12 / 6-20-90 JD ①

ANL/TM 455
REVISION 2

# Using the Central VAX 8700 Computer

**at ANL**

Argonne National Laboratory
Computing and Telecommunications Division
Operated by The University of Chicago for the U.S. Department of Energy under Contract W-31-109-Eng-38

This report has been reproduced from the best available copy.

Available from the
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

Price: Printed Copy A06
Microfiche A01

ANL/TM 455
REVISION 2

/

# Using the Central VAX 8700 Computer

ANL/TM--455-Rev.2

DE90 010402

**at ANL**

By

David A. Lifka

Edited by

Torri A. Bennington
Merri Ann Savage

May 1990

Computing and Telecommunications Division
Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439-4801

(Intended primarily for internal distribution)

## ACKNOWLEDGMENTS

Pete Bertoncini, Tom Canfield, Ken Greenlee, Dave Leibfritz, Barry Miller, Fred Moszur, Rich Raffenetti, Mike Thommes, and Linda Winkler reviewed one or more drafts of *Using the Central VAX 8700 Computer at ANL.*

Some information on printing and transferring files came from *Guide to VAX/VMS NJE Networking at ANL* (ANL/TM 444), written by Rich Raffenetti.

Diane Lark and Jim Patterson wrote the original document (August 1987), which was edited by Cliff Caruthers and Chris Opitz. Diane Lark revised the document (September 1988), also edited by Cliff Caruthers.

We have created this document with Waterloo Generalized Markup Language (GML) and Waterloo Script (Version 86.1) for printing on the Linotype L300P typesetter.

# PREFACE

*Using the Central VAX 8700 Computer at ANL* (ANL/TM 455) is a local reference companion to the following manuals from Digital Equipment Corporation: *Introduction to VMS* (AA-LA04A-TE), *Guide to Using VMS* (AA-LA05A-TE), and *Guide to VMS Files and Devices* (AA-LA06-TE). Users of the ANL central VAX 8700 computer should be thoroughly familiar with *Using the Central VAX 8700 Computer at ANL* as well as with *Introduction to VMS* (AA-LA04A-TE). When appropriate, *Using the Central VAX 8700 Computer at ANL* points to information that you can find the vendor documents listed above and in other documents.

Unless otherwise stated, all documents cited in *Using the Central VAX 8700 Computer at ANL* are available for purchase at the Document Distribution Counter (Building 221, Room A-134) or through the mail (call extension 2-5405 to order).

Refer to *Recommended Documentation for Computer Users at ANL* (ANL/TM 379) for a complete list of VAX/VMS documentation available at the Document Distribution Counter. The basic DEC manuals are the *Guide to Using VMS* (AA-LA05A-TE) and *Introduction to VMS* (AA-LA04A-TE)

*Using the Central VAX 8700 Computer at ANL* employs the "^X" notation as shorthand for the terminal keyboard control key and another character ("X") that you are to press while you hold the control key down. The "X" may be either uppercase or lowercase.

For clarity of information, *Using the Central VAX 8700 Computer at ANL* observes the following textual conventions:

1. While the general text of this document is printed in the Times font, examples of system prompts and other system responses appear in `regular Courier` font.

2. Examples of user-issued commands appear in **`Courier-Bold`** font. VMS system commands (including the entire command line) appear in **`UPPERCASE`** (or, at some levels, in **`lowercase`** ), except that user-supplied parameters always appear in **`lowercase`**:

   ```
   $  SEND PRINT filename.filetype nodename username
   ```

   Note that in the example above, "$" is a system prompt followed by a user-issued command line.

3. All references to a user's computer account or identification number (in commands, examples, etc.) will appear as **`username`**.

4. For clarity in referring in text to user options (sometimes referred to as qualifiers), we enclose such options in quotes when these options are in lowercase or include characters other than letters (e.g., "login.com," "/MAIL"). When these options consist of only uppercase letters, we do not enclose them in quotes, because the captial letters themselves establish the distinction clearly (e.g., WASTEBASKET). The only exception to these procedures is in "where" clauses following examples of command lines. In the "where" clauses we always put quotation marks around options, because we are directly quoting portions of the preceding command line.

# CONTENTS

## TABLES

## FIGURES

# CHAPTER 1

# THE CENTRAL VAX CLUSTER: WHAT IT IS AND HOW IT WORKS

The Argonne central Virtual Address Extension (VAX) cluster is a group of Digital Equipment Corporation (DEC) VAX computers in Building 221 that access a common set of disks and tapes. Presently, the cluster consists of:

- A VAX 8700 computer (nodename ANLCV1), for general Virtual Memory System (VMS) interactive and batch use and for communication with other nodes on the ANL Transmission Control Protocol/ Internet Protocol (TCP/IP) and DECnet.

- A VAX 8250 computer (nodename ANLVG), for general VMS interactive use but has few of the licensed application programs that are available on the VAX 8700. It also functions as a file server for Macintosh computers connected to the Lab-wide AppleTalk network.

- A VAX 11/750 computer (nodename ANLVMS), which functions primarily as a high-speed gateway to the Magnetic Fusion Energy network (MFEnet) Cray computers. The VAX 11/750 is not available for general VMS interactive or batch use.

The VAX 8700, the VAX 8250, and the VAX 11/750 share a common file system through a high-speed intelligent controller called a Hierarchical Storage Controller (HSC). These computers (identified by the nodenames cited above) are connected to the Lab-wide DECnet, TCP/IP, and NJE networks and, through these Laboratory networks, to BITnet and other national and international networks. These computers are also accessible from the Argonne Private Branch Exchange (PBX) by using a terminal and a modem.

VT-type terminals (e.g., the VT100) are the most appropriate for communicating with the VAX. You can use Digital Equipment terminals, personal computers with terminal emulating software, or workstations that provide VT100 emulation.

Print queues have been established for a number of printers in several divisions, and the cluster has access to divisional print facilities through the Lab-wide NJE network (see "Transferring Files" and "Printing Files" in Chapters 4 and 6, respectively). For the current configuration of the central VAX cluster, see Figure 1.

## CAPABILITIES OF THE VAX/VMS OPERATING SYSTEM

The DEC VAX/VMS operating system is an interactive and batch operating system that provides four gigabytes (four billion bytes) of virtual address space. By using this virtual address space, users are able to run very large programs with a relatively small amount of physical memory.

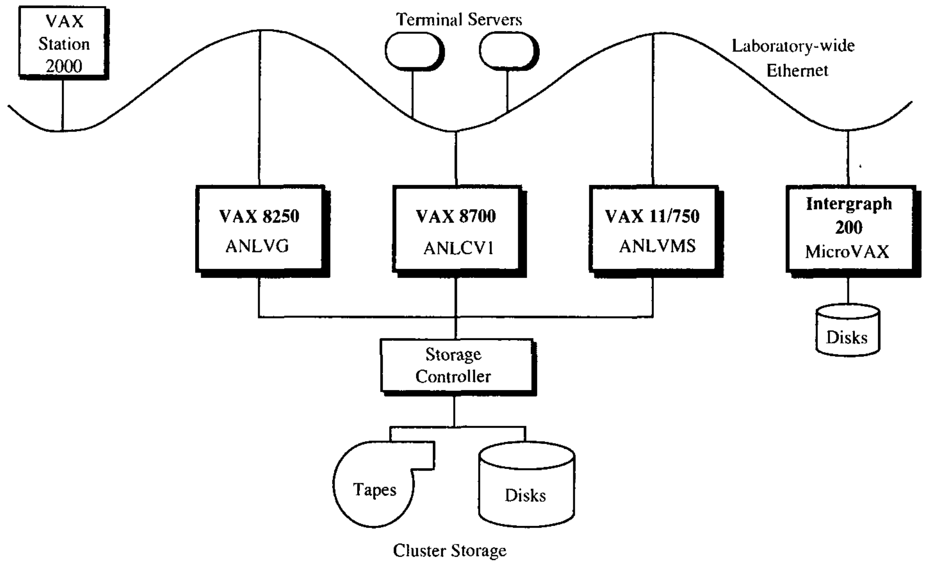


Figure 1: **The Central VAX Computers**

The user's interface to DEC VAX/VMS is the Digital Command Language (DCL). Through DCL, application programs, and vendor supplied software, a user can accomplish a variety of tasks, including:

- Route jobs to the printers in Graphic Arts.
- Access files and initiate interactive sessions on nodes connected to the Lab-wide DECnet and the Internet.
- Create, run, and debug programs.
- Create word processing documents and spreadsheets.
- Format manuals and memoranda and typeset with TeX.
- Create graphs, charts, and documents.
- Perform work in a Unix environment.
- Use the central VAX cluster as a Macintosh file server.
- Send and receive electronic mail.
- Use the VMS PHONE utility to carry on a keyboard conversation with a user logged on to any system connected to the VAX cluster through DECnet.
- Obtain reports of monthly Computing and Telecommunications (CTD) charges.

- Query the Argonne telephone directory.
- Consult the online index of published Argonne computing information.
- Query the documents available through the Document Distribution Counter (Building 221, Room A-134).
- Read CTD online NEWS articles.

