HEWLETT PACKARD

Getting Started With RTE-A

Software Technology Division
11000 Wolfe Road
Cupertino, CA 95014-9804

Manual Part No. 92077-90039
U0189

Printed in U.S.A. August 1987
Update 1 January 1989

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARs 252.227.7013.

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

First Edition	Jun 1983	
Update 1	Dec 1983	
Reprint	Dec 1983	Update 1 incorporated
Update 2	Jan 1985	
Reprint	Jan 1985	Update 2 incorporated
Update 3	Jan 1986	
Reprint	Jan 1986	Update 3 Incorporated
Second Edition	Aug 1987	Rev. 5000 (Software Update 5.0)
Update 1	Jan 1989	Rev. 5010 (Software Update 5.1)
Reprint	Jan 1989	Update 1 incorporated and index revised

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Preface

This manual is a tutorial guide for the first-time user of the HP 1000 A-Series computer system, which consists of an A400, A600/600+, A700, A900, or A990 processor with a hard disk drive. The manual will introduce you to the RTE-A operating system and its interactive commands. You will learn how to

- Start up the system and log on
- Use the Command Interpreter
- Manipulate files using the editor
- Create and run programs

You will quickly become comfortable with RTE-A if you try each of the examples at your terminal keyboard as you read through this manual. In the examples, your responses are underlined for clarity.

For information about the keyboard and display screen, refer to the terminal owner's manual supplied with your system.

For detailed reference material about your system, refer to the *RTE-A Index and Glossary*, part number 92077-90036, which contains a summary description of all manuals in the RTE-A Operating System manual set. The *RTE-A User's Manual*, part number 92077-90002, provides more detailed information on using all of the interactive features of RTE-A.

This manual is intended for systems operating from hard-disk media. Memory-based systems or systems with only flexible disk storage are not configured to run the examples given.

Your system may include the optional 92078A Virtual Code Package known as VC+. Since VC+ is required for multiuser environments, this manual describes the use of the additional features where appropriate. If you are working with the standard RTE-A system, you may skip those paragraphs that are labeled with (VC+ Systems) following the paragraph title.

In the process of getting acquainted with RTE-A, if you cannot recall the meaning of a command or a term, go to the index at the back of the manual. Terms and commands used with RTE-A are listed there, with the page number or numbers on which they are defined.

Table of Contents

Chapter 1
Beginning

Chapter 2
The Command Interpreter

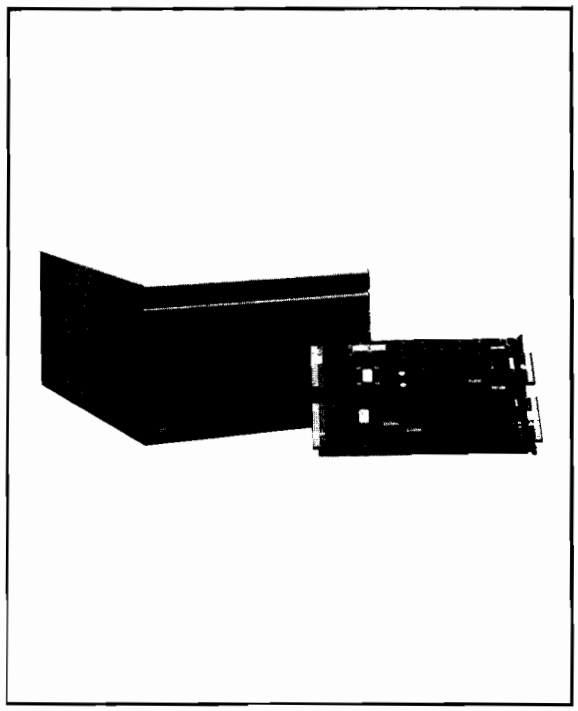
Chapter 3
The File System

Chapter 4
Using the Editor

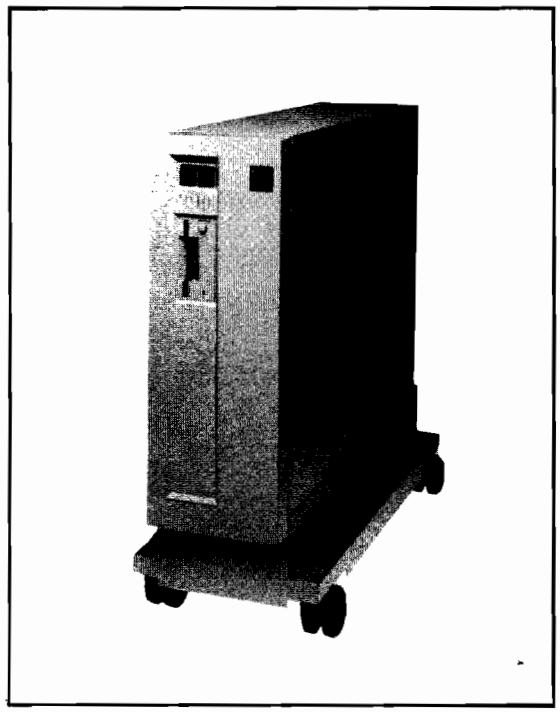
Chapter 5
Developing a Program

Appendix A
System Management

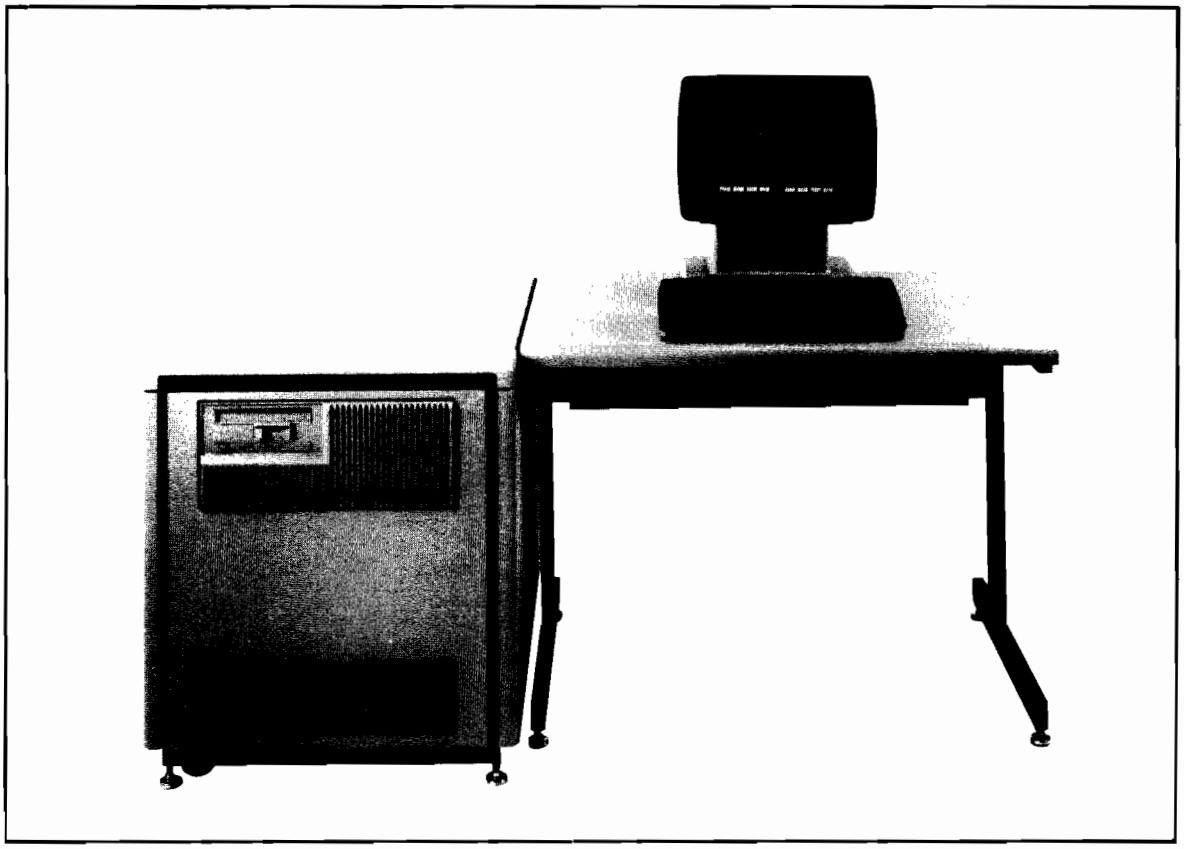
List of Illustrations



A400 and Micro 14 Package



Micro/1000 System



Model 26/27/29 with CS/80 Disc and Cartridge Tape Drive

1

Beginning

Talking to the System

You communicate with the system by entering command codes at your keyboard. These codes are stored in the memory section of your terminal. When you press the carriage return after entering the command, this signals the system to accept your command line and begin acting on it.

If you make a typing mistake when entering the code line, use the BACKSPACE key to correct the error before pressing the carriage return. Do not use the left arrow key to move the cursor. BACKSPACE removes the characters from your terminal memory and the terminal screen. (On some older systems, the backspace key removes characters from the terminal memory but they are still displayed on the screen.) Just backspace the cursor over to the error, correct the typing mistake, and then retype the remainder of the line. (You must retype all remaining characters since the backspacing erased these characters from their locations in memory.) When you are satisfied with the command line, press the carriage return to send the command to the system.

If you have a long command line or if your typing error is at the beginning of the line, you can delete the entire line prior to pressing the carriage return by using the DEL key. (On some terminals, delete is carried out by the SHIFT-DEL key combination; refer to your terminal user's manual.) Pressing DEL will erase the entire line from its location in memory. The line remains on the screen for your reference, and a backslash is printed at the end of the line. The cursor moves to the left margin of the next line, ready for input. You now can enter the line correctly and press the carriage return to send the command to the system.

To summarize the procedure for communicating with the system through your keyboard: Enter the command line following the system prompt; BACKSPACE to correct a typing error in a command code line, or DEL (or SHIFT-DEL) to cancel the line for retyping; press the carriage return to send the command to the system.

Starting the System

If you will be regularly starting from a powered-off system, you should read the next two paragraphs. If you will not be responsible for powering up the system, you may skip to the section "Logging On".

Powering Up the System

The A-series computer comes in a variety of system packages. Refer to the System Installation Guide for the power-up procedure for your system.

Booting the System

After power has been turned on and the system completes its self tests, the system console screen will display the prompt VCP> to show that the Virtual Control Panel program is active. The VCP program allows you to control the processor from your system console. For example, to zero all memory locations, enter the %C command. VCP will clear all memory registers, display the results of the program action, and return to the VCP> prompt. The sequence is:

```
VCP> %C
CLEAR MEMORY
VCP> _
```

The boot-up process consists of running a program, BOOTEX, from disk. This program automatically loads the operating system and initializes any peripheral devices. Various system configurations require a different VCP command to accomplish this task; your system manager can provide the command string for your current system. Write the command string here as a reminder for the next time you boot the system. (Use a pencil; the system manager can change the command string at any time.)

```
VCP>_____________________________
```

Once the command string is entered, the system will begin booting up. This process may take a minute or more, depending on how many peripherals are being initialized. A series of status messages may be displayed as the process proceeds. The last message will be:

```
Finished
```

At this point, press the carriage return to get your system prompt.

Logging On

The RTE-A system is now ready for your use, with the flashing cursor positioned at the spot on the screen where you will begin typing commands. In this manual, the cursor position is represented with a single underline character (_) in the examples. Feel free to experiment with any of the commands described in this manual. The worst consequence of your actions would be to ruin one of your files or perhaps find yourself logged off.

The system manager can tell you if you are working with the standard RTE-A operating system or the RTE-A/VC+ operating system (RTE-A with Virtual Code, described below). If yours is the standard system, you will not need to log on; you will see the CI> prompt (the Command Interpreter user interface prompt) immediately after the system is booted up. If the prompt does not appear, hit any key to bring it up. Skip to the next section, "Logging Off".

The VC+ option is designed to provide you with a multi-user environment approaching that of a personal computer. Used properly, it will prevent other users from affecting your work on the system and vice-versa. If you have a VC+ system, you will need to log on before you can begin using the Command Interpreter. You will see something similar to the following prompt:

```
Please log in: _
```

User.Group Identification

You can log on by entering a user and group identification at the logon prompt. The user and group identification is established for you by the System Manager or installer, and it has the following format, where the period (.) separates the user name from the group name:

```
user.group
```

Every user in a multi-user system has a unique user name and is a member of one or more groups. For example, in the following user.group identification, the user is Sandi and the group is Lab:

```
Please log on: Sandi.Lab
```

The *.Lab* portion of the user.group identification is optional, because every user is assigned a default logon group when their user account is set up. For example, if the default logon group for user Sandi is group QA and Sandi logs on without entering a group identification, she will be logged on as *Sandi.QA*. If Sandi wants to be logged on in group Lab, she must include the *.Lab* group identification.

To determine your default logon group, log on without specifying a group name and do a SET command from CI to display all the CI variables. The LOGON variable contains the user and associated group you are logged onto in the form:

```
USERNAME.GROUPNAME
```

(Refer to the description of the SET command in your RTE-A User's Manual.) The default logon group can be changed by the System Manager, a superuser, or a user with a high enough capability, through the Group and User Management Program (GRUMP).

Password

After the user.group identification is entered, the system prompts for a password.

```
Password? _
```

At this point, enter your password. If you do not know what your password is, contact your System Manager. Your entry is not displayed on the terminal screen to prevent others from seeing your password. If you do not have a password, just enter a carriage return at the *Password?* prompt.

Following a successful logon sequence, you will see the prompt:

```
CI> _
```

You are now ready to use the Command interpreter, CI. (The Command Interpreter is described in the next chapter.)

If log on is unsuccessful, you can try to log on again by repeating the process. After two or three unsuccessful attempts, you should contact your System Manager for assistance.

Log On Error Messages

The system program that handles both the log on and log off processes is the LOGON program. Error messages displayed during log on and log off are proceeded by one of three prompts:

```
LOGOX:

LOGON:

LOGOF:
```

If the user name in the user.group identification entered is invalid, the following error message is displayed on your terminal, where xxx is the user name entered:

```
LOGOX: No such user xxx
```

If the group name entered is invalid, the following error message is displayed, where xxx is the group identification entered:

```
LOGOX: Unable to access group xxx
```

If the password entered is incorrect, the following message is displayed:

```
LOGOX: Incorrect password
```

When accounts are set up, the System Manager allots a certain amount of CPU usage and connect time for each group and for each user within a group. Each time a user logs on, the system checks the accrued CPU usage and connect time for the group as a whole and for the user within the group. If the CPU usage or connect time has been exceeded for either the group account or for your account within the group, an error message will be displayed and you will be denied access to the system. The possible error messages follow, where xxx is the group identification entered:

```
LOGOX: Your CPU time within xxx has been exceeded

LOGOX: Your connect time within xxx has been exceeded

LOGOX: xxx CPU time limit has been exceeded

LOGOX: xxx Connect time limit has been exceeded
```

Of the error messages listed above, the first two refer to situations in which an individual user's limits have been exceeded. The second two messages refer to situations when the group's limits have been exceeded. If you receive any of these messages, contact your System Manager for assistance.

Logging Off

The proper procedure for logging off is to enter the following command:

```
CI> ex
Finished
```

For RTE-A operating systems with the VC+ option, you enter the same command (EX), but there are additional steps involved in the log off process. If a logoff program or command file has been defined for your user.group logon, it is scheduled or transferred to following the EX command. The *Finished* message is then printed and you receive additional system messages that provide such information as the session number, date, time of log off, connect time, and CPU usage. The following is an example of logging off with a command file that prints the message *Bye World!*:

```
CI> ex

Bye World!

Finished

Session 103 finished. Tue Jul 14, 1987  5:49:52

                Connect time          CPU usage
Session:        0 hr 0 min 12 sec     0 hr 0 min 4 sec 740 msec
Cumulative      31 hr 54 min 27 sec   1 hr 38 min 41 sec 560 msec
```

This closes any open files you might have been using and exits the current session.

In the VC+ multiuser environment, if you have any programs that are still running, you will be prompted to continue with your session, terminate the programs or force them to execute in a background mode. You will see the message sequence:

```
CI> ex
Your programs:
MATH1
APPL2
Continue, Log-off, Background, or ? [C]? _
```

If you now enter "?", you will see a message explaining each of the options. More information on background programs can be found in the RTE-A User's Manual.

The HELLO Program

The first thing you should do when getting started with RTE-A is to run the HELLO program. This program is designed to provide intensive interaction with RTE-A through a series of instructional screens and exercises. Information in this manual does not correspond directly to the contents of the HELLO program. This manual leads you through the various system functions most frequently used. The HELLO program presents "menus" from which you can select any topic you are interested in learning.

HELLO guides you through the use of RTE-A from fundamentals, file management and editing, to developing, loading and running programs. Each instruction segment ends with a short quiz to let you know how well you have absorbed the material presented. HELLO is available on your system for your use as a basic learning tool and as a refresher for individual segments of the operating system. To run HELLO, just type *hello* after the CI> prompt and press the carriage return. If the HELLO program has not been loaded on your system, the following message is displayed:

```
No such file HELLO
```

The Help Command

CI provides a special help command you can execute to obtain information on a command or utility program. Enter "?" at the CI> prompt to obtain a list of the commands and programs for which help is available. Enter "?" followed by the command or program name to display a summary of the command or program functions. For commands, Help lists the command string and defines all parameters in the string.

2

The Command Interpreter

The Command Interpreter (CI) is a program that accepts your input, interprets it, and translates it into instructions to the operating system. The command EX is one you are familiar with already. This tells the operating system to terminate the copy of CI that you are using, which logs you off in a VC+ environment.

This chapter will cover a sample of the RTE-A CI commands you can use to become familiar with the system. The CI commands consist of two or more letters, and are usually mnemonics to aid in recall. Many CI commands are actually programs. For this reason, it is important to avoid using any of the CI command names when naming your programs.

Run a Program – RU

RU is the CI command used to run a program. It is formally entered "ru programName", but CI assumes the "RU" if the program name entered does not match another CI command. For example, to run the EDIT/1000 program (the RTE-A editor), you could enter either of the following:

```
CI> ru edit

CI> edit
```

Because the RU is implied, you should avoid using any of the CI command names when naming your programs. For example, if you name a program CO and enter merely CO to run it, CI will interpret the entry as a copy command and will display the CO command syntax.

Directory List – DL

To list the contents of the current directory (in this example, your directory is named george), enter the DL (Directory List) command:

```
CI> dl
directory ::GEORGE
J.TXT     PROGA.FTN    SPOT.RUN     TEMP.TEXT     W.PAS
CI> _
```

If there are no files in your directory, CI will respond with the following message:

```
Directory is empty ::GEORGE
CI> _
```

Show System Time – TM

Another commonly used command is TM, which displays the current time. Enter:

```
CI> tm
Mon June 8, 1987  11:37:13 am
CI> _
```

CI shows the current date and time, then returns a CI> prompt.

Show System Status – WH

A useful command is WH, which gives you active program information for your terminal (or your session in a multiuser VC+ environment). To find your status, enter:

```
CI> wh
```

WH will respond with the program status in a display similar to the following (the specific information for your active programs will, of course, replace the information in the example):

```
Program                    DataPartition   CodePartition
Name    Prio  PC   Seg     Size   Status   Size    Status   Program   Status
-------------------------------------------------------------------------------

Session   44 Capability 20  User  GLADYS
CI        51 14523           32     in                      waiting for WH
WH         5  5640           15     in 5                    scheduled
-------------------------------------------------------------------------------

Sun May 31, 1989 12:10 pm
```

This example shows WH running in a session (VC+) environment, with session 44 identifying GLADYS (capability level 20) as the user. At this point, the important information for you to recognize is the program status. The command interpreter CI for Gladys' session (44) is temporarily suspended waiting for WH to complete. The term "scheduled" here means that WH is now running.

Show Input/Output Device Status – IO

Another common command is IO, which asks the operating system to report the functional state of the peripheral devices (disks, terminals, printers, tape drives) connected to your system, I/O configuration organized by LU (logical unit) number, or I/O configuration organized by select code grouping. Each device is assigned an LU number which the operating system uses to identify the device. If you know the LU number assigned to a device, you can get the status of that specific device by entering the command and the LU number (entering the command without an LU number will give you the status of all peripheral devices in the system). As an example, if your system printer is identified as LU 6, you can get the device status by entering:

```
CI> io 6

lu device name status

 6 printer     idle

Thu, Apr 28, 1989, 12:03 PM

CI> _
```

This tells you that the printer is "idle"; that is, the printer is operational but is not in use. If the status is listed as "busy", the printer is currently in use. A status of "down" means that the printer is not currently operational; it may have been placed off-line (not under control of the computer), or it may not be working for some reason. If you find that a device is listed as down, try using the "UP" command to put that device back into operation. If, for example, the printer device (LU 6, above) status is shown as down, enter the following:

```
CI> up 6
CI> _
```

If the operation is successful, you will see the CI> prompt. If the operation is not successful, in most cases a device-status error message will appear followed by the CI> prompt. When this happens, you should refer the problem to the System Manager.

Command Stack – /

The command stack editor lets you enter command stack commands, display and modify previously entered command lines using the terminal editing keys, and re-enter previously entered command lines as new ones.

Each command line entered is remembered in the "stack." The most recently entered commands are pushed onto the top of the stack, and the oldest commands fall off the bottom. Duplicate command lines are removed from the stack.

Command stacks can be saved in files. At logon, a file called CI.STK is searched for on the home directory first. If it is not found there, it is searched for on the working directory, or in the FMGR disk cartridges, if there is no working directory. If the command stack file is found, it is opened and its contents used to initialize the command stack. If it does not exist, the default file CI.STK

is created on the working directory, and the contents of the command stack are posted there at logoff.

When command stack commands are entered from the CI> prompt, a "frame" of previous command lines is displayed. Frame refers to the maximum number of lines to be displayed in a command stack window. In CI, this is specified in the FRAME_SIZE variable.

In the following example, the variable FRAME_SIZE is equal to 20. The banner that precedes the command stack window shows the starting line number of the window, the total number of lines selected for this display, and the name of the current stack file. Here the window starts at line 20 out of a total of 320 lines in the stack.

```
CI> /
--020/320-- Commands: [/DEXTER/CI.STK]
? ex
dl
ru dl
ss spot.run
go spot.run
br spot.run
tm
wh
io 6
up 6
io 8
wh al
? wh
? pu
pu spot.lst
pu spot.dbg
wh pa
edit testprog.ftn
co testprog.ftn backup.ftn
li testprog.ftn
_
```

The cursor is now positioned below the last command entered. If you want to repeat a command, just move the cursor to that line and press the carriage return. You can also edit any command in the command stack using the local terminal keys, and then execute the command by pressing the carriage return. When the carriage return key is pressed, the commands below the command executed will disappear from the screen (but not from your stack).

In Case of Trouble

It is possible to interrupt the system while it is working. For example, if you are waiting for something like a COPY operation to complete on a long file, you could interrupt the system by pressing any key on your keyboard.

A copy of CI, called CM, is kept waiting to handle such situations. If the system is interrupted while CI is busy performing a task, the standby program CM is run and you will see the following prompt on your screen

```
CM>
```

CM is similar to CI except that CM does not support the following commands: TR, EX, IF-THEN-ELSE-FI, and WHILE-DO-DONE. Also, CM executes only one command before returning to CI. Just press the carriage return in response to the CM> prompt and the interrupted program will resume communication with your terminal through CI.

If you suspect that a program is in trouble, you can use the commands described in the following section to remedy the situation.

Restarting or Terminating a Program – SS, GO, BR, OF

There are three ways for you to stop a program that is currently executing. You can suspend execution until a later time (SS command), you can halt execution in an orderly way (BR command), or you can halt execution abruptly (OF command) without internal table or file clean up.

In all cases you stop program execution by pressing any key on the keyboard and, when a prompt appears, entering the program halt command with the program name.

When you stop execution with the SS command, for example:

```
CI> ss proga
```

any input/output operations in progress will continue to completion, and the program file will remain open.

You can resume execution at a later time using the GO command. When you use the GO command to restart your suspended program, enter:

```
CI> go proga
```

and the program will begin executing at the point at which you suspended it.

You can set a flag to break the program execution using the BR command, as follows:

```
CI> br proga
```

The BR command merely sets a break flag. It is up to the program to check the break flag. If the program does not check the flag, this command will not break the program.

When you stop execution with the BR command, normal clean-up operations such as closing files used by your program and ending input/output to the file are performed, and the program exits. Note, however, that a program will respond to the BR command only if that program includes a call to a subroutine to test for the BR command. (For more information about BR, refer to the RTE-A User's Manual.)

When you stop execution with the OF command, the normal clean-up operations are not performed. The OF command is used mainly to halt a program that is in trouble. Note:

1. Only the System Manager can use the OF command with programs loaded as system utilities.
2. If you OF your copy of CI, you will have to log on again to continue.

3

The File System

The File Structure

Files are cataloged in directories that contain file names and file locations on the disk, as well as other information such as file size and type. RTE-A has a "hierarchical" file structure, in which directories can also contain similar information about other directories, called subdirectories. Subdirectories have the same characteristics as directories; the term subdirectory means only that it is cataloged in the next higher level directory or subdirectory. Throughout this manual the terms "directory" and "subdirectory" are used interchangeably.

This file system organization can be thought of as a tree, with a root and branches. Figure 3-1 illustrates this tree structure, with the root (called the global directory) at the top. RTE-A can support multiple global directories that can branch to any level, limited only by the system storage capacity.

The tree structure provides a clear search path for the system in finding a specified file, since each file can belong to only one directory. Referring to Figure 3-1, if file DATALOG is to be accessed, the search path is from directory GEORGE, through subdirectory APPLICATIONS, then through subdirectory MEASCONTRL to file DATALOG. The search path must be spelled out in the command line you enter to specify the file.

File Descriptors

Each file has a descriptor that uniquely identifies that file. The descriptor includes the directory path in which the file is cataloged, the file name and extension, the file type, size, and the length of the records in the file. The system supplies default values for the type, size, and record-length of the file. In getting acquainted with RTE-A, you will be working with general-purpose text files, and the default values supplied are proper for these files. (Refer to the RTE-A User's Manual for information on file type, size and record length.)

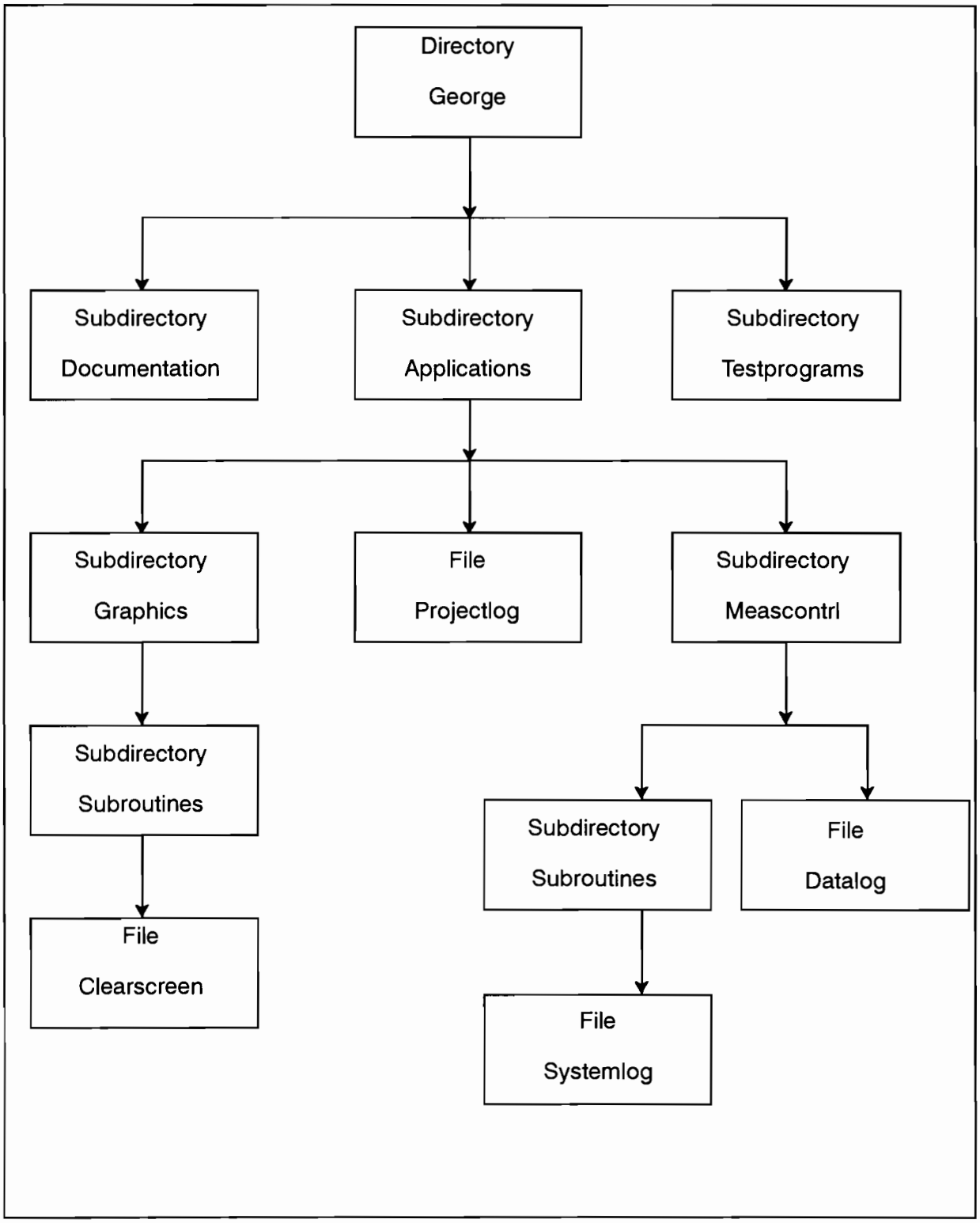


Figure 3-1. RTE-A File Structure

File Name and Type Extension

The name you assign to your file should clearly define the contents of the file; that is, it's better to name a file "LetterToMom" rather than "ltm". You can use uppercase, lowercase, or a combination of both in constructing your file name; alpha characters are stored in uppercase. The name can have up to 16 characters in any meaningful combination, subject only to the following rules:

1. File names must not start with a number. If the first character is a number it is interpreted as an LU number identifying an input/output device, and not a file. File names can, however, contain a number – july14report would be a valid file name.

2. File names must not contain the characters / @ . – >] These characters have a special meaning to CI. Any other punctuation is allowed, although you should be aware that other existing and future HP products may give special meaning to certain punctuation within a file name. Alphabetic characters and numbers will always be safe to use.

File type extensions, while not always required, are highly recommended as another means of clearly defining the contents of a file. Extensions identify the type of information in the file: text, FORTRAN or Pascal programs, data, etc. They consist of a period and up to four characters added to the end of a file name. It is important to remember that when you assign an extension to a file name, you must always use that extension as part of the file reference in a command line.

Some of the most commonly used extensions are:

.txt	text file
.cmd	CI command file
.ftn	FORTRAN source code file
.pas	Pascal source code file
.lod	LINK command file
.dat	data file
.lst	listing file
.rel	relocatable file
.run	program file

Refer to the RTE-A User's Manual for a complete list of extensions.

The following are all valid file names:

LetterToMom.txt
CalculateTax.ftn
DATALOG
SparesInventory.dat
july14report

The following file names are not valid:

LetterToMomWrittenOnFriday.txt	(file name >16 characters)
14julyreport.txt	(begins with a number)
CostOfBagels@15.dat	(contains reserved character)
TaxOwedOuch.fortran	(extension >4 characters)

Directory Name

Directory names are subject to the same restrictions as file names: a limit of 16 characters, may not begin with a number, and may not include reserved characters. Directory names are assigned the type extension .DIR by default. You cannot specify the extension .DIR in a command line where the entry is followed with a slash. For example, the following command line would be illegal:

```
co /george/applications.dir/prog 6
```

You must, however, specify the .DIR type extension when purging a directory. You may specify the .DIR type extension in a command such as:

```
dl @.dir
```

which requests a listing of all directories under the current directory. In this command, the at sign (@) reflects the use of the RTE-A mask feature, described later in this chapter.

When referencing a file in a command line, you should specify the directory (and subdirectory, if applicable) in which the file is cataloged. All directory and subdirectory references must precede the file name in the command line entry. (This is done implicitly when you reference your working directory.)

When referencing a global directory in a command line, you must use a leading slash (/), as follows:

```
/george
```

Without the slash, CI interprets the entry (in this case, GEORGE) as a subdirectory or file and assumes that it can be found in your working directory. (This assumption is always made when the global directory is not specifically named.) For example, if you want to change your working directory to the global directory GEORGE, you enter:

```
CI> wd /george
```

Since GEORGE is a global directory and not a subdirectory or file cataloged in the working directory, if the slash was omitted, CI would return the error message:

```
No such directory GEORGE.
```

This is an important distinction: when the name is preceded with a slash (/GEORGE), it describes a search path starting at a global directory; when it is not preceded with a slash (GEORGE), it describes a subdirectory or file in your working directory.

Entering the Command Line

A command line to CI may contain a maximum of 256 characters, including all delimiters (spaces, slashes, etc.) When your command line contains a file reference through several levels of subdirectories, the line length can approach this limit. You can use the command stack feature of RTE-A if you repeatedly use the same command during the period you are logged on to the system.

CI expects the file reference portion of your command line to be entered in the following format:

```
/directory/subdirectory/filename.ext
```

The leading slash (/) is required if the first element is a global directory name, and slashes are required to separate the elements of the entry. If the first element of the entry is a subdirectory of the working directory, the required format is:

```
subdirectory/filename.ext
```

Where there are several levels of subdirectories, they must all be specified. Using the example of Figure 3-1, the following command line would specify opening file DATALOG for an edit operation:

```
CI> edit /george/applications/measconTrl/datalog
```

This will access the file DATALOG which is cataloged in subdirectory MEASCONTRL, which in turn is cataloged in subdirectory APPLICATIONS, which is cataloged in the global directory GEORGE. (The command-line length here is 44 characters, including the blank space after the edit command and the slash delimiters.) There must be a space or a comma between the command and the file specification.

To edit the file PROJECTLOG in subdirectory APPLICATIONS, the command line format is:

```
CI> edit /george/applications/projectlog
```

Directory Path Abbreviations

The file system on RTE-A has as one of its main features a tree structure linking directories together. When accessing a file or directory in the tree, you enter the complete path from global directory down through all the subdirectories. If the current working directory is a subdirectory in the path, you can use it as a starting point and specify the subdirectories below the working directory down to the file. For example, to access the file CLEARSCREEN on path /GEORGE/APPLICATIONS/ GRAPHICS/SUBROUTINES, the entire file descriptor may be given:

```
/george/applications/graphics/subroutines/clearscreen
```

But if the current directory is /GEORGE/APPLICATIONS/GRAPHICS, then the descriptor SUBROUTINES/CLEARSCREEN describes the same file.

This shows that it is very easy to move down the tree structure. The character sequence '..' (two dots or periods in a row), allows you to move up the tree structure. It means *use the parent*

directory name in this position. For example, if your working directory is /GLOBAL/SUB1/SUB2 and you want to access file /GLOBAL/SUB1/TOP. You could specify the entire path (/GLOBAL/SUB1/TOP), or you could specify ../TOP. The '..' tells the system to *back up to the parent of the current directory*, and look for TOP there. Of, if your working directory is /GLOBAL/SUB1 and you want to access file TOP on the directory /GLOBAL/SUB2, you could specify /GLOBAL/SUB2/TOP, or use the '..' feature and specify ../SUB2/TOP. The '..' means *use my parent directory*, which in this case is /GLOBAL. Then starting from that point, the system picks up subdirectory SUB2 and its file TOP.

The character '.' means *the current directory*. For instance, the descriptor './TOP' refers to the file TOP on the current directory. Specifying './TOP' forces TOP to be found on the current directory. This is important to note, because in some cases (for example, when you are using user-definable search paths), entering only 'TOP' does not guarantee that the current directory will be searched for the file TOP.

Creating a Directory – CRDIR

The CRDIR command allows you to create a directory. For example:

```
CI> crdir /fred
```

will create the global directory FRED. Note the slash preceding the directory name. This is necessary to identify FRED as a global directory. (In creating a directory or subdirectory, the type extension .DIR is used as the default file type extension.) If you enter the command as follows:

```
CI> crdir fred
```

FRED will be created as a subdirectory in your working directory. To create a subdirectory in a directory other than your working directory, you must specify the entire directory path in the command, as follows:

```
CI> crdir /fred/games
```

GAMES will be created as a subdirectory in directory FRED.

Defining a Working Directory – WD

CI allows you to define a working directory using the command WD. A working directory is convenient when you are repeatedly accessing a file or files that have long path names. For example, referring to Figure 3-1, the command line to print the file DATALOG is:

```
CI> print /george/applications/measconrtl/datalog
```

You can specify MEASCONTRL as your working directory with the command:

```
CI> wd /george/applications/measconrtl
```

MEASCONTRL will remain as the working directory until you change it with a new WD command or you log off the system.

If you define MEASCONTRL as your working directory, you can print the file DATALOG with the command:

```
CI> print datalog
```

When CI sees the above command, it assumes that the file is cataloged in the working directory.

It is important to note that in a non-VC+ environment all users share the same working directory. You should therefore consider the needs of other users before using the WD command.

In a VC+ environment, each user has an independent working directory. Each time you log on to the system, your default directory (in Figure 3-1, /GEORGE) is initialized as the working directory. If you want a specific directory to be the working directory, you can change it after log on using the WD command, or you can have your System Manager change the account definition so that each time you log on, a new default working directory is initialized.

You can display the name of your current working directory by entering the WD command without specifying a directory:

```
CI> wd
Working directory is /GEORGE/APPLICATIONS/MEASCONTRL
CI>
```

Creating a File – CR

Although most files are created by the editor or by other programs, CI allows you to create a file using the CR command.

The CR command creates an empty file; that is, a directory entry is written and space is reserved for the file. Unless you specify the directory and the file type and size, CR will create a general-purpose text file entry in your working directory and will reserve 24 blocks of space on your disk for the file. You can specify the directory and the subdirectory where you want your file to be created, as long as the directories and subdirectories entered already exist. When you specify directory path information, the command line must follow the directory path format defined above:

```
CI> cr /george/applications/newfile
```

Use the command now to create a file on your working directory. Notice that CI only returns the CI> prompt after the file has been created. Enter the DL command to confirm that the file has been created:

```
CI> dl /george/applications/
directory  /GEORGE/APPLICATIONS.DIR
JGRAPHICS.DIR    MEASCONTRL.DIR    NEWFILE    PROJECTLOG
```

The file that you created, NEWFILE, is now listed as being in the directory. Note that files are listed alphabetically.

Purging and Unpurging – PU, UNPU

The PURGE command allows you to delete both files and directories. The UNPURGE command allows you to reverse the action of the purge command to recover a purged file, but not a purged directory. Because the Purge command frees up the disk area occupied by your file, the Unpurge command will reverse the Purge only as long as that disk area is not reused by you or another user.

Purging and Unpurging a File

When you purge a file, the file entry in the directory is marked as being purged. The disk space used by the purged file is made available for other users. However, the contents of the file are not destroyed and the file may be recovered with the unpurge command. This feature is very useful should you accidentally purge the wrong file. Keep in mind, however, that the disk space is available for other users and could be overwritten at any time, destroying your file. If you purge the wrong file, use UNPU immediately to recover that file. For example, if you purge the file NEWFILE then decide you really want to keep it, issue the UNPU command after the purge command. The sooner the UNPU command is issued, the better chance you have of recovering the purged file. For example:

```
CI> pu newfile
Purging NEWFILE ... [ok]
CI> unpu newfile
Unpurging NEWFILE ... [ok]
```

Note that if the disk space is re-allocated before the UNPU command is used, the UNPU command will fail. Multiple files can be purged and unpurged by using a file-name mask when specifying the file to purge. (File masking is discussed later in this chapter.)

Purging a Directory

You can purge (but not unpurge) a directory. However, you can only purge an empty directory; all files and subdirectories in that directory must be moved or purged before the directory itself can be purged. Use the MO command or the CO command (described in the following section) to remove files from the directory, or purge all files using the command:

```
CI> pu /fred/@.@.s ok
Purging A::FRED:3:24 ... [ok]
Purging B::FRED:3:48 ... [ok]
Purging C::FRED:4:24 ... [ok]
```

This method of specifying the file uses file-mask "wildcard" characters. These are described in the section "File Masking" later in this chapter. After you have entered the command, enter a DL to see if the directory is empty. You may have to repeat the PU command if you have multiple levels of subdirectories (use the command stack feature). When your DL command response is:

```
CI> dl /fred/
Directory is empty  ::FRED
```

you can then purge the directory. The sequence is:

```
CI> pu /fred
Purging /FRED.DIR ... [ok]
```

You must include the .DIR extension when purging a subdirectory. For example:

```
CI> pu /fred/games.dir
Purging /FRED/GAMES.DIR ... [ok]
```

Moving and Copying – MO and CO

RTE-A provides two commands, move and copy, that allow you to move or copy files to another directory or device, and to rename a moved file.

Both the CO and MO commands require the definition of a source and destination. Whenever you enter a command that has a source and a destination for the operation, be sure to enter a space or a comma as a delimiter between the source and the destination:

```
CI> co source destination

CI> co source,destination
```

With the copy command you can make a copy of your file and place it in another directory, or you can copy your file to a device such as a tape drive. The original file entry remains in the original directory. If you omit the directory name in the CO command, CI assumes that your file is in the working directory. You can also direct the copy operation to your terminal or to a printer by specifying the LU number assigned to the device (the system manager can tell you the assigned LU or you can use the IO command as described previously). The following examples use /NEWDIR as the destination directory. If you want the file to reside in a specific subdirectory, you must enter the full path. If you want the copied file to have the same name, just enter a slash (/) at the end of the command line without retyping the name, or enter:

```
/dir/sub1/sub2/sub3/@.@
```

If you want to change the file name, enter the new name preceded with a slash as the last entry.

```
CI> co /dir/sub1/sub2/sub3/file.ext /newdir/newfilename.txt
```

To copy a file to another directory, keeping the same name, enter:

```
CI> co /dir/sub1/sub2/sub3/file.ext /newdir/@.@
```

To copy a file from your working directory to a device (in this example a printer, LU 6) enter:

```
CI> co filename,6
```

The move command is similar to the copy command except that the source file entry no longer exists after being moved. MO only moves the directory entry – the file remains where it is on your disk. Because the file itself does not move, the MO operation is much faster than CO.

Because the file does not move and the file must be on the same volume (LU) as the directory entry, files cannot be moved across disk volumes (LUs). If you want to move files across volumes, the CO command can be used with the 'p' option (purge source after copy) to move the files.

Move also allows you to change the file name at the destination.

To move a file to another directory, keeping the same name, enter:

```
CI> mo /dir/sub1/sub2/sub3/file.ext,newdir/@.@
```

To move a file from your working directory to another directory and change its name, enter:

```
CI> mo filename /newdir/newfilename
```

Time-Stamping Files

All CI files are automatically time-stamped by CI when the file is created, each time the file is accessed, and each time it is updated. This feature of RTE-A is particularly convenient in determining if you are working with the latest revision of a file. The DL command, described in Chapter 2, can be used to list the file time stamps. For this purpose, DL includes three options: create time (c), time last accessed (a), and last update (u).

For example, to see when HIFILE.CMD in directory /DEBBIE was created, enter:

```
CI> dl hifile.cmd c
directory  ::DEBBIE

   name              create time

HIFILE.CMD     Mon  May 18, 1987 11:00:31 am
```

More than one option can be specified in the DL command. To see when the file was last accessed and when it was last updated, enter:

```
CI> dl hifile.cmd au
directory  ::DEBBIE

HIFILE.CMD
access time  Fri July 3, 1987  8:47:23 am
update time  Mon June 29, 1987 11:00:31 am
```

The *access time* and *update time* will not always be the same, since you may open a file many times for reading the contents without making any changes to its contents.

File Masking

File masking is a powerful feature that allows you to specify more than one file in a single command line. Some of the commands that accept file masks are DL, PU, UNPU, MO, CO, and LI.

Masking is done by using "wildcard" characters as part of the file name. As the name suggests, a wildcard character will match any character in a file name. CI recognizes two wildcard characters:

- – The dash will match any one character in that position in the file name.
- @ The "at" sign will match zero or more characters in that position in the file name.

With wildcard characters, you can address multiple files with the following commands:

`CI> pu prog@`	This command will purge *all* files whose first four characters are PROG (PROG, PROGA, PROGRESSREPORT, PROGRAM, PROGRAM.FTN, PROGENY, etc.). This use of the wildcard character will *not* purge files whose names are less than four characters (for example, the file PRO will not be purged).
`CI> unpu prog-`	This command will unpurge (restore) all files with file- character names beginning with PROG (PROGA, PROGZ, PROG2, etc.). This use of the wildcard character will *not* unpurge any files whose names begin with PROG but are more or less than five characters (PROGZZ, PROGRESSREPORT, PROG, PROGENY, etc.) or files with non-blank type extensions (PROGA.PAS, etc.).
`CI> dl pr--.@`	This command will list all files with four-character names beginning with PR *and* with an extension of zero or more characters (PROG.DBG, PROB.PAS, PREP.TXT, PREP, etc.). This use of the wildcard characters will *not* list files whose names begin with PR but are more or less than four characters (PROTEST, PRO, PRV.FTN, PROGZZ.DBG, etc.).
`CI> dl @.@`	This command will list all files with blank or non-blank type extensions.

CI file masking also enables you to use mask qualifier fields to perform more complex file selection operations. Mask qualifiers can be used to select open files or temporary files, for example, or select files by creation date or within a range of dates. Mask qualifiers can also direct the search through selected portions of the directory structure.

For more information on file masking, refer to the RTE-A User's Manual.

Ownership and Protection (VC+ Systems)

In the VC+ multiuser environment, you usually "own" your global directory and all others that you create. Ownership allows you to change the protection status (read/write capability) of your files as described below. No other user (except a superuser or a user with a capability level of 31) may override the read or write protection of your files.

If you are not sure if you are the owner of a directory, you can use the OWNER command to find the name of the assigned owner. In the following example, user George owns the directory and Lab is the associated group:

```
CI> owner /applications.dir
Owner of APPLICATIONS is GEORGE.LAB
CI>
```

For each file, you can set the read/write protection to protect the file against unauthorized access. Read/write protection can be set for you (the owner), for the members of the associated group, and for all other users on the system. Generally, you will want to have both read and write access to your own files. You may, however, want to specify write protection for your important files to prevent inadvertent modification of the contents. Normally, your files will be write-protected or read/write-protected from members of the associated group and all other system users, depending on your needs.

To change a file's protection status, use the PROT command. The syntax for the command is:

```
PROT filename ownerRW/groupRW/systemRW
```

For example, the following command will allow the owner and members of the associated group to have both read and write access to file REPORT.TXT, but other system users have only read access to the file.

```
CI> prot report.txt rw/rw/r
```

Note that only the owner of a file, a superuser, or a user with a capability level of 31 can change the protection status of a file.

Directories also can be protected. Write-protected directories prevent the creation of files or subdirectories in the directory. Read-protected directories prevent others from listing the directory (DL) or accessing the files. To protect the files and subdirectories in directory GEORGE from any modification, specify:

```
CI> prot /george.dir.d r/r/r
```

This form of the command specifies write protection (read-only access) for the subdirectories and files in the directory. (The .d mask qualifier following the type extension is a search directive to search this directory, as described in the RTE-A User's Guide.) In this example, the directory owner, members of the associated group, and all others are allowed read-only access.

When files or directories are created, they assume the protection of the directory they are in. To find the current protection status of any file or directory, enter the PROT command with no protection field:

```
CI> prot /george
directory /
   name        prot
GEORGE.DIR     r/r/r
```

4

Using the Editor

The EDIT program provides an easy means of creating a file or modifying an existing file. The editor includes two modes of operation: line mode and screen mode. In line mode you create or edit files one line at a time. In screen mode you can create or edit files in blocks of lines on your terminal screen. (When you initially enter the editor, you are in line mode.) This chapter first describes the screen-mode control commands, then the line-mode control commands.

When a file is created, EDIT writes that file to disk. When you later modify the file, a copy of the file is written to the edit workspace. After the file is closed, EDIT then writes the modified copy to disk, overwriting the original file. (Refer to the section "Saving the Modified File" for more details on the edit workfile.)

Entering and Leaving the Editor

To start the editor program and edit a new file, enter:

```
CI> ru edit
```

or, to use the implied run, enter:

```
CI> edit
```

The editor responds with the message:

```
EDIT : Use ? for help
FI,<file name> specifies file to edit.
EOF
/_
```

The slash (/) is the editor prompt character, indicating that the editor is waiting for input. At this point, you can bring in an existing file for modification or create a new file.

To edit an existing file, call the editor with the runstring:

```
CI> ru edit oldfile.txt
```

or call the editor as above and enter the FI command and the file name, as follows:

```
EDIT: Use ? for help
FI,<file name> specifies file to edit.
EOF
fi,oldfile.txt
opened file OLDFILE.TXT::GEORGE:3:16:24
345 lines read
/_
```

At the end of the edit session, enter the ER (Exit and Replace) command and your edited file will be stored, overwriting the previous copy of the file.

There are three ways to create a file with the editor: You can enter the new file name in the edit runstring (edit filename), you can enter the file name with the editor FI command, or you can call the editor, type in text, and then use the editor EC (Exit and Create) command when you close your new file.

The prompt and response sequence for creating a file in line mode with the FI command is:

```
CI> edit
EDIT: Use ? for help
FI,<file name> specifies file to edit.
EOF
 /fi,/george/newfile.txt
No such file NEWFILE.TXT
An ER or the first WR will create it
EOF
/ enter line of text
/ enter line of text
        .
        .
/ enter last text line
/er
created file NEWFILE.TXT::GEORGE:16:4:2
closed file NEWFILE.TXT::GEORGE:16:4:2
CI>
```

The prompt and response sequence for creating a file with the EC command is:

```
CI> edit
EDIT: Use ? for help
FI,<file name> specifies file to edit.
EOF
/ enter file line
      .
      .
      .
/ enter last file line
/ec,/george/newfile.txt
created file NEWFILE.TXT::GEORGE:16:4:2
closed file NEWFILE.TXT::GEORGE:16:4:2
CI>
```

There is no danger of losing the file lines you have entered. The editor won't let you close the file with an ER command (Exit and Replace the previous copy of the file), nor will it let you use the EC command without giving a file name.

Enter the editor and create the file NEWFILE.TXT, then close it.

The Help Command

EDIT includes a Help command you can use in line mode. Just enter "h" or "?" following the / prompt and EDIT will respond with a summary of all available edit commands. By entering a command code with the "h" or "?", you can get a more detailed description of that command. For example, to obtain information about screen editing, enter either of the following commands:

```
/?S

/hs
```

Screen Mode Editing

In working with the editor, you will most likely use the screen mode, which allows you to modify or create a block of lines using the local terminal editing keys. Enter screen mode with the S command, as follows:

```
/S
```

The editor clears your screen and displays a block of 21 lines, or the entire file if it has less than 21 lines. Each screen is bracketed by two markers. The top marker shows the line number of the first line of the screen and the bottom marker shows the line number of the last line of the screen and the file name. The top marker also shows the commands you can use to get out of screen mode:

```
>>******* line  45 ******* ctrl U reads ***ctrl U ctrl U aborts ******<<
>>------ line  64 ---------- NEWFILE.TXT::GEORGE:16:4:2 ------------<<
```

If you are creating a file, the top and bottom markers will be on adjacent lines at the top of your screen and the bottom marker will show EOF rather than a line number. Use the terminal edit key INSERT LINE to open up the screen for entering lines.

Now you can use all of the terminal editing and positioning keys to create or modify the file. Note that any line longer than 78 characters is continued on the next line. In this case you will see two dots (..) after the 78th character. This signifies a continued line.

The following control key combination commands can be used when editing a screen of text:

CTRL-U	Quit screen mode (use on terminals with Xon/Xoff handshake protocol)
CTRL-Q	Quit screen mode (do not use on terminals with Xon/Xoff handshake protocol)
CTRL-F	Go Forward to the next screen
CTRL-P	Go to the Previous screen
CTRL-T	Start next screen at current cursor position (use on terminals with Xon/Xoff handshake protocol)
CTRL-S	Start next screen at current cursor position (do not use on terminals with Xon/Xoff handshake protocol)
CTRL-A	Go to the beginning of the current line
CTRL-Z	Go to the end of the current line

These commands are entered by holding down the CTRL key and then pressing the appropriate key before pressing the carriage return key. Each command causes the system to read the screen and to replace the text in the workfile with your changes entered on the screen. Remember that the editor works only with a copy of your original file; until you replace the file by writing it back to the disk, nothing changes in the original file.

If you do not want to save the text changes made to the current screen, you can enter a command twice, for example, CTRL-F CTRL-F. Doing this performs the function described above without reading the screen, leaving the work file unchanged.

Once you have learned how to start a screen mode edit, how to move from screen to screen, and how to get out of the screen mode, you can proceed to use the cursor positioning and editing keys to modify your file.

Saving the Modified File

Now that you have modified a copy of your file with the editor, the file with all your changes must be saved. This can be done either by replacing the old file with the modified file (ER command) or creating a new file (EC command), as described in the section "Entering and Leaving the Editor".

Until you write the file to the disk, the original copy of your file on disk remains unchanged. The workfile (in memory) does not replace the original file until you give an ER or WR command and press the carriage return. If you do not want to lose the original copy of your file (as stored on disk), use EC or WC to create a new file and write the work file to it.

For example, you could call up FILE2.FTN for editing. After making extensive changes to it, you might decide that you wish to retain FILE2 in its existing form, but retain the changed file as well. You could then store the edited version under a new name, say FILE3.FTN, by specifying:

```
/ec,/george/file3.ftn
```

If you do not wish to save the changes made to your work file, terminate the EDIT session with the Abort (A) command. If changes were made to the file, EDIT will prompt you to verify that you do, indeed, wish to abort EDIT, losing your changes:

```
/a
OK? y
EDIT aborted by user
end of edit
CI>
```

To find out more about these and other capabilities of the editor, refer to the *EDIT/1000 User's Manual.*

5

Developing a Program

In this chapter you will be directed through the process of writing, compiling, linking and running a simple FORTRAN program. The RTE-A primary system includes the editor (EDIT), the FORTRAN 7X compiler (FTN7X) and the linkage editor (LINK) utility programs so that you can develop and run a FORTRAN program. The languages Pascal and BASIC also are available as optional products. These are described in their respective reference manuals.

Before beginning your program, you should determine which programs and libraries are available for your use. Use the DL command to get a listing of the contents of the /LIBRARIES and /PROGRAMS directories. To see all programs, enter the command (the listing will be similar to that illustrated below):

```
CI> dl /programs/
directory  ::PROGRAMS
AB2MI.RUN    ARSTR.RUN  ASAVE.RUN  AUTOR.RUN  BUILD.RUN
CI.RUN       CIX.RUN    COMND.RUN  COPYL.RUN  CSYS.RUN
DL.RUN       DRTR.RUN   EDIT.RUN   FC.RUN     FMGR.RUN
FORMC.RUN    FPACK.RUN  FPUT.RUN   FREES.RUN  FST.RUN
FTEST.RUN    FTN7X.RUN  FVERI.RUN  HELLO.RUN  GRUMP.RUN
INSTL.RUN    IO.RUN     IS.RUN     LIF.RUN    LINDX.RUN
LINK.RUN     LOGON.RUN  MACRO.RUN  MERGE.RUN  MI2AB.RUN
OLDRE.RUN    PATH.RUN   PRINO.RUN  PRINT.RUN  PROMT.RUN
RS.RUN       RTAGN.RUN  TF.RUN     TIME.RUN   WH.RUN
WHOSD.RUN
CI> _
```

To see all of the libraries available for developing programs, enter the command (the listing will be similar to that illustrated below):

```
CI> dl /libraries/
directory  ::LIBRARIES
$BGCDS.LIB      $BIGDS.LIB       $BIGLB.LIB
$CDSLB.MLB      $CDSONOFF.MLB    $FNDLB.LIB
$MACLB.MLB      $SYSA.LIB        $SYSLB.LIB
BIGNS.LIB       BIGNS_CDS.LIB    PASCAL.LIB
PASCAL_CDS.LIB  PASCAL_ERR.REL   PASCAL_FMGR_ALT.LIB
SEC1000.LIB     SEC1000CDS.LIB
CI> _
```

Program development in FORTRAN consists of writing, compiling and linking a program. Each phase has a corresponding utility program that moves the application program along the path to executable machine instructions. The editor (EDIT) produces the source file, the FORTRAN compiler (FTN7X) generates the relocatable file, and LINK produces the executable program.

The source code is the first file you create when you develop an application program. The source file contains the program name, FORTRAN statements, and comments describing the statements. Remember, because many CI commands are actually programs, it is important to avoid using CI command names when naming your programs.

In RTE-A, a FORTRAN source program is identified by the extension .FTN following the program name. The program you are about to develop is called PROGA; the source file is PROGA.FTN. You must assign this extension when you create the file.

The relocatable program, also referred to as a relocatable module, contains machine code produced by the FTN7X compiler from the source code. The compiler produces a type 5 (binary) relocatable file. The compiler names the relocatable for you by creating a new file whose name is the same as the source file with the .FTN extension changed to .REL (PROGA.REL).

LINK finds and attaches any system routines your program needs and then makes a type 6 (memory-image) file. This is the executable program, and is given the .RUN type extension when LINK creates the program file.

Unless directed otherwise, LINK takes the name from the PROGRAM statement in the file (see line two of Figure 5-1) and uses it to name the executable program. The program you are about to write is short and simple. When you run it, it asks you for some numbers, then displays the sum of the numbers and the sum of the squares of the numbers.

Creating the Source File

Run the editor and start writing PROGA.

```
CI> edit
EDIT: Use ? for help
FI,<filename> specifies file to edit.
EOF
/s
```

(The "s" places you in screen mode. Use the INS LINE key to insert several blank lines, then type in the text for your program.)

```
FTN7X,L
.
.
.
etc.
```

If you make an error, use the EDIT commands given in the previous chapter to make the correction.

Figure 5-1 shows the complete source program. The first line of the program is FTN7X,L. The "L" is an optional compiler control statement that specifies a source listing.

```
FTN7X,          L
      PROGRAM PROGA (3,99),Example Program
      IMPLICIT INTEGER (a-z)
      DIMENSION program_name(3), number_array(0:50)
      DATA sum /0/, square /0/

* Get the numbers

      k = 0    $    number_array(0) = 32767
      WRITE (1, *) 'Enter nonzero integers, or zero to quit.'
      DO WHILE ((k .LT. 50) .AND. (number_array(k) .NE. 0))
            k = k + 1
            high_bound = k
            WRITE (1, '(3X, "integer #", I2, " (0 to stop): _")') k
            READ (1, *) number_array(k)
      END DO

* Don't include the 0 entry as valid data

      IF (high_bound .LT. 50) high_bound = high_bound - 1

* Add the numbers and square the sum

      DO  k = 1, high_bound
           sum = sum + number_array(k)
           square = square +(number_array(k) * number_array(k))
      END DO

* Report the results

      WRITE (1, *) 'The sum of the numbers is', sum , '.'
      WRITE (1, *) 'The sum of the squares is', square, '.'

      END  ! PROGA
```

Figure 5-1. Program PROGA Source Code

The second line, the PROGRAM statement, includes program type and priority (3,99). You can default these values by using just the empty parentheses (), followed by a comma to delimit the comment that follows on the same line. If there is no comment, the parentheses and the comma are not required. Continue copying the program shown in Figure 5-1.

Compiling the Relocatable Program

When you have the source program correct, use the EC (exit and create) command to exit the editor, saving the source under the name PROGA.FTN.

```
/ec proga.ftn
created file PROGA.FTN::GEORGE:16:3:2
closed file PROGA.FTN::GEORGE:16:3:2
end of edit
CI> _
```

Run the compiler (FTN7X) by entering the command:

```
CI> ftn7x,proga.ftn,,-
```

Following the compiler name, the first runstring parameter is the name of the source file. The second parameter is the name of the list file, in this case it is defaulted by a double comma to LU 1 (your terminal display screen). You can also send the listing to a file by specifying the file name or a hyphen (–) for the default of proga.lst. The third parameter is the name of the output file, where the compiler will put the relocatable code. For the output file, the hyphen (–) means use the default name, which is the same as the source file name. The compiler automatically changes the file type extension .FTN to .REL (identifying the file now as relocatable code).

If the compiler finds statements in the source file that it cannot resolve, it displays error messages to aid you in correcting the source file. When the relocatable module is complete, the compiler concludes with the display shown in Figure 5-2.

```
Module PROGA      No errors    Program:  302    Blank Common: None
FTN7X 2326/830509No warnings  Save:     None    Local Ema:    None

$END FTN7X: No disasters, No errors, No warnings.
CI>
```

Figure 5-2. Compiler FTN7X Display

If FTN7X finds a file named PROGA.REL in directory /GEORGE, it overwrites that file with the new output.

When you have successfully compiled the source file, use the DL command to verify that the relocatable file, PROGA.REL, exists in your working directory.

Linking the Program

LINK can be run in more than one way. This example shows how to use LINK commands interactively to link PROGA. The only commands you need right now are the RE (relocate) and the EN (end) commands.

Run LINK as shown in Figure 5-3. You can use the BACKSPACE key for correcting errors in LINK, just as you do in CI, as long as you have not pressed the carriage return key. When you receive the LINK prompt:

```
link:
```

enter the commands as shown in Figure 5-3. After you enter the RE,PROGA.REL command, LINK displays the name of the relocatable file.

As soon as you enter the EN command, LINK executes the commands you entered. LINK displays the externals (modules) referenced by the program, the program load map in memory, and the name of the program file. The output shown in Figure 5-3 includes only a few of over 30 modules attached to PROGA by LINK. The second column of the load map gives the starting location for each module in memory.

If you enter a command LINK cannot understand, you will receive the following message:

```
Unknown command; use ? for help
link:
```

To abort LINK at any time without generating a program file, use the command AB (abort):

```
link: ab
aborting link
CI> _
```

```
CI> link
link version 5000    Use ? for help
link: re,proga.rel
   PROGA
link: en
   PNAME    XREIO  REIO  LOGLU  $CVT1   $CVT3   .PACK  .OPN?    .FION
   .UFMP    .EIO.  .FIO. .IIO.  .FMIN   .FMCN   .FMCV  .FMFP    .FMLD
   .FMIO    .FMUI  .FMO? .FMER  .FMGB   .SST    .IOER  .PAU.E   ERO.E
 MOD
Load map:

   PROGA      2000        302        Example program
   PNAME      2501        24         92071-1X210 REV.2041  800409
   XREIO      2531        104        92077-1X422 REV.2326  <841029.1608>
   REIO       2701        96         92077-1X562 REV.2326  <841029.1608>
   LOGLU      3041        20         92077-1X205 REV.2326  <841029.1608>
   $CVT1      3065        7          92071-1X321 REV.2041  800530
      .         .         .               .          .         .
      .         .         .               .          .         .
      .         .         .               .          .         .
   .IOER      12227       102        24998-1X321 REV.2140 810506
   PAU.E                  12375      1           24998-1X254       REV.2
001 750701
   ER0.E      12176       1          24998-1X249 REV.2001 750701

Main          2000 - 12426          4735.  words

Program PROGA.RUN::6:49 ready; 6 pages
Runnable only on an RTE-A system
CI>
```

Figure 5-3. LINK Display

LINK overwrites a program file of the same name if it already exists. If PROGA.RUN exists, the concluding message from LINK is:

```
Purging old file:PROGA.RUN:::6:41
Program PROGA.RUN:::6:41 ready: 6 pages
CI>
```

You now have a new type 6 executable program file on disk called PROGA.RUN. Note that LINK uses the name specified in the PROGRAM statement, unless otherwise specified. If you want to rename the executable program, specify the new name in the EN command, as follows:

```
link: en newfilename.run
```

When you have successfully linked PROGA, you can use DL to verify its existence.

```
CI> dl,proga.@
directory ::GEORGE
PROGA.FTN  PROGA.REL  PROGA.RUN
```

Because you specified the PROGA file name with the wildcard character @ following the period, DL listed all files of that name. You can see that you still have your source program (.FTN) and your relocatable program file (.REL) in addition to the executable program file (.RUN).

For more information, refer to the RTE-A LINK User's Manual.

Running the Program

With PROGA in an executable (type 6) file on disk, you can run the program from CI. In the example below, the RU command is implied:

```
CI> proga
Enter nonzero integers, or zero to quit.
integer # 1 (0 to stop):1
integer # 2 (0 to stop):3
integer # 3 (0 to stop):5
integer # 4 (0 to stop):7
integer # 5 (0 to stop):0
The sum of the numbers is       16.
The sum of the squares is       84.
CI> _
```

A

System Management

System management is an important aspect in the computer environment. Each computer system should have a person designated as the system manager. This person will be a central source for current system information, and is responsible for maintaining a secure, viable system.

The duties of the system manager include generating new operating systems as needed to add peripherals or to tune system performance; controlling disk use and packing disk volumes when required; and backing up disk volumes as needed in case of accidental loss of files or catastrophic disk failure.

In the VC+ environment, the system manager is a "superuser" and has unrestricted access to all user files, system programs, and user account files. The system manager also will have a normal user account for times when he is not performing system management duties.

More information on the various system management functions can be found in the System Manager's Manual, the System Generation and Installation Manual, the User's Manual, and the System Design Manual.

Index