditional execution (i.e., EQUAL, ELSE, END).
- Conversions (i.e., OCTAL, DECIMAL).

Pseudo-macro	Expands to:
IACL	ACL of a file.
IASCII	Character corresponding to value.
IDATE	System date.
IDECIMAL	Decimal value of octal number.
IDEFACL	Current default ACL.
IDIRECTORY	Working directory.
IELSE	Conditional.
IEND	Conditional.
IEQUAL	Conditional.
IEXPLODE	Expands argument with spaces.
INEQUAL	Conditional.
IOCTAL	Converts decimal to octal.
IPID	Process ID.
IREAD	Displays text and accepts argument.
ISEARCHLIST	Your searchlist.
ITIME	System time.
IUSERNAME	Your username.

Table 9.A Commonly Used Pseudo-macros

**Example 1**  
**WRITE [!DATE]**  
 16-MAY-82

**Example 2**  
**WRITE [!SEARCHLIST]**  
 :UTIL, :UTIL:INFOS, :UDD:RYAN

**Example 3**  
**WRITE [!OCTAL 999]**  
 1747

**Example 4**  
**WRITE [!DECIMAL 777]**  
 511

**Example 5**  
**WRITE [!ASCII 207]**  
 (You will hear a beep tone <CTRL-G>)

**Example 6**  
**WRITE [!USERNAME] IS P.I.D. [!PID]**  
 RYAN IS P. I. D. 14

**Example 7**  
**SEARCHLIST**  
 :UTIL, :UTIL:INFOS  
**SEARCHLIST,[!SEARCHLIST],:UDD:PROJECT\_X**  
**SEARCHLIST**  
 :UTIL, :UTIL:INFOS, :UDD:PROJECT\_X

**Example 8**  
**DIRECTORY**  
 :UDD  
**DIRECTORY [!DIRECTORY]:MODULE\_9**  
**DIRECTORY**  
 :UDD:MODULE\_9

**Example 9**  
 The !ACL pseudo-macro always requires an argument.

**ACL OLDFILE**  
 RYAN,OWARE,+ .JOE,RE  
**ACL,OLDFILE,[!ACL,OLDFILE],+,E**  
**ACL OLDFILE**  
 RYAN,OWARE,JOE.RE,+ ,E

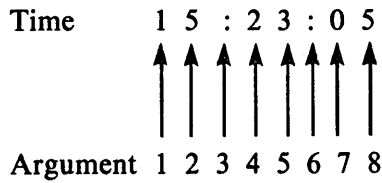
**Example 10**  
**CREATE/I DELMAC.CLI**  
**)DELETE/V/C [!READ DELETE WHAT FILE?]**  
**)WRITE DELETIONS COMPLETED AT [!TIME] ON [!DATE]**  
**)**  
**DELMAC**  
 DELETE WHAT FILE? +.BU  
 =SCRIPT.BU? Y  
 DELETED =SCRIPT.BU  
 =GUIDE.BU? Y  
 DELETED =GUIDE.BU  
 DELETIONS COMPLETED AT 16:16:13 ON 22-JAN-82

**Example 11**

The following macros illustrate how to use the time of day as input to a program that requires a random number as input.

```
CREATE/I RANDO.CLI
)RANDOM [!EXPLODE [!TIME]]
)WRITE DONE
))
```

This line creates a CLI command that looks like this:  
RANDOM 1 5 : 2 3 : 0 5



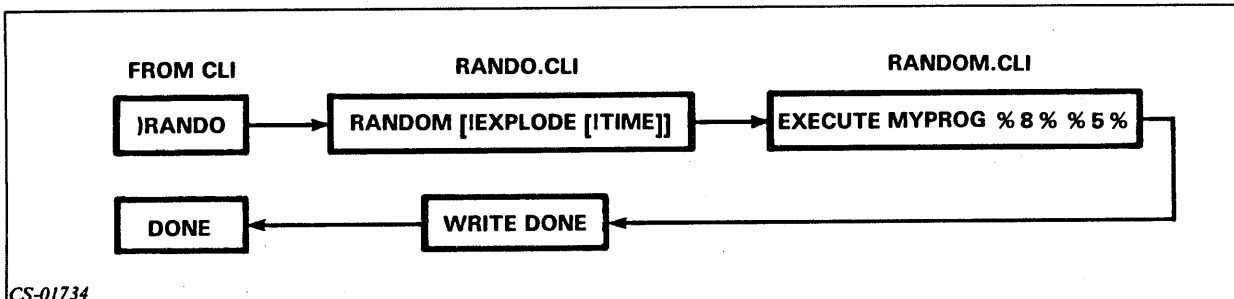
This will call a macro named RANDOM.CLI and pass eight arguments to it. Each of the numbers is an argument, and each colon is also an argument.

```
CREATE/I RANDOM.CLI
)XEQ MYPROG %8% %5%
))
```

This will execute a program named MYPROG and pass a random number in the range of 00 to 99 to it. The number is composed of the digit portion of the seconds from the time of day and the digit portion from the minutes.

To execute:

**RANDO**



CS-01734

Figure 9.3 Flow of Control of RANDO Macro



*Conditional pseudo-macros* allow selected execution of commands.

**Example 1**

```
CREATE/I AM_PM.CLI
)[!EQUAL %1% 0]
)WRITE MORNING
)[!ELSE]
)WRITE AFTERNOON
)[!END]
))
AM_PM 0
MORNING
AM_PM 1
AFTERNOON
```

**Example 2**

```
CREATE/I SAFETY.CLI
)[!EQUAL [!USERNAME] MANAGER]
)XEQ ACCOUNTS
)[!ELSE]
)WRITE *** EXECUTION ABORTED INVALID USER ***
)[!END]
))
```

For any user not logged on as username MANAGER, the program accounts will not execute. This method can be used to add a level of security beyond simply using ACLs.

Now do Exercise 9-2 on the next page.



## Exercise 9-2

**Directions:** Fill in the appropriate returned value, given the following conditions:

USERNAME: TRICIA  
TIME: 10:45 PM  
DATE: 10-24-82  
WORKING DIRECTORY: :UDD:MACRO\_DIR  
PID: 21

**Example**

**WRITE** [!TIME] 22:45:00 (Remember the proper format.)

1. **WRITE** [!DATE] \_\_\_\_\_
2. **WRITE YOUR WORKING DIRECTORY IS** [!DIRECTORY]  
\_\_\_\_\_
3. **WRITE** [!USERNAME] **IS PROCESS I.D.** [!PID]  
\_\_\_\_\_
4. **WRITE** [!EXPLODE [!USERNAME]]  
\_\_\_\_\_
5. **WRITE THIS IS THE END** \_\_\_\_\_
6. In the space below, write a macro to write the time of day if you enter a T and write the date otherwise. Call the macro TIDA.CLI.

Check your answers on the next page.

## Exercise 9-2

### Answers

1. 24-OCT-82
2. YOUR WORKING DIRECTORY IS :UDD:MACRO\_\_DIR
3. TRICIA IS PROCESS I.D. 21
4. T R I C I A
5. THIS IS THE END
6. CREATE/I TIDA.CLI  
)[!EQUAL %1%.T]  
)WRITE [!TIME]  
)[!ELSE]  
)WRITE [!DATE]  
)[!END]  
))

or:

```
CREATE/I TIDA.CLI  
)[!EQUAL %1%T]  
)TIME  
)[!ELSE]  
)DATE  
)[!END]  
))
```

If you answered all of the questions correctly, continue to Lab Activity 9-1. Otherwise, review the material and do this exercise again before you continue.

## Lab Activity 9-1

**Directions:** Enter the commands necessary to accomplish each of the following tasks.

1. Write and execute a macro that displays the time, date, and working directory. Call the macro WHEN\_WHERE.
2. Write and execute a CLI command line that displays the time with a space between each character.
3. Enter and execute a macro that prints OPTION 1 if you enter a 1 as an argument, and prints INVALID if you enter any other character.
4. Back up all of the files in your directory on magnetic tape.
5. Get a list of the files on the tape, but do not replace them in your directory.
6. Create a macro to do Steps 4 and 5. Include in the macro steps that tell you what is happening as it happens.
7. Examine some macros on your system. Make :UTIL your working directory. List the files in :UTIL that have the .CLI extension. Type some of the files that are of interest to you and to which you have read access.

Now check your answers on the next page.

## Lab Activity 9-1

### Answers

1. **CREATE/I WHEN\_WHERE.CLI**  
    **)TIME**  
    **)DATE**  
    **)DIRECTORY**  
    **)**
2. **WRITE [!EXPLODE [!TIME]]**
3. **CREATE/I OPTION.CLI**  
    **[!EQUAL %1%,1]**  
    **WRITE OPTION 1**  
    **[!ELSE]**  
    **WRITE INVALID**  
    **[!END]**
4. **DUMP @ MTA0:0** (See your system manager for correct magnetic tape drive).
5. **LOAD/N @MTA0:0**
6. **CREATE/I SUPERDUMP.CLI**  
    **DUMP @MTA0:0**  
    **WRITE FILES ARE DUMPED**  
    **LOAD/N @MTA0:0**  
    **WRITE MACRO COMPLETE**
7. **DIRECTORY :UTIL**  
    **FILESTATUS +.CLI**  
    **TYPE ....CLI**

If you completed this Lab Activity successfully, continue to the Module 9 Test. If you had any difficulty, review the material and try again before continuing.

## Module 9 Test

**Directions:** Answer the following questions.

1. Which of the following is in the correct format?
  - a. `{!DATE}`
  - b. `[!DATE]`
  - c. `{DATE}`
  - d. `[DATE]`
2. Which of the following can be used to create a back-up file?
  - a. `LOAD`
  - b. `DUMP`
  - c. `WRITE`
  - d. `SAVE`
3. Which of the following can be used to copy a back-up file into your directory?
  - a. `LOAD`
  - b. `DUMP`
  - c. `WRITE`
  - d. `SAVE`
4. Which of the following commands converts an octal 111 to decimal?
  - a. `WRITE [!DECIMAL 111]`
  - b. `WRITE [!DECIMAL 73]`
  - c. `WRITE [!OCTAL 111]`
  - d. `WRITE [!OCTAL 73]`
5. To access the macro name within a macro, which would you use?
  - a. `%0%`
  - b. `%1%`
  - c. `%2%`
  - d. `%3%`

6. Write a macro called QUIZ6\_6.CLI that executes a program called PROG1 and prints a message when completed.

---

---

---

---

7. Write a macro that executes PROG2 if you enter a 1 as an argument, and executes PROG2 otherwise.

---

---

---

---

---

---

8. Using pseudo-macros, write a command line that adds the directory :UTIL to your searchlist.

---

9. Write a series of macros called MAC1, MAC2, that prints out the seconds portion of the time of day.

---

---

---

Now check your answers to the Module 9 Test in Appendix A. If you answered all the questions correctly, continue to Module 10. Otherwise, go back and review the material in Module 9 and take the Module 9 Test again.

This concludes Module 9.



# Module 10

## The SPEED Editor

### Introduction

This module explains the elementary features of the SPEED text editor (AOS, AOS/VS). Although not all of the features of SPEED are discussed, this module allows you to use enough of the features to develop and edit files for use in program development. For a more detailed discussion of all of the features, refer to *SPEED Text Editor (AOS and AOS/VS) User's Manual*.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. List the steps involved in a typical editing session.
2. Identify and use the SPEED commands that perform the following functions:
  - a. Open a file for input.
  - b. Create and open a file for output.
  - c. Change the location of the Character Pointer.
  - d. Insert text into the edit buffer.
  - e. Search for text in the edit buffer.
  - f. Display text in the edit buffer.
  - g. Change text in the edit buffer.
  - h. Delete text from the edit buffer.
  - i. Read text into the edit buffer.
  - j. Move text to the output file.
  - k. Create temporary output files.
  - l. Create back-up files.
  - m. Close files.
  - n. Return to the CLI.

## Resources

To complete this module, you will need:

- Module 10 audiotape.
- Module 10 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

Module 10 discusses the following topics:

1. SPEED concepts
  - a. Units of text
  - b. Files and the edit buffer
  - c. Editing steps
  - d. Command structure
  - e. Character Pointer
2. Sample editing session
  - a. Opening an output file
  - b. Inserting text
  - c. Character Pointer commands
  - d. Editing text
  - e. Moving text to the output file
  - f. Returning to the CLI
  - g. Opening and reading from an input file
3. Advanced commands
  - a. Page and window mode
  - b. Commands that perform several functions
  - c. Creating a temporary output file
  - d. Creating a back-up file

## SPEED Concepts

### Units of Text

*Text* is a sequence of one or more ASCII characters.

Unit of Text	Definition	Example
Character	A single ASCII alphanumeric character. SPEED uses the full upper-case and lower-case ASCII character set.	A
String	A sequence of ASCII characters. A string can contain any ASCII character, except delimiters such as carriage returns.	THIS IS A STRING
Line	A sequence of characters up to and including a carriage return.	THIS IS A LINE }
Page	A sequence of characters ending in a new page character: CTRL-L. A page has no size limit.	(See Figure 10.1.)
Window	A sequence of characters divided into a specific number of lines.	(See Figure 10.1.)

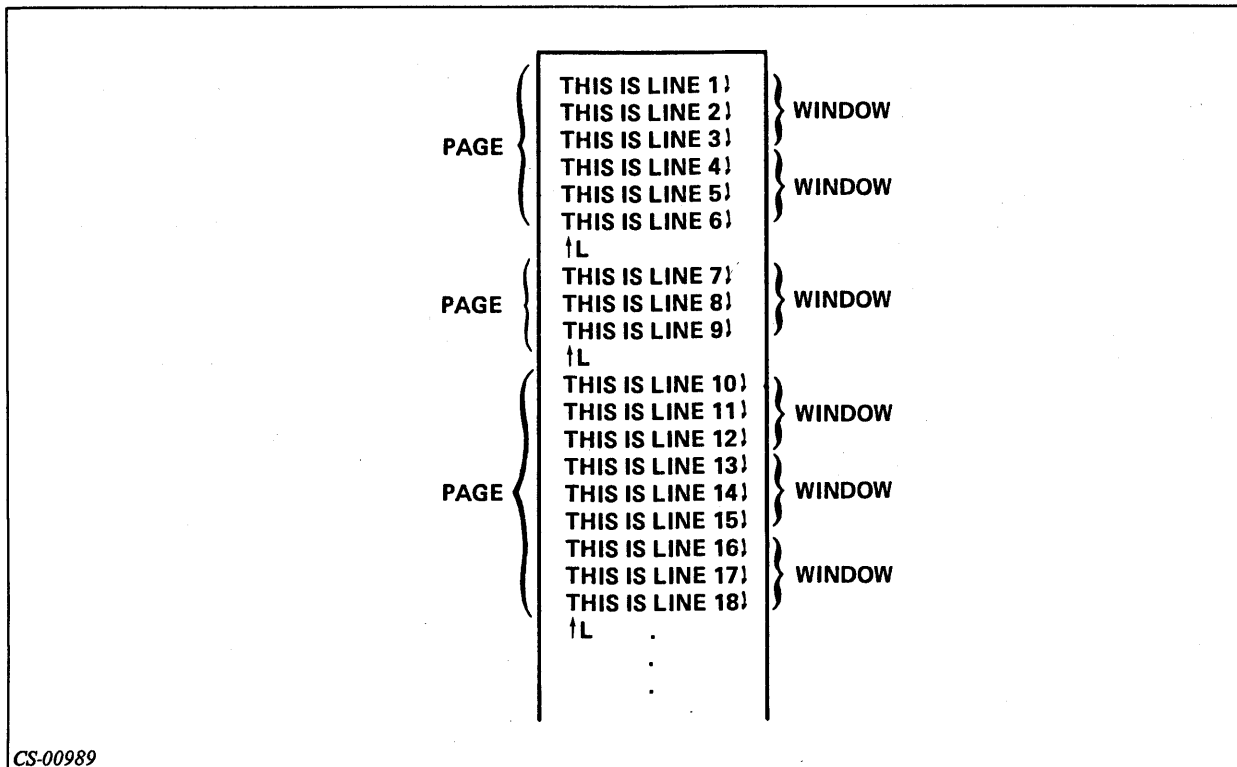
Table 10.A Units of Text that SPEED Recognizes

Special-purpose characters:

- Command terminators: CTRL-D = \$\$
- String delimiter: ESCape = \$
- Command separator: ESCape = \$

### Edit Buffer

- Area of memory where SPEED manipulates your text.
- Limited by memory size.
- 36 buffers available.



CS-00989

Figure 10.1 Pages and Windows

## Editing Steps

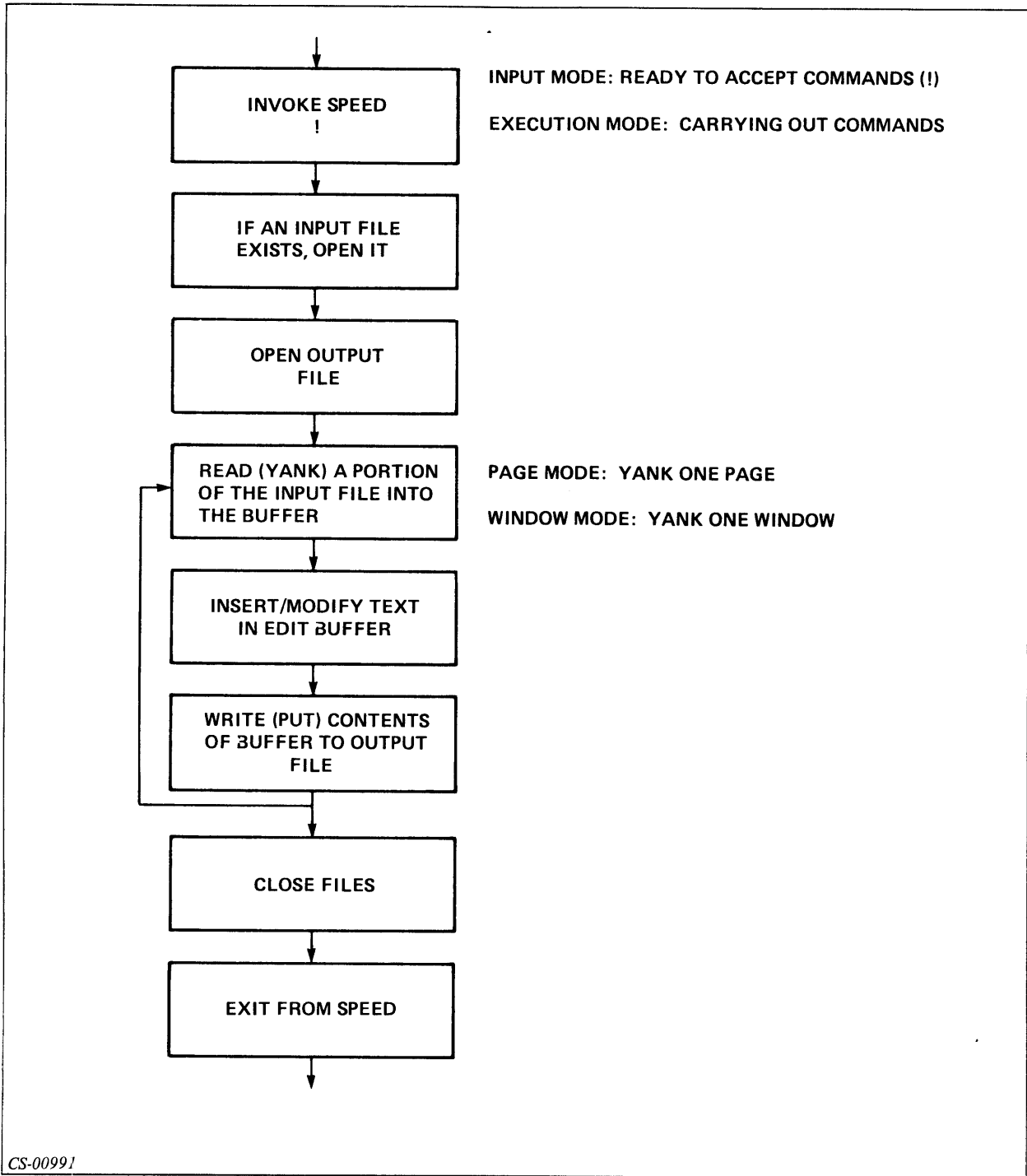
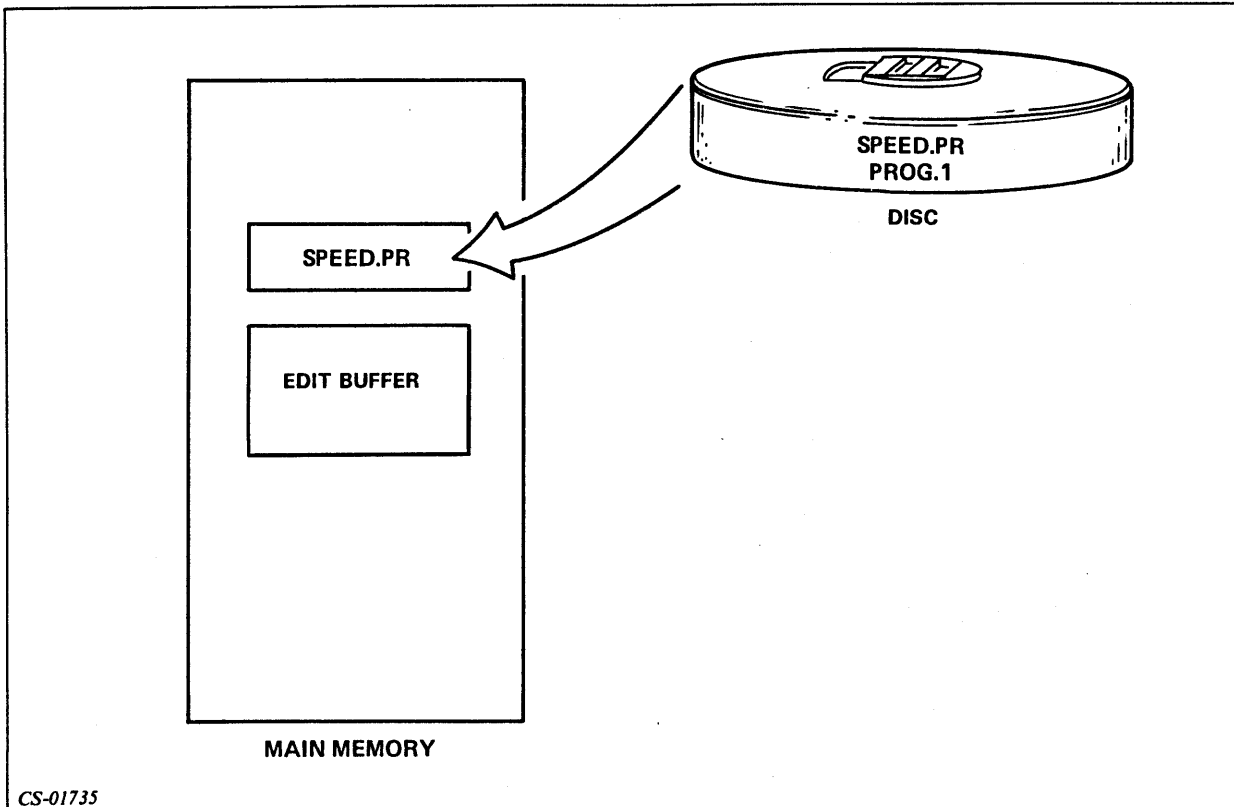


Figure 10.2 Editing Steps

More than one of the steps shown in Figure 10.2 may be performed with a single command.

Step 1: When you issue the command to execute SPEED, two things happen:

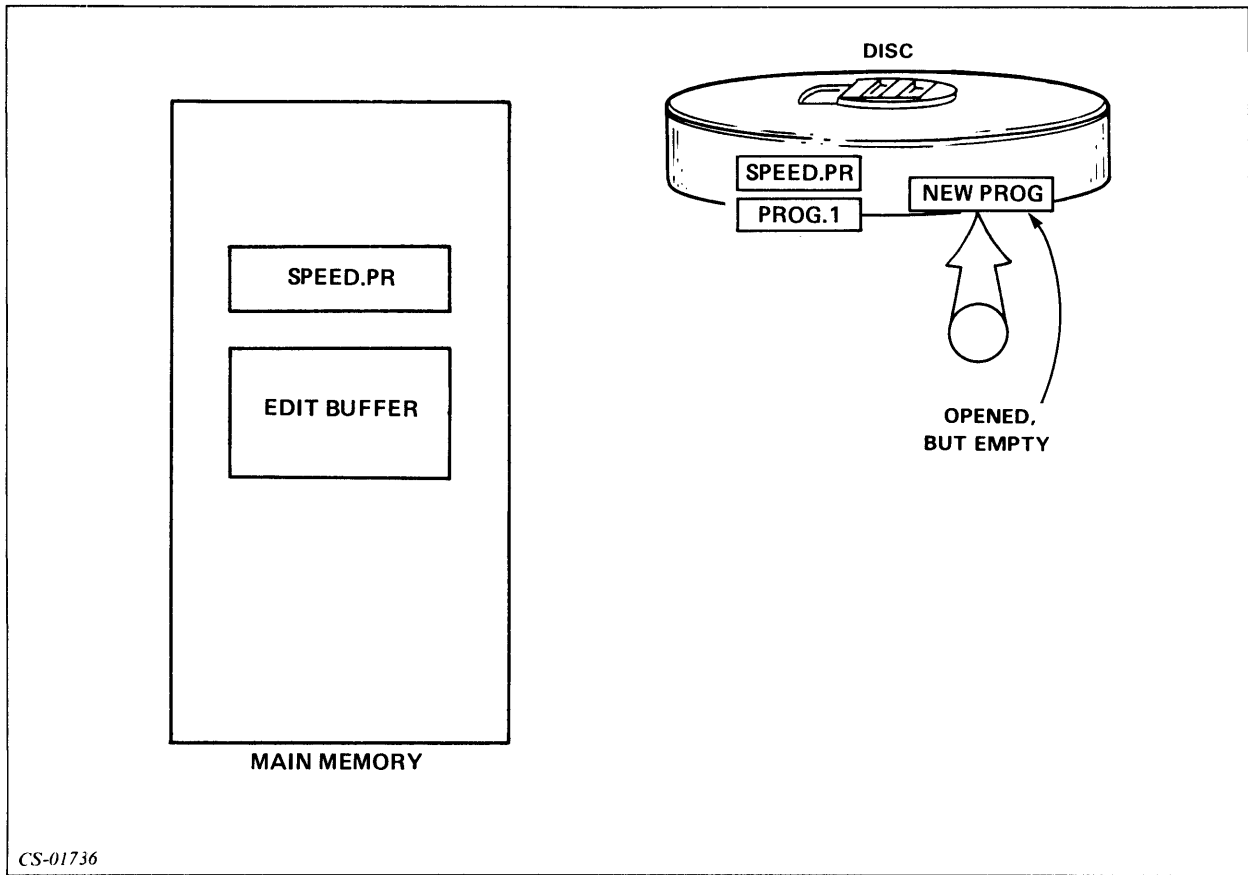
- The program is called in from the disc.
- An edit buffer is created.



CS-01735

Figure 10.3 SPEED is Invoked

Steps 2 and 3: Open files on the disc. Files remain unchanged.

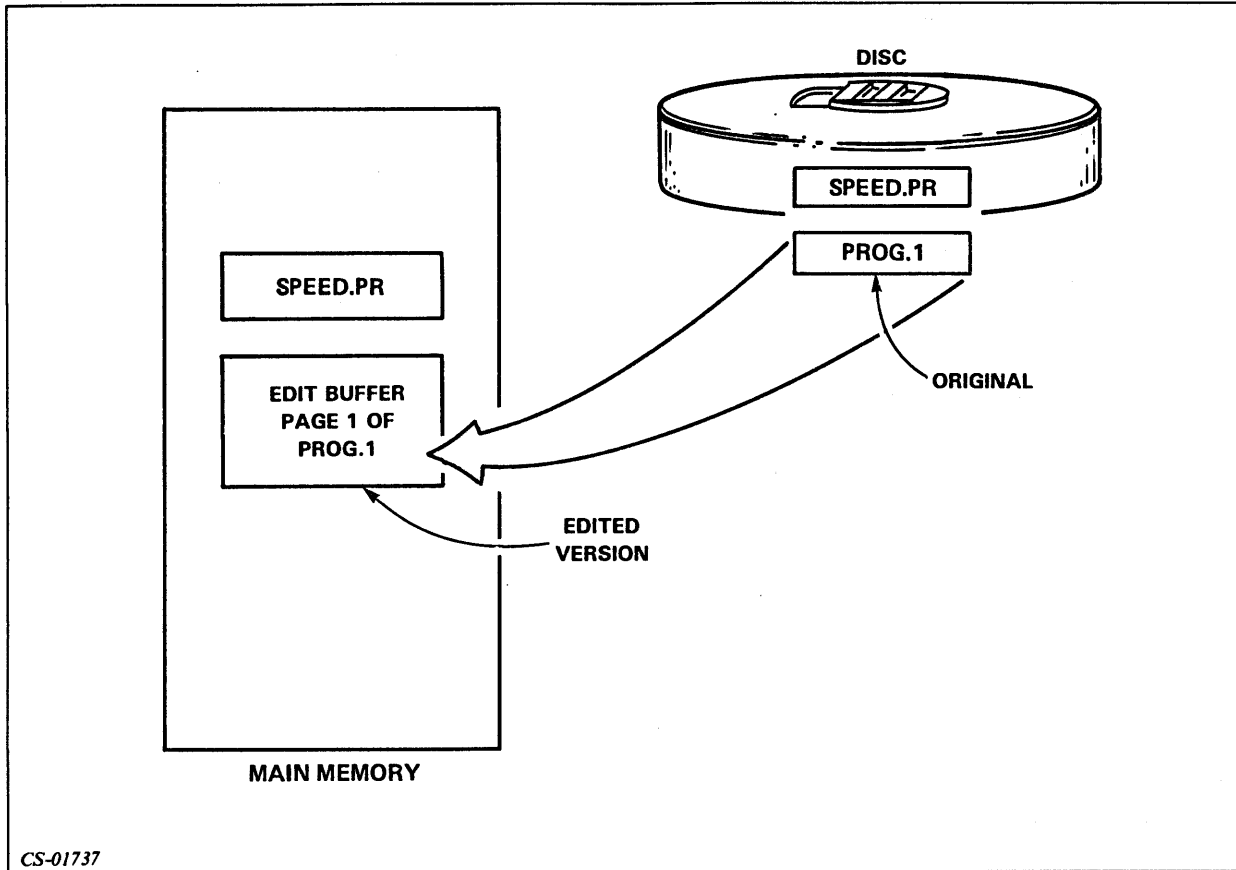


CS-01736

Figure 10.4 Open Files for Output and Input

Step 4: Data is moved into the edit buffer when a READ command is issued.

Step 5: Add, change, or delete data in the edit buffer.

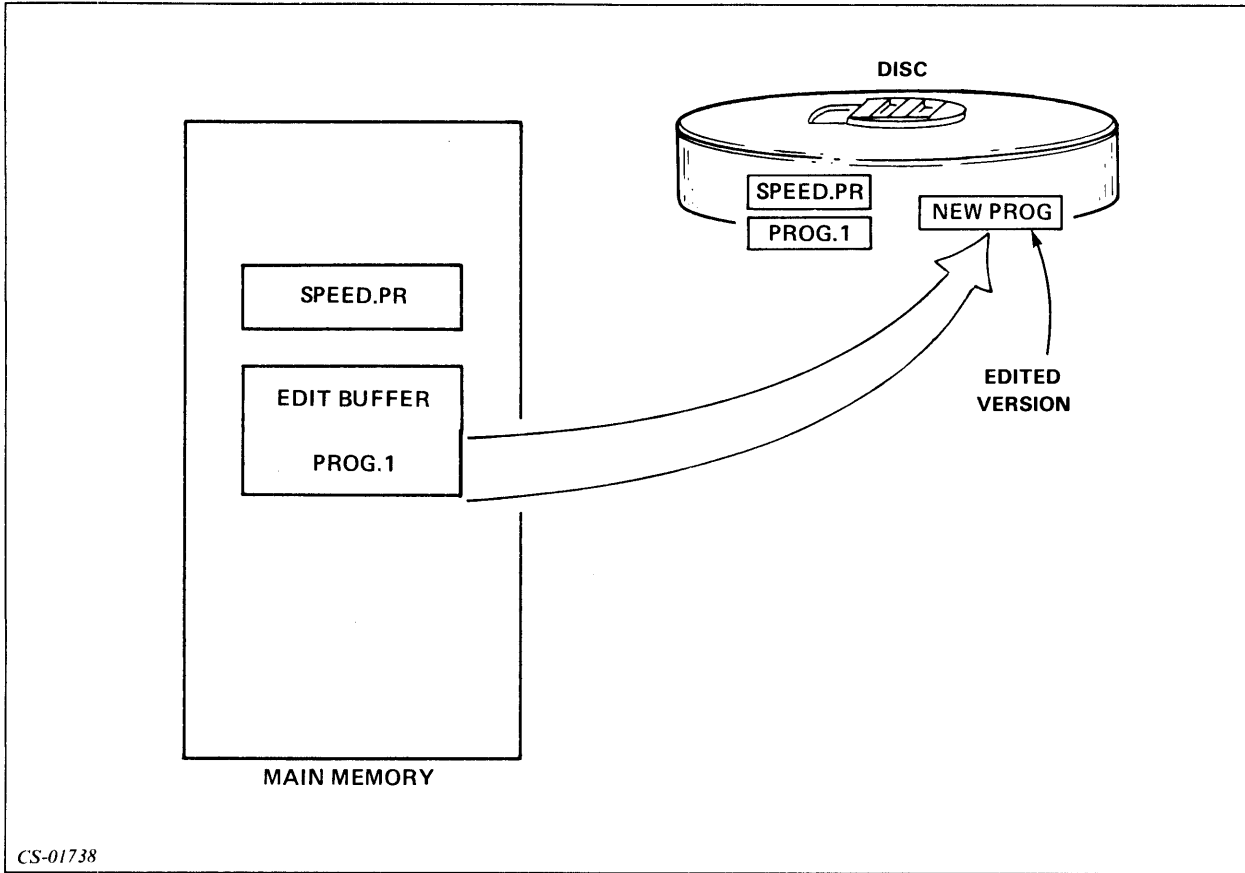


CS-01737

Figure 10.5 Read a Page of PROG.1 for Editing



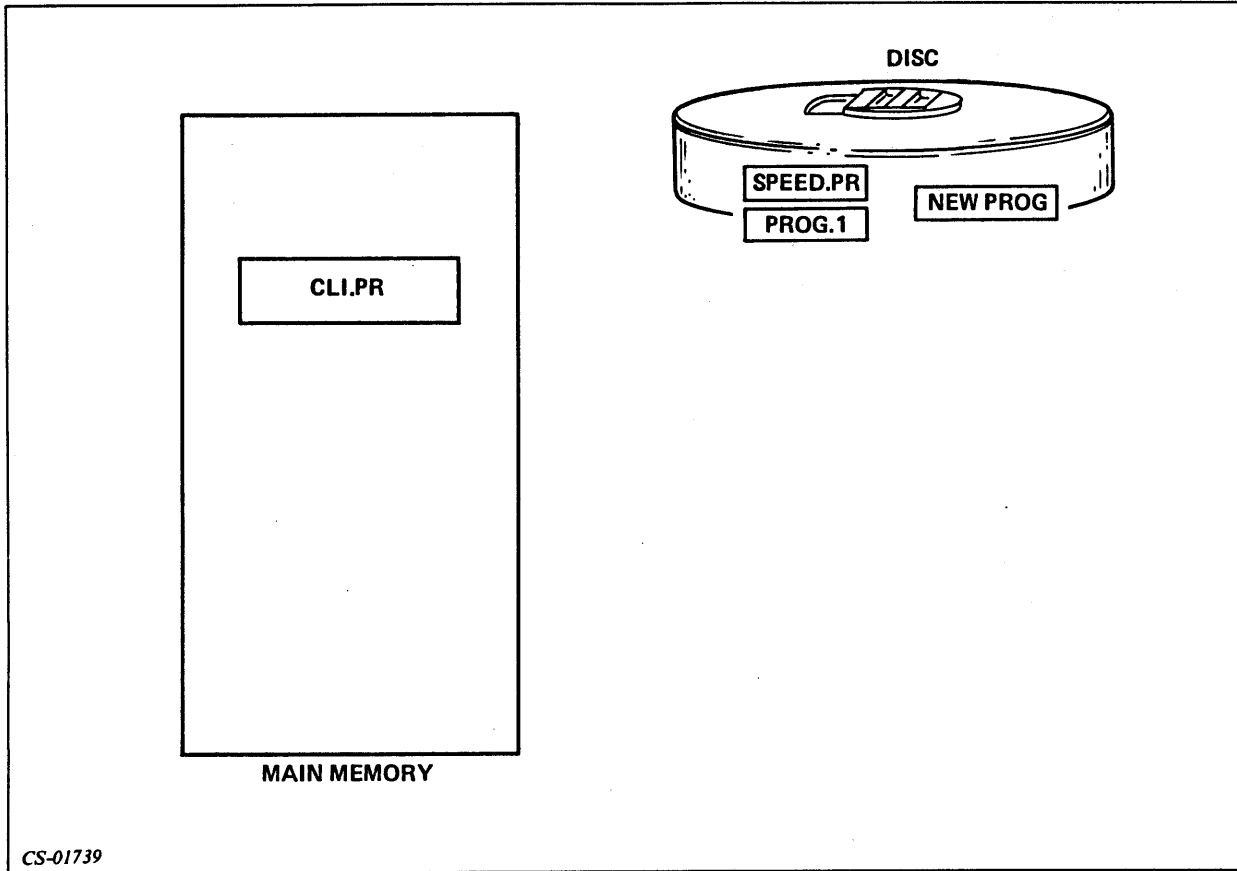
Step 6: WRITE commands move data to output file.  
Step 7: Close the files.



CS-01738

Figure 10.6 WRITE the Edited Version Out to Disc

Step 8: Return to CLI.



CS-01739

Figure 10.7 Terminate SPEED; Reinvoke CLI

**Console Control Procedures**

CTRL-U deletes an entire SPEED command line.

**Example 1**

\$\$

I THIS IS AN EXAMPLE OF CTRL-U  
(CTRL-U)

**Example 2**

\$\$

I HELLO, THIS IS A TEST  
(CTRL-U)

CTRL-C, CTRL-A deletes a multiple-line SPEED command.

**Example 1**

\$\$

I THIS IS AN EXAMPLE OF THE  
CTRL-C, CTRL-A COMBINATION  
(CTRL-C, CTRL-A)

**Example 2**

**\$\$**

**I THIS IS AN EXAMPLE OF THE  
CTRL-C, CTRL-A COMBINATION  
THIS IS AN EXAMPLE  
OF WHAT IS REQUIRED  
TO DELETE A SEGMENT  
OF TEXT, WHICH SPANS SEVEN LINES  
THE TEXT WILL NOW BE DELETED  
(CTRL-C,CTRL-A)**

Now do Exercise 10-1 on the next page.

## Exercise 10-1

### Part 1

**Directions:** Match the letter of the correct definition with the term on the right.

- |                   |                                  |
|-------------------|----------------------------------|
| 1. ____ Character | a. A group of pages.             |
| 2. ____ String    | b. Ends with NEW LINE.           |
| 3. ____ Line      | c. Occupies one position.        |
| 4. ____ Page      | d. Sequence of ASCII characters. |
| 5. ____ Window    | e. Ends with CTRL-L (NEW LINE).  |

**Directions:** Mark the following statements true or false.

6. \_\_\_\_ A string is a sequence of characters up to and including a carriage return.
7. \_\_\_\_ A page is a sequence of characters ending in a CTRL-L.
8. \_\_\_\_ You can only edit text if it resides in the edit buffer.
9. \_\_\_\_ A SPEED command line is terminated by entering CTRL-D.

### Part 2

**Directions:** Given the following steps in an edit cycle, list them in their usual order:

- a. Edit text in the edit buffer.
- b. Open files for input and output.
- c. Close the input and output files.
- d. Invoke SPEED.
- e. Exit from SPEED.
- f. Write text from the edit buffer to the output file.
- g. Read text from the input file to the edit buffer.

1. \_\_\_\_ 2. \_\_\_\_ 3. \_\_\_\_ 4. \_\_\_\_ 5. \_\_\_\_ 6. \_\_\_\_ 7. \_\_\_\_

### Part 3

**Directions:** Choose the letter of the response that best completes the statement.

1. To delete only the last character typed, press:
  - a. RUBOUT or DELETE
  - b. CTRL-U
  - c. CTRL-D
  - d. ESCAPE
  - e. CTRL-C, CTRL-A

2. To delete one command line (that does not contain any new-line characters), press:
  - a. RUBOUT or DELETE
  - b. CTRL-U
  - c. CTRL-D
  - d. ESCAPE
3. To terminate a command line and allow SPEED to execute it, you press:
  - a. RUBOUT or DELETE
  - b. CTRL-U
  - c. CTRL-D
  - d. ESCAPE
  - e. Repeated rubouts or deletes
4. SPEED command termination (CTRL-D) is echoed as:
  - a. \$\$
  - b. \$
  - c. !
  - d. ?

Now check your answers on the next page.

## **Exercise 10-1**

### **Answers**

#### **Part 1**

1. c.
2. d.
3. b.
4. e.
5. a.
6. F
7. T
8. T
9. T

#### **Part 2**

1. d.
2. b.
3. g.
4. a.
5. f.
6. c.
7. e.

#### **Part 3**

1. a.
2. b.
3. c.
4. a.

If you answered all the questions correctly, continue with Module 10 by restarting the Module 10 audiotape. Otherwise, review the material and do this exercise again before you continue.

## Manipulating Files with SPEED

There are two ways that you can enter SPEED.

### Example 1

Specify no file.

```
XEQ,SPEED
SPEED REV 3.00
!
```

(! is the SPEED prompt.)

### Example 2

Specify file to be edited.

```
XEQ,SPEED,MYFILE
```

There are two possible responses.

If the file exists:

```
SPEED REV 3.00
!
```

If the file does not exist:

```
SPEED REV 3.00
CREATE NEW FILE? (Respond Y or N)
!
```

## File Management

### Opening Input Files

The FR command opens an existing file for input.

#### Example 1

```
FRFILEA$$
```

Opens FILEA for input.

#### Example 2

```
FRFILEB$$
```

```
ERROR: FILE DOES NOT EXIST
```

```
FRFILEB
```

### Opening Output Files

The FW command opens and creates a file for output.

**Example 1**  
**FWFILEA\$\$**

**Example 2**  
**FWFILEB\$\$**  
**ERROR: FILE ALREADY EXISTS**  
**FWFILEB**

### Opening Input and Output Files

The FO command opens a file for updating and reads in first page.

**Example 1**  
**FOFILEA\$\$**

**Example 2**  
**FOFILEB\$\$**  
**ERROR: FILE DOES NOT EXIST**  
**FOFILEB**

Command	Type	Example	Error Message	Features
FR	Input	FR FILE1	NO FILE	Reads first page.
FW	Output	FW FILE1	FILE PRESENT	
FO	Input/Output	FO FILE1	NO FILE	

Table 10.B File Opening Summary

### Closing Files

The FC command closes all files.

**Example**  
**FC\$\$**

The FU command updates and closes files, clears the buffer area, and copies the input file to output file.

**Example**  
**FU\$\$**

The FB command closes and backs up files. This command performs the same function as the command FU and creates a back-up file.

**Example**  
**FB\$\$**

Before SPEED editing:  
 FILE1          Input to SPEED

After SPEED editing:  
 FILE1          Updated file  
 FILE1.BU      Back-up file



Command	Arguments	Effect on Buffers	Effect on Input File	Special Features
FC	None	None	None	None
FU	None	Copies to output file.	Copies to output file.	None
FB	None	Copies to output file.	Copies to output file.	Creates back-up file.

Table 10.C File Closing Summary

## Read Commands

The Y command reads one page from input file into the buffer and destroys the current contents of the buffer.

### Example

Y\$\$

The A command appends a page to the current buffer and does not affect the current contents of the buffer.

### Example

A\$\$

## Write Commands

The P, or put, command writes the current buffer to output file, appends a form-feed character at the end of output data, and does not destroy contents of the buffer.

### Example

P\$\$

The  $nP$  command writes  $n$  lines of the buffer to the output file, appends a form-feed character at the end of output data, and does not destroy contents of the buffer.

### Example

2P\$\$

BEFORE

Buffer contains:

THIS IS THE FILE  
THIS IS THE FILE  
TO BE EDITED  
IT CONTAINS LOTS AND LOTS  
OF IMPORTANT  
INFORMATION.

Output:

AFTER

Buffer contains:

THIS IS THE FILE  
TO BE EDITED  
IT CONTAINS LOTS AND LOTS  
OF IMPORTANT  
INFORMATION.

Output:

THIS IS THE FILE  
TO BE EDITED (Form-feed)

The PW command writes the current buffer to output files, does not append a form-feed character at the end of data, and does not destroy the contents of the buffer.

The E, or eject, command writes buffer and remainder of input file to output file.

The R, or roll, command writes the current buffer to the output file and reads next page of input file.

**Example**  
**RSS**

**BEFORE**  
Input file contains:

THIS IS A FILE THAT CONTAINS  
MORE THAN ONE PAGE OF INFORMATION  
YOU ARE NOW LOOKING AT PAGE 1. (Form-feed)

THIS IS THE BEGINNING OF PAGE TWO  
PAGE TWO HAS ONLY TWO LINES. (Form-feed)

PAGE THREE IS THE LAST PAGE OF  
THIS FILE. (Form-feed)

Buffer contains:

THIS IS A FILE THAT CONTAINS  
MORE THAN ONE PAGE OF INFORMATION  
YOU ARE NOW LOOKING AT PAGE 1  
THIS IS THE END OF PAGE 1. (Form-feed)

Output file is empty.

**AFTER**  
Input file contains:

THIS IS A FILE THAT CONTAINS  
MORE THAN ONE PAGE OF INFORMATION  
YOU ARE NOW LOOKING AT PAGE 1  
THIS IS THE END OF PAGE 1. (Form-feed)

THIS IS THE BEGINNING OF PAGE TWO  
PAGE TWO HAS ONLY TWO LINES. (Form-feed)

PAGE THREE IS THE LAST PAGE OF  
THIS FILE. (Form-feed)

Buffer contains:

THIS IS THE BEGINNING OF PAGE TWO  
PAGE TWO HAS ONLY TWO LINES. (Form-feed)

Output file contains:

THIS IS A FILE THAT CONTAINS  
MORE THAN ONE PAGE OF INFORMATION  
YOU ARE NOW LOOKING AT PAGE 1  
THIS IS THE END OF PAGE 1. (Form-feed)

## Status Command

The F? command lists all files currently open.

### Example

XEQ SPEED

SPEED REV 3.00

F?

GLOBAL:

INPUT FILE-NONE

OUTPUT FILE-NONE

LOCAL:

INPUT FILE-NONE

OUTPUT FILE-NONE

FRFILE1

FWFILE2

F?

GLOBAL:

INPUT FILE-FILE1

OUTPUT FILE-FILE2

LOCAL:

INPUT FILE-NONE

OUTPUT FILE-NONE

FC

F?

GLOBAL:

INPUT FILE-NONE

OUTPUT FILE-NONE

LOCAL:

INPUT FILE-NONE

OUTPUT FILE-NONE

## Terminating the Editing Session

The H command allows you to exit from SPEED.

### Example 1

If the buffer is empty:

!HSS

)

**Example 2**

If the buffer is not empty:

```
!H$$  
CONFIRM?Y  
)
```

or

```
!H$$  
CONFIRM?N  
!
```

**Summary of SPEED Commands**

**Invoke and exit**

**XEQ SPEED**

Invoke SPEED.

**XEQ SPEED,FILE1**

Invoke and open file.

**H\$\$**

Exit SPEED.

**Open and close files**

**FR** File read

**FW** File write

**FC** Files close

**FO** File open

**FU** File update

**FB** File backup

**F?** File status

**Read and write files**

**Y** Yank

**P** Put

**A** Append

**E** Eject

**R** Put and yank

Now do Exercise 10-2 on the next page.

## Exercise 10-2

**Directions:** Match the commands in the right column with the correct result in the left column. (Note that there are two sections to this matching exercise. Each section is exclusive of the other section.)

- |  |       |
|--|-------|
| 1. ____ Open input.  | a. FU |
| 2. ____ Open output.   | b. FR |
| 3. ____ Open input and output.   | c. FC |
| 4. ____ File write and back up.  | d. FW |
| 5. ____ Close files.   | e. FO |
| 6. ____ Close and write.   | f. FB |
|  |       |
| 7. _____ Write then read.  | a. H  |
| 8. _____ Read from input.  | b. P  |
| 9. _____ Halt and return to CLI.   | c. A  |
| 10. _____ Write buffer to output.<br>(Do not affect the rest of the file.) | d. Y  |
| 11. _____ Append.  | e. E  |
| 12. _____ Write rest of file to output.                                    | f. R  |

Check your answers on the next page.

## **Exercise 10-2**

### **Answers**

1. b.
2. d.
3. e.
4. f.
5. c.
6. a.
7. f.
8. d.
9. a.
10. b.
11. c.
12. e.

**If you answered all the questions correctly, continue with Module 10 by restarting the Module 10 audiotope. Otherwise, review the material and do this exercise again before you continue.**

## The Character Pointer (CP)

### Example 1

CP on DASHER® CRT

#T\$\$

\*THIS IS THE SAMPLE TEXT.  
LINE TWO\*OF SAMPLE.  
THE THIRD LINE OF THE SAMPLE.  
FOURTH AND LAST LINE.\*!

### Example 2

CP on DASHER® hard-copy terminal

(^)THIS IS A SAMPLE TEXT  
LINE TWO OF SAMPLE  
THE THIRD LINE OF THE SAMPLE  
FOURTH AND LAST LINE

## The L Command

*n*L\$\$      Move the CP *n* lines from current position.

Current buffer:

THIS IS A SAMPLE TEXT  
LINE TWO OF SAMPLE  
THE THIRD LINE OF THE SAMPLE  
FOURTH AND LAST LINE

(\* is the Character Pointer.)

### Example 1

L\$\$

Move CP to beginning of the current line.

T\$\$

\*THIS IS THE SAMPLE TEXT.} ———— Display the current line.

### Example 2

3L\$\$

Move CP three lines forward.

T\$\$

\*FOURTH AND LAST LINE.} ———— Display the current line.

### Example 3

-2L\$\$

Move CP two lines backwards.

T\$\$

\*LINE TWO OF SAMPLE.} ———— Display the current line.

## The J Command

**nJ\$\$** Move the CP to the *n*th line in the edit buffer.

Current buffer:

THIS IS THE SAMPLE TEXT  
 LINE TWO OF SAMPLE  
 THE THIRD LINE OF THE SAMPLE  
 FOURTH AND LAST LINE

(\* is the Character Pointer.)

### Example 1

**2J\$\$** Move the CP to line two.  
**T\$\$**  
**\*LINE TWO OF SAMPLE.}** ————— Display the current line.

### Example 2

**J\$\$** Move the CP to the beginning of the buffer.  
**T\$\$**  
**\*THIS IS THE SAMPLE TEXT.}** ————— Display the current line.

### Example 3

**ZJ\$\$** Move the CP to the end of the buffer.  
**T\$\$**  
**\*| }** ————— Display the current line.

## The M Command

**nM\$\$** Move the CP *n* character positions from the current location.

Current buffer:

THIS IS A SAMPLE TEXT  
 LINE TWO OF SAMPLE  
 THE THIRD LINE OF THE SAMPLE  
 FOURTH AND LAST LINE

### Example 1

**5M\$\$** Move the CP five characters to the right.  
**T\$\$**  
**LINE \*TWO OF SAMPLE.}** ————— Display the current line.

### Example 2

**-3M\$\$** Move the CP three characters to the left.  
**T\$\$**  
**LINE \*TWO OF SAMPLE.}** ————— Display the current line.



## The T Command

Command	Result
T\$\$	Display the line with the character pointer.
<i>n</i> T\$\$	Display <i>n</i> lines, starting from the CP line.
#T\$\$	Display all lines.

Table 10.D The T Command

Current buffer:

THIS IS THE SAMPLE TEXT  
 LINE TWO OF SAMPLE.  
 THE THIRD LINE OF THE SAMPLE.  
 FOURTH AND LAST LINE.

### Example 1

T\$\$  
 \*LINE TWO OF SAMPLE.) } ————— Display the current line.

### Example 2

2T\$\$  
 LINE TWO OF SAMPLE.  
 THE THIRD LINE OF THE SAMPLE.) } ————— Display two lines starting from current line.

### Example 3

#T\$\$  
 THIS IS THE SAMPLE TEXT.  
 LINE TWO OF SAMPLE.  
 THE THIRD LINE OF THE SAMPLE.  
 FOURTH AND LAST LINE.) } ————— Display entire buffer.

## The I Command

**ITEXT-STRING\$\$**      Insert text-string at current location of the CP.

Current buffer:

THIS IS THE SAMPLE TEXT.  
LINE TWO OF SAMPLE.  
THE THIRD LINE OF THE SAMPLE.  
FOURTH AND LAST LINE.

### Example 1

**IT**THIS IS THE SAMPLE TEXT.      Insert the text.

LINE TWO OF SAMPLE.

THE THIRD LINE OF SAMPLE.

FOURTH AND LAST LINE.\$\$

**T\$\$**

FOURTH AND LAST LINE.\*I}

————— Display the current line and the CP.

### Example 2

**I**

Insert a character at the current location of CP.

**\$\$**

**#T\$\$**

THIS IS THE SAMPLE TEXT.

LINE TWO OF SAMPLE.

THE THIRD LINE OF THE SAMPLE.

FOURTH AND LAST LINE.

} ————— Display CP and line.

### Example 3

**3J\$\$**

Move the CP to line 3. Insert a line.

**IT**THIS FITS BETWEEN 3 AND 2

**\$\$**

**#T\$\$**

THIS IS THE SAMPLE TEXT.

LINE TWO OF SAMPLE.

THIS FITS BETWEEN 3 AND 2

THE THIRD LINE OF THE SAMPLE.

} ————— Display the whole buffer.

## The S Command

STEXT-STRING\$\$ Search for text-string.

Current buffer:

THIS IS THE SAMPLE TEXT.  
 LINE TWO OF SAMPLE.  
 THIS FITS BETWEEN 3 AND 2  
 THE THIRD LINE OF THE SAMPLE.  
 FOURTH AND LAST LINE.

### Example 1

STHIRD\$\$ Search for "THIRD".  
 T\$\$ }  
 THE THIRD\* LINE OF THE SAMPLE. } Display the current line.

### Example 2

STEXT\$\$ Search for "TEXT".  
 ERROR: UNSUCCESSFUL SEARCH }  
 STEXT\$\$ } Error message.

### Example 3

3JT\$\$ Move to line 3.  
 (^)THIS FITS BETWEEN 3 AND 2 Display the CP.  
 -3SSAMPLE TEXT\$\$ Search backwards three lines.  
 T\$\$ }  
 THIS IS THE SAMPLE TEXT(^). } Display the new CP.

### Example 4

\$\$  
 0STHIS\$\$ Search backward.  
 T\$\$ }  
 THIS(^) IS THE SAMPLE TEXT. } Display the CP.

### Example 5

T\$\$ }  
 THIS(^) IS A SAMPLE TEXT. } Display the CP.  
 1,100SFIT\$\$ Search for "FITS".  
 T\$\$ }  
 THIS FITS(^) BETWEEN 3 AND 2 } Display the new CP.

Command	Result
<i>n</i> STEXT-STRING\$\$	Search for text-string starting from CP and going <i>n</i> lines back toward the beginning of the buffer.
0TEXT-STRING\$\$	Search for text-string, starting from CP and going back to the beginning of the current line.
<i>n,z</i> STEXT-STRING\$\$	Search for text-string, starting from character position <i>n</i> + 1 and continuing to character position <i>z</i> .

Table 10.E The S Command

## The N Command

**NTEXT-STRING\$\$** Search for text-string throughout the entire input file.

Current buffer:

THIS IS THE SAMPLE TEXT.  
 LINE TWO OF SAMPLE.  
 THIS FITS BETWEEN 3 AND 2.  
 THE THIRD LINE OF THE SAMPLE.  
 FOURTH AND LAST LINE.

### Example 1

**J\$\$** Move CP to beginning of buffer.  
**NTHIRD LINE\$\$** Search for "THIRD LINE".  
**T\$\$**  
 THE THIRD LINE\* OF THE SAMPLE. } ——— Display current line.

### Example 2

**\$\$** Search for "TEXT".  
**NLINE TWO\$\$**  
 ERROR: NO OPEN FILE } ————— Error message.  
 NLINE TWO

## The C Command

**COLD-TEXT\$NEW-TEXT\$\$** Search for old-text.  
 Delete old-text.  
 Insert new-text.  
 Leave CP after new-text.

Current buffer:

THIS IS THE SAMPLE TEXT.  
 LINE TWO OF SAMPLE.  
 THIS FITS BETWEEN 3 AND 2.  
 THE THIRD LINE OF THE SAMPLE.  
 FOURTH AND LAST LINE.

### Example 1

**J\$\$** Move CP to beginning of buffer.  
**CSAMPLESPRACTICE\$\$** Change "SAMPLE" to "PRACTICE".  
**T\$\$**  
 THIS IS THE PRACTICE\* TEXT. } ——— Display current line.

### Example 2

**CTHIS IS\$LINE ONE OF \$\$** Change "THIS IS" to "LINE ONE OF".  
 ERROR: UNSUCCESSFUL SEARCH } ————— Error message.  
 CTHIS IS\$

**Example 3**

J\$\$ Move CP to line one.  
 CTHIS IS \$LINE ONE OF \$\$  
 T\$\$ }  
 LINE ONE OF\*THE PRACTICE TEXT } ——— Display the current line.

**Example 4**

CTEXT-STRING\$\$ Search for text-string.  
 Delete text-string.  
 Leave CP after text-string.

C3 AND 2\$\$  
 T\$\$  
 THIS FITS BETWEEN (^)

**The D Command**

*n*D\$\$ Delete *n* characters to the right of the CP.  
 -*n*D\$\$ Delete *n* characters to the left of the CP.

Current buffer:

LINE ONE OF THE PRACTICE TEXT.  
 LINE TWO OF SAMPLE.  
 THIS FITS BETWEEN  
 THE THIRD LINE OF THE SAMPLE.  
 FOURTH AND LAST LINE.

**Example 1**

4D\$\$ Delete four characters following CP.  
 \$\$  
 T\$\$ }  
 \*ONE OF THE PRACTICE TEXT. } ——— Display current line.

**Example 2**

STWO\$\$ Move the CP to "TWO" in line.  
 T\$\$  
 LINE TWO\* OF SAMPLE.  
 -3D\$\$ Delete three characters preceding CP.  
 T\$\$ }  
 LINE\* OF SAMPLE } ——— Display results.

## The K Command

- n*K\$\$** Delete *n* lines following CP.
- n*K\$\$** Delete *n* lines preceding CP.

Current buffer:

LINE ONE OF THE PRACTICE TEXT.  
LINE TWO OF SAMPLE.  
THIS FITS BETWEEN  
THE THIRD LINE OF THE SAMPLE.  
FOURTH AND LAST LINE.

### Example 1

<b>2J\$\$</b>	Move CP to line 2.
<b>1K\$\$</b>	Delete (kill) one line following CP.
<b>#T\$\$</b>	
ONE OF THE PRACTICE TEXT.	} ——— Display entire buffer.
THIS FITS BETWEEN	
THE THIRD LINE OF THE SAMPLE.	
FOURTH AND LAST LINE.	

### Example 2

<b>2J\$\$</b>	Move CP to line 2.
<b>-1K\$\$</b>	Delete one line preceding CP.
<b>#T\$\$</b>	
THIS FITS BETWEEN	} ——— Display buffer.
THE THIRD LINE OF THE SAMPLE.	
FOURTH AND LAST LINE.	

### Example 3

<b>STHIRD\$\$</b>	Move CP to the middle of a line.
<b>T\$\$</b>	
THE THIRD* LINE OF THE SAMPLE	
<b>K\$\$</b> —————	Delete the characters in the front of the line.
<b>T\$\$</b>	
* LINE OF THE SAMPLE.	

## Summary of Text Commands

### CP Movers

L	Line move
J	Jump to a line
M	Move character positions

### Searchers

S	Search
C	Change
N	Nonstop search

### Insert

I	Insert
C	Change

### Delete

C	Change
D	Delete characters
K	Kill lines

Now do Exercise 10-3 on the next page.





## Exercise 10-3

**Directions:** Match the commands on the left with the functional descriptions on the right.

1. \_\_\_\_\_ C      a. Delete lines of text.
2. \_\_\_\_\_ J      b. Search for string, delete it, insert another string.
3. \_\_\_\_\_ D      c. Insert string at CP location.
4. \_\_\_\_\_ L      d. Delete characters.
5. \_\_\_\_\_ S      e. Search for string in buffer, put, yank (if there are open files).
6. \_\_\_\_\_ K      f. Search for string in buffer.
7. \_\_\_\_\_ T      g. Move CP character positions.
8. \_\_\_\_\_ M      h. Move CP to beginning of specified line, relative to start of buffer.
9. \_\_\_\_\_ N      i. Move CP to beginning of a line, relative to current line.
10. \_\_\_\_\_ #T      j. Display contents of entire buffer.
11. \_\_\_\_\_ I      k. Display contents of current line with CP.

Check your answers on the next page.

## **Exercise 10-3**

### **Answers**

1. b.

2. h.

3. d.

4. i.

5. f.

6. a.

7. k.

8. g.

9. e.

10. j.

11. c.

If you answered all the questions correctly, continue to Lab Activity 10-1 and the Module 10 Test. Otherwise, review the material and do the exercise again before continuing.

## Lab Activity 10-1

1. Invoke SPEED and create a file called SPEED.LAB.
2. Display the status of your files.
3. Input the following text:  

```
SPEED IS A TEXT EDITOR THAT YOU CAN  
USE WITH WITH AOS OR AOS/VS.  
YOU CAN USE IT TO CREATE AND MODIFY FILES.
```
4. Write this data to the output file, appending an end of page character.
5. Insert this text:  

```
MODULE 10 OF THE AOS, AOS/VS USERS COURSE IS  
DEVOTED TO TEACHING YOU TO USE THE SPEED  
EDITOR. THIS LAB ACTIVITY WILL INDICATE  
YOUR ABILITY TO USE THE EDITOR.
```
6. Repeat Step 4.
7. Insert this text:  

```
SUCCESSFUL COMPLETION OF THIS EXERCISE  
INDICATES YOU HAVE MASTERED THE BASICS OF SPEED.
```
8. Write this text to the output file.
9. Close the files.
10. Exit from SPEED and return to the CLI.
11. Invoke SPEED once again. This time do not specify a file to edit.
12. Check filestatus.
13. Open SPEED.LAB for both input and output.
14. Display the contents of the buffer.
15. Change "CREATE" on the first page to "BUILD."
16. Insert the following line after the second line:  

```
YOU CAN RUN IT ON A DATA GENERAL COMPUTER.
```
17. Delete the fourth line.
18. Display the buffer.

19. Write out the buffer to the output file.
20. Read forward to the third page.
21. Display the buffer to verify the success of the previous step.
22. Return to the CLI.

Check your answers on the next page.

## Lab Activity 10-1

### Answers

1. X SPEED SPEED.LAB
2. F?
3. ISPEED IS A TEXT EDITOR THAT YOU CAN USE WITH AOS OR AOS/V.S. YOU CAN USE IT TO CREATE AND MODIFY FILES.  
\$\$
4. P
5. IMODULE 10 OF THE AOS, AOS/V.S USERS COURSE IS DEVOTED TO TEACHING YOU TO USE THE SPEED EDITOR. THIS LAB ACTIVITY WILL INDICATE YOUR ABILITY TO USE THE EDITOR.  
\$\$
6. P
7. ISUCCESSFUL COMPLETION OF THIS EXERCISE INDICATES YOU HAVE MASTERED THE BASICS OF SPEED.  
\$\$
8. P
9. FC
10. H\$\$
11. X SPEED
12. F?
13. FO SPEED.LAB
14. #T
15. CCREATES\$BUILD\$\$
16. 3J  
IYOU CAN RUN IT ON A DATA GENERAL COMPUTER.
17. 4J  
1K
18. #T
19. P
20. PY OR R
21. #T
22. FU  
H

If you completed this Lab Activity successfully, continue to the Module 10 Test. If you had any difficulty, review the material and try again before continuing.

## Module 10 Test

**Directions:** List in order the eight steps involved in editing with SPEED.

1. \_\_\_\_\_
2. \_\_\_\_\_
3. \_\_\_\_\_
4. \_\_\_\_\_
5. \_\_\_\_\_
6. \_\_\_\_\_
7. \_\_\_\_\_
8. \_\_\_\_\_

**Directions:** Fill in the blanks.

8. The FR command is used to \_\_\_\_\_ a file for \_\_\_\_\_.
9. The S command will \_\_\_\_\_ for a particular string of text.
10. To move the Character Pointer to the beginning of the buffer, use the \_\_\_\_\_ command.
11. A form of the \_\_\_\_\_ command is used to display text.
12. The I command is used to \_\_\_\_\_ text.
13. To open a file for both input and output, the \_\_\_\_\_ command can be used.
14. To close a file that was opened for both input and output, the \_\_\_\_\_ command must be used.
15. The two commands that can be used to read from the input file into the buffer are \_\_\_\_\_ and \_\_\_\_\_.
16. To write text from the buffer into a file, the \_\_\_\_\_ or \_\_\_\_\_ command can be used.
17. The command used to substitute one string of text for another is \_\_\_\_\_.

Now check your answers to the Module 10 Test in Appendix A. If you have answered all the questions correctly, continue to Module 11. Otherwise, go back and review the material in Module 10 and take the Module 10 Test again.

This concludes Module 10.

# Module 11

## Program Development Cycle

### Introduction

This module is concerned with the steps involved in the development of your application programs. First, you will look at the program development cycle in a general way, and then you will see how to develop Assembly Language, FORTRAN, COBOL, and BASIC programs.

### Module Objectives

Upon successful completion of this module, you should be able to:

1. Identify the steps involved in program development.
2. Identify the meaning of the following terms:
  - Translator
  - Compiler
  - Assembler
  - Interpreter
3. Write the CLI command line that assembles (or compiles) an Assembly Language (or FORTRAN or COBOL) program.
4. Write the CLI command line that creates an executable file from an object file.
5. Write the CLI command line that executes an application program.
6. If you are a BASIC programmer, show the commands to execute the BASIC interpreter, save an application program on disc, read it back into main memory, and execute it.

## Resources

To complete this module, you will need:

- Module 11 audiotape.
- Module 11 of your *Student Guide*.
- Audiotape playback unit.

## Module Outline

Module 11 discusses the following topics:

1. Program development cycle
  - a. Development steps
  - b. Translation phase
  - c. Loading phase
2. Development examples
  - a. BASIC programming
  - b. Assembly Language programming
  - c. FORTRAN programming
  - d. COBOL programming

Now start the Module 11 audiotape. As you listen, follow along in Module 11 of your *Student Guide*.



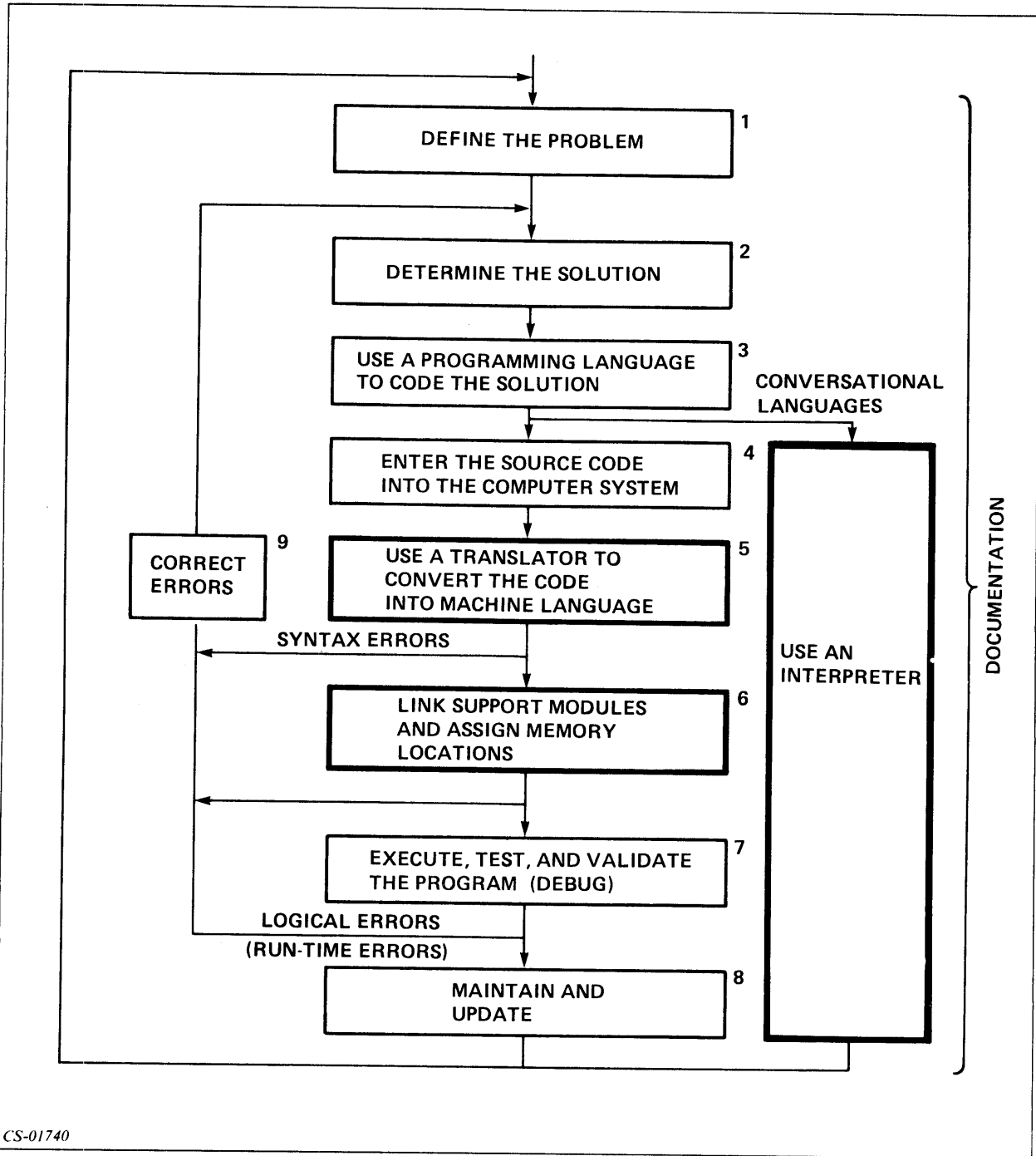
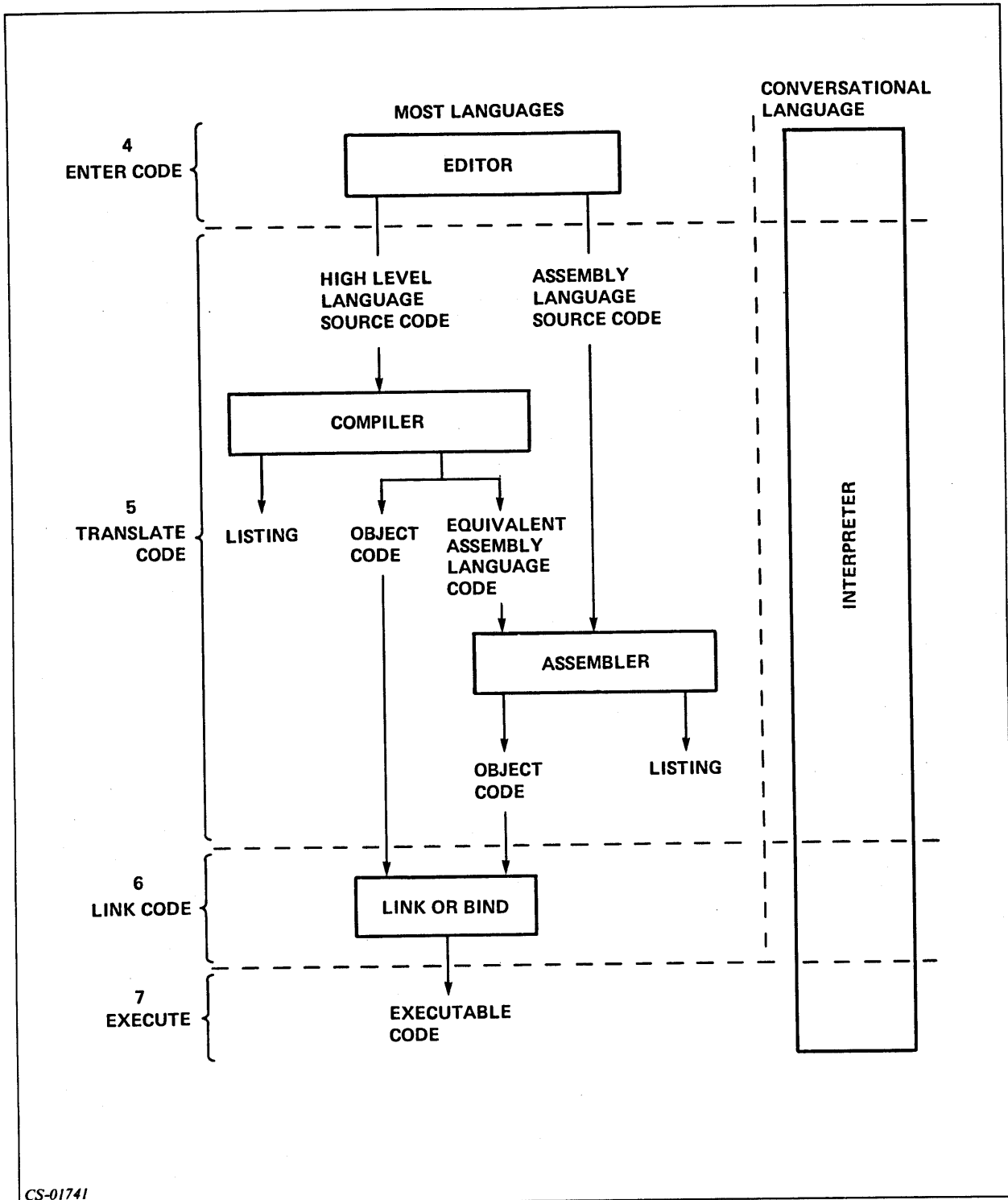


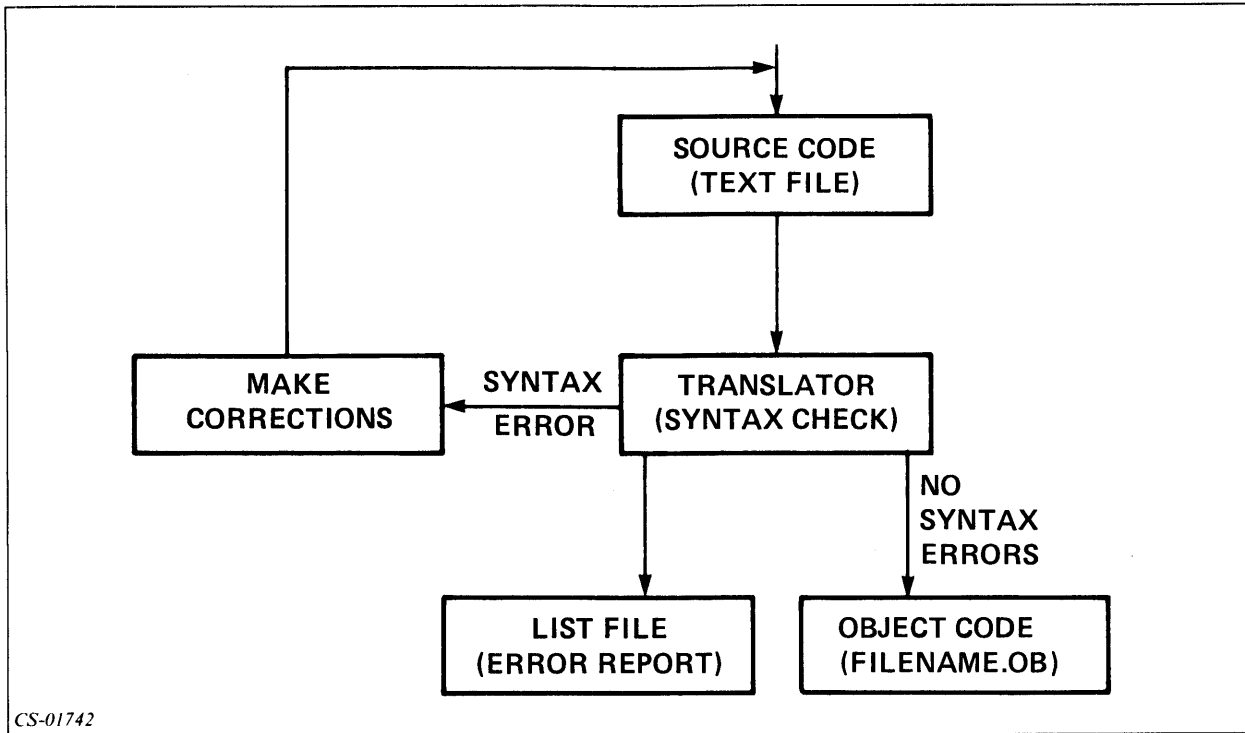
Figure 11.1 The Program Development Cycle

At Step 4 in Figure 11.1, use a text editor.  
 At Step 5, use a compiler or an assembler.  
 At Step 6, use the LINK or BIND utilities.



CS-01741

Figure 11.2 Types of Translators



CS-01742

Figure 11.3 Syntax Errors and Corrections

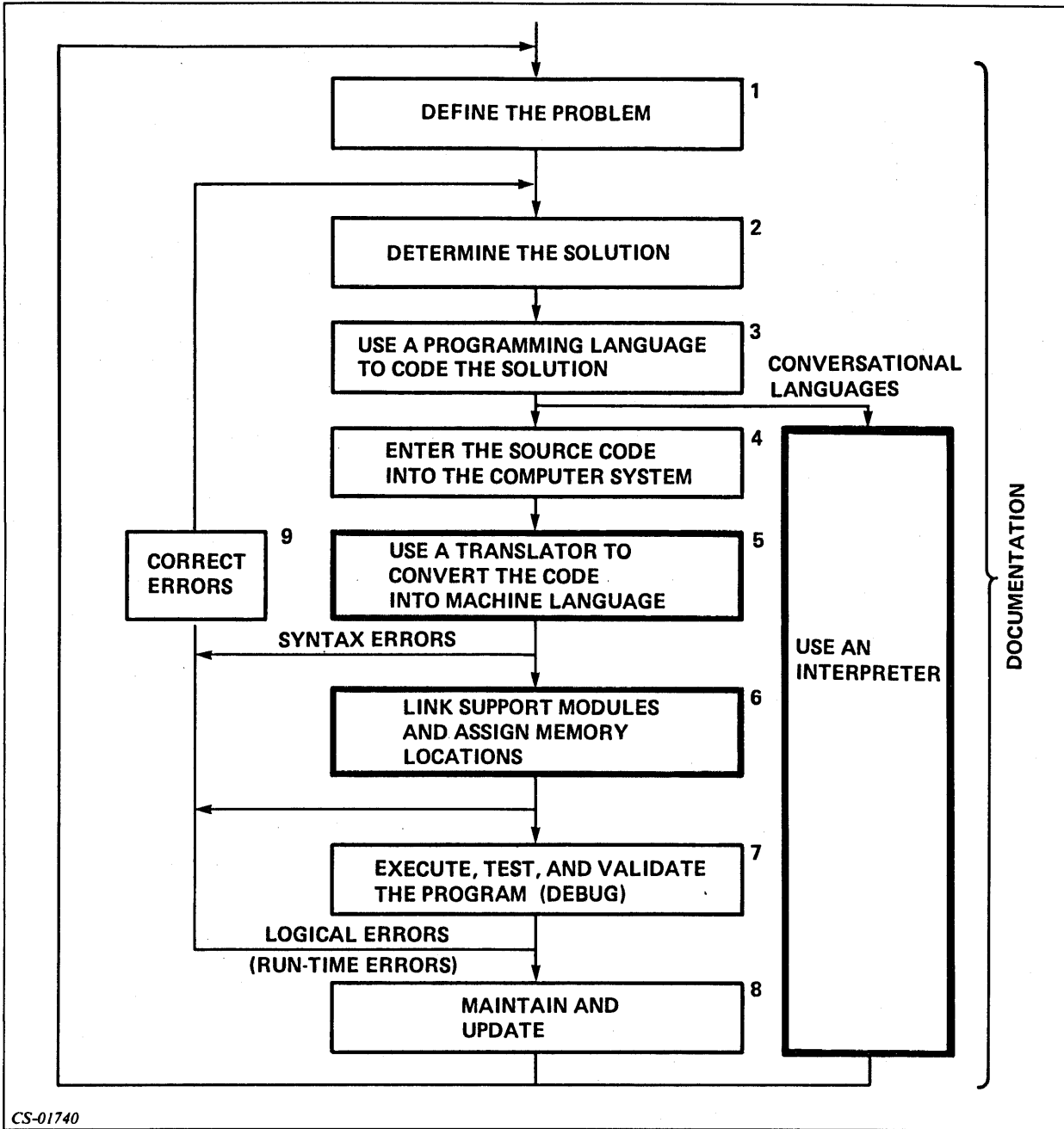


Figure 11.4 The Program Development Cycle

## Where Do I Go from Here?

1. Stop the Module 11 audiotape, and do Exercise 11-1 in this *Student Guide*. If you have difficulty with any of the questions, go back and review the material and do the exercise again before continuing.
2.
  - a. If you are a BASIC programmer, read the section titled "BASIC Programming."
  - b. If you are an Assembly language programmer (or if you want to be able to complete the Lab Activity in this module), read the section titled "Assembly Language Programming."
  - c. If you are a COBOL programmer, read the section titled "COBOL Programming."
  - d. If you are a FORTRAN programmer, read the section titled "FORTRAN Programming."
  - e. If you program in another language, such as PL/1 or ALGOL, read either "COBOL Programming" or "FORTRAN Programming."
3. Do Lab Activity 11-1.
4. Take the Module 11 Test.

Now do Exercise 11-1 on the next page.



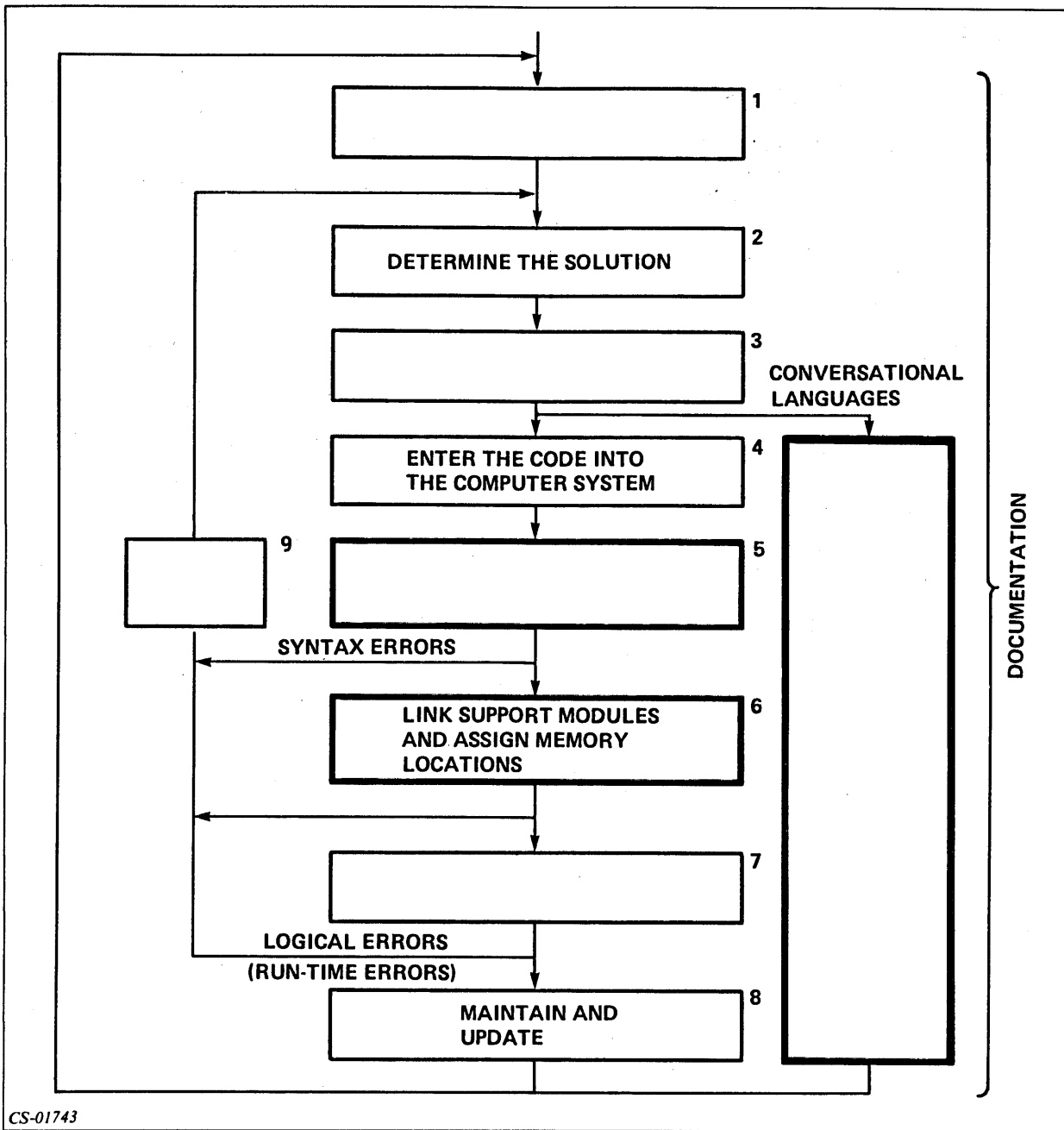
## Exercise 11-1

**Directions:** Mark the following statements true or false.

1. \_\_\_\_\_ A translator accepts source code as input and produces relocatable object code for output.
2. \_\_\_\_\_ The LINK or BIND combines several object modules into a single executable file.
3. \_\_\_\_\_ A library file is a collection of executable, or save, files.
4. \_\_\_\_\_ An interpreter, used with BASIC, combines several program development steps into one.

5. Fill in the boxes in Figure 11.5 on the next page, using the choices below.

- DEFINE THE PROBLEM
- CORRECT ERRORS
- USE A PROGRAMMING LANGUAGE TO CODE THE SOLUTION
- USE AN INTERPRETER
- EXECUTE, TEST, AND VALIDATE THE PROGRAM
- USE A TRANSLATOR TO CONVERT THE CODE INTO MACHINE LANGUAGE



CS-01743

Figure 11.5

Check your answers on the next page.



## **Exercise 11-1**

### **Answers**

1. True
2. True
3. False
4. True
5. See Figure 11.6.

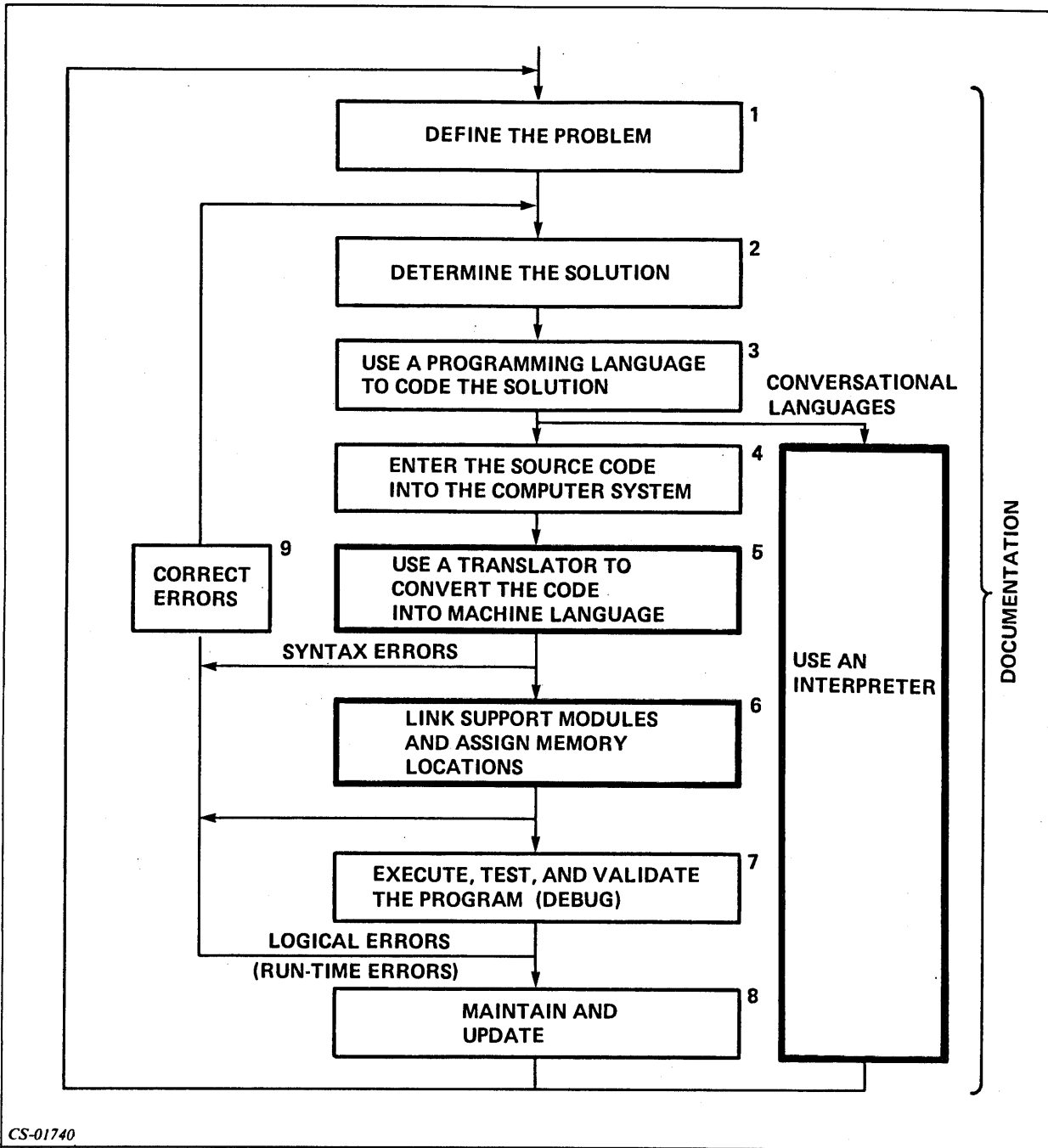


Figure 11.6

If you answered all the questions correctly, continue to the appropriate section of this module. Otherwise, review the material and do this exercise again before you continue.

## BASIC Programming

This section leads you through a sample session in BASIC. It assumes that you have some experience with the BASIC language.

To create a BASIC program:

1. Invoke the BASIC interpreter and get into BASIC by typing the CLI command:

**XEQ BASIC**

2. Write a series of BASIC program statements. BASIC has its own editor and an interactive compiler that rejects bad syntax as you type each statement.

3. Run the program with the BASIC command:

**RUN**

4. If the program runs correctly, you're done! Save the program on disc with the LIST command. Type **BYE** to get back to the CLI.
5. If your program contains runtime errors, fix it using erroneous output or BASIC runtime error messages. Go to Step 3.

### Writing BASIC Programs

You write a BASIC program as a series of statements, which you must begin with a number between 1 and 9999. Each statement includes a BASIC command.

At various points, you can examine the statements in your program with the LIST command, or tell BASIC to execute the statements with the RUN command. BASIC's error messages will help you correct errors; you can correct offending statements by typing their line numbers, then the new text. When you're satisfied with a program, save it on disc with the command LIST "FILENAME"; later, you can read it back into memory with the command ENTER "FILENAME". To print it on the line printer, type LIST "@LPT". To start work on another program, type NEW, then proceed. To sign off BASIC and return to the CLI, type BYE.

You can execute a BASIC program only from BASIC; you cannot do it from the CLI. The BASIC interpreter accepts both upper-case and lower-case characters, and translates lower-case letters to upper-case.

### Example: BASIC Program

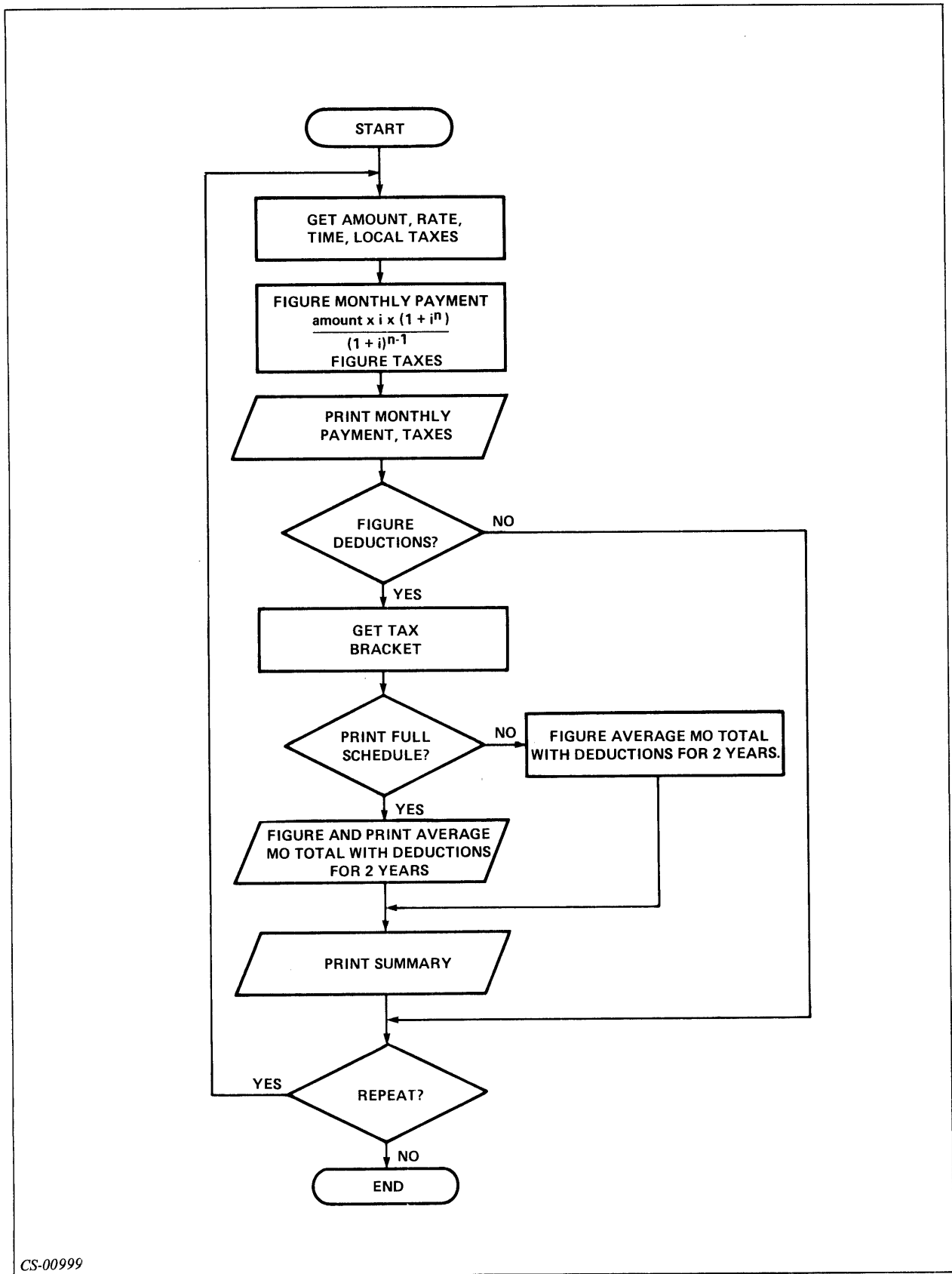
The BASIC example in this section is a simple program to calculate home mortgage payments, taxes, and deductions in a general way, and write its computations to the console. Figure 11.7 shows a flowchart of the program, and Figure 11.8 shows the program itself.

You can enter the program using the BASIC interpreter. The BASIC interpreter checks the syntax of each line as you type it in.

You can examine the lines you've typed by typing LIST. To list a portion of the lines, type LIST number comma number, where each number is a line number (e.g., LIST 10,100).

Periodically as you type the program in, and when you're done, type **LIST** "MORTGAGE.BA" to write the program to disc; you can also get a hard-copy listing by typing **LIST**, or **LIST "@LPT"** if you have a line printer.

Examine the flowchart (Figure 11.7) and program (Figure 11.8) before proceeding to the next section.



CS-00999

Figure 11.7 MORTGAGE Program Flowchart

```

0010 REM PROGRAM MORTGAGE.BA, COMPUTES MORTGAGE PAYMENTS, HAS TAX SUBROUTINE.
0020 PRINT "<12> I CALCULATE MORTGAGE PAYMENTS, INTEREST, AND TAXES."
0030 PRINT "TYPE AMOUNT OF PRINCIPAL, INTEREST RATE IN WHOLE NUMBERS,"
0040 PRINT "MORTGAGE LIFE IN YEARS, AND ANNUAL PROPERTY TAX BILL FOR HOUSE."
0050 PRINT "SEPARATE ENTRIES WITH A COMMA; FOR EXAMPLE 40000,10.5,25,2000."
0060 PRINT
0070 PRINT " AMOUNT? RATE? YEARS? TAXES?"
0080 INPUT " ? ",A,R1,Y,T
0090 REM GET MONTHLY RATE R (41/12), MAKE INTO FRACTION AS R1 WAS WHOLE NUMBER
0100 LET R=R1/1200
0110 REM GET NUMBER OF MONTHS M FOR LOAN.
0120 LET M=12*Y
0130 REM COMPUTE MONTHLY PAYMENT.
0140 LET P=A*R*(1+R)^M/((1+R)^M-1)
0150 REM DEFINE FORMAT F$ THAT ROUNDS NUMBERS TO NEAREST WHOLE CENT.
0160 LET F$="-----.##"
0180 REM PRINT TOTALS AND GIVE OPTION FOR TAX SUBROUTINE.
0190 PRINT "MONTHLY PAYMENT:      TAXES:      HIDEOUS TOTAL:"
0200 PRINT USING F$,P," ",T," ",P+T
0210 PRINT
0220 PRINT "WANT TO COMPUTE THE TRUE COST AFTER U.S. TAX DEDUCTIONS ON THE"
0230 PRINT "INTEREST AND TAXES? YOU MUST ITEMIZE TO QUALIFY."
0240 INPUT "ANSWER Y (YES) OR N (NO).      ",Q$
0250 REM $ SPECIFER STRING INPUT (E.G."Y") INSTEAD OF NUMERIC.
0260 IF Q$="Y" THEN GOSUB 1000
0270 PRINT
0280 INPUT "TYPE Y (YES) TO RUN PROGRAM AGAIN, ANYTHING ELSE TO STOP. ",Q$
0290 IF Q$="Y" THEN GOTO 0060
0300 STOP
1000 REM TAX DEDUCTION COMPUTATION SUBROUTINE.
1010 INPUT "WHAT IS YOUR TAX BRACKET, IN WHOLE NUMBERS?      ",B1
1020 LET B=B1/100
1030 PRINT
1040 PRINT "SHOULD I LIST PAYMENTS FOR THE FIRST TWO YEARS? I HAVE"
1050 INPUT "TO FIGURE THE INTEREST ANYWAY. ANSWER Y (YES) OR N (NO). ",Q$
1060 REM SET UP VARIABLES A1 (PRINCIPAL PD PER MONTH) AND I1 (FOR TOTAL INTEREST)."
1070 LET A1=A
1080 LET I1=0
1090 IF Q$<>"Y" THEN GOTO 1110
1100 PRINT "      MONTH      PRIN.      INT.      INT. TOTAL"
1110 REM FOR-NEXT LOOP COMPUTES (OPTIONALLY LISTS) FIGURES BY MONTH AND TOTALS.
1120 FOR J=1 TO 24
1130 LET P1=A1*R/((R+1)^M-1)
1140 LET I1=I1+(P-P1)
1150 LET A1=A1-P1
1160 IF Q$<>"Y" THEN GOTO 1180
1170 PRINT USING F$,J,P1,P-P1,I1
1180 NEXT J
1190 PRINT
1200 REM GET DEDUCTIONS D FOR 1 YEAR, T(AXES) + I1/2 (HALF OF 2 YRS. INTEREST)
1210 LET D=T+I1/2
1220 PRINT "ANNUAL MORTGAGE-RELATED DEDUCTIONS ARE:"
1230 PRINT USING F$,D
1240 PRINT "BUT I MUST SUBTRACT THE $3200 STANDARD (O BRACKET) DEDUCTION"
1250 PRINT "BUILT INTO THE TAX TABLES. THE TRUE MONTHLY COST IS:"
1260 LET D1=D-3200
1270 REM GET REAL MO. COST. (TOTAL MO. PAY = P+T12) - ((BRKT * ADJ. DEDS)/12)
1280 LET C=(P+T/12)-(B*D1/12)
1290 PRINT USING F$,C
1300 PRINT "      ****SUMMARY ****"
1310 PRINT "      LIFE:  AMOUNT:  RATE:  CASH PAY:  BRKT:  TRUE COST:"
1320 PRINT USING F$,Y,A,R1,P+T/12,B1,C
1330 RETURN

```

CS-01744

Figure 11.8 MORTGAGE Program with Errors

## Running the BASIC Program

To run the program, get into BASIC (if you're not already there), and type:  
**NEW**

**ENTER "MORTGAGE.BA"** (or whatever name you gave it.)  
**RUN**

```
I CALCULATE MORTGAGE PAYMENTS,
INTEREST, AND TAXES.
TYPE AMOUNT OF PRINCIPAL, INTEREST RATE
IN WHOLE NUMBERS.
MORTGAGE LIFE IN YEARS, AND ANNUAL
PROPERTY TAX BILL FOR HOUSE.
SEPARATE ENTRIES WITH A COMMA; FOR
EXAMPLE 400000, 10.5, 25, 2000.
AMOUNT? RATE? YEARS? TAXES?
?
```

Respond with some plausible figures:

**40000,10.5,25,2000**

The program responds:

```
MONTHLY PAYMENT:   TAXES:   HIDEOUS TOTAL:
      377.67      2000.00      2377.67
WANT TO COMPUTE THE TRUE COST...
... ITEMIZE TO QUALIFY
ANSWER Y (YES) OR N (NO).
```

This seems a little high - over \$2,000 per month. We forgot to convert the yearly tax figure to months. (Depending on the precision generated on your BASIC system, your answers may differ slightly.) It would be meaningless to proceed, so type:

**N**

```
TYPE Y (YES) TO RUN PROGRAM AGAIN.
ANYTHING ELSE TO STOP.
STOP AT 300
```

The problem is in line 200, and you can fix it by dividing the tax variable, T, by 12. Replace the existing line 200 by typing:

**200 PRINT USING F\$,P,"[] []"n,T/12m"["],P+T/12**

and run it again:

```
RUN
I CALCULATE...I
```

```
AMOUNT? RATE? YEARS? TAXES?
40000,10.5,25,2000
MONTHLY PAYMENT:   TAXES:   HIDEOUS TOTAL:
      377.67      166.67      544.34
WANT TO COMPUTE...
ANSWER Y (YES) OR N (NO).
```

These figures are more reasonable, so you can proceed with the program:

Y

WHAT IS YOUR TAX BRACKET, IN WHOLE NUMBER?

The tax bracket issue is explained later in this section. For now, try 25, a typical bracket:

25

SHOULD I LIST PAYMENTS FOR ...

...ANSWER Y (YES) OR N (NO). Y

MONTH	PRIN.	INT.	INT. TOTAL
1.00	27.67	350.00	350.00
2.00	27.65	350.02	700.02
3.00	27.64	350.04	1050.06
4.00	27.62	350.06	1400.11

MORTGAGE-RELATED DEDUCTIONS ARE:

6202.63

BUT I MUST SUBTRACT THE \$3200...

...TRUE MONTHLY COST IS:

481.79

\*\*\*\*SUMMARY \*\*\*\*

LIFE: AMOUNT: RATE: CASH PAY: BRKT: TRUE

COST:

25.00 40000.00 10.50 544.34 25.00 81.79

TYPE Y (YES) TO RUN PROGRAM AGAIN...

It works! At least it appears to work. (Clearly, if the TRUE PAY figure exceeds the CASH PAY figure, it may not pay you to itemize.) Let's examine the interest schedule again though, because the program bases its deductible figure on the last value of variable I1 in this schedule.

Unfortunately, there is a problem in this schedule - the amount of interest paid each month increases while it should decrease. This must be wrong, and being wrong, it voids the entire deduction figure. Looking over the FOR-NEXT loop that computes the monthly interest, note that we forgot to decrement the month indicator, N, for each circuit through the FOR-NEXT loop. You can fix this easily. First, stop the program:

STOP AT 300

Having incremented line numbers by 10, you can easily insert a new statement:

1155 LET M = M-1



Now RUN it again, giving the same figures (40000, 10.5, 25, 20, and 2000) and tax bracket (25), to compare the results. The monthly schedule now says:

MONTH	PRIN.	INT.	INT. TOTAL
1.00	27.67	350.00	350.00
2.00	27.92	349.76	699.76
3.00	28.16	349.51	1049.27
4.00	28.41	349.27	1398.54

MORTGAGE-RELATED DEDUCTIONS ARE:

6164.34

BUT I MUST SUBTRACT...

...TRUE MONTHLY COST IS:

482.58

\*\*\*\*\*SUMMARY\*\*\*\*\*

LIFE: AMOUNT: RATE: CASH      PAY: BRKT: TRUE  
COST:

25.00 40000.00 10.50 544.34 25.00 482.58

TYPE Y (YES) TO RUN PROGRAM AGAIN...

Eureka! The interest is now declining, and the program works correctly. (The difference may not seem significant here, but if the FOR-NEXT loop covered 60 months instead of 24 (i.e., J = 1 to 60), it would be immense.)

You've fixed the program, so you can write it to disc under its original name; this overwrites the old, erroneous version:

**LIST "MORTGAGE.BA"**

You can also print it (**LIST "@LPT"**) if you have a line printer, or type it (**LIST**). To learn **BASIC** and return to the **CLI**, type:

**BYE**

All programs you write via the **BASIC** interpreter reside in the **BASIC** directory; the interpreter automatically goes to this directory when you invoke it.

## Itemized Deductions and Tax Bracket

For simplicity, this sample **BASIC** program assumes you are married, and, when you acquire this mortgage, you will start itemizing deductions instead of taking the standard deduction. When you itemize, the IRS allows you to deduct only the itemized amount over the standard deduction (\$3200 if married and filing jointly, \$2200 if single). The standard deduction is already figured into the tax tables. This is why line 420 of the program subtracts the standard deduction from the mortgage-related deductions. If you are, in fact, moving to itemized deductions, you can deduct much more than mortgage-related expenses (e.g., medical expenses, casualty losses), but the program doesn't deal with these. It figures the **TRUE COST** amount as if you were deducting only the mortgage-related amounts.

Thus, if the mortgage moves you from the standard deduction to itemized deductions, your real cost per month will be less than the TRUE COST figure. If this is true for you, you can modify the program to compute the TRUE figure more accurately. The critical figure is the "3200" in line 1260. Your program statements should get all mortgage-unrelated deductions (medical, contributions, casualty losses, etc.) and put them in a variable; let's say Q. Then it should add Q to D in line 1260; e.g.,  $LET D1 = D + Q = 3200$ .

In any case, if you are single, change the 3200 in line 1260 to 2200; e.g.,  $LET D1 = D - 2200$ .

## **FORTRAN Programming**

This section leads you through a sample session in FORTRAN. It assumes that you have some experience with the FORTRAN programming language.

Follow these steps to run a FORTRAN program:

1. Create and edit the source code using a text editor, for example the SPEED text editor used in Module 10.
2. Compile the source file with the CLI command:

**FORT4 FILENAME** (AOS only)

or

**FORT5** (depending on your compiler.)

3. Correct any compilation errors by returning to Step 1, if necessary.
4. Link or bind the object file:

For AOS (FORTRAN IV): **XEQ BIND FILENAME,[SUBROUTINE NAME...],FORT.LB**

For FORTRAN 5 (AOS or AOS/VS): **F5LD FILENAME**

5. Test the program:

**XEQ FILENAME**

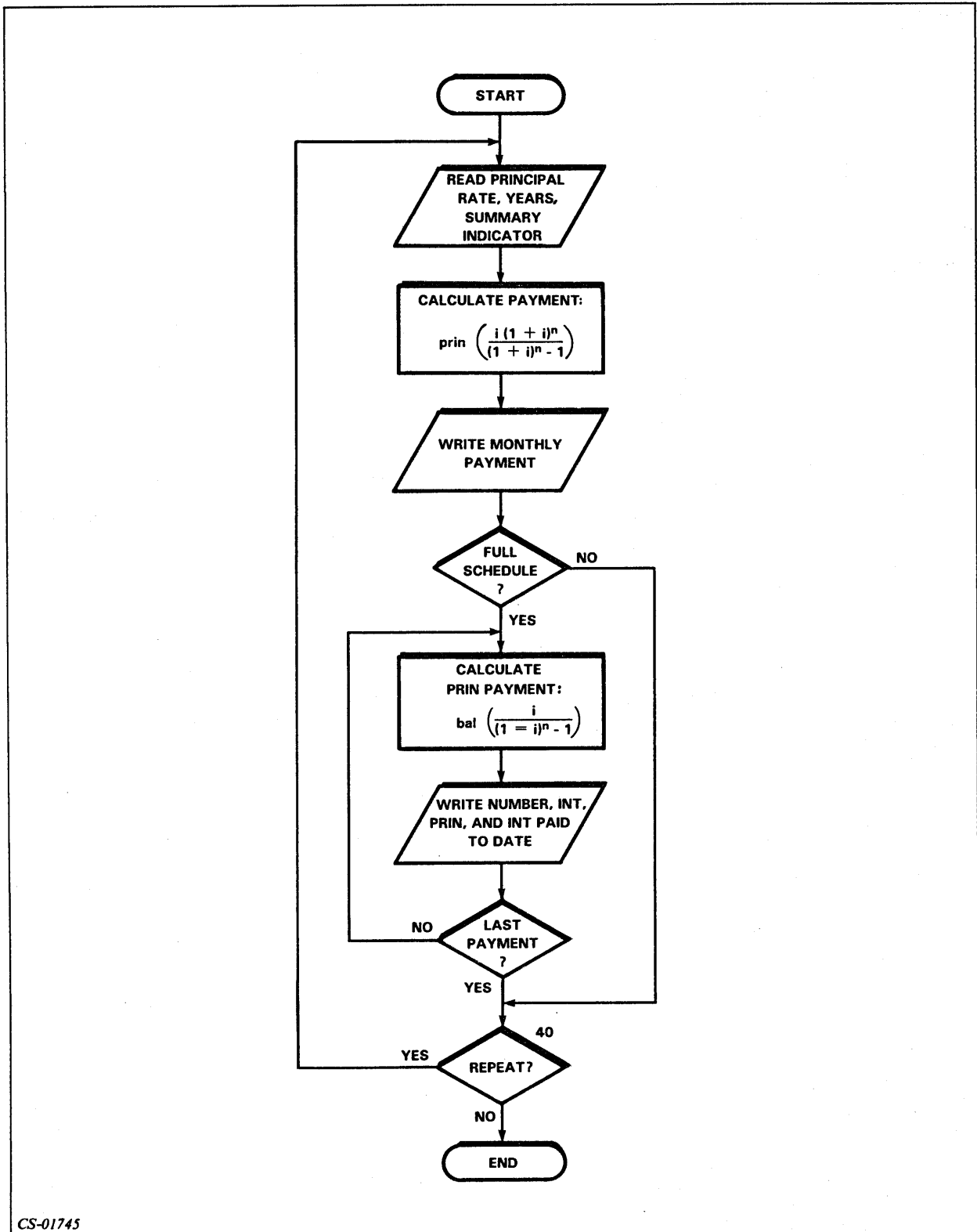
6. If the program is correct, you are finished. If not, find the errors and go to Step 1 to make corrections.
7. Pat yourself on the back, you are finished!

### **Writing the FORTRAN Source Program**

The example in this section is a FORTRAN program to calculate mortgage payments. It produces a month by month schedule of principal and interest. It does this using only simple arithmetic operations and calls no subroutines. The program uses two formulas that you do not need to understand mathematically.

The program reads input from the console and writes the results back to your console.

Review the flowchart (Figure 11.9) and program (Figures 11.10 and 11.11) before proceeding to the next section.



CS-01745

Figure 11.9 MORTGAGE.FR Program Flowchart

```

      REAL I, INT, ITD, LB
5     WRITE (10) " ENTER PRINCIPAL, RATE, YEARS,"
      WRITE (10) " AND 0 FOR SUMMARY OR 1 FOR"
      WRITE (10) " FULL SCHEDULE. SEPARATE EACH"
      WRITE (10) " PARAMETER WITH A COMMA."
      WRITE (10) " TERMINATE INPUT WITH A NEW LINE."
      READ (11) P, R, L, IFULL
C     CHANGE YEARLY RATE (R) TO MONTHLY RATE (I)
      I = R/12.
C     CHANGE YEARS (L) TO MONTHS (N)
      N = L*12
C     CALCULATE MONTHLY PAYMENT (PAY)
      PAY = P*I*(1+I)**N/((1+I)**N-1)
      WRITE (12,110) P, R, L
110    FORMAT (1H1,"PRINCIPAL = $",F9.2,/, " INTEREST RATE =",
-       F7.4,/, " LOAN LIFE IS",I4, " YEARS",/,/)
      WRITE (12,120) PAY
120    FORMAT (1H0,"MONTHLY PAYMENT = $",F10.2)
      IF (IFULL .LE.0) GO TO 40
C     INITIAL LOAN BALANCE (LB) EQUALS PRINCIPAL
      LB = P
C     CREATE COUNT (NN) FOR USE IN OUTPUTTING SCHEDULE
      NN = N
      ITD = 0.
      WRITE (12,130)
130    FORMAT (1H0," NUM",7X,"INTEREST",5X,"PRIN.PAY   PRIN."
-       " BAL",6X,"INTEREST PAID TO DATE",/)
      DO 30 NP=1,NN
C     CALCULATE PRINCIPAL PORTION (PN)
      PN = LB*I/((1+I)**N-1)
      N = N-1
C     CALCULATE INTEREST PORTION (INT)
      INT = PAY-PN
C     UPDATE LOAD BALANCE
      LB = LB-PN
C     UPDATE INTEREST PAID TO DATE
      ITD = ITD+INT
      WRITE (12,140) NP, INT, PN, LB, ITD
140    FORMAT (1H0,I3,7X,"$",F9.2,"   $",F9.2,"   $",F9.2,
-       8X,"$",F9.2)
      30 CONTINUE
      40 WRITE (10) " TYPE 1 TO REPEAT, 0 TO STOP
      READ (11) IR
      IF (IR .GT. 0) GO TO 5
      END

```

CS-01746

Figure 11.10 MORTGAGE Program (Initial Version with Errors)

## Compiling the Program

Now that you have written and entered the program into a file, it is time to compile the program. The command to compile the program is:

**FORT4 MORTGAGE** (For FORTRAN IV)

**F5 MORTGAGE** (For FORTRAN 5)

When you look at the output of the compilation, you will see that there were a few errors. You will see the error code and an associated line number where the error occurred. The error may have actually occurred one or two lines before the point where the compiler detected it. Also, one error may generate more than one error message.

Refer to the appropriate reference manual, AOS or AOS/VS, for more information about compilation errors.

Figure 11.11 lists the corrected version of the program.

When these errors are corrected, you can compile the program again and then proceed to the next step.

```

5   REAL I, INTEREST, ITD, LB ←
    WRITE (10) " ENTER PRINCIPAL, RATE, YEARS, "
    WRITE (10) " AND 0 FOR SUMMARY OR 1 FOR"
    WRITE (10) " FULL SCHEDULE. SEPARATE EACH"
    WRITE (10) " PARAMETER WITH A COMMA."
    WRITE (10) " TERMINATE INPUT WITH A NEW LINE."
    READ (11) P, R, L, IFULL
C   CHANGE YEARLY RATE (R) TO MONTHLY RATE (I)
    I = R/12.
C   CHANGE YEARS (L) TO MONTHS (N)
    N = L*12
C   CALCULATE MONTHLY PAYMENT (PAY)
    PAY = P*I*(1+I)**N/((1+I)**N-1)
    WRITE (12,110),P, R, L
110  FORMAT (1H1,"PRINCIPAL = $",F9.2,/, " INTEREST RATE =",
    -F7.4,/, " LOAN LIFE IS",I4," YEARS",/,/)
    WRITE (12,120) PAY
120  FORMAT (1H0,"MONTHLY PAYMENT =$",F10.2)
    IF (IFULL .LE. 0) GO TO 40
C   INITIAL LOAN BALANCE (LB) EQUALS PRINCIPAL
    LB = P
C   CREATE COUNT (NN) FOR USE IN OUTPUTTING SCHEDULE
    NN = N
    ITD = 0.
    WRITE (12,130)
130  FORMAT (1H0," NUM",7X,"INTEREST",5X,"PRIN.PAY   PRIN."
    -" BAL",6X,"INTEREST PAID TO DATE",/)
    DO 30 NP-1,NN
C   CALCULATE PRINCIPAL PORTION (PN)
    PN = LB*I/((1+I)**N-1)
    N = N-1
C   CALCULATE INTEREST PORTION (INTEREST) ←
    INTEREST = PAY-PN ←
C   UPDATE LOAD BALANCE
    LB = LB-PN
C   UPDATE INTEREST PAID TO DATE
    ITD = ITD+INTEREST ←
    WRITE (12,140) NP, INTEREST, PN, LB, ITD
140  FORMAT (1H0,I3,7X,"$",F9.2,"   $",F9.2,"   $",F9.2,
    -8X,"$",F9.2)
    30  CONTINUE
    40  WRITE (10) " TYPE 1 TO REPEAT, 0 TO STOP " ←
        READ (11) IR
        IF (IR .GT. 0) GO TO 5
    END

```

} INT IS A  
PREDEFINED  
FUNCTION

ADD CLOSING  
QUOTES

CS-01747

Figure 11.11 MORTGAGE Program (Corrected Version) (Arrows Point to Correct Lines.)

## Binding or Linking the Program

Before you can execute the program, we must bind or link the object program. You can do that with the following CLI commands:

For AOS (FORTRAN IV): XEQ BIND MORTGAGE,FORT.LB

For FORTRAN 5 (AOS or AOS/VS): **F5LD MORTGAGE**

F5LD MORTGAGE is a macro that invokes the LINK or BIND utilities and includes the appropriate libraries.

## Executing the Program

We now proceed to the next step, executing MORTGAGE.PR. To execute this or any other program, simply type the XEQ command with the name of the program and follow it with a new-line character. Typing XEQ MORTGAGE gives us the following message on the console:

```
ENTER PRINCIPAL , RATE , YEARS ,  
AND 0 FOR SUMMARY OR 1 FOR  
FULL SCHEDULE . SEPARATE EACH  
PARAMETER WITH A COMMA .  
TERMINATE INPUT WITH A NEW LINE .
```

We then respond with a request for the summary information (monthly payment only), given a mortgage of \$20,000 at 9% for 25 years.

```
20000,.09,25,0
```

Since MORTGAGE writes the summary on the line printer, we won't see the output until we terminate the program.

MORTGAGE prompts:

```
TYPE 1 TO REPEAT , 0 TO STOP  
1
```

Since we responded with 1, the same initial instructions appear on the console.

This time we repeat the same arguments but ask for a detailed schedule:


```
20000,.09,25,1
```

And get the following results.

PRINCIPAL = \$ 20000.00  
 INTEREST RATE = 0.0900  
 LOAN LIFE IS 25 YEARS

MONTHLY PAYMENT = \$ 167.84

NUM	INTEREST	PRIN. PAY	PRIN. BAL	INTEREST PAID TO DATE
1	\$ 150.00	\$ 17.84	\$ 19982.16	\$ 150.00
2	\$ 149.87	\$ 17.98	\$ 19964.18	\$ 299.87
3	\$ 149.73	\$ 18.11	\$ 19946.07	\$ 449.60
4	\$ 149.60	\$ 18.25	\$ 19927.82	\$ 599.19
5	\$ 149.46	\$ 18.38	\$ 19909.43	\$ 748.65
6	\$ 149.32	\$ 18.52	\$ 19890.91	\$ 897.97
7	\$ 149.18	\$ 18.66	\$ 19872.24	\$ 1047.15
8	\$ 149.04	\$ 18.80	\$ 19853.44	\$ 1196.20
9	\$ 148.90	\$ 18.94	\$ 19834.50	\$ 1345.10
10	\$ 148.76	\$ 19.08	\$ 19815.41	\$ 1493.86
11	\$ 148.62	\$ 19.23	\$ 19796.19	\$ 1642.47
12	\$ 148.47	\$ 19.37	\$ 19776.82	\$ 1790.94



CS-01748

Figure 11.12 Sample of Extended Schedule from MORTGAGE



## COBOL Programming

This section leads you through a sample session in COBOL. It assumes that you have some experience with the COBOL programming language.

Follow these steps to run a COBOL program:

1. Create and edit the source code using a text editor, for example the SPEED text editor used in Module 10.

2. Compile the source file with the CLI command:

**COBOL FILENAME**

3. Correct any compilation errors by returning to Step 1, if necessary.

4. Link or bind the object file:

For AOS: **CBIND FILENAME,[SUBROUTINE NAME...]**

For AOS/VS: **CLINK FILENAME,[SUBROUTINE NAME...]**

5. Test the program:

**XEQ FILENAME**

6. If the program is correct, you are finished. If not, find the errors and go to Step 1 to make corrections.

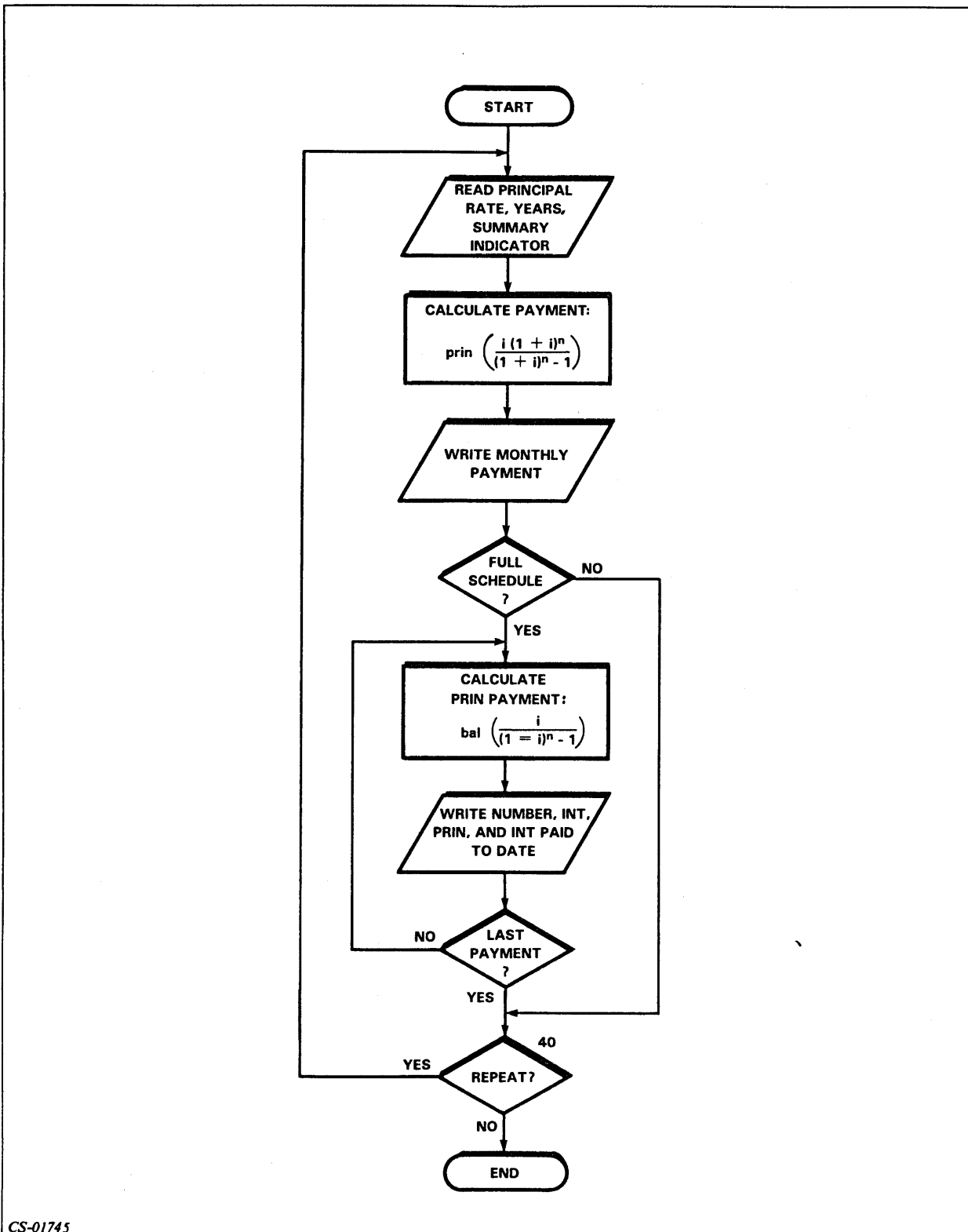
7. Pat yourself on the back; you are finished!

### Writing the COBOL Source Program

The example in this section is a COBOL program to calculate mortgage payments. It produces a month by month schedule of principal and interest. It does this using only simple arithmetic operations and calls no subroutines. The program uses two formulas that you do not need to understand mathematically.

The program reads input from the console and writes the results back to your console.

Review the flowchart (Figure 11.13) and program (Figure 11.14) before proceeding to the next section.



CS-01745

Figure 11.13 MORTGAGE Program Flowchart

```

IDENTIFICATION DIVISION.
PROGRAM-ID. MORTPROG.
AUTHOR. JOE SCHMOE.
DATE-WRITTEN. 6 MARCH 1982
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. ECLIPSE.
OBJECT-COMPUTER. ECLIPSE.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT OUTFILE, ASSIGN TO PRINTER.
DATA DIVISION.
FILE SECTION.
FD      OUTFILE, BLOCK CONTAINS 512 CHARCTERS.
01      OUTREC.
        02 OUT-PAYMT-NUM          PIC ZZ9.
        02 FILLER                 PIC X(6).
        02 OUT-MON-INT            PIC $(4)9.99.
        02 FILLER                 PIC X(3).
        02 OUT-MON-PRIN          PIC $(6)9.99.
        02 FILLER                 PIC X(3).
        02 OUT-BALANCE           PIC $(6)9.99.
        02 FILLER                 PIC X(8).
        02 OUT-INT-TO-DATE       PIC $(6)9.99.
        02 FILLER                 PIC X(10).

WORKING-STORAGE SECTION.
01      CRT-INPUTS.
        02 PRINCIPAL              PIC 9(6)V99.
        02 PERCENT                PIC 99V99.
        02 YEARS                  PIC 99.
        02 FUNCTION               PIC 9.
        02 REPEAT-FLAG           PIC 9.

01      TEMPS.
        02 MONTHLY-INT-RATE      USAGE COMP-1.
        02 MONTHS                 PIC 9(4).
        02 MONTHS-LEFT           PIC 9(4).
        02 MONTHLY-PAYMT         PIC 9(4)V99.
        02 LOAN-BAL              PIC 9(6)V99.
        02 INT-TO-DATE           PIC 9(6)V99.
        02 PAYMT-NUM             PIC 9(4), USAGE COMP.
        02 INT-PAYMT             PIC 9(4)V99.
        02 PRIN-PAYMT            PIC 9(4)V99.

01      SUMMARY-LINE1.
        02 FILLER                 PIC X(16), VALUE "PRINCIPAL = "
        02 SUMMARY-PRIN, PIC $(6)9.99.
        02 FILLER                 PIC X(50), VALUE SPACES.
01      SUMMARY-LINE2.
        02 FILLER                 PIC X(20), VALUE "INTEREST RATE = "
        02 SUMMARY-RATE, PIC 9.9(4).
        02 FILLER                 PIC X(50), VALUE SPACES.
01      SUMMARY-LINE3.
        02 FILLER                 PIC X(18), VALUE "LOAN LIFE = "
        02 SUMMARY-YEARS, PIC Z9.
        02 FILLER                 PIC X(6), VALUE " YEARS".
        02 FILLER                 PIC X(50), VALUE SPACES.
01      SUMMARY-LINE4.
        02 FILLER                 PIC X(18), VALUE "MONTHLY PAYMENT = "
        02 SUMMARY-PAYMT, PIC $(4)9.99.

```

CS-01749

Figure 11.14 MORTGAGE Program

```

01      02 FILLER          PIC X(50), VALUE SPACES.
      HEADLINE           PIC X(80),
      VALUE " NUM       INTEREST   PRIN. PAY PRIN.
      " BAL             INTEREST PAID TO DATE".
-
PROCEDURE DIVISION.
INIT.  OPEN OUTPUT OUTFILE.
OPERATOR.
      DISPLAY "ENTER PRINCIPAL:  $" WITH NO ADVANCING.
      ACCEPT PRINCIPAL.
      DISPLAY "INTEREST RATE (%): " WITH NO ADVANCING.
      ACCEPT PERCENT.
      COMPUTE MONTHLY-INT-RATE = PERCENT / 100 / 12.
      DISPLAY "YEARS TO PAY:    " WITH NO ADVANCING.
      ACCEPT YEARS.
      COMPUTE MONTHS = YEARS * 12.
      DISPLAY "FUNCTION (0=SUMMARY, 1-FULL SCHEDULE): "
      WITH NO ADVANCING.

      ACCEPT FUNCTION.
      COMPUTE MONTHLY-PAYMT ROUNDED =
      PRINCIPAL * MONTHLY-INT-RATE *
      (1 + MONTHLY-INT-RATE) ** MONTHS /
* -----
      ((1 + MONTHLY-INT-RATE) ** MONTHS - 1).
      PERFORM SUMMARY-OUTPUT.
      IF FUNCTION NOT = 0,
      PERFORM DETAIL-OUTPUT.
      DISPLAY "TYPE 1 TO REPEAT, 0 TO STOP: " WITH NO ADVANCING.
      ACCEPT REPEAT-FLAG.
      IF REPEAT-FLAG NOT = 0, GO TO OPERATOR.
      CLOSE OUTFILE.
      STOP RUN.
SUMMARY-OUTPUT.
      MOVE PRINCIPAL TO SUMMARY-PRIN.
      WRITE OUTREC FROM SUMMARY-LINE1 BEFORE ADVANCING 1.
      COMPUTE SUMMARY-RATE = PERCENT / 100.
      WRITE OUTREC FROM SUMMARY-LINE2 BEFORE ADVANCING 1.
      MOVE YEARS TO SUMMARY-YEARS.
      WRITE OUTREC FROM SUMMARY-LINE3 BEFORE ADVANCING 2.
      MOVE MONTHLY-PAYMT TO SUMMARY-PAYMT.
      WRITE OUTREC FROM SUMMARY-LINE4 BEFORE ADVANCING 2.
DETAIL-OUTPUT.
      MOVE PRINCIPAL TO LOAN-BAL.
      MOVE MONTHS TO MONTHS-LEFT.
      MOVE 0 TO INT-TO-DATE.
      WRITE OUTREC FROM HEADLINE BEFORE ADVANCING 2.
      MOVE SPACES TO OUTREC.
      PERFORM DO-DETAIL-LINE
      VARYING PAYMT-NUM FROM 1 BY 1
      UNTIL PAYMT-NUM > MONTHS.
DO-DETAIL-LINE.
      COMPUTE PRIN-PAYMT ROUNDED =
      LOAN-BAL * MONTHLY-INT-RATE /
* -----
      ((1 + MONTHLY-INT-RATE) ** MONTHS-LEFT - 1).
      SUBTRACT 1 FROM MONTHS-LEFT.
      COMPUTE INT-PAYMT = MONTHLY-PAYMT - PRIN-PAYMT.
      SUBTRACT PRIN-PAYMT FROM LOAN-BAL.
      ADD INT-PAYMT TO INT-TO-DATE.
      MOVE PAYMT-NUM TO OUT-PAYMT-NUM.
      MOVE INT-PAYMT TO OUT-MON-INT.
      MOVE PRIN-PAYMT TO OUT-MON-PRIN.

```

CS-01750

Figure 11.15 MORTGAGE Program (Continued)

## Compiling the Program

Now that you have written and entered the program into a file, it is time to compile the program. The command to compile the program is:

**COBOL MORTGAGE**

When you look at the output of the compilation, you will see that there were no errors. If there were, you would have seen an error code and an associated line number where the error occurred. The error may have actually occurred one or two lines before the point where the compiler detected it. Also, one error may generate more than one error message. Refer to the appropriate reference manual, AOS or AOS/VS, for more information about compilation errors.

When the errors have been corrected, you can compile the program again and then proceed to the next step.

## Binding or Linking the Program

Before you can execute the program, you must link or bind the object program. You can do that with the following CLI commands:

For AOS: **CBIND MORTGAGE**

For AOS/VS: **CLINK MORTGAGE**

## Executing the Program

To execute our COBOL program, or any other program, simply type the XEQ command followed by the program name and a new-line character. For our COBOL program, then, type:

**XEQ MORTPROG**

This produces the following dialog on the console:

```
ENTER PRINCIPAL : $25000.00
INTEREST RATE (%) : 8.75
YEARS TO PAY : 20

FUNCTION (0=SUMMARY, 1=FULL SCHEDULE) :
1
```

For each of these above three queries, we respond with an appropriate answer. In this example, we want the monthly payment table for a \$25,000 mortgage at 8.75% for 20 years. We also want the full payment schedule.

Since the MORTPROG program writes the payment table out to the line printer, we won't see the output until we terminate the program.


After the MORTPROG program completes its calculations for the principal, rate, and term that we selected, it outputs the message:

```
TYPE 1 TO REPEAT, 0 TO STOP
```

If we type **1**, the program will repeat the queries for principal, rate, and years to pay and function. Then, once again, the program will calculate the table of monthly payments, but not print it.

If we type 0, the output file is closed and the program is terminated. Then, any monthly payment tables which we requested from the MORTPROG program will be output to the line printer, which we assigned as the output file. We have included a partial listing of the monthly payment schedule for a \$25,000 mortgage at 8.75% for 20 years in Figure 11.16.

PRINCIPAL = \$25000.00				
INTEREST RATE = 0.0875				
LOAD LIFE = 20 YEARS				
MONTHLY PAYMENT = \$220.93				
NUM	INTEREST	PRIN. PAY	PRIN. BAL	INTEREST PAID TO DATE
1	\$182.29	\$38.64	\$24961.36	\$182.29
2	\$182.01	\$38.92	\$24922.44	\$364.30
3	\$181.73	\$39.20	\$24883.24	\$546.03
4	\$181.44	\$39.49	\$24843.75	\$727.47
5	\$181.15	\$39.78	\$24803.97	\$908.62
6	\$180.86	\$40.07	\$24763.90	\$1089.48
7	\$180.57	\$40.36	\$24723.54	\$1270.05
8	\$180.28	\$40.65	\$24682.89	\$1450.33
9	\$179.98	\$40.95	\$24641.94	\$1630.31
10	\$179.68	\$41.25	\$24600.69	\$1809.99
11	\$179.38	\$41.55	\$24559.14	\$1989.37
12	\$179.08	\$41.85	\$24517.29	\$2168.45



CS-01751

Figure 11.16 Sample Payment Schedule

## Assembly Language Programming

This section leads you through a sample session in Assembly Language. It does not attempt to teach you how to program using the Data General Assembly Language.

Follow these steps to run an Assembly Language program:

1. Create and edit the source code using a text editor, for example, the SPEED text editor used in Module 10.

2. Assemble the source file with the CLI command:

```
XEQ MASM FILENAME
```

3. Correct any compilation errors by returning to Step 1, if necessary.

4. Link or bind the object file:

```
For AOS: XEQ BIND FILENAME,[SUBROUTINE NAME...]
```

```
For AOS/VS: XEQ LINK FILENAME,[SUBROUTINE NAME ...]
```

5. Test the program:

```
XEQ FILENAME
```

6. If the program is correct, you are finished. If not, find the errors and go to Step 1 to make corrections.

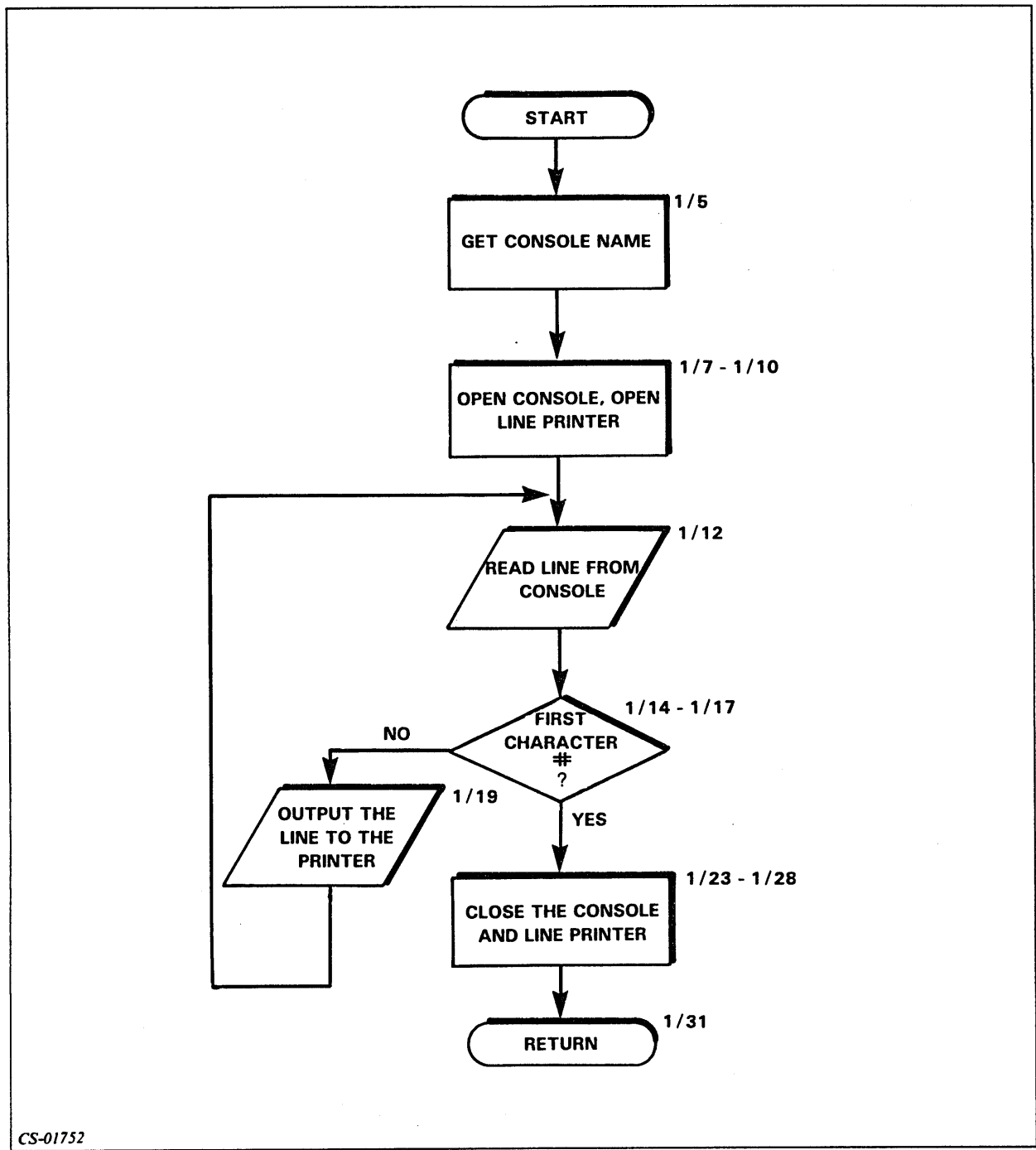
7. Pat yourself on the back; you are finished!

### Writing the Assembly Language Source Program

The example program, entitled WRITE, is a short Assembly Language program that goes through the standard I/O cycle and uses two devices: the console and the line printer. This program lets you type one or more lines on your console; then it writes these on the line printer. When you type a special terminating line (#), the program returns to the CLI. Figure 11.17 shows you a flowchart of this program's activity.

Upon any exceptional condition, the program reports the condition by using ?RETURN. The system displays on the console all lines typed on the console. (The system always performs this service.) Lines output to the printer are not printed until it is closed, whereupon the system prints them with a standard header prefix.

Study the flowchart and source listing in Figure 11.17 before continuing to the next section.



CS-01752

Figure 11.17 WRITE Flowchart



```

0001 WRITE AOS ASSEMBLER REV 01.07                                09:40:44 01/13/78
      .TITL  WRITE
02      .ENT  START
03      .NREL
04
05      START: ?GTMES  GTMFK  ;GET ORIGINAL CLI COMMAND
06 00004'000447      JMP  ERTN  ;EXCEPTION RETURN
07      ?OPEN  CONSOLE ;OPEN CONSOLE FOR READING
08 00011'000442      JMP  ERTN  ;REPORT ERROR
09      ?OPEN  LINPT   ;OPEN LINE PRINTER FOR WRITING
10 00016'000435      JMP  ERTN  ;REPORT ERROR
11
12      CONT:  ?READ  CONSOLE ;READ A RECORD
13 00023'000430      JMP  ERTN  ;REPORT ERROR
14 00024'020434      LDA  0,TERM ;CHECK FOR "#" TERMINATOR
15 00025'024545      LDA  1,CONSOLE+3 ;GT BYTEPTR FROM CONSOLE PKT
16 00026'126710      LDB  1,1   ;GET 1ST CHARACTER
17 00027'106415      SUB  # 0,1,SNR ;SKIP IF 1ST CHAR. NOT #
18 00030'000407      JMP  CLOSE ;TERMINATOR, CLOSE DEVICES.
19      ?WRITE  LINPT   ;WRITE THE RECORD TO THE PRINTER
20 00035'000416      JMP  ERTN  ;REPORT ERROR
21 00036'000761      JMP  CONT  ;READ ANOTHER CONSOLE RECORD
22
23      CLOSE: ?CLOSE  CONSOLE ;CLOSE THE CONSOLE
24 00043'000410      JMP  ERTN  ;REPORT THE ERROR
25      ?CLOSE  LINPT   ;CLOSE THE LINE PRINTER
26 00050'000403      JMP  ERTN  ;REPORT ERROR
27 00051'152620      SUBZR  2,2   ;SET "GOOD RETURN" FLAG
28 0;0052'000402      JMP  .+2   ;AND SKIP OVER ERROR FLAG SETTINGS
29
30 00053'030404 ERTN:  LDA  2,FLAGS ;GET ERROR FLAGS
31      ?RETURN  ;RETURN TO CLI
32 00056'000775      JMP  ERTN  ;TRY TO REPORT ?RETURN ERROR
33
34 00057'150000 FLAGS: ?RFEC+?RFCF+?RFER ;?RETURN FLAGS
35 00060'000043 TERM:  043   ;ASCII CODE FOR "#"
36 00061'000074 IBUF:  .BLK  60. ;120 CHARS MAXIMUM PER LINE
37      ;PARAMETER PACKETS
38
39      ;?GTMES PACKET
40
41 00155'000003 GTMPK: ?GARG
42 00156'000001      1       ;GET THE CONSOLE NAME
43 00157'000000      0
44 00160'000342"      CON*2  ;BYTE POINTER TO ADRS OF CONSOLE NAME
45
46      ;END OF ?GTMES PACKET
47
48 00161'000006 CON:  .BLK  6   ;AREA TO RECEIVE CONSOLE NAME
49
50      ;READ PACKET
51
52 00167'000000 CONSOLE:0 ;PACKET TO OPEN AND READ CONSOLE
53 00170'000022      ?RTDS+?OFIN ;DATA SENSITIVE READS
54 0;0171'000000      0
55 00172'000142"      IBUF*2  ;BYTE POINTER TO READ BUFFER
56 00173'000000      0
57 00174'000170      120.    ;120 CHAS. MAX IN EACH RECORD
58 00175'000000      0
59 00176'000000      0
60 00177'000000      0

```

CS-01753

Figure 11.18 WRITE

```

10002 WRITE
01 00200'000342"      CON*2      ;BYTE POINTER TO CONSOLE NAME
02 00201'177777      -1
03 00202'177777      -1
04                      ;END OF ?READ PACKET
05
06                      ;START OF ?WRITE PACKET
07
08 00203'000000      LINPT: 0
09 00204'000012      ?RTDS+?OFOT
10 00205'000000      0
11 00206'000142"      IBUF*2      ;BYTE POINTER TO RECORD BUFFER

12 00207'000000      0
13 00210'000170      120.      ;WRITE 120 RECORDS MAXIMUM
14 00211'000000      0
15 00212'000000      0
16 00213'000000      0
17 00214'000436"      LINP*2
18 00215'177777      -1
19 00216'177777      -1
20                      ;END OF ?WRITE PACKET
21
22 00217'040114      LINP  .TXT  "@LPT"
23      050124
24      000000
25
26
27                      .END      START ;START AT THE BEGINNING

**00000 TOTAL ERRORS, 00000 PASS 1 ERRORS

```

```

0003 WRITE

CLOSE 000037'      1/18      1/23
CON   000161'      1/44      1/48      2/01
CONSO 000167'      1/08      1/13      1/15      1/24      1/52
CONT  000017'      1/12      1/21
ERTN  000053'      1/06      1/08      1/10      1/13      1/20      1/24      1/26
      1/30      1/32
FLAGS 000057'      1/30      1/34
GTMPK 000155'      1/06      1/41
IBUF  000061'      1/36      1/55      2/11
LINP  000217'      2/17      2/22
LINPT 000203'      1/10      1/20      1/26      2/08
START 000000' EN   1/02      1/05      2/27
TERM  000060'      1/14      1/35
?CLOS 002277 MC    1/23      1/25
?GTME 002424 MC    1/05
?OPEN 002254 MC    1/07      1/09
?READ 002322 MC    1/12
?RETU 002447 MC    1/31
?WRIT 002345 MC    1/19
?XCAL 000001      1/06      1/08      1/10      1/13      1/20      1/24      1/26
      1/32

```

CS-01754

Figure 11.19 WRITE (Continued)

## Assembling the Program

Now that you have written and entered the program into a file, it is time to compile the program. The command to compile the program is:

```
XEQ MASM WRITE
```

When you look at the output of the compilation, there should be no errors. If there are errors, you will see them flagged with an error indicator following the line number of the line where the error occurred. For more information on errors and Assembly Language programming in general, refer to the reference manuals.

### **Binding or Linking the Program**

Before you can execute the program, you must link or bind the object program. You can do that with the following CLI commands:

For AOS: **XEQ BIND WRITE**

For AOS/VS: **XEQ LINK WRITE**

### **Executing the Program**

To try out our program, we type the following:

**XEQ WRITE @CON13**

After waiting briefly for our process to be created, we then type in a test line:

**WRITE RUNS RIGHT,Q.E.D.**

and follow this with a terminating line: **#**

These lines echo on the console, and our program ends and reinvokes the CLI. Our test line is printed on the line printer.

## Lab Activity 11-1

The Lab Activity for this module is to execute the program in the programming section that you read. You should complete the steps necessary to enter, compile, correct, link or bind, and execute the program. Of course, if you are a BASIC programmer, you will not do all of these steps.

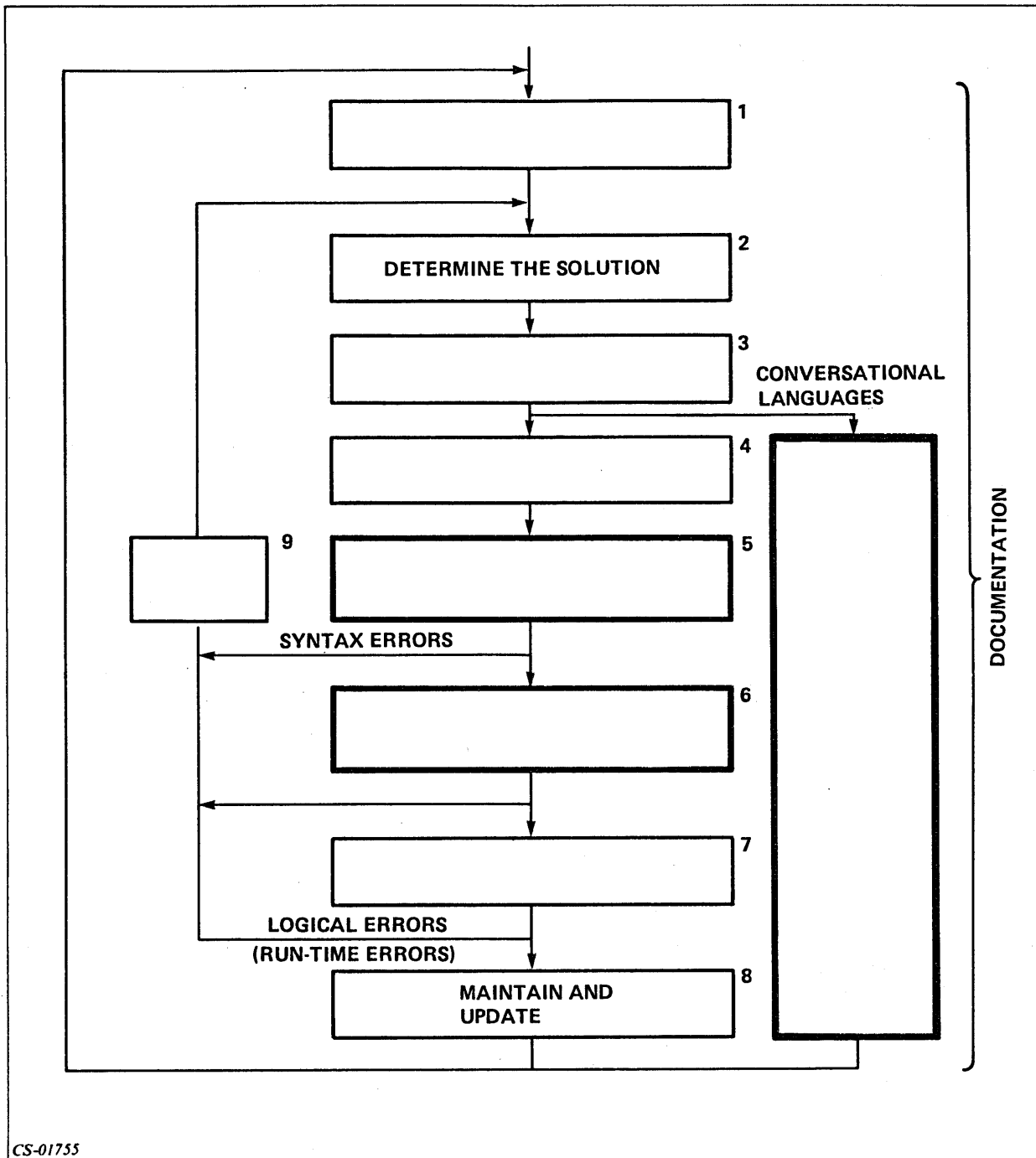
There is no solution listed for this Lab Activity. You will know that you completed it successfully if the program gives you the correct results. Then, continue to the Module 11 Test on the next page.

## Module 11

### Test

**Directions:** Complete the following sentences:

1. The utility program that converts a high-level language into machine code is called a(n) \_\_\_\_\_.
2. The utility program that converts Assembly Language into machine code is called a(n) \_\_\_\_\_.
3. The general term for the program that converts a source language into machine language is \_\_\_\_\_.
4. The file that is input to the program in Question 3 is in \_\_\_\_\_ format.
5. A(n) \_\_\_\_\_, used with conversational languages, combines several program development steps into one.
6. The program that combines several separately translated sections of code, links them, and produces an executable file is called \_\_\_\_\_.
7. The files that are input to the program in Question 6 are in \_\_\_\_\_ code.
8. The CLI command used to test a program called TEST1 that has been compiled and linked is \_\_\_\_\_.
9. The next command that you must issue after you have successfully compiled a program called TEST1 is \_\_\_\_\_.
10. What command would you use to (choose one unless you program BASIC, then skip this question) assemble a program, compile a FORTRAN program, compile a COBOL program named TEST1?
11. Fill in the boxes in Figure 11.20 using the information below.
  - CORRECT ERRORS
  - EXECUTE, TEST, AND VALIDATE THE PROGRAM
  - USE A TRANSLATOR TO CONVERT THE CODE INTO MACHINE LANGUAGE
  - USE A PROGRAMMING LANGUAGE TO CODE THE SOLUTION
  - ENTER THE CODE INTO THE COMPUTER SYSTEM
  - LINK SUPPORT MODULES AND ASSIGN MEMORY LOCATIONS
  - USE AN INTERPRETER
  - DEFINE THE PROBLEM



CS-01755

Figure 11.20

Now check your answers in Appendix A of your *Student Guide*. If you did not answer all the questions correctly, review the material and do the Test again.

This concludes the *AOS, AOS/VS User Self-Study Course*.

# Appendix A

## Module Tests Answers

### Module 1

1. a.
2. c.
3. c.

### Module 2

1. b.
2. c.
3. f.
4. d.
5. a.
6. b.
7. g.
8. e.
9. c.
10. a.
11. **DATE**
12. **TIME**
13. **WHO**
14. **BYE**
15. Press **NEW LINE**.  
Enter your username, followed by **NEW LINE**.  
Enter your password, followed by **NEW LINE**.  
Wait for the system prompt.

### Module 3

1. A 4  
B 5  
C 3  
D 2
2. I (No space after the command.)
3. V
4. V
5. I (No spaces between switches.)
6. V
7. I
8. V
9. I
10. I
11. V
12. V
13. V
14. V
15. I
16. V
17. DELETE/V/C,XYZ
18. TYPE,NEWS\_FILE
19. RENAME,GOOD\_STUFF,OLD\_JUNK
20. CREATE/I,TERMINAL\_DATA
21. COPY,PERSONNEL,PAYROLL
22. HELP/V DELETE
23. COPY/A,MASTER\_FILE,FILE1A,FILE1B,FILE2A,FILE2B
24. TYPE,USELESS\_FILE  
DELETE,USELESS\_FILE
25. TYPE,FILE1,FILE2,TEST1,TEST2
26. DELETE FILE1.OLD,FILE2.OLD,FILE3.OLD
27. TYPE,FILE1  
TEST,TEST2



**Module 4**

1. :
2. @ or PER
3. UTIL
4. UPD
5. UDD
6. :UDD:USER1:DIRA:DIRA2:FILEC
7. :UDD:USER1:DIRB:FILE1
8. :UDD:USER2:FILED
9. :UDD:USER3
10. :UDD:USER3:STUFF:IMPORTS
11. PAYROLL+ OR PAYROLL-
12. PAYROLL\*
13. + FORMS
14. -
15. +2+
16. CREATE/DIRECTORY,SUPER\_DIR
17. CREATE/DIRECTORY/MAXSIZE=10,LITTLE\_C.P.D.
18. FILESTATUS
19. DIRECTORY
20. DIRECTORY,:UTIL
21. SEARCHLIST
22. SEARCHLIST,:UTIL
23. PATHNAME,FORTRAN.PR

**Module 5**

1. The superuser privilege allows the user access to any file. It overrides checking of any file's ACL. When superuser is in effect, the user's prompt changes from ) to \*).
2. ACL,FILE\_1
3. ACL,FILE\_1,JOHN,OWARE,MARY,R
4. DEFACL
5. DEFACL,+,OWARE

6. SUPERUSER,ON

7. SUPERUSER,OFF

8.

Access	Abbreviation	Nondirectory File	Directory File
Execute	E	User can execute the file.	User can use the directory in a path-name.
Read	R	User can read (examine) data in the file.	User can examine the list of files.
Append	A	N/A	User can insert new files in directory.
Write	W	User can modify the contents of the file.	User can insert and delete files and change ACLs of files in the directory.
Owner	O	User can change file's ACL or delete files.	User can change directory's ACL or delete the directory.

Table 5.D Module 5 Test Answers: ACLs

### Module 6

1. b.
2. Sequence  
job
3. QPRINT/NOTIFY,PRINT\_1
4. QPLOT/COPIES=5/NOTIFY,PLOT\_FILE
5. QDISPLAY/TYPE=PLOT
6. QCANCEL,453
7. QHOLD,29
8. QUNHOLD,90
9. QBATCH,INDEPENDENT\_1

### Module 7

1. c.
2. a.
3. b.
4. d.
5. c.
6. c.
7. b.
8. d.

9. a.
10. c.
11. **PUSH**
12. **POP**
13. **LEVEL**
14. **CURRENT**
15. **PROMPT,TIME**
16. **LISTFILE,TEMPORARY\_OUTPUT**
17. **SUPERUSER**
18. **SUPERPROCESS,ON**
19. **SQUEEZE,ON**
20. **CLASS2,IGNORE**

## **Module 8**

1.
  - a. Resident
  - b. Preemptible
  - c. Swappable
2.
  - a. Eligible
  - b. Ineligible
  - c. Blocked
3. a.
4. (In any order:)  
Unique ID  
Username  
Memory  
Priority  
Privileges  
State  
Type  
Program

5.

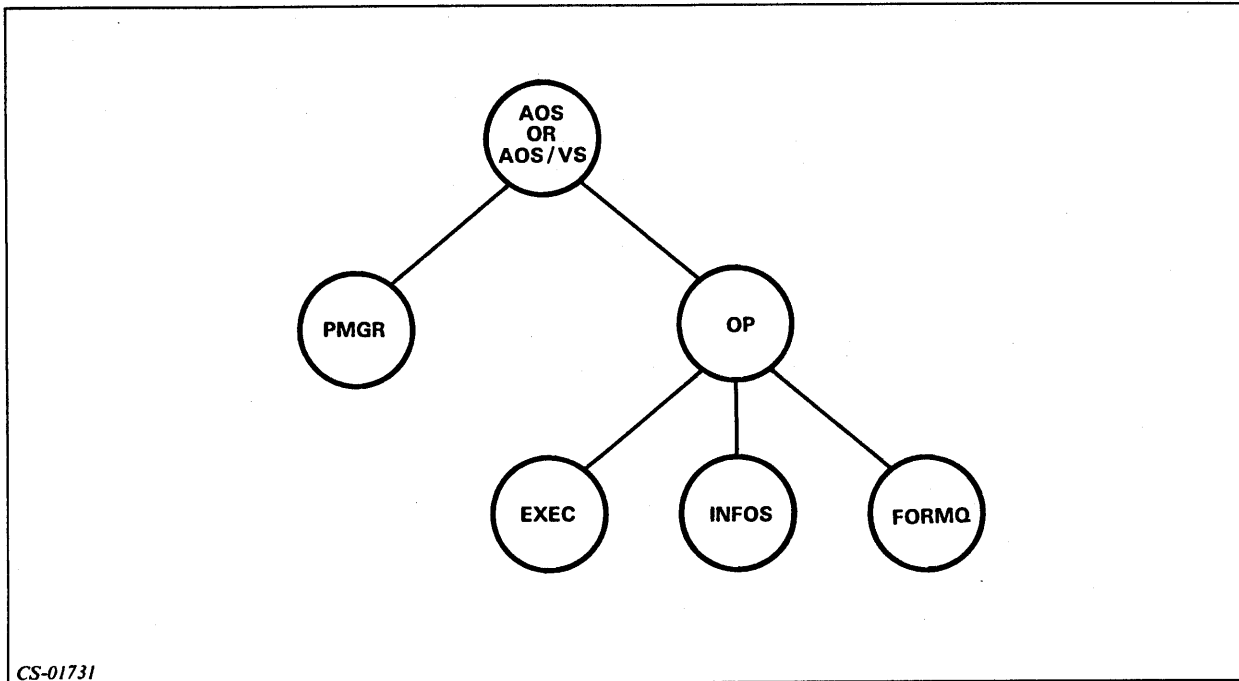


Figure 8.18

- 6. a.
- 7. a.
- 8. b.
- 9. c.
- 10. d.
- 11. TREE,14
- 12. TERMINATE,14
- 13. SUPERPROCESS,ON
- 14. WHO,13
- 15. PROCESS/BLOCK/IOC,GOOD\_STUFF
- 16. PROCESS MODULE\_EIGHT\_CONCLUSION

## Module 9

1. b.
2. b.
3. a.
4. a.
5. a.
6. 

```
CREATE/I QUIZ6_6
)EXECUTE PROG1
)WRITE ALL FINISHED
))
```
7. 

```
CREATE/I QUIZ6_7
)![EQUAL % 1%,1]
)EXECUTE PROG1
)![ELSE]
)EXECUTE PROG2
)![END]
```
8. 

```
SEARCHLIST [!SEARCHLIST]:UTIL
```
9. 

```
CREATE/I MAC1.CLI
)MAC2 [!EXPLODE[!TIME]]
))

CREATE/I MAC2.CLI
)WRITE % 7% % 8%
))
```

## Module 10

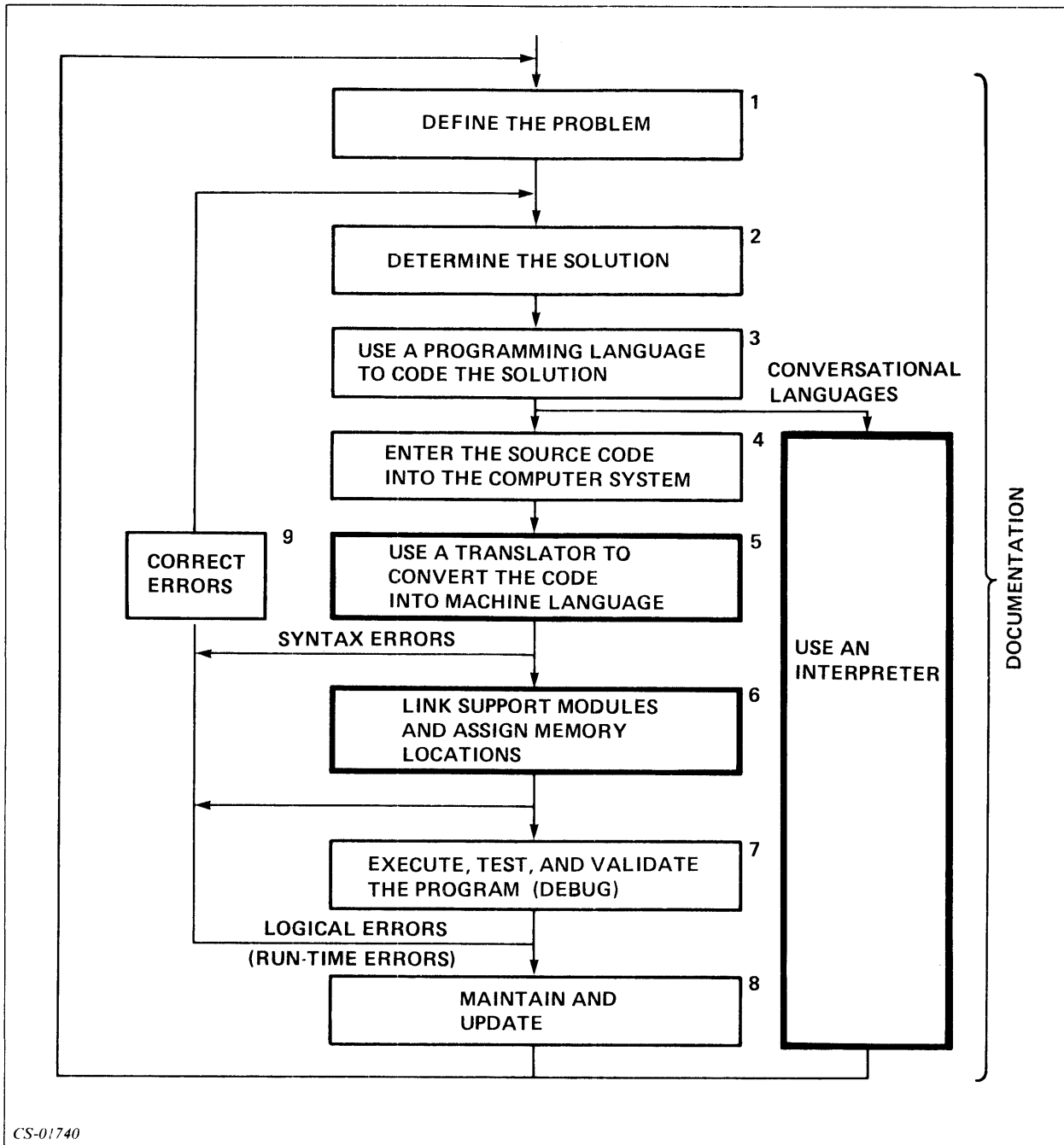
1. Invoke SPEED.
2. Open files for input.
3. Open files for output.
4. Read a portion of the file into the buffer.
5. Insert or modify text.
6. Write buffer to output.
7. Close files.
8. Exit SPEED.
9. Open, input
10. Search
11. L
12. T

13. Insert
14. FO
15. FU
16. Y, A
17. P, E
18. C

## Module 11

1. Compiler
2. Assembler
3. Translator
4. Text or source code
5. Interpreter
6. Link or bind
7. Object code
8. XEQ TEST1
9. For AOS: XEQ BIND TEST1  
For AOS/VS: XEQ LINK TEST1 or F5LD TEST1 (FORTRAN5)  
For COBOL:  
For AOS: CBIND TEST1  
For AOS/VS: CLINK TEST1
10. For COBOL: COBOL TEST1  
For FORTRAN: FORT4 TEST1 or F5 TEST1  
For ASSEMBLER: XEQ MASM TEST1

11.



CS-01740

Figure 11.21





# Appendix B

## AOS, AOS/VS Reference Manuals

### Advanced Operating System (AOS)

069-000016	<i>Introduction to the Advanced Operating System</i>
069-000018	<i>Learning to Use Your Advanced Operating System</i>
069-000020	<i>AOS Software Documentation Guide</i>
069-000030	<i>ECLIPSE® Data Systems: Software for Commercial Applications</i>
093-000120	<i>AOS Programmer's Manual</i>
093-000122	<i>Command Line Interpreter User's Manual (AOS, AOS/VS)</i>
093-000150	<i>AOS Console User's Handbook</i>
093-000190	<i>AOS Binder User's Manual</i>
093-000191	<i>AOS Shared Library Builder User's Manual</i>
093-000192	<i>AOS Macroassembler Reference Manual</i>
093-000193	<i>AOS System Manager's Guide</i>
093-000194	<i>AOS Operator's Guide</i>
093-000195	<i>AOS Debugger and File Editor User's Manual</i>
093-000197	<i>SPEED Text Editor User's Manual (AOS, AOS/VS)</i>
093-000198	<i>AOS Library File Editor User's Manual</i>
093-000217	<i>How to Load and Generate Your AOS System</i>
093-000218	<i>AOS LINEDIT Text Editor User's Manual</i>
093-000230	<i>Plastic ring binder for AOS Console User's Handbook</i>
093-000254	<i>AOS Link User's Manual</i>
093-000259	<i>Running AOS on Your ECLIPSE® MV/8000 Computer (MV/8000)</i>

### Advanced Operating System/Virtual Storage (AOS/VS)

093-000122	<i>Command Line Interpreter User's Manual (AOS,AOS/VS)</i>
093-000197	<i>SPEED Text Editor User's Manual (AOS,AOS/VS)</i>
093-000241	<i>AOS/VS Programmer's Manual</i>
093-000242	<i>AOS/VS Macroassembler Manual</i>
093-000243	<i>Managing AOS/VS</i>
093-000244	<i>AOS/VS Operator's Guide</i>
093-000245	<i>AOS/VS Link and File Editor User's Manual</i>
093-000246	<i>AOS/VS Debugger User's Manual</i>
093-000247	<i>AOS/VS System Management binder for 93-122, -243, -244</i>
093-000248	<i>AOS/VS SED Text Editor (6053) template for DASHER®D2</i>

- 093-000249 *AOS/VS SED Text Editor User's Manual*
- 093-000256 *AOS/VS SED Text Editor (6108/6109) template for DASHER® D200*
- 093-000276 *AOS/VS Debug/FED (6052/6053) template for DASHER® D2*
- 093-000278 *AOS/VS Debug/FED (6108/6109) template for DASHER® D200*

### **FORTRAN IV (AOS, RDOS, DOS, RTOS, SOS, \*CB)**

- 069-000029 *Data General's FORTRANs: A Technical Comparison*
- 093-000053 *FORTRAN IV User's Manual*
- 093-000068 *FORTRAN IV Runtime Library User's Manual (NOVA® )*
- 093-000142 *FORTRAN IV Runtime Library User's Manual (ECLIPSE® )*
- 093-000239 *FORTRAN QCALLs Reference Manual (AOS)*

### **FORTRAN IV (AOS, MP/OS)**

- 093-400004 *MP/FORTRAN IV Programmer's Reference*

### **FORTRAN 5 (AOS, AOS/VS, RDOS)**

- 069-000029 *Data General's FORTRANs: A Technical Comparison*
- 093-000085 *FORTRAN 5 Reference Manual*
- 093-000154 *FORTRAN 5 Programmer's Guide (AOS, AOS/VS)*
- 093-000227 *FORTRAN 5 Programmer's Guide (RDOS)*

### **BASIC (AOS, MP/OS)**

- 093-400005 *MP/BASIC Programmer's Reference*

### **Business BASIC (AOS, AOS/VS, RDOS, DOS)**

- 069-000028 *A Guide to Using Business BASIC*
- 093-000226 *Business BASIC Directory*
- 093-000228 *Business BASIC System Management*
- 093-000212 *File/Screen Maintenance 8-key template for DASHER® D2*
- 093-000213 *File/Screen Maintenance 3-key template for DASHER® D2*
- 093-000265 *Business BASIC 15-key template for DASHER® D200*

### **Commercial BASIC (CB)**

Model 3896 includes DOS and Business BASIC manuals.

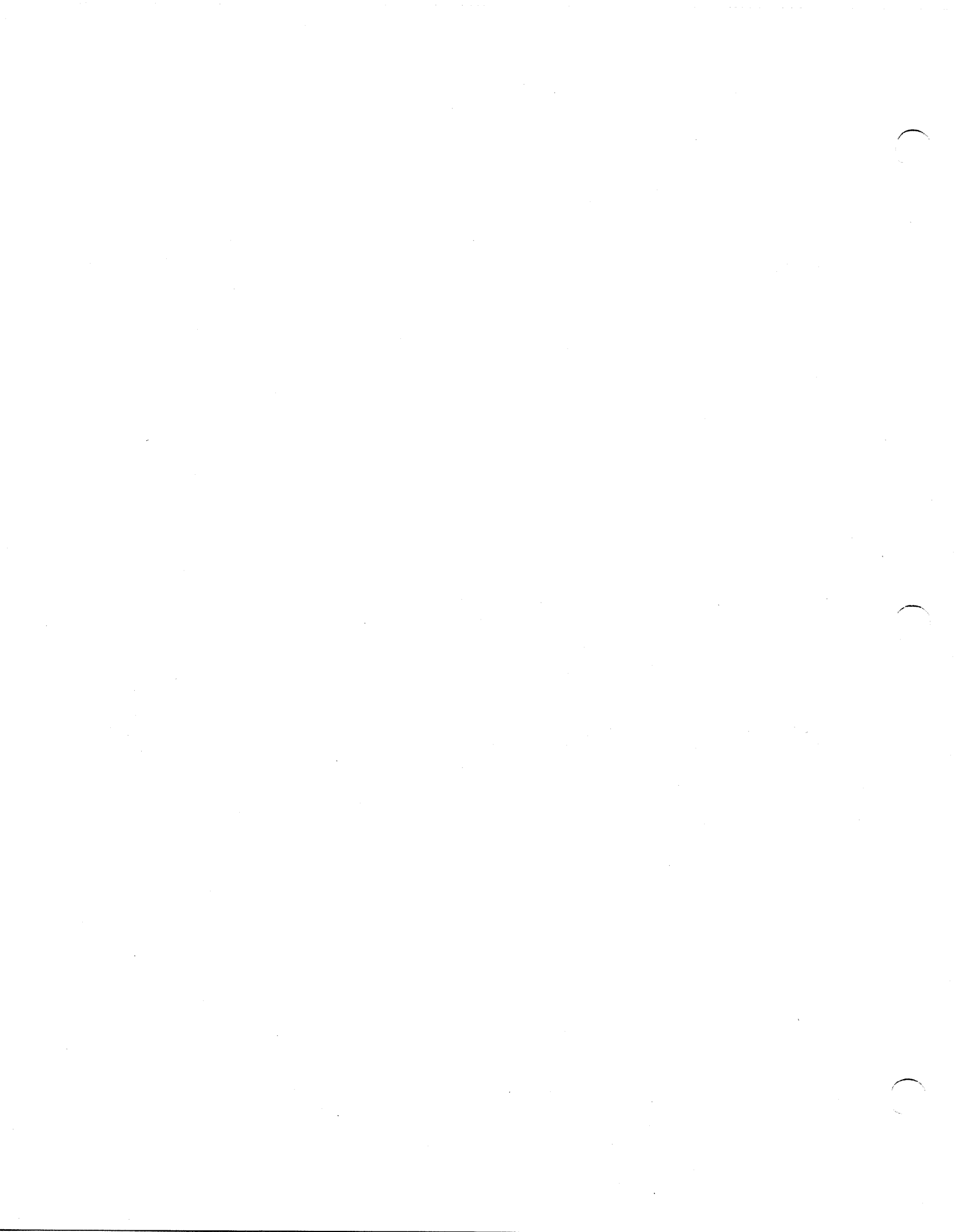
Model 3897 includes RDOS, RDOS Sort/Merge, and Business BASIC manuals.

### **Extended BASIC (AOS, AOS/VS, RDOS, DOS, SOS\*)**

- 069-000003 *basic BASIC (AOS, AOS/VS, RDOS, DOS)*
- 093-000065 *Extended BASIC User's Manual*
- 093-000119 *Extended BASIC System Manager's Guide*

**COBOL (AOS, AOS/VS, RDOS)**

093-000180	<i>COBOL Reference Manual (RDOS)</i>
093-000200	<i>COBOL Pocket Reference (RDOS)</i>
093-000223	<i>COBOL Reference Manual (AOS, AOS/VS)</i>



# Addendum

## The SED Editor

### Introduction

The SED Editor Addendum is a self-paced text only module that explains the elementary features of the SED text editor. Although not all the features of SED are discussed, this addendum allows you to use enough of the features to develop and edit files for use in program development. For a more detailed discussion of all the features, refer to your *SED Text Editor User's Manual*.

### Module Objectives

Upon successful completion of this addendum, you should be able to:

1. List the steps involved in a typical editing session.
2. Use arguments to SED commands in a SED editing session.
3. Use the SED commands that perform the following functions:
  - a. Add text to a file
  - b. Modify text in a file
  - c. Change the line position
  - d. Locate text in a file
  - e. Move and duplicate text
  - f. Delete text
  - g. Update files
  - h. Access the CLI
  - i. Terminate SED.

### Resources

You only need this addendum; there is no accompanying audiotape.

## Module Outline

The SED Editor Addendum discusses the following topics:

1. SED Overview
  - Introduction to SED
  - Aids for using SED
2. Arguments to SED Commands
  - Address
  - Range
  - Source
  - Destination
  - Searchstring
3. SED Commands
  - Adding Text
    - APPEND
    - INSERT
  - Modifying Text
    - MODIFY
    - REPLACE
    - SUBSTITUTE
    - CUT
    - PASTE
    - SPLIT
    - JOIN
  - Displaying Information
    - HELP
    - DISPLAY\_STATUS
    - LIST
    - VIEW
  - Locating Text
    - POSITION
    - FIND
    - BACKFIND
  - Moving and Duplicating Text
    - MOVE
    - DUPLICATE
  - Deleting Text
    - DELETE
    - UNDO
  - Accessing the CLI
    - DO
    - CLI
  - Updating Files
    - SAVE
    - BYE

## SED Overview

### Introduction to SED

SED is a screen-oriented text editor, which allows you to create and modify text files from your terminal. SED is known as a screen-oriented text editor because it deals with lines, pages, and screens of text.

SED can be used with most Data General Corporation terminals. On a DASHER<sup>®</sup> D1 terminal, however, it can be used only with certain limitations. The DASHER D1 terminal is uppercase only and does not have all the special function keys that SED allows.

Figure 1 illustrates an overview of a typical SED editing session. The first step is to execute the SED utility from your working directory. The second step is to create a new file or bring in an old file for modifications. The third step is to make the modifications using SED features. These SED features are explained later in this addendum. The last step is to update the file and terminate the SED session and return to the CLI.

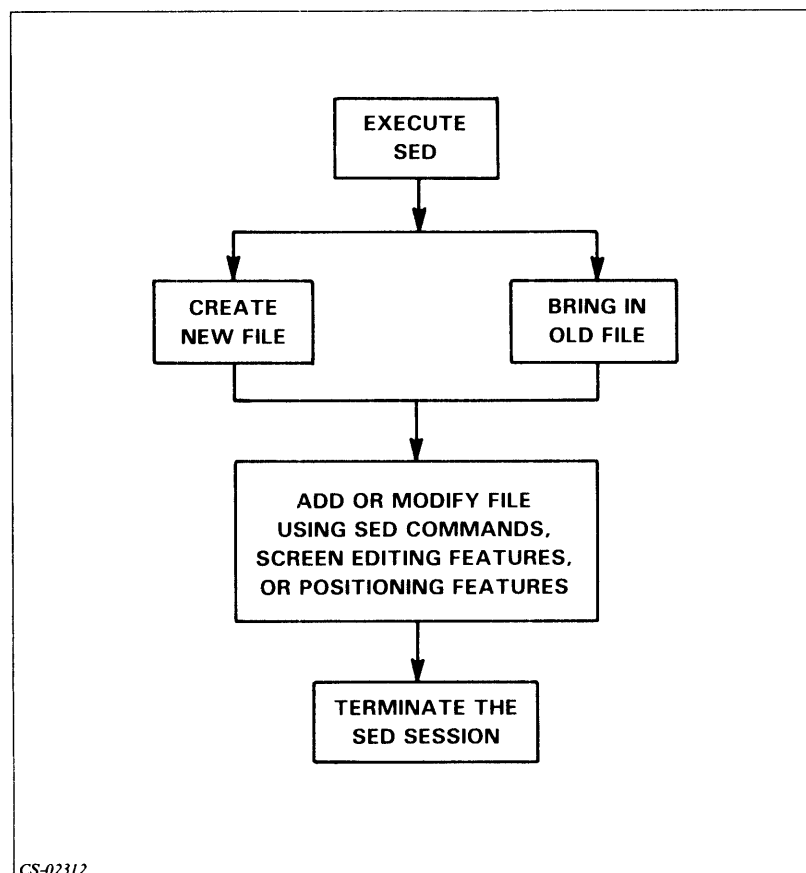


Figure 1 SED Editing Overview

Now let us look at an actual editing session. There are two ways to execute SED. You can give the CLI command, `xed sed`, *with* or *without* specifying a file to be edited.

If you give the command, `xed sed`, *without* specifying a file, SED asks for the file. You must respond with a file pathname. The pathname may specify an existing file or a new one to be created by the SED utility. The file pathname must follow the CLI naming conventions. Example 1 shows how to execute SED to edit a file called `samplefile` without specifying the file.

#### Example 1

```
) xed sed
```

```
Name of the file to edit: samplefile
```

If you give the command `xed sed` *with* a filename, SED will search for the file. If the file exists, the editing session begins. If the file does not exist, then SED will ask if you wish to create the file. If you answer `y` or `yes`, then the file will be created and the SED editing session begins. If you answer `n` or `no`, then you can enter another pathname. Example 2 shows how to execute SED with a filename of a newly created file called `samplefile`.

#### Example 2

```
)xed sed samplefile
```

```
Do you want SAMPLEFILE to be created? y
```

When SED is ready to begin the editing session, it displays its prompt, an asterisk (\*), at the top of your screen above a line. You can now use SED commands, screen editing features, and screen positioning features to add to or modify your file.

There are two modes you may use during a SED editing session: the keyboard input mode and the command mode. In the keyboard input mode you enter text into the file. SED displays a cursor to mark where text entries start; this is called the current position. In the command mode you can issue the SED commands that modify files. The last command given is displayed next to the SED asterisk prompt and is called the current command. You start a SED session in the command mode, but will ordinarily move back and forth between modes several times in a session.

Figure 2 shows the CRT display from a short SED editing session. SED displays two commands: the current command and the command given prior to the current command. The current SED command is displayed next to the asterisk above the line. The command given immediately prior to the current command is shown above the current command. In Figure 2 the current command is `APPEND`. The `APPEND` command allows you to input text into the file. The command given immediately prior to the `APPEND` command is the `VIEW` command. The `VIEW` command is explained later. The current position line and page is shown above the line to the right. Text is entered below the line. SED will automatically number lines as the text is entered.

To conclude an appending session press the `ESCAPE` key, `<ESC>`. Press the `ESCAPE` key only after entering a `NEW-LINE`, `<NL>`, at the end of the text; otherwise, the last line of text will be lost. The SED prompt will return. To terminate the SED session and return to the CLI type the command, `BYE`, after the SED prompt.



```

view
*append
page 1 line 1
1  this is the text.
2

```

Figure 2 Short Editing Session

Example 3 demonstrates how to use SED to create a new file called `text` with the message **This is the text.** in it.

### Example 3

<code>)xeq sed text</code>	Starts SED naming the file <code>text</code> to be edited.
<code>Do you want TEXT to be created? y</code>	Creates a new file called <code>text</code> .
<code>*append</code>	Allows you to enter text into the file.
<code>This is the text. &lt;NL&gt;</code>	Entered text is placed at the current line.
<code>&lt;ESC&gt;</code>	Ends APPEND session and returns you to command mode.
<code>*bye</code>	Terminates SED session and returns you to the CLI.
<code>)</code>	

This section has introduced the SED editor and described a simple editing session. Answer the questions in Exercise 1 and do Lab Activity 1 before continuing with the rest of the addendum.



**Exercise 1**

1. List the four steps in a typical SED editing session.
  - a.
  - b.
  - c.
  - d.
2. Match each command with its effect.
  - a. XEQ SED
  - b. APPEND
  - c. BYE
  1. \_\_\_\_\_ Ends SED session.
  2. \_\_\_\_\_ Initiates SED session.
  3. \_\_\_\_\_ Used to enter text.

**Directions**

Now check your answers with those on the next page.

## Exercise 1

### Answers

1.
  - a. Execute SED.
  - b. Create a new file or bring in an old file for editing.
  - c. Add or modify a file.
  - d. Terminate the session.
2.
  1. c
  2. a
  3. b

### Directions

If you answered all the questions correctly, continue to the next page. Otherwise, review the material and do this exercise again before you continue.

## Lab Activity 1

### Directions

Follow the instructions below to create a new file called **text** with the message **this is the text.** in it.

1. Execute SED from your working directory.  
`)xeq sed text`
2. Create a new file called **text** by answering **y** for **yes** to SED's request.  
`Do you want TEXT to be created? y`
3. The SED screen will appear. The asterisk indicates the location where commands are displayed. Issue the APPEND command to enter text.  
`*append`
4. The number 1 will brighten indicating the position of the current line. Text entered at the keyboard will be placed in the current line. Enter the text, **this is the text.**, followed by `<NL>`. After pressing `<NL>` a bright 2 will appear on the left to indicate that line 2 is now the current line.  
`1 this is the text. <NL>`  
`2`
5. Press `<ESC>` to end APPEND mode. Do not press `<ESC>` at the end of a line of text. Always enter the text first by pressing `<NL>`; otherwise, the last line of text will be lost.  
`<ESC>`
6. Terminate the SED session by entering the BYE command. You will return to your working directory and the CLI prompt will appear.  
`*bye`  
`)`

### Directions

After you have successfully completed Lab Activity 1, continue to the next page.



**Table A** Cursor and Line Control Characters

Control Character	What It Does
CTRL-A	Moves the cursor to the end of the line. In command mode, repeats the last command issued.
CTRL-B	Moves cursor backward to the last letter in each word.
CTRL-E	Lets you insert one or more characters before the cursor. Pressing <NL> or CTRL-E terminates the insert mode and displays the edited line.
CTRL-F	Moves the cursor forward to the first letter of each word.
CTRL-H	Moves the cursor to the beginning of the line.
CTRL-K	Erases all characters to the right of the cursor.
CTRL-X →	Moves the cursor to the right one character.
CTRL-Y ←	Moves the cursor to the left one character.
CTRL-U	Deletes all the characters in the line.
CTRL ↑ TAB key	Moves the cursor to the next tab stop; tab stops are in columns 9, 17, 25, 33, 41, 49, 57, 65, and 73.
DEL	Erases the character to the left of the cursor and closes up the line.
ESC	Terminates APPEND, INSERT, MODIFY, or REPLACE modes. If you press <ESC> before you press <NL>, you will enter the line with no changes.

Figure 2 shows the usual or default format settings for a SED editing session. Unless otherwise formatted SED will:

- Make the current line bright and all others dim.
- Display line numbers before each line of text.
- Move the cursor up or down one line in the same column when you press the up arrow or down arrow keys.
- Show 21 lines of text on the screen, 10 lines before and 10 lines after the current line.
- Remove any blanks at the end of a line after you press <NL>.

You can find out the present settings for your system by issuing the DISPLAY command from the command mode. Your SED manual describes how you can change the format settings using the SET and CLEAR commands.

The HELP command gives you information about all SED commands and vocabulary. If you type HELP while in the command mode, SED will display a table of commands and keywords. If you type HELP followed by a command or keyword, SED will display information about that word. The use of the HELP command is shown in Figure 4.

```

help
*
page 1 line 1
-----
***** C O M M A N D S *****
-----
ESCAPES  ADD TEXT  CHANGE TEXT  DELETE TEXT  LISTINGS  POSITIONING
-----
EXECUTE  APPEND    MODIFY      DELETE      LIST      POSITION
HELP     INSERT    REPLACE     MOVE        VIEW      FIND
SAVE     DUPLICATE  SUBSTITUTE  JOIN        PRINT
          UNDO      SPLIT
          CUT
          PASTE

EXITING  MISC      HELP WORDS
-----
ABANDON  CLEAR     CURSOR__CONTROL  ADDRESS
BYE      DIRECTORY RANGE          SOURCE
CLI      DISPLAY   SEARCH__STRING  DESTINATION
DO       SET       KEYS           SYNTAX
          SPELL     SWITCHES

Type HELP followed by a command or keyword you want to know about.

```

Figure 4 The HELP Command

The SPELL command provides spelling assistance during an editing session. SED lists words that begin with a string of letters you specify. Figure 5 shows the use of the SPELL command to determine the spelling of *liaison* when only the first three letters are known.

```

spell lia
*
page 1 line 1
-----
liabilities
liability
liable
laise
liaison
liaisons
liana
liane
liang
liangs
lianoid
liar
liars
lias
- No More -

```

Figure 5 The SPELL Command



## Command Abbreviations

SED allows you to abbreviate any command if your abbreviation is unique. For example, SED will recognize any of the following abbreviations as the APPEND command:

```
AP
APP
APPE
APPEN
```

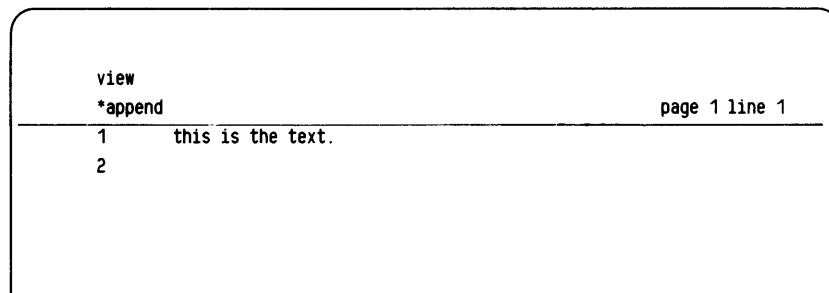
APPEND cannot be abbreviated to "A" however, since there is another SED command, ABANDON, which begins with A.

## Lab Activity 2

### Directions

Follow the instructions below to practice using the HELP and SPELL commands and command abbreviations.

1. Execute SED from your working directory.  
    `)xeq sed text`
2. The SED screen will appear with the text you have previously entered in Lab Activity 1. Figure 6 shows the screen.



```
view
*append                                page 1 line 1
1  this is the text.
2
```

Figure 6 SED Screen

3. Enter the **HELP** command. SED displays a table of commands and keywords about which you can get more information. Figure 7 shows the display.

```

help
*
page 1 line 1
-----
***** C O M M A N D S *****

ESCAPES  ADD TEXT  CHANGE TEXT  DELETE TEXT  LISTINGS  POSITIONING
-----  -----  -----  -----  -----  -----
EXECUTE  APPEND    MODIFY      DELETE      LIST      POSITION
HELP     INSERT    REPLACE     MOVE        VIEW      FIND
SAVE     DUPLICATE  SUBSTITUTE  JOIN        PRINT
          UNDO
          SPLIT
          CUT
          PASTE

EXITING  MISC      HELP WORDS
-----  ----  -----
ABANDON  CLEAR    CURSOR__CONTROL  ADDRESS
BYE      DIRECTORY  RANGE            SOURCE
CLI      DISPLAY   SEARCH__STRING  DESTINATION
DO       SET       KEYS             SYNTAX
          SPELL   SWITCHES

Type HELP followed by a command or keyword you want to know about.

```

Figure 7 The HELP Display

4. Type **help spell**. SED displays information on the use of the **SPELL** command. Figure 8 shows the use of the **HELP** command to get information on the command, **SPELL**.

```

help spell
*
page 1 line 1
-----
SPELL <STRING>

THE SPELL COMMAND WILL TRY TO OPEN SED.DICTIONARY.
THIS MUST BE A LIST OF WORDS, EACH WORD IN LOWER CASE
FOLLOWED BY A NEW LINE. THIS LIST SHOULD BE SORTED
IN ALPHABETICAL ORDER. SED.DICTIONARY COULD BE A
LINK TO A FILE CONTAINING THE WORD LIST.

GIVEN THIS LIST, THE SPELL COMMAND WILL SEARCH THE
WORD LIST FOR WORDS BEGINNING WITH 'STRING'. 'STRING'
MAY BE UPPER OR LOWER CASE.

```

Figure 8 HELP SPELL

- Determine the spelling of the word *abbreviation* using only the first three letters of the word by typing **spell abb**. Figure 9 shows SED's response.

```
spell abb
*
page 1 line 1
-----
abbacy
abbasid
abbatial
abbe
abberations
abness
abbevillian
abbey
abbot
abbott
abbreviate
abbreviated
abbreviates
abbreviating
abbreviation
abbreviations
abbreviator
abby
- No More -
```

Figure 9 SPELL Abb

- Demonstrate that SED will allow the use of command abbreviations by typing **he** instead of **help**. SED will recognize **he** as **help** and display the HELP screen as in Figure 7.
- Try to type in the abbreviation **sp** for **spell**. SED will respond with the message:  
Command not unique, correct the command:  
SED will not accept **sp**, because it is not a unique abbreviation. There are two commands **SPLIT** and **SPELL** that begin with **SP**.
- Terminate the SED session by typing **bye**. You will return to the CLI.

### Directions

After you have successfully completed Lab Activity 2, continue to the next page.

## Arguments to SED Commands

SED commands tell the computer what actions to take. Arguments to SED commands tell the computer what to act on. You can specify page numbers, line numbers, searchstrings, or pathnames as arguments to certain SED commands. Five categories of arguments: address, range, source, destination, and searchstring are described below.

### Address

Address specifies a particular line or page. It consists of an optional word, PAGE or LINE, followed by a modifier. If you omit the word PAGE, SED searches for the LINE within the current page. The modifier can be a number without a plus or minus sign, or one of the following keywords: CURRENT, PREVIOUS, NEXT, or LAST. If you omit the number or keyword modifier, SED assumes the current line (or page). Example 4 shows several uses of the address argument with the SED command, POSITION, which sets the current line according to the specified address.

#### Example 4

POSITION PAGE 3	Set current line to line 1 of page 3.
POS LINE 47	Set current line to line 47 of current page.
POS PAGE LAST	Set current line to line 1 of last page.
POS PAGE NEXT	Set current line to line 1 of next page.
POS 2	Set current line to two lines previous.
POS 5	Set current line to line 5 on current page.

### Range

Range specifies a contiguous group of lines or pages. You can enter a range in numbers or with the keywords ALL, REMAINING, TO or FOR. The range argument may be formatted in the following ways:

```
ALL [pages]
REMAINING [pages]
address
address [TO] address
address [FOR] address
```

Example 5 shows several uses of the range argument with the SED command, DELETE, which removes text from the file.

#### Example 5

DELETE ALL	Delete all lines on the current page.
DELETE REMAINING	Delete all lines from the current line to the last line on the current page.
DELETE 3	Delete line 3.

**DELETE 3 7** Delete lines 3 through 7 in current page.  
**DELETE 4 FOR 3** Delete lines 4, 5, and 6.

## Source

Source specifies a particular place from which to get text. If you specify a pathname, it must already exist and you must precede it with **FROM**. If you omit a pathname, SED uses your keyboard as a default source for text. Example 6 shows uses of the source argument with the **APPEND** command.

### Example 6

**APPEND FROM OTHERFILE** Append the entire contents of file **OTHERFILE** to the end of the current page.  
**APPEND** Append text typed from the keyboard source to the end of the current page.

## Destination

Destination specifies a new location for a block of text. It has two forms:

1. Command range **ONTO** pathname  
Places a block of text onto the file specified by the pathname. If the file does not exist, SED will create it.
2. Command range (**BEFORE** or **AFTER**) address  
Places a block of text before or after the specified address.

Example 7 shows the use of the destination argument with the command **MOVE**, which moves text within or between files.

### Example 7

**MOVE 3 10 ONTO FILEA** Move lines 3 through 10 into the file, **FILEA**, creating the file if it does not exist.  
**MOVE 9 BEFORE 5** Move line 9 before line 5 in the current page.

## Searchstring

Searchstring specifies a particular word or phrase in the text. You must enclose the searchstring in quotation marks if it contains blank spaces, symbols other than a dollar sign or underscore, or if you want to distinguish between uppercase and lowercase in your search. You can enclose the searchstring in either quotation marks or apostrophes. Use apostrophes if the searchstring contains quotation marks, and vice versa. Example 8 shows the use of the searchstring argument with the commands `FIND` and `BACKFIND`, which find a text string after and before the current line, respectively.

### Example 8

<code>FIND EDITOR</code>	Locate the searchstring <code>EDITOR</code> after the current page.
<code>BACKFIND "this is it"</code>	Locate the searchstring "this is it" before the current line.

## Exercise 2

1. Match each argument with its use.
  1. Address \_\_\_\_\_ a. Specifies a new location for a block of text.
  2. Range \_\_\_\_\_ b. Specifies a particular line or page.
  3. Source \_\_\_\_\_ c. Specifies a particular word or phrase.
  4. Destination \_\_\_\_\_ d. Specifies a particular place to get text from.
  5. Searchstring \_\_\_\_\_ e. Specifies a contiguous group of lines or pages.
  
2. What SED command do you issue to get information on the SPELL command displayed on your terminal?
  
3. Write the SED command to find the correct spelling of *ardvark* if you only know that it begins with AA.

### Directions

Check your answers with those on the next page.

## Exercise 2

### Answers

1. 1. b
  2. e
  3. d
  4. a
  5. c
- 
2. help spell
  3. spell aa

### Directions

If you answered all the questions correctly, continue to the next page. Otherwise, review the material and do this exercise again before you continue.



## SED Commands

SED contains many commands that allow you to easily create and edit text. Table B lists several SED commands by category. Lab Activity 3 will then take you step-by-step through an example illustrating the use of each command. Review the table and then try the lab. Do not try to understand all the commands in the table your first time through. The lab will give you practice with each command. For further information on SED commands use the HELP utility or the SED manual.

**Table B SED Commands**

Command	Description
<b>Adding Text</b>	
APPEND [FROM pathname] [range]	Opens a file for editing.
INSERT [address] [FROM pathname] [range]	Inserts text before a location in the page. Text comes from the terminal or another file.
<b>Modifying text</b>	
MODIFY [range]	Revises a line or range of text that you retype.
REPLACE [range]	Deletes text and replaces it with new text that you type at the terminal.
SUBSTITUTE "searchstring" [FOR] "searchstring" [[IN] range]	Substitutes a word or phrase for another word or phrase throughout a range of text.
CUT address column	Splits a line into two lines.
PASTE range [OR] address	Merges a range of lines into a single line.
SPLIT [address]	Sets a page break in the file.
JOIN [address]	Removes a page break in the file.
<b>Displaying information</b>	
HELP [word]	Displays information about commands and keywords on the terminal screen.
DISPLAY_STATUS	Displays file status information.
LIST [range]	Lists a range of text on the terminal screen.
VIEW [number]	Lists a range of text surrounding the current line of the terminal screen.
<b>Locating text</b>	
POSITION address	Moves the current line position to an address in the file.
FIND "searchstring" [[IN] range]	Finds a word or phrase in a range of text beginning at any line on the page and working forwards.
BACKFIND "searchstring"	Finds a word or phrase in a page of text beginning at the line before the current line and working backwards.
<b>Moving and duplicating text</b>	
MOVE [BEFORE address] [AFTER address] [ONTO pathname]	Moves text from one location in the page to another, or onto another file.
DUPLICATE [BEFORE address] [AFTER address] [ONTO pathname]	Copies text from one location in the page to another, or onto another file.
<b>Deleting and restoring text</b>	
DELETE [range]	Removes a range of text from the file.
UNDO	Restores the most recently deleted text to the file.
<b>Using CLI commands</b>	
DO CLI-command	Executes a CLI command and then returns to the editing session.
CLI	Temporarily suspends editing session, creates a new process, SON OF SED, and allows you to execute CLI commands.
<b>Closing or updating files</b>	
SAVE	Updates a copy of the file, including changes, without concluding the session.
BYE	Concludes the editing session and updates the file with the new changes.

## Directions

This concludes the text of this addendum. Now try Lab Activity 3 on the next page.

### Lab Activity 3

Follow the directions below to create and modify a text file using SED commands. The figures show the effect of the instructions.

1. Invoke SED from the CLI and create a file called `samplefile`.  
`)xeq sed samplefile`  
 and then respond `y` to the SED prompt.

```

SED Rev 1.65.00.00;Input file - SAMPLEFILE
Do you want SAMPLEFILE to be created? Y

```

Figure 10 Create Samplefile

2. Use the APPEND command to open the file for adding text.  
`*append`
3. SED will display the line number 1. Enter the text as shown in Figure 11.

```

view
*append
page 1 line 1
1 this is line 1.
2 this is line 2.
3 this is line 3.
4 this is line 4.
5 this is line 5.

```

Figure 11 Enter Text

4. Press `<NL>` following the last line of text.
5. End the APPEND session by pressing `<ESC>`. Do not press `<ESC>` however, until you have pressed `<NL>` following the last line of text; otherwise, you will lose the last line of text. If you make a typing mistake before you press `<ESC>` you may use the cursor, line control characters and the arrow keys to move the cursor to the mistake. Then you may type over the mistake, use the DEL key, or insert text using CTRL-E. If you see a mistake after pressing `<ESC>`, reenter the keyboard input mode by using the APPEND command and then exit again using the `<ESC>` key.

6. Add a new line between lines 2 and 3 by using the INSERT command. You must specify the address where the new text is to be placed. SED will displace the text from that line until you have finished the insert. Insert the text **this is the new line 3.** and then end the insert session by pressing <ESC>. (Do not forget the <NL> following the text.)

```
insert 3
*
page 1 line 4
1  this is line 1.
2  this is line 2.
3  this is the new line 3.
4  this is line 3.
5  this is line 4.
6  this is line 5.
```

Figure 12 INSERT Command

7. The INSERT command can also be used to add a specified range of text from another file. Between lines 3 and 4 add the text from the file, text, which you created earlier.

```
insert 4 from text
*
page 1 line 5
1  this is line 1.
2  this is line 2.
3  this is the new line 3.
4  this is the text.
5  this is line 3.
6  this is line 4.
7  this is line 5.
```

Figure 13 INSERT Text

- If you want to change text, you may use the **MODIFY** command followed by the address of the line to be modified. Use the **MODIFY** command to change the "4" in line 6 to a "6." Note that the **MODIFY** command places the cursor at the beginning of the line. To change the 4 you must move the cursor to it and type 6. To move to the 4 you may use the arrow key or press **CTRL-F** three times.

```
insert 4 from text
*modify 6                                     page 1 line 5
-----
1  this is line 1.
2  this is line 2.
3  this is the new line 3.
4  this is the text.
5  this is line 3.
6  this is line 6.
7  this is line 5.
```

Figure 14 MODIFY Command

- Delete line 5 and replace it with the line, **this is replacement line 5.** using the **REPLACE** command.

```
replace 5
*                                             page 1 line 6
-----
1  this is line 1.
2  this is line 2.
3  this is the new line 3.
4  this is the text.
5  this is replacement line 5.
6  this is line 6.
7  this is line 5.
```

Figure 15 REPLACE Command

10. Capitalize the first letter of each instance of the word, "this," using the SUBSTITUTE command. The first argument, the replacement searchstring, must be in quotation marks, since you want to distinguish between uppercase and lowercase. The second argument, the searchstring, which is searched for does not have to be in quotation marks, since you are not concerned with which case it is in before replacement.

```
substitute 'This' for this all
*
page 1 line 7
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is the text.
5 This is replacement line 5.
6 This is line 6.
7 This is line 5.
```

Figure 16 REPLACE Command

11. Line 4 may be cut into two lines using the CUT command. You may cut the line using the CUT command followed by the line number and column number as an argument or, in the keyboard input mode, move the cursor to the point where you want to cut the line and press the CUT function key. To cut line 4 into two lines between "this" and "is" requires the arguments 4 and 5. The argument "4" refers to line 4. The argument "5" refers to the location of the cut, column 5.

```
cut 4 5
*
page 1 line 4
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This
5 is the text.
6 This is replacement line 5.
7 This is line 6.
8 This is line 5.
```

Figure 17 CUT Command

12. Lines 4 and 5 can be merged into one line using the PASTE command. You can paste lines in two ways: you can type PASTE followed by the number of the first of two lines you want to paste or the range of lines you want to paste; or you can make the first of two lines to be pasted the current line and press the PASTE function key.

```
paste 4 5
*
page 1 line 4
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is the text.
5 This is replacement line 5.
6 This is line 6.
7 This is line 5.
```

Figure 18 PASTE Command

13. Page breaks can be created using the SPLIT command. To cause a page break between lines 3 and 4 use the SPLIT command with an address argument. The page break occurs at the line before the address. The address then becomes the first line of the new page.

```
split 4
*
page 2 line 1
1 This is the text.
2 This is replacement line 5.
3 This is line 6.
4 This is line 5.
```

Figure 19 SPLIT Command

14. Create a third page beginning with line 3.

```

split 3
*
page 3 line 1
-----
1 This is line 6.
2 This is line 5.

```

Figure 20 Third Page

15. Page breaks may be removed by the JOIN command. JOIN requires that the current line position be at line 1 of the page you wish to join to the preceding page. You can set this position by moving to it with cursor control characters or by using "1" as an argument to JOIN. Merge pages 2 and 3 by issuing the JOIN command from line 1 of page 3.

```

join
*
page 2 line 3
-----
1 This is the text.
2 This is replacement line 5.
3 This is line 6.
4 This is line 5.

```

Figure 21 JOIN Command

16. You can change the current line to any line or page in the file using the POSITION command. If you give a line location as an argument, that line becomes the current line. If you give a page location, the first line of that page becomes the current line. Move to page 1 by issuing the POSITION command with page 1 as the argument.

```

position page 1
*
page 1 line 1
-----
1 This is line 1.
2 This is line 2.
3 This is the new line 3.

```

Figure 22 POSITION Command



17. Move to page 2 with the POSITION command.

```
po page 2
*
page 2 line 1
1 This is the text.
2 This is replacement line 5.
3 This is line 6.
4 This is line 5.
```

Figure 23 Move to Page 2

18. Merge pages 1 and 2 with the JOIN command.

```

join
*
----- page 1 line 4
1   This is line 1.
2   This is line 2.
3   This is the new line 3.
4   This is the text.
5   This is replacement line 5.
6   This is line 6.
7   This is line 5.

```

Figure 24 Merge Pages 1 and 2

19. SED file status information can be shown using the DISPLAY\_STATUS command. For details on the displayed items consult the SED manual.

```

display_status
*
----- page 1 line 4
Edit File Name - :UDD:SARAH.S209:SAMPLEFILE

Current Page:    1      Last Page:    1
Current Line:   4      Last Line:    7

View range:     10
Display mode:   0

Typer mode:     ON
Blank mode:     OFF
Preload old line: ON
Skip white space: ON

Upper mode:     OFF
Escape mode:    OFF

New_line is view: ON
Escape is view:  ON

Line numbers being displayed

```

Figure 25 DISPLAY\_STATUS Command

20. Portions of a file may be displayed on the screen using either the LIST or VIEW commands. The LIST command displays a range of text on your terminal screen. The range must be within a page. Issue the LIST command to display lines 3, 4, and 5.

```
list 3 5
*
page 1 line 4
3 This is the new line 3.
4 This is the text.
5 This is replacement line 5.
```

Figure 26 LIST Command

21. The VIEW command displays a screen of lines around the current line. Issue the VIEW command to display the entire text.

```
view
*
page 1 line 4
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is the text.
5 This is replacement line 5.
6 This is line 6.
7 This is line 5.
```

Figure 27 VIEW Command

22. A given word or phrase in text may be located by the FIND or BACKFIND commands. Text after the current line may be located using the FIND command. Text before the current line may be located using the BACKFIND command. Use the FIND command to locate the word *replacement*.

```
find replacement
*
page 1 line 5
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is the text.
5 This is replacement line 5.
6 This is line 6.
7 This is line 5.
```

Figure 28 FIND Command

23. Use BACKFIND to locate the phrase *new line*. Remember that a phrase searchstring must be enclosed in quotation marks.

```
backfind "new line"
*
page 1 line 3
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is the text.
5 This is replacement line 5.
6 This is line 6.
7 This is line 5.
```

Figure 29 BACKFIND Command

24. The **MOVE** command lets you move blocks of text. The lines specified are deleted from their original location and inserted at the destination. The destination can be within the current page or another file altogether. To move the text from one page to another, you must move the text onto a new file and then insert that text into the new destination using the **INSERT** command. Move lines 6 and 7 before line 4.

```
move 6 7 before 4
*
page 1 line 6
-----
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is line 6.
5 This is line 5.
6 This is the text.
7 This is replacement line 5.
```

Figure 30 MOVE Command

25. To copy text to the same page or to the end of another file without deleting the original text use the **DUPLICATE** command. Duplicate lines 2 and 3 following line 6.

```
duplicate 2 3 after 6
*
page 1 line 3
-----
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is line 6.
5 This is line 5.
6 This is the text.
7 This is line 2.
8 This is the new line 3.
9 This is replacement line 5.
```

Figure 31 DUPLICATE Command

26. Text may be removed with the DELETE command. Delete lines 7 through 9.

```
delete 7 9
*
page 1 line 6
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is line 6.
5 This is line 5.
6 This is the text.
```

Figure 32 DELETE Command

27. Delete line 5.

```
delete 5
*
page 1 line 5
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is line 6.
5 This is the text.
```

Figure 33 Delete Line 5

28. The last block of text deleted may be restored with the UNDO command. Undo the last DELETE command.

```
undo
*
page 1 line 5
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is line 6.
5 This is line 5.
6 This is the text.
```

Figure 34 UNDO Command

29. UNDO restores only the last DELETE command. Entering UNDO twice in a row results in an error message.

```

No deleted text to 'UNDO', correct the command:
*undo                                     page 1 line 5
-----
1      This is line 1.
2      This is line 2.
3      This is the new line 3.
4      This is line 6.
5      This is line 5.
6      This is the text.

```

Figure 35 UNDO Error

30. The DO and CLI commands allow you to execute CLI commands leaving SED. The DO command invokes a subordinate CLI process to execute the command, then terminates this CLI process and returns the SED prompt. Use the DO command to execute the CLI FILESTATUS command.

```

do filestatus
*                                     page 1 line 5
-----
DIRECTORY :UDD:SARAH.S209

SAMPLEFILE.S2   TEXT          SAMPLEFILE   SAMPLEFILE.SC
TEXT.ED         SAMPLEFILE.ED

```

Figure 36 DO Command

31. The CLI command temporarily suspends the editing session and creates a CLI process called SON OF SED. Issue the CLI command.

```

cli
*                                     page 1 line 5
-----
AOS/VS CLI REV 01.64.255.255 11-APR-83   15:25:28
Son of Sed)

```

Figure 37 CLI Command

32. From SON OF SED issue the FILESTATUS command.

```

c11
*
page 1 line 5
-----
AOS/V5 CLI REV 01.64.255.255 11-APR-83 15:25:28
Son of Sed) filestatus

DIRECTORY :UDD:SARAH.S209

SAMPLEFILE.S2 TEXT SAMPLEFILE SAMPLEFILE.SC
TEXT.ED SAMPLEFILE.ED
Son of Sed)

```

Figure 38 Filestatus

33. To leave SON OF SED and return to SED, enter bye.

```

c11
*
page 1 line 5
-----
AOS/V5 CLI REV 01.64.255.255 11-APR-83 15:25:28
Son of Sed) filestatus

DIRECTORY :UDD:SARAH.S209

SAMPLEFILE.S2 TEXT SAMPLEFILE SAMPLEFILE.SC
TEXT.ED SAMPLEFILE.ED
Son of Sed) bye
AOS/V5 CLI TERMINATING 11-APR-83 15:27:22

```

Figure 39 Terminating SON OF SED



34. Changes to files may be lost if the system fails during an editing session. To prevent loss it is good practice to update your files from time to time. The **SAVE** command makes a copy of the file, including all changes in the current editing session. The copy is saved with the same filename and the extension **.SV**. Issue the **SAVE** command.

```

save
*
page 1 line 5
1 This is line 1.
2 This is line 2.
3 This is the new line 3.
4 This is line 6.
5 This is line 5.
6 This is the text.

```

Figure 40 The SAVE Command

35. Use the **DO** command to do a filestatus. Note the file, **samplefile.sv**. This is the file created by the **SAVE** command. The file **samplefile.ed** is created by SED and stores display settings. Since you did not specify any display settings, the settings are default values. The files with the extensions **.S2** and **.SC** are deleted when SED is terminated.

```

do filestatus
*
page 1 line 5
DIRECTORY :UDD:SARAH.S209
SAMPLEFILE.S2 SAMPLEFILE.SV TEXT SAMPLEFILE
SAMPLEFILE.SC TEXT.ED SAMPLEFILE.ED

```

Figure 41 DO Command

36. Terminate SED with the **bye** command.

```

bye
*
page 1 line 5
Output file - :UDD:SARAH.S209:SAMPLEFILE
)

```

Figure 42 BYE Command

37. Use the CLI filestatus command to determine the filestatus. The temporary files with the .S2 and .SC extensions are gone. The file created by the SAVE command is still present. It does not contain any editing changes made since the last SAVE command.

```
) filestatus  
  
DIRECTORY :UDD:SARAH.S209  
  
SAMPLEFILE.SV   TEXT       SAMPLEFILE   TEXT.ED  
SAMPLEFILE.ED  
)
```

Figure 43 Filestatus

### Directions

After you have successfully completed Lab Activity 3 turn to the Addendum Quiz.

## SED Addendum Quiz

1. List the four steps in a typical SED editing session.
  - a.
  - b.
  - c.
  - d.
2. Use SED to create a file called `practice`. Input the text in Figure 44 exactly as shown.

```

append
*
page 1 line 11
1 Data General's SED is a screen-oriented test editor that allows
2 you to modify existing test files or to create new ones. The
3 test may be as varied as reports, computer programs, business
4 correspondence, and novels. With SED commands you can: display
5 existing test, add new test, change existing test, delete old
6 test, move pieces of test, copy pieces of test.
7 SED allows you to choose among several ways of displaying your test.
8 Congratulations!
9 You can define function keys to execute commands that meet your
10 editing needs and copy test in your current file onto other files.
11

```

Figure 44 PRACTICE Text

3. Use SED to create a second file called `finis`. Input the text in Figure 45 exactly as shown.

```

view
*append
page 1 line 1
1 You have completed the SED addendum.
2

```

Figure 45 FINIS Text

4. Use SED to make the following changes to the file practice. Refer to Figure 46 as you make your changes.

```

append
*
page 1 line 11
1 Data General's SED is a screen-oriented test editor that allows
2 you to modify existing test files or to create new ones. The A
3 test may be as varied as reports, computer programs, business
4 correspondence, and novels. With SED commands you can:1display
5 existing test,2add new test,3change existing test,4delete old
6 test,5move pieces of test,6copy pieces of test.
7 SED allows you to choose among several ways of displaying your test.
8 Congratulations!
9 You can define function keys to execute commands that meet your
10 editing needs and copy test in your current file onto other files.
11
B

```

Figure 46 Changes to Text

- a. Use SED commands or function keys to split the sentence labeled A at points 1, 2, 3, 4, 5, and 6 in Figure 46. Continue only after your screen matches Figure 47.

```

view
*
page 1 line 6
1 Data General's SED is a screen-oriented test editor that allows
2 you to modify existing test files or to create new ones. The
3 test may be as varied as reports, computer programs, business
4 correspondence, and novels. With SED commands you can:
5 display existing test,
6 add new test,
7 change existing test,
8 delete old test,
9 move pieces of test,
10 copy pieces of test.
11 SED allows you to choose among several ways of displaying your test.
12 Congratulations!
13 You can define function keys to execute commands that meet your
14 editing needs and copy test in your current file onto other files.
15

```

Figure 47 PRACTICE Screen

- b. Delete the last sentence (labeled B).
- c. Use the **SUBSTITUTE** command to locate each instance of the word *test* and change it to the correct word **text**.
- d. Add the text from the file **finis** to the end of the file **practice**.

**Directions**

Check your answers with those on the next page.

## SED Addendum Quiz Answers

1. a. Invoke SED  
b. Create a new file or bring in an old file for editing  
c. Add or modify the file  
d. Terminate the session
4. b. One method is to use the command, delete 13 14.  
c. One method is to enter substitute "text" for test all.  
d. One method is to enter insert 12 from finis. Your screen should now be similar to Figure 48.

```
insert 12 from finis
*
page 1 line 14
1 Data General's SED is a screen-oriented text editor that allows
2 you to modify existing text files or to create new ones. The
3 text may be as varied as reports, computer programs, business
4 correspondence, and novels. With SED commands you can:
5 display existing text,
6 add new text,
7 change existing text,
8 delete old text,
9 move pieces of text,
10 copy pieces of text,
11 SED allows you to choose among several ways of displaying your text.
12 Congratulations!
13 You have completed the SED addendum.
```

Figure 48 The End

Congratulations! If you have answered all the questions correctly, you have completed this addendum. Otherwise, review the material and try the questions again.

Please complete this questionnaire and return it to us after finishing the course. We will then send you an *Official Certificate of Course Completion!* Fill in the charts below to rate the course on the items listed. (Check one box for each item.)

Items	Excel- lent	Good	Fair	Poor
1. Organization and Style 2. Technical Content: Level of Detail 3. Ease of Listening to the Audio 4. Pace of the Audio 5. Quality of the Audio 6. Pace of the Student Guide 7. Usefulness of Quizzes 8. Effectiveness of Exercises 9. Effectiveness of Illustrations 10. Technical Reference Materials				

Items	Too Little Detail	Too Much Detail	Unclear	Other
11. If you said that technical content was fair or poor, what did you mean? 12. If you said the pace of the audio was fair or poor, what did you mean? 13. If you said the pace of the Student Guide was fair or poor, what did you mean?				

Items	Always	Almost Always	Rarely	Never
14. Was the information in the course interesting? 15. Was the manner of presenting the information interesting? 16. Did you understand what you were supposed to learn? 17. Were the materials directly related to the objectives? 18. Were there enough practice exercises? 19. Were the practice exercises appropriate? 20. Was there enough information explaining the answers to the practice exercises? 21. Did the module quizzes really measure your performance of the objectives? 22. Was the information in the audio consistent with the information in the Student Guide?				

23. Have you any comments on specific modules?

\_\_\_\_\_

24. What did you like best about the course?

\_\_\_\_\_

25. What did you like least about the course?

\_\_\_\_\_

26. Would you use another self study on other Data General products?

\_\_\_\_ Yes    \_\_\_\_ No    If No, why? \_\_\_\_\_

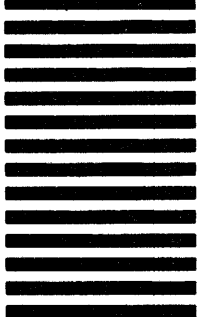
27. Suggestions for improvement: (Write your comments here and on the back of this page.)

Name \_\_\_\_\_ Address \_\_\_\_\_


**BUSINESS REPLY MAIL**  
FIRST CLASS      PERMIT NO. 26      SOUTHBORO, MASS. 01772



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



Postage will be paid by addressee:

 **Data General**  
Educational Services Department  
Mail Stop F019  
4400 Computer Drive  
Westboro, Massachusetts 01581-9973

ATTENTION: Quality Assurance



1

2

3