## CAPABILITIES OF THE CENTRAL VAX 8700 COMPUTER

Most people who wish to use the VAX/VMS software available through the cluster will be using the central VAX 8700 computer. The VAX 8700 is a general purpose computer equipped with floating-point hardware that uses all four VAX floating point formats (F, D, G, and H). Because of the following factors, the VAX 8700 may offer you greater productivity than earlier VAX computers:

- **A faster processor:** The central processing unit of a VAX 8700 is approximately six times faster than that of a VAX-11/780. Programs with intensive processing (not limited by access to other resources, such as file access) can finish approximately six times faster.
- **More memory:** The central VAX 8700 currently has 128 megabytes of physical memory; it can handle large programs more efficiently (with less paging) than VAXes with less memory.
- **Compatibility for file transfer:** Because internal formats are the same in all VAXes, you can transfer files without having to reformat or otherwise change files. The time it takes to move files to the VAX 8700 will be more than offset by the faster execution times on the VAX 8700.
- **Remote file access:** Files can be accessed by programs running on VAXes through the Lab-wide DECnet.
- **A familiar operating system:** If you are accustomed to working with the DEC VAX/VMS operating system, you will not have to learn a new operating system.
- **Extensive software available:** A large selection of software is available on the VAX 8700 (see Chapters 9 and 10).
- **Powerful Interactive Debugger:** VAX users can easily locate run-time programming or logic errors by using the interactive debugging tool, which can be tailored to individual needs.

## AVAILABLE NETWORKS AND PROTOCOLS

For the benefit of Argonne computer users, CTD has implemented standard protocols that permit ANL to connect to national protocols. A network is a collection of at least two computers that can transfer information between them. A protocol is a set of conventions or rules for the format and timing of information sent and received on networks by computers. The networks and protocols described in the following sections are accessible by Argonne VAX/VMS users on the VAX 8700. For general information on networks and protocols, see *Electronic Mail at ANL* (ANL/TM 431, REVISION 2).

## The Laboratory-Wide TCP/IP Network

The Laboratory-wide Transmission Control Protocol/Internet Protocol (TCP/IP), uses the widely adopted standard TCP/IP network protocol defined by the Department of Defense for communication among different computer systems.

The central VAX (ANLCV1) runs the TGV Multinet communications software, with TELNET and RLOGIN, for connecting to TCP/IP nodes (e.g., the Cray X-MP/14 with nodename "xmp.ctd.anl.gov" and the IBM CMS with nodename "anlvm.ctd.anl.gov."). Multinet also has FTP for transferring files to and from TCP/IP nodes. You can also use TELNET to access the Advanced Computing Research Facility (ACRF) computers (e.g., nodename "alliant.mcs.anl.gov") and other computers on the Lab-wide TCP/IP network.

## The Laboratory-Wide NJE Network

The Lab-wide NJE network is a computer-to-computer file transfer network through which users on the central Argonne computers, on most divisional VAXes, and on some divisional NBI systems can transmit data to other users, computers, and remote output devices. This network allows both file transfer and electronic mail communication with other users on the worldwide BITnet (see Figure 2).

The Lab-wide NJE network uses the IBM Network Job Entry (NJE) communications protocol. Argonne has developed software that emulates this protocol in VMS and provides the means for IBM to DEC communication at ANL. NJE also makes BITnet communications possible. When you receive your VAX 8700 account you will be able to communicate with the IBM systems and BITnet. You will also be enrolled as an NJE user.

For further information on using the Lab-wide NJE network with VAX/VMS, see *Guide to VAX/VMS NJE Networking at ANL* (ANL/TM 444).

## Internet

The Internet is a network of computers at universities, research facilities, and other institutions encompassing DDN, MILnet, NSFnet, and CSnet. Hosts on the Internet are accessible from any Argonne host computer connected to the Lab-wide TCP/IP network.

## BITnet

BITnet is a worldwide network of computers at universities and other research institutions. You can communicate with other users on BITnet through electronic mail and NJE commands. For more information on accessing BITnet, see "Sending and Receiving Mail" in Chapter 2. See also *Guide to VAX/VMS NJE Networking at ANL* (ANL/TM 444) and *Electronic Mail at ANL* (ANL/TM 431, REVISION 2). For online information on BITnet, enter:

```
    $  HELP BITNET
```

To view a current list of BITnet sites on your terminal screen, enter:

```
    $  TYPE/PAGE SYS_ANLDATA:BITNET.NODES
```

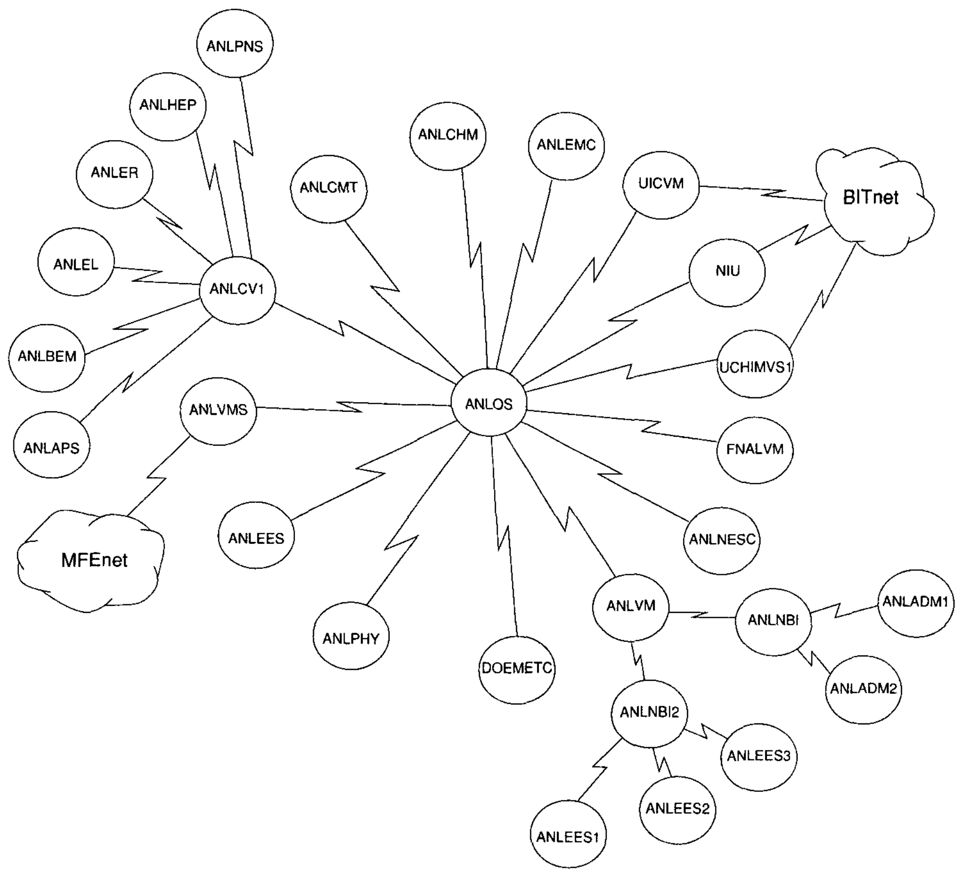


Figure 2: **The Laboratory-Wide NJE Network (February 1990)**

## The Digital Equipment Corporation Network (DECnet)

The Digital Equipment Corporation Network (DECnet) is a network architecture developed by DEC for file transfer and other communication among DEC computers. Many of Argonne's divisional DEC computers (VAXes, PDP-11s, and MicroVAXes), as well as the central VAX computers, connect to the Lab-wide DECnet. For specific information on using DECnet, refer to *Guide to DECnet-VAX Networking* (AA-LA47A-TE). For a description of how to access a DECnet node, refer to "Using DECnet" in Chapter 2.

## The Magnetic Fusion Energy Network (MFEnet)

The Magnetic Fusion Energy Network (MFEnet) allows remote users to access energy research computer systems, including the Cray computers at the National Magnetic Fusion Energy Computer Center (NMFECC) in Livermore, California, and the Cray Computer at Florida State University (FSU). MFEnet is being decommissioned in FY1990 and is being replaced by ESnet.

## ESnet

The DOE Office of Energy Research (OER) under the management of the OER Scientific Computing staff has initiated a new effort in data communications networking designated the Energy Sciences network (ESnet). The ESnet concept brings together the various data communication efforts of the different research programs funded by OER into a single network that would use protocols being developed by the International Standards Organization for Open Systems Interconnect. A phased implementation of ESnet is now underway at ANL. (Currently, ANL connects to ESnet.) By using Cisco routers, ESnet handles dual protocols--TCP/IP (with gateways planned to connect to the Internet) and DECnet (for the HEPnet backbone). A T1 backbone is being installed connecting major nodes. Argonne has T1 connections to Fermi National Accelerator Laboratory and to Princeton Plasma Physics Laboratory and will eventually connect to Oak Ridge National Laboratory.

# CHAPTER 2

# GETTING STARTED WITH THE VAX 8700 COMPUTER AND VAX/VMS

To use the central VAX 8700 computer and VAX/VMS effectively, you should be familiar with the procedures covered in this chapter. You should also be aware of the "Training and Other Available Assistance" described in Chapter 3.

## ESTABLISHING A VMS ACCOUNT

Contact Account Services (Building 221, Room A-147, extension 2-5425) to complete and sign a "Computing and Telecommunications Request for Authorization of Computer Account" (CTD-111). Your signature signifies that you will adhere to the established policy for accounts, which appears on the back of the form (and in Appendix A of this manual).

The standard VMS account on the central VAX 8700 provides access to:

- 10 megabytes of virtual memory.
- 40 megabytes of permanent disk file space consisting of 81,920 blocks of 512 bytes each ("SYS$LOGIN").
- Approximately 250 megabytes of temporary disk file space consisting of 500,000 blocks of 512 bytes each ("SYS$SCRATCH").

You can request larger (or smaller) memory and disk file space quotas at the time you establish your account. We will allocate up to 370 megabytes of virtual memory and as many as 102,400 blocks of disk file space upon request. Approval from your group manager may be necessary for disk file space quotas greater than 102,400 blocks. We will fulfill such requests if sufficient disk resources are available.

You can access temporary disk space through the logical name "SYS$SCRATCH." Note that any files you store on this temporary disk will be automatically deleted seven days after their latest modification. Account Services will charge you daily for any space you use on this scratch disk as well as on the permanent disk. The Computer Operations section performs backups nightly on all permanent and temporary disks.

Your VMS username normally will be Bnnnnn, where "nnnnn" is your payroll badge number; however, it can be a vanity name in some cases. (Account Services use badge numbers as usernames on multiple vendor systems as a means of providing integrated accounting information, establishing authorizations, and protecting tapes and datasets.) CTD will assign you an initial expired password. After logging on for the first time, change your password immediately with the **SET PASSWORD** command. Directions for changing your password appear in the "Logging On" section later in this chapter. If you do not change your password before logging off for your first session, you will be unable to logon a second time.

Account Services has provided you with an initial "login.com" file. It is located in your ROOT directory (the highest level directory of your account). Enter the **DIR** command at the standard VMS "$" prompt to verify the existence of this file. VMS automatically executes this command file whenever you logon to your account. It is comparable to an IBM CMS PROFILE EXEC file, a Unix Bourne shell ".profile," or a C shell ".login" file and should be used to tailor your personal system environment. You will modify this file to create your own VMS environment. Some examples of tasks you might want to add to your login file are (1) setting up your terminal device type, (2) defining function keys, (3) setting symbols or logicals equivalent to command strings or device names, and (4) setting up commonly used products. For instance, if you add the commands lines

```
$  SET TERMINAL/DEVICE=VT100
$  DEFINE/KEY PF3 "SHOW TIME"
$  HOME :== SET DEFAULT SYS$LOGIN
```

to your "login.com" file, the commands will execute each time you logon to your account. The first command line sets your terminal device type to a VT100. The second command line defines the string SHOW TIME to the PF3 key. The third command line creates a symbol HOME to be used as a command whenever you want to set your default directory to the "SYS$LOGIN" logical name. You can learn more about the default directory and logical names in "Managing Files" in Chapter 4.

When you use the VMS MAIL utility, other files are created in your root directory. These files have the filename of "MAIL.MAI." The VMS MAIL utility requires these files. *Do not delete files with an MAI filetype.* To reduce the number of MAI files as much as possible, enter (within the MAIL utility):

```
MAIL>  COMPRESS
```

You can also use the following command from within the MAIL utility

```
MAIL>  SET MAIL_DIRECTORY [username.subdirectory]
```

to specify that all your MAI files be moved from your previous MAIL directory (which by default is "SYS$LOGIN:") to the specified subdirectory.

A READER subdirectory (filename "READER.DIR") has been created for you as well. This subdirectory receives files transferred to your account from other systems. *Do not delete the READER subdirectory.*

## REQUESTING A DECNET PROXY

If you want to access files from the central VAX 8700 over DECnet when using a divisional VAX system, contact Account Services to set up your VAX cluster account with a DECnet proxy. A DECnet proxy on the VAX 8700 will allow easy access to your VAX 8700 files from another VAX system without having to specify your VAX 8700 username and password in the access control string. You may use any file manipulation command. For instance, you can enter

```
$  COPY ANLCV1::[username]filename.filetype
```

rather than

```
$  COPY ANLCV1"username password"::[username]filename.filetype
```

Conversely, if you want to copy files to a divisional VAX from your VAX 8700, contact your divisional VAX manager and request a DECnet proxy on your divisional VAX.

## ACCESSING THE APPROPRIATE VAX CLUSTER COMPUTER

The VAX 8700 computer is available for general interactive and batch use. The VAX 8250 functions as a print server for the Macintosh computers that are connected to the Lab-wide AppleTalk network. The VAX 11/750 is available as a gateway computer to the MFEnet Cray computers. Table 1 lists VAX cluster services and the appropriate access command, depending on where you are coming from (e.g., a terminal server, a DECnet host, a TCP/IP host).

Table 1

**VAX Cluster Services and Access Alternatives**

| *Capabilities* | *Origin* | *Access Command* |
|---|---|---|
| Central VAX 8700 | ASCII Terminals (dial 972-8700) | `connect anlcvl` |
| (Interactive and Batch VMS Services) | Host on the Lab-wide DECnet | `set host anlcvl` |
| | Host on the TCP/IP network | `telnet anlcvl.ctd.anl.gov` |
| Central VAX 8250 | ASCII Terminals (dial 972-8700) | `connect anlvg` |
| | Host on the Lab-wide DECnet | `set host anlvg` |

The VAX 8700 computer is the general purpose computer within the VAX cluster that most people will use. The following examples illustrate numerous ways to access the VAX 8700 computer.

### Configuring Your Terminal

To access a VAX system (e.g., the VAX 8700) and communicate with other VAX systems on the network, you must configure your terminal to the proper settings. If you are using a DEC VT100, VT200, or VT300 series terminal or a VT-compatible terminal, you should not have to modify the manufacturer's settings (except perhaps from numeric keypad to application keypad on the general setup screen). If you do need to configure your terminal, you should use the following settings:

- Full duplex
- No echo
- No parity
- 8 data bits, 1 stop bit
- Flow control (XON/XOFF) on

Kermit users should set the handshake to none by using the Kermit command:

```
Kermit-MS>  set hand none
```

If you have difficulties with configuring your terminal to the proper settings, refer to the user guide to your terminal. If you are still unable to configure the terminal properly, call the consultants at extension 2-5405. When you phone the consultants, please have the user guide in hand for quick reference and be near your terminal.

## Logging On

After establishing an appropriate connection to VMS, follow these four steps:

1. Connect to VMS by pressing the RETURN key. When the system prompts you for your VMS username, enter:

   ```
   Username:  Bnnnnn
   ```

   where "Bnnnnn" is your badge number. If you have a vanity account, enter your assigned username.

2. Next, enter your VMS password:

   ```
   Password:
   ```

   Though you have entered your password, it will not appear on your terminal screen. After the system accepts your password, you will see the VMS welcome message, followed by the standard DCL "$" prompt. Note that you can logoff at any time by following the steps defined in "Logging Off" later in this chapter.

3. When you logon for the first time, you must immediately change your password. Enter the **SET PASSWORD** command and follow the prompts:

   ```
   $  SET PASSWORD
      Old Password:
      New Password:
      Verification:
   ```

   Passwords will not echo to your terminal. Enter the new password again, for verification. Passwords must be at least six characters in length. Avoid using your name, initials, nickname, or any other word easily associated with you. You must change your password at least every six months.

4. Now that you are logged on, you should define your terminal. Use the **SET TERMINAL** command. To see the available terminal types, enter:

   ```
   $  HELP SET TERMINAL/DEVICE_TYPE
   ```

   Several device types exist, but the VT100 is the most common and is widely emulated. To define a DEC VT-type terminal, enter:

   ```
   $  SET TERMINAL/DEVICE_TYPE=vt-type
   ```

   where "vt-type" is VT100, VT200, or VT300 series terminal. The **SHOW TERMINAL** DCL command will display the current VMS settings for your terminal.

### *Using Dial-up Services*

If you have your terminal or personal computer connected to the PBX through an ADI, you can dial extension 2-8700, press the ACCESS button on your telephone, and then press the "#" button twice. Then press the RETURN key on your terminal several times, until the terminal server responds with the "Local>" prompt. For further information refer to *Asynchronous Data Communications User Guide With ADI/ACI 100 Operating Instructions* (ANL/TM 469).

Use the following procedure to connect to the central VAX cluster through a terminal server from a dial-up line and modem:

1. Dial extension 2-8700 in the manner appropriate for your modem.
2. Press the RETURN key until you have the attention of the terminal server. Then logon to the terminal server by selecting the central VAX computer at the terminal server "Local>" prompt:

   ```
   Local>  CONNECT nodename
   ```

   where "nodename" is "ANLCV1" for the VAX 8700 or "ANLVG" for the VAX 8250.

   A range of services is available on the terminal server. You can view a list of the services by entering **HELP** following the "Local>" prompt. Entering **SHOW SERVICES** at the "Local>" prompt will display the nodes available from the terminal server. **SHOW SESSIONS** will display your sessions (you can have up to four VMS sessions running through the terminal server at any one time).
3. Follow the standard login procedure for VAX/VMS (described in "Logging On" above).

### *Using DECnet*

To use the Lab-wide DECnet, you must already have logged on to a computer connected to it. Then (if your computer is a VAX running VMS) connect to the central VAX 8700 by entering:

```
$  SET HOST ANLCV1
```

Once you have established the connection, follow the standard VAX/VMS login procedure (described in "Logging On" above).

You can obtain a log of your session by using the "/log=filename.filetype" option of the **SET HOST** command. For instance,

```
$  SET HOST ANLCV1 /LOG=SESSION.NOTES
```

will log your DECnet session and save it with the filename "SESSION.NOTES" in the current directory of your original VMS session.

### *Using TCP/IP*

To begin a central VAX 8700 session through TCP/IP, follow these procedures:

- Enter (at the DCL prompt):

  ```
  $  TELNET
  ```

You will receive the "TELNET>" prompt.

- For more information about TELNET commands, enter a question mark (at the "TELNET>" prompt):

```
TELNET>  ?
```

- To establish a connection with a TCP/IP node, enter:

```
TELNET>  CONNECT tcp/ip-nodename
```

  or

```
TELNET>  CONNECT tcp/ip-number
```

  where "tcp/ip-nodename" is the name of the node you are trying to access and "tcp/ip-number" is the number you are trying to access. You will then receive its login prompt. (The system will return you to the DCL level when you logoff the remote host.)

- To access ANLVM from the VAX 8700 enter:

```
TELNET>  CONNECT ANLVM.CTD.ANL.GOV
```

  Multinet provides a default keyboard mapping file of the IBM keyboard functions:

```
SYS_ANLDATA:MAP3270.DAT
```

  To create your own keyboard mapping file with your own key definitions, copy and revise the above file and tell Multinet to use your keyboard mapping file rather than the provided default with the following command:

```
$  define map3270 "@filename.filetype"
```

- To return to the DCL level (from the "TELNET>" prompt), enter:

```
TELNET>  EXIT
```

To transfer files between the central VAX 8700 and another TCP/IP node, use the file transfer program (FTP), which implements the TCP/IP File Transfer Protocol:

1. Enter:

```
$  FTP
```

   You will receive the "FTP>" prompt.

2. For information on the available commands, enter:

```
FTP>  HELP
```

   or

```
FTP>   ?
```

3. To establish a connection to a TCP/IP node, enter:

```
FTP>  CONNECT tcp/ip-nodename
```

or

```
FTP>  CONNECT tcp/ip-number
```

Use the **GET** or **PUT** FTP commands to transfer files to or from the remote system and the VAX 8700 computer. Note that when transferring files between the VAX 8700 computer and CMS, CTD recommends that you FTP from CMS to the VAX 8700 computer.

4. To return to the DCL level, enter:

```
FTP>  EXIT
```

For further examples of FTP file transfer, refer to "Transferring Files" in Chapter 4. TCP/IP online help is available by entering:

```
$    HELP TELNET
```

or

```
$    HELP FTP
```

Follow the standard login procedure described in "Logging On" above.

### *Using an IBM Personal Computer or an Apple Macintosh Computer with Kermit*

Versions of Kermit distributed at the CTD Document Distribution Counter emulate a DEC VT102 (full screen) terminal.

To run Kermit on an Apple Macintosh computer, double click on the Kermit application which is labeled appropriately for the baud rate of the modem you use (e.g., VAX1200, VAX2400, or VAX19200). For further information on setting the parameters on Kermit for the Apple Macintosh, see the *Kermit User Guide.*

To run Kermit on an IBM personal computer with a modem or ADI, use the following procedure. Load DOS and run Kermit. Modify the default settings (e.g., the baud rate) where necessary. For instance, to run Kermit and set the defaults, enter:

```
        A>  KERMIT

Kermit-MS>  TAKE VAX.INI
```

CTD provides the initialization file "VAX.INI" on Kermit disks available at the Document Distribution Counter. This file contains the necessary settings for a VMS system.

Set the baud rate appropriate to your configuration:

```
Kermit-MS>  TAKE VAX.INI

Kermit-MS>  SET BAUD 19200   (default is 9600)
```

```
Kermit-MS> SET HAND NONE

Kermit-MS> STATUS   (displays current settings)
```

The appropriate settings for connecting to VMS are (1) set local echo off, (2) set handshake to none, (3) no parity, (4) flow control of XON/XOFF, and (5) terminal emulation of the VT102.

```
Kermit-MS> CONNECT
```

The MS Kermit program maps the VT100 keypad (used in most VMS editors) to the function keys of the personal computer keyboard. Since the personal computer keyboard differs from the VT terminal keyboard, you must translate. For instance, any references to the VT100 PF1 key should be understood as references to the personal computer F1 key. The VT100 keypad 5 key corresponds to the personal computer F10 key. The VT100 keypad 6 key corresponds to the personal computer combination of the shift key and the F1 key. Figure 3 compares the IBM personal computer, the Apple Macintosh II, and the DEC numeric keypads. See Table 2 for corresponding key functions.

CTD created a program--"GOLDKEY.COM," which enables MS Kermit to use the numlock key on the IBM personal computer enhanced keyboard as the VT-terminal Gold key. This program is useful because the numlock key on the IBM personal computer is in the same position as the Gold key on a VT-terminal. MS Kermit alone cannot redefine the numlock key but, by using "GOLDKEY.COM" first, you can redefine it in your MS Kermit keypad initialization file or by using the one provided with "GOLDKEY.COM" called "DECPAD.KEY." To use this tool, enter:

```
A:\ GOLDKEY

A:\ KERMIT

Kermit-MS> TAKE DECPAD.KEY
```

Now you can start your VAX session. If you want to use the numlock function after you leave your VAX session, press shift-numlock to restore it.

MS Kermit "DECPAD.KEY" and "GOLDKEY.COM" are available on the MS Kermit diskettes at the Document Distribution Counter (Building 221, Room A-134) or through the mail (by calling extension 2-5405 and requesting copies) or by downloading them directly from the VAX 8700 from the SYS_PUBLIC Directory. You can access this Directory by entering:

```
$  SET DEFAULT SYS_PUBLIC
```

Then use FTP in binary mode to transfer them from this directory to your IBM personal computer.

For further information on running Kermit on an Apple Macintosh Computer and on an IBM personal computer, see the *Kermit User Guide*.

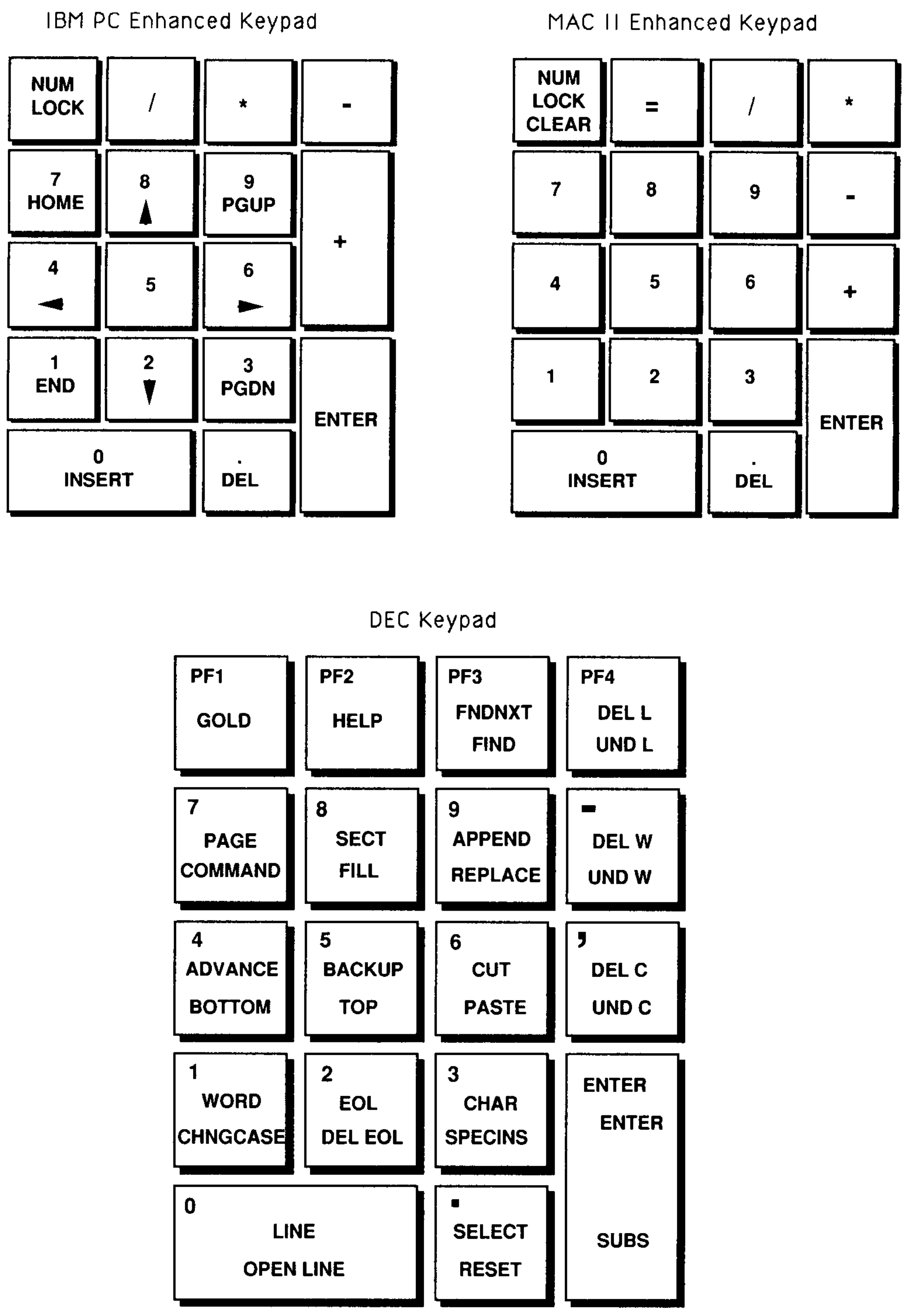


Figure 3: **Personal Computer Numeric Keypads in Contrast with DEC Keypads**

Table 2

**VT100 Terminal Keypad Emulation for the IBM Personal Computer and Apple Macintosh Computer with Kermit and Telnet**

| **EDT Keypad Functions** | **Equivalent Keystrokes** | |
|---|---|---|
| | Apple Macintosh Extended Keyboard | IBM Personal Computer |
| GOLD | numlock clear | F1 |
| HELP | = | F2 |
| FNDNXT | / | F3 |
| DEL L | * | F4 |
| PAGE | 7 | F5 |
| SECT | 8 | F6 |
| APPEND | 9 | F7 |
| DEL W | - | F8 |
| ADVANCE | 4 | F9 |
| BACKUP | 5 | F10 |
| CUT | 6 | shift F1 |
| DEL C | + | shift F2 |
| WORD | 1 | shift F3 |
| EOL | 2 | shift F4 |
| CHAR | 3 | shift F5 |
| ENTER | enter | shift F6 |
| LINE | 0 | shift F7 |
| SELECT | . | shift F8 |
| **To execute the following commands enter the GOLD key first** | | |
| FIND | / | F3 |
| UND L | * | F4 |
| COMMAND | 7 | F5 |
| FILL | 8 | F6 |
| REPLACE | 9 | F7 |
| UND W | - | F8 |
| BOTTOM | 4 | F9 |
| TOP | 5 | F10 |
| PASTE | 6 | shift F1 |
| UND C | + | shift F2 |
| CHNGCASE | 1 | shift F3 |
| DEL EOL | 2 | shift F4 |
| SPECINS | 3 | shift F5 |
| OPEN LINE | enter | shift F6 |
| RESET | 0 | shift F7 |
| SUBS | . | shift F8 |

## Logging Off

To logoff the VAX, enter:

```
$   LOGOUT
```

VMS will respond by printing usage information followed by the date and time.

Users who connect to the central VAX system through a dial-up line and modem must also logoff the terminal server by entering an additional **LOGOUT** after the "Local>" prompt.

For Kermit users, the logoff procedure is the same as the above **LOGOUT** procedure. Once you have entered the **LOGOUT** procedure, you can attract the local Kermit's attention by pressing Ctrl-] and then C. Then (after the "Kermit-MS>" prompt) enter:

```
Kermit-MS>  QUIT
```

## Reconnecting after Inadvertent Disconnects

CTD has set VAX 8700 terminal attributes to allow users who were logged on through a terminal server to reconnect to disconnected sessions. If you are inadvertently disconnected from a VAX 8700 session because of communication difficulties, you have 15 minutes from the disconnect time to resume that session. After 15 minutes, VMS will terminate your session.

To determine the state of your sessions, enter the terminal server command (at the "Local>" prompt):

```
Local> SHOW SESSIONS
```

If your session was truly disconnected, the system will display the following message on the terminal screen:

```
Port 1: username Local Mode
        Current Session NONE
```

You must then connect to the VAX 8700 and follow the standard login procedure. Once you have logged in, the system will notify you that you have a disconnected session and will ask if you would like to restore that session. The system will display the following message on your terminal screen:

```
You have the following disconnected process
Terminal  Process Name  Image Name
LTAXX:      username       XXXXX
Connect to above listed process [YES]:
```

Press the RETURN key and the system will reconnect your session. Pressing the RETURN key again will display the VMS "$" prompt (if that is where you left off). If you were using a full screen editor prior to disconnection, you must also press Ctrl-W to refresh your screen.

If your connection was broken but not disconnected, the system will respond to the **SHOW SESSIONS** command by displaying the following message on your terminal screen:

```
Port 1: username Local mode
        Current Session 1
-Session 1: connected
        Interactive ANLCV1
```

You can then enter:

```
Local>  RESUME [1]
```

The optional number is required only if you have multiple active sessions. CTD terminal servers permit up to four active sessions.

## Using Digital Command Language

The Digital Command Language (DCL) allows you to communicate with the VMS operating system. This language consists of commands, parameters, and qualifiers (sometimes referred to as options). DCL commands are entered at the "$" prompt. When you see this prompt, you are at the DCL level, and the operating system is awaiting your instructions. A command line interpreter (CLI) reads and translates your commands. DCL commands are words that describe the task you want VMS to perform. A parameter is what will be affected by the command. The command qualifiers are the options available for that command. For instance, a DCL command with a parameter would be:

```
$  SHOW TIME
```

where "SHOW" is the DCL command and "TIME" is the parameter. This command will display the date and time to your terminal. The following sample command line contains a DCL command with a qualifier and a parameter:

```
$  SUBMIT/QUEUE=W_BATCH PROCEDURE
```

where "SUBMIT" is the DCL command, "/QUEUE=W_BATCH" is the qualifier, and "PROCEDURE" is the parameter (referring to the command procedure named "PROCEDURE.COM"). This command will instruct DCL to add the command procedure "PROCEDURE.COM" to the "W_BATCH" queue.

A command entered with parameters and qualifiers at the DCL level is referred to as a command string. This command string must follow specific format rules or syntax. The correct syntax for each DCL command is available through the online HELP utility by entering (at the DCL level):

```
$  HELP command
```

Refer to "Online Help" later in this chapter for details on using the VMS HELP utility.

You can use DCL commands to write command procedures which are files containing DCL commands. You can execute these files, and the commands in them will be invoked automatically. These procedures can be structured to resemble a high-level language by equating logical names, symbols, and lexical functions to variables. DCL expressions can manipulate the variables.

A logical name typically represents a file, directory, command, or device. It is used to achieve device independence. For instance, you can assign a filename to the logical name for your terminal ("SYS$OUTPUT"):

```
$  ASSIGN OUTPUT.DAT SYS$OUTPUT
```

Any commands you issue following this assignment that would cause output to be written to your terminal would be written instead to a file named "OUTPUT.DAT."

You can equate a character string or integer value with a symbol in DCL. For instance, you can equate the symbol "SHOQUE" to the command:

```
$  SHOQUE=="SHOW QUEUE/ALL SYS$BATCH"
```

When you enter this symbol at the DCL level, the command string will be substituted and then executed. Symbols can be used as variables in expressions, or as parameters being passed to and from command procedures. You can create local symbols (current command level) by using the assignment statement (=), and you can create global symbols (all command levels) by using two equal signs:

```
$  SS = "SHOW SYMBOL"
$  HOME == "SET DEFAULT SYS$LOGIN"
```

When you enter the symbol "SS," the command **"SHOW SYMBOL"** will be executed, and the system will prompt you for a symbol. This symbol is active for the current command level; when this command level becomes inactive, the symbol will be deleted. Global symbols are used when you do not want the symbols to be deleted.

Local symbols are commonly defined in a command procedure and used only in that command procedure. When the command procedure completes, the symbol will be deleted. When you enter the symbol HOME, your default string will be changed to your root directory ("SYS$LOGIN"). This symbol is available for the entire process unless you explicitly delete it with a **DELETE/SYMBOL** command.

Lexical functions are also available. You can use them in command language expressions. DCL evaluates the function and replaces the function with the returned value. Each function returns one value. The value can be an integer or a character string depending on the function. When used in an expression, the symbol is equated to the result of the expression:

```
$  NMBR = F$LENGTH("LENGTH_OF_WORD") + 1
$  SHOW SYMBOL NMBR
         NMBR = 15  Hex = 0000000F  Octal = 00000000017
```

The symbol NMBR will be equal to the value returned from the lexical function "F$LENGTH" (in the case of the example above, the integer 15, which is the length of the string within quotes plus one). You can show the value of a symbol by entering:

```
$  SHOW SYMBOL symbol-name
```

If the value is an integer, it will be displayed to your terminal screen in decimal, hexadecimal, and octal representations.

Note, do not define symbols that have the same names as DCL commands because you will lose the functionality of those DCL commands for the life of that session. For instance, if you redefine the **DIRECTORY** command to the **SHOW DEFAULT** command, **DIRECTORY** will display your current default string instead of listing the the files in the current directory.

For detailed information on DCL and command procedures, refer to the DEC Version 5.0 manuals *DCL Concepts Manual* (AA-LA10A-TE) and *Guide to Using VMS Command Procedures* (AA-LA11A-TE).

## EDITING FILES WITH THE EVE EDITOR

The Extensible VAX Editor (EVE) is available by using the EDIT "/TPU" option. With EVE, you can define your own editing environment, and windows are available. To learn more about the EVE editor, see the *VAXTPU Reference EVE* (AA-LA14A-TE) or the online tutorial, by entering:

```
$  RUN EVECAI
```

To invoke the EVE editor, enter:

```
$  EDIT/TPU filename.filetype
```

The editor will check your terminal device type. Note that EVE is designed for use with a VT200 or VT300 keyboard. If your device type is defined as a VT100, EVE maps the VT200 and VT300 keys (e.g., DO, Prev Screen, Next Screen) to the VT100 keypad. The VT200 DO key, for example, is mapped to the VT100 PF4 key. A display of VT200 (and VT300) mappings to the VT100 keypad is available on the VT100 by pressing the PF2 key. If your device type is defined as a VT200 or a VT300, you simply use the keys normally (e.g., press the Next Screen key to view the next screen of the file).

You can also use EVE with EDT editor keypad definitions. To do this, create a file called "EVEINIT.EDT" in your root directory with the following lines in it:

```
SET KEYPAD EDT
SET CURSOR BOUND
SET SCROLL MARGINS 15% 15%
```

Then add the following line to your "login.com":

```
$  EVE =="EDIT /TPU /INITIAL=SYS$LOGIN:EVEINIT.EDT"
```

To use EVE in the EDT mode, enter:

```
$  EVE
```

To issue EVE commands, press the DO key, which will give you the "Command:" prompt. Help is available for EVE commands by entering **HELP** at this prompt.

Using the **EXIT** command (at the "Command:" prompt) to terminate your session will save your file; using the **QUIT** command will terminate your session without saving your file.

EVE also allows you to:

- Edit multiple files in multiple buffers.
- Define multiple windows of different sizes.
- Execute DCL commands within an edit session.
- Create a customized editing environment.

## EDITING FILES WITH THE EDT EDITOR

Another editor available in VMS is EDT. This editor is the default editor when you enter the **EDIT** command without any options:

```
$  EDIT filename.filetype
```

The above command will invoke EDT in line-editing mode. (HELP files are available at this level by entering **HELP** at the asterisk prompt.) Enter **QUIT** or **EXIT** to terminate your EDT session and to return to the DCL prompt. If you are working on a VT100-compatible, a VT200-compatible, or a VT300-compatible terminal, you can use the full screen editor. To invoke the full screen mode, enter (following the asterisk prompt):

```
*  CHANGE
```

Help (including a description of the function of each keypad key) is available by pressing PF2. To terminate the session, press Ctrl-Z (to return to the "*" prompt) and then enter **EXIT** or **QUIT**. The **EXIT** command will save your changes and return you to the DCL prompt; the **QUIT** command will not save your changes and return you to the DCL prompt. To change your EDT editor default setting, create a file called "EDTINI.EDT" in your root directory by entering:

```
SET MODE CHANGE
SET SCREEN 72
SET NOTRUNCATE
SET WRAP 72
SET TAB 8
SET TEXT END "--[bottom]--"
```

You can also redefine keys in this file by entering:

```
define key gold D as "DATE"
```

Then add the following line to your "login.com" file:

```
$  EDIT  :=="EDIT/COMMAND=SYS$LOGIN:EDTINI.EDT"
```

While working in the full screen mode of any VMS editor, you may occasionally receive broadcast messages (i.e., notification of mail reception) that may obliterate portions of information from your screen. To refresh the screen, press Ctrl-W.

## GETTING ONLINE HELP

An extensive online HELP utility is available for the VAX utilities and the DCL commands. See Table 3 for a listing of available information.

Table 3

**Available Help in DCL**

| *Information Needed* | *Command to Enter* |
|---|---|
| A listing of all DCL commands | **HELP** |
| Exit from HELP and return to the DCL "$" prompt | Press RETURN key at the "topic?" prompt or press Ctrl-C. |
| Specific commands or topics | **HELP** topic |
| Kermit, the utilities available on the central VAX cluster, CRJOB, CHARGES, or SETUP, USING TAPES, AND TCP_IP | **HELP @CVCUTIL** |
| Locally written tools that are distributed to other divisional VAXes | **HELP @ANLUTIL** |
| Locally written graphics tools | **HELP @GRAPHICS** |
| File transfers, electronic mail, and printing for IBM printers | **HELP @NJE** |
| Magnetic Fusion Energy software (MFE) | **HELP @MFE** |
| Numerical Algorithms Group (NAG) Scientific Mathematical Library of Routines | **HELP @NAG** |
| IMSL | **HELP @IMSL** |
| Cray Super Computer Gateway access | **HELP @CRAY** |

## USING THE SETUP COMMAND

The main purpose of the **SETUP** command is to reduce login time and minimize logical symbol conflicts. The user's global symbol table installed in VMS does not include all possible symbols (representing command strings) or system logical names (representing system directories and subdirectories) available on the central VAX 8700.

You can view a list of products whose symbols and logical names are *not* included in the global symbol table by entering the **SETUP LIST** command. Currently, these include ANSYS, CERNLIB, Disspla, ESnet, GNUEMACS, Ingres, MACSYMA, Mass-11, NAG, PME, Prolog, SAS, Tellagraf, TeX, Tool Pack,and WYLVAX. Before invoking these products you must create the symbols and logical names that define the products' environment. To do this you must add the symbols and logicals to the global symbol table with the **SETUP** command. For instance, if you need to run TeX on a file, you must first enter:

```
$  SETUP TEX
$  TEX  filename
```

You do not need to reissue the **SETUP** command each time you run the program; once per session or batch job is sufficient. If you use a product often, you can put the **SETUP** command with the appropriate product name in your "login.com" file so that it is set up automatically each time you login.

To learn more about **SETUP,** enter:

```
$  HELP SETUP
```

## CONTROLLING YOUR VAX SESSION

VMS provides special control keystroke sequences in VMS to control your interactive session. You can enter these keystrokes from DEC terminals, non-DEC CRT (screen) terminals, and hardcopy ASCII terminals.

Occasionally, a program or system command that you have executed will produce no feedback and appear to have stalled. Enter ^T and the system will display information about the executing command and accumulated processor time in seconds. You can enter another ^T and compare the processing times to see if execution is continuing. You can then use this information to help determine your next action, whether to let your program continue or to cancel it.

To interrupt a program (or command) that you have initiated, use ^Y which will return you to the DCL prompt level. At this point you may issue (a) the VMS **CONTINUE** command, (b) a VMS command that will permit you to resume the interrupted program later, or (c) a VMS command that will not permit you to resume the interrupted program later. Whether you can or cannot continue your program depends upon the command you issue at the DCL level. If that command requires that another image (or program) be loaded into memory, then the system will overwrite your interrupted program, and you will be unable to continue. For instance, reading MAIL would not allow you to continue, but displaying the contents of your directory through the DCL **DIR** command would allow you to continue. The **SPAWN** and **SET** commands will also allow you to continue. They do not require loading in special programs over your current program.

The **CONTINUE** command will cause the interrupted program to begin at the point where the interruption occurred. The ^Y request interrupts and stores the entire context (state) of the running program.

Use the DCL **EXIT** command to terminate a suspended program (the system will delete the image of the interrupted program). Termination will occur if you log off from a suspended system. Table 4 lists the keystrokes and their corresponding functions.

Table 4

**Keystrokes and Commands for Controlling VAX Sessions**

| *Keystrokes* | *Function* |
|---|---|
| ^T | Display Status of Currently Executing Commands |
| ^Y, ^C | Suspend an Executing Program |
| ^Z | Terminate Program Input |
| ^W | Refresh the Terminal Screen |
| ^S | Stop Scrolling on Terminal Screen |
| ^Q | Resume Scrolling on Terminal Screen |
| **CONTINUE** | Resume Interrupted Program |
| **EXIT** | Terminate Interrupted Program |

The ^C is another type of interrupt directive. When you enter ^C while a program is executing, the ^C will either behave like a ^Y, or it will exhibit some program-specific behavior. For example, the command line

```
$  TYPE LOGIN.COM;*
```

displays all versions of your login command file on your terminal. If there were multiple versions and you did not wish to view the rest of the version currently being displayed, you could enter ^C. The **TYPE** command would then begin processing the next most recent version. The ^C interrupt is available to programmers; see language-specific documentation such as the *VAX Fortran User Manual* (AA-D035E-TE) to learn how to use the ^C interrupt in programs that you run. If no specific action is programmed, then ^C behaves like the ^Y interrupt.

Another type of stop directive (or alternately the command **$EOD**) is ^Z, which signifies end-of-input to programs that are reading from the terminal input device (keyboard), either with or without a program-specific input prompt. Two typical examples are the DCL **CREATE** command and the MAIL utility. The **CREATE** command creates a file containing the records that you enter at the terminal after you enter **CREATE.** There is no prompting; each input record becomes a part of the file until you signal end of input with the ^Z directive. A session would appear as follows:

```
$  CREATE FILE.TMP
This is record one.
 . . .
last record
^Z
$ . . .
```

In most utility programs that prompt you for input from the terminal, you can end the session by either pressing ^Z or entering **EXIT**. For example, after you send a message with the MAIL utility in interactive mode, the program prompt appears again. To end the session, answer with ^Z or **EXIT**:

```
MAIL>  ^Z
$  . . .
```

While working in the full screen mode of any VMS editor, you may occasionally receive broadcast messages (i.e., notification of mail reception) that may obliterate portions of information from your screen. To refresh the screen, press ^W.

Sometimes information may scroll up your terminal screen faster than you can read it. Press ^S to stop the scrolling; press ^Q to resume the scrolling. Some keyboards have a convenient scroll/noscroll key that alternatively issues ^S and ^Q to the attached computer (e.g., the HOLD SESSION key on a VT100 or the scroll lock on an IBM personal computer).

## EDITING COMMAND LINES

Occasionally, command lines that users enter do not execute because of typographical errors. To avoid retyping the command line, use the VMS **RECALL** command to retrieve the last command, edit it to fix the error, and reissue the command.

### Recalling the Command Line

To recall previously entered command lines:

- If you have a Digital VTnnn terminal or any other terminal or personal computer that emulates a Digital VTnnn terminal, use the UP cursor key to recall the last command. You can recall the last 20 unique commands (not including successive identical commands, null commands, or the **RECALL** command). You can also use the DOWN cursor key in case you back up past the command line you need.

- If your terminal has no cursor keys and no CRT display (e.g., the TI Silent 700), enter the VMS **RECALL** command to display either all 20 previous command lines, one specific command line (by its relative position number within the last 20 commands), or one of the last 20 command lines (by matching a few of the initial characters in the line). To obtain further information, enter:

  ```
  $  HELP RECALL
  ```

## Editing the Recalled Command Line

You can select either the "/INSERT" or "/OVERSTRIKE" option on the **SET TERMINAL** command line, depending on your preference. In "/INSERT" mode, characters enter to the left of the cursor and do not replace existing characters on the line. In "/OVERSTRIKE" mode, characters enter and replace the character at the cursor position. You may toggle the current "/INSERT" or "/OVERSTRIKE" mode by pressing ^A.

Use the LEFT cursor key (←) and the RIGHT cursor key (→) to move the cursor within the command line. Use the BACKSPACE key to move the cursor to the beginning of the current command line. Use the DELETE key to delete the character to the left of the cursor.

You may edit recalled command lines after you have activated the command line editing feature by entering:

```
$  SET TERMINAL/LINE_EDIT/edit-option
```

where "edit-option" is either "/INSERT" or "OVERSTRIKE" to reflect your line editing preference.

## Alternate Keystroke Sequences

You can use certain two-key keystrokes to obtain additional line-editing features and to substitute for cursor movement keys and other special keys that may not be available on your terminal keyboard. These keystrokes consist of typing a character while holding the Control key down. Table 5 lists command line editing keystroke sequences and their functions.

Table 5

**Command Line Editing Keystroke Sequences**

| *Single Keystroke* | | *Action* | *Alternate Keystrokes* |
|---|---|---|---|
| Cursor Up | (↑) | Recall previous command line | ^B |
| Cursor Right | (→) | Move cursor to right | ^F |
| Cursor Left | (←) | Move cursor to left | ^D |
| Backspace | | Move cursor to start of line | ^H |
| -- | | Delete characters to start of line | ^X |
| Line Feed | | Delete characters back to a separator | ^J |
| -- | | Move cursor to end of line | ^E |
| -- | | Switch between OVERSTRIKE and INSERT modes | ^A |

## SENDING AND RECEIVING MAIL

You can invoke the MAIL utility by entering:

```
$ MAIL
```

When you logon to VMS, the system will notify you of any unread messages. Please note, however, that occasionally messages or files sent to you from other systems (e.g., BITnet) are incompatible with the MAIL file structure and are saved in your "[.READER]" subdirectory. VMS will not notify you (unless you are logged on at the time) when such files have been sent to your "[.READER]." Files transferred from other IBM nodes (ANLVM and ANLOS) will not appear in MAIL either. Files sent with the CMS NOTE utility will appear as new messages in the VMS MAIL utility.

Adding the following statement to your "login.com" file will display to your terminal any files in your [".READER"] subdirectory each time you login:

```
$ DIRECTORY/DATE [.READER]
```

Any messages received by the MAIL utility will be stored in folders. Three folders are used by MAIL on a regular basis. The NEWMAIL folder contains any unread messages received in MAIL. After you have read a message with the **READ** command, MAIL will file it in your MAIL folder, unless you file it in some other folder with the **FILE foldername** command, where "foldername" is the name of the folder the message is to be filed in. MAIL will automatically create this folder if it does not already exist.

After reading your message, you may want to discard it with the **DELETE** command. The message is then transferred to your WASTEBASKET folder, which is emptied when you **EXIT** MAIL. The system saves your messages in your WASTEBASKET folder if you **QUIT** MAIL. You can display the folders containing messages by entering:

```
MAIL> DIRECTORY/FOLDERS
```

The system groups and stores folders in a file named "MAIL.MAI." The system creates other ".MAI" files as the volume of your saved messages increases. You should never delete these files. You can instruct the MAIL utility to save these files in a subdirectory with the following example of the **SET MAIL_DIRECTORY** command:

```
MAIL> SET MAIL_DIRECTORY [.dummy]
```

where "dummy" is the name you have given to a subdirectory in which the MAIL utility will store its files. This command is useful when you are using the **DIRECTORY** DCL command in your root directory, because it prevents numerous ".MAI" files from appearing on your screen.

Another parameter you can set within the MAIL utility is your name. Whenever you mail a note to someone, your username is used rather than your personal name. To identify yourself by a nickname or a full name, enter:

```
MAIL> SET PERSONAL_NAME "first-name last-name"
```

To access messages, use the **READ** command at the "MAIL>" prompt. The **READ** command displays the messages to your screen sequentially. You can select specific messages in your folder by entering:

```
MAIL> READ num
```

where "num" is the number of the message within the folder. Entering just the number of the message (at the "MAIL>" prompt) will work as well.

You can read messages from other folders by entering:

```
MAIL>  SELECT foldername
```

where "foldername" is the name of the folder that contains the message(s) you would like to read. The system then notifies you of the number of messages contained in that folder. To display information about the messages contained in the current folder, enter:

```
MAIL>  DIRECTORY
```

The **SEND** and **MAIL** commands are used to send messages through the MAIL utility: When you issue these commands (at the "MAIL>" prompt), the system prompts you for the "To:" and the "Subject:."

To send a message to a user on the same VAX node, enter:

```
To:  username
```

To send a message to a user on another VAX node, enter:

```
To:  node::username
```

Use this format from within the MAIL utility. For instance, to send a message to user ABCDE at node BNLDAG (on BITnet), enter:

```
To:  GATEWAY::"ABCDE@BNLDAG"
```

where "ABCDE" is the name of the user and node "BNLDAG" is Brookhaven National Laboratory.

To send a message to a user in CMS, enter:

```
To:  ANLVM::username
```

To send the same message to a user with an Internet address, use the Gateway Convention:

```
To:  GATEWAY::"B36857@XMP.CTD.ANL.GOV"
```

or to a user on the Alliant computer in the ACRF:

```
To:  GATEWAY::"B36857@ALLIANT.CTD.ANL.GOV"
```

To send the same message to several users simultaneously, use the **SEND/CC_PROMPT** command:

```
MAIL> SEND/CC_PROMPT

To:   GATEWAY::"B36857@ANLVM"

CC:   B12345,GATEWAY::"B54321@XMP.CTD.ANL.GOV"
```

For further information on accessing Gateways from VMS MAIL, see *Electronic Mail at ANL* (ANL/TM 431, REVISION 2). Online information for BITnet MAIL is available by entering:

```
MAIL> HELP BITNET_MAIL
```

After you have responded to the "To:" and "Subject:" prompts, you will be in a line editor to create your message. The **MAIL/EDIT** command (at the DCL prompt) will invoke the EDT editor, from which you can select the full screen editor (at the EDT "*" prompt) through the **CHANGE** command. For the **CHANGE** command to execute successfully, you must set your device type to VT100 or some other full screen VT terminal when you log on. The **MAIL** command (without the "/EDIT" qualifier) will provide a line editor without any keypad functions, unless you have the command **SET MODE CHANGE** in your "EDTINI.EDT" file. To avoid using the "/EDIT" qualifier, add the following statement to your "login.com" file:

```
$ MAIL :== mail/edit=(reply=extract,forward,send)
```

This symbol will provide the EDT editor with full screen capabilities whenever you issue the **REPLY, FORWARD, SEND,** or **MAIL** command at the "MAIL>" prompt. Additionally, when you reply to or forward a message, the message will be brought into your editor buffer for manipulation. You can refer to the message while you compose your answer and then delete it, or you can use it as part of your response. Press ^Z and then enter (at the asterisk prompt) **EXIT** to send your message or **QUIT** to cancel your message. Entering the **EXIT** command or pressing ^Z at the "MAIL>" prompt will return you to the DCL level.

A wide range of features are available within this utility (e.g., you can set a default editor, forward your 8700 mail to your divisional VAX, or other TCP/IP, ANL NJE, or BITnet nodes, and print mail messages). For more details, enter **HELP** at the "MAIL>" prompt.

An introduction to the MAIL utility is included in the online *Introduction to VMS* tutorial. To receive mail, tell users your electronic mail address (ANLCV1) and your username. The actual gateway address format they will use depends on the convention of the originating mail system (e.g., ARPAnet, BITnet, MFEnet). See *Electronic Mail at ANL* (ANL/TM 431, REVISION 2).

## DETERMINING VAX CHARGES

Several methods are available for reviewing your computing charges for the VAX 8700 computer.

To view the charges for the current process in either interactive or batch modes, enter:

```
$ CHARGES
```

To view information about your monthly charges for multiple vendor systems, including those for central VAX sessions, enter:

```
$  USERDETAIL
```

Then follow the prompts.

To obtain a month-to-date summary of charges for each userid in your cost center, enter:

```
$  COSTCSUM
```

Note that while you can run each of these programs at any time, the charges for running them may be substantial. We recommend that you run them infrequently--normally once a month at the end of the month.

# CHAPTER 3

# TRAINING AND OTHER AVAILABLE ASSISTANCE

Argonne VAX users can receive assistance by referring to VMS documentation, tutorials, VAX/VMS classes, consulting services, computer operations services, and other tools.

## VMS DOCUMENTATION

The most comprehensive sources of technical information on VAX/VMS are the various DEC VMS manuals and this local supplemental manual. For assistance in selecting VAX/VMS manuals that will best meet your needs, refer to *Recommended Documentation for Computer Users at ANL* (ANL/TM 379). Documents listed in this manual can be purchased at the Document Distribution Counter (Building 221, Room A-134) or by calling extension 2-5405 and requesting a copy through the mail.

To receive online information on documents currently available at the Document Distribution Counter (and their prices), enter:

```
$   DOCUMENT topic
```

where "topic" is a portion of the title of a document, the ANL/TM number of a document, or a topic for which you want a list of relevant documents. To view a complete listing of documents available, substitute **all** for the "topic" parameter. The **DOCUMENT** command will send a request to the IBM CMS computer system for the document information. You will receive a file in your READER subdirectory (named "DOCUMENT.TOPIC") containing your requested information.

To view master index entries (a compilation of index entries from all available Argonne technical memoranda and the *Argonne Computing Newsletter)* on a given topic, enter:

```
$   LOOKUP topic
```

where "topic" is the name of a topic for which you want page references in available documents. The **LOOKUP** command will send a request to the IBM CMS system for the document information. You will receive a file in your READER subdirectory (named "LOOKUP.TOPIC") containing your requested information.

Information about VMS commands and ANL-provided tools is also available through the VMS HELP facility. This online information can be useful whether you are a VMS novice or an experienced user. To learn how to use the VMS HELP utility, see "Getting Online Help" in Chapter 2.

## TUTORIALS

Several online tutorials are available on the central VAX cluster, including *Introduction to VAX/VMS, Introduction to the Extensible VAX Editor (EVE), Datatrieve for Users, Datatrieve for Programmers,* and *Using the VAX Language Sensitive Editor (LSE).*

VMS file structure and MAIL are two of the topics covered in *Introduction to VAX/VMS. Introduction to the Extensible VAX Editor* covers how to create and save text with EVE and how to define and create your own environment. *Datatrieve for Users* covers how to create and access a database by using the VAX Datatrieve language. *Datatrieve for Programmers* introduces you to Datatrieve and teaches you how to design complex data structures with it. *Using the VAX Language Sensitive Editor* teaches you how to use LSEDIT, an editor that will enable you to develop and document software systems written in any VMS language. Allow yourself several hours for each tutorial; you may interrupt your training session and resume it at another time. You must have a VT-type emulator (VT100, VT200, or VT300 series terminal) to use the tutorials.

Table 6

**Training Sessions for Tutorials**

To begin the training sessions, logon to the central VAX 8700 and enter the appropriate command:

- For the *Introduction to VAX/VMS* course, enter:

```
$ RUN VMSCAI
```

- For the *Using the VAX Language Sensitive Editor* course, enter:

```
$ RUN LSECAI
```

- For the *Introduction to the Extensible VAX Editor* course, enter:

```
$ RUN EVECAI
```

- For the *Datatrieve for Users* course, enter:

```
$ RUN DTRCAI
```

- For the *Datatrieve for Programmers* course, enter:

```
$ RUN DTRPCAI
```

## VAX/VMS CLASSES

Several times each year, CTD offers a number of introductory VMS computing courses. *Introduction to VAX/VMS* is for first-time VAX/VMS users who need an overview of the features available in VAX/VMS. Attendees will become familiar with available VMS documentation and will learn how to login to VMS, create files, set up subdirectories, compile and link programs, and use online HELP facilities. *Programming in VAX/VMS* acquaints VMS users with features of VMS. Topics include programming VAX Fortran, writing Digital Command Language (DCL) procedures, reviewing VMS internals, and using the VMS system debugger, the run-time library, and system services. Announcements of course offerings appear in the *Argonne Bulletin* and the monthly *Argonne Computing Newsletter*. To register for a course, call or visit the CTD Consulting Office (Building 221, Room A-139, extension 2-5405). Registration is necessary so that we can gauge the size of classes and notify attendees of schedule changes. CTD may cancel or reschedule classes with fewer than six registered attendees.

## CONSULTING SERVICES

Direct your requests for VMS assistance to the Consulting Office (Building 221, Room A-139, extension 2-5405). If you call after normal working hours, leave a message on the answering machine or send a brief description of your difficulties and an address (or extension) where you can be reached through VMS electronic mail. Address your mail message to CONSULT:

```
$ MAIL
Mail>  SEND
To:    CONSULT
```

## COMPUTER OPERATIONS SERVICES

The Computer Operations Section provides file backup, file restoration, and tape management.

### File Backup

To protect your data from inadvertent loss or destruction, the CTD Operations staff copies disk files to magnetic tape. A schedule of weekly full disk volume backups and daily incremental backups ensures that all files that are in your account for more than one day will be protected from loss. If a disk volume is destroyed, the data before the last backup can be recovered completely. At most, the changes that occurred during the hours following the backup will not be recoverable. The magnetic tapes for the full backup operations are cycled each three weeks, with one of the cycles stored at a different location. Backup tapes are normally used to protect the files on a full disk volume but are also used to restore selected files.

To retrieve files older than three weeks, you should maintain your own file backups. See information on the BACKUP utility in "Deallocating the Tape Drive" in Chapter 5.

### File Restoration

If you delete a central VAX cluster file by mistake that existed in the system at least one day earlier, you can recover it by calling Account Services (Building 221, Room A-147, extension 2-5425), and requesting restoration of the file (provide the exact file specification). See the Computing Services Rate Sheet for up-to-date charges on restoring files.

### Tape Management

For detailed information on tape management, see Chapter 5.

## MISCELLANEOUS ARGONNE TOOLS

To obtain an ANL employee's telephone number, location, and badge number, enter:

```
$  ANLPHONE employee
```

where "employee" is the employee's last name or ANL badge number. This command will send a request to the IBM CMS system which will create a file called "ANLPHONE.EMPLOYEE" in your READER subdirectory. To obtain the information you want, type out the file on your terminal screen.

To obtain a list of current special CTD online announcements, enter:

```
$  NEWS
```

The system will then prompt you for the topic of interest.

To create a report detailing your monthly central computing activity and charges, enter:

```
$  USERDETAIL
```

Then follow the prompts. After prompting, the system will submit a job to IBM MVS batch and route your output to a local output device (i.e., a printer located in Building 221), unless you specify otherwise during the session.

To view summary charges for an entire cost center, enter:

```
$  COSTCSUM
```

Then follow the prompts. After prompting, the system will submit a job to IBM MVS batch and route your output to a local output device, unless you specify otherwise during the session.

To submit an NQS batch job to the Cray from the VAX 8700 (through the Lab-wide NJE network), enter:

```
$  CRJOB nqsjob-filename
```

where "nqsjob-filename" is the name of the file that contains your NQS batch and UNICOS commands.

## GROUP MANAGERS

To learn if your cost center has a group manager, enter the VMS command:

```
$  SHOW LOGICAL/GROUP GRP_MANAGER_NAME
```

If a group manager has been assigned, the system will display the group manager's VMS username. To learn more about the cost center's computing environment, contact your group manager. Your group manager may be able to locate a printer in your vicinity, provide information about special tools, logical names, and symbols that are available to you.

# CHAPTER 4

# USING THE VAX/VMS FILE SYSTEM

This chapter complements the *Guide to Using VMS* (AA-LA05A-TE) by focusing on procedures for managing, protecting, and transferring files efficiently.

## MANAGING FILES

The *Guide to Using VMS* (AA-LA05A-TE) covers procedures for creating, modifying, copying, renaming, and deleting files. However, you will find the following sections useful for referencing files, using Directory Manager, using default strings, using logical filenames, and using wildcard characters.

### Referencing Files

Detailed information on the VMS file structure appears in the *Introduction to VAX/VMS* online tutorial, which you can invoke by entering:

```
$  RUN VMSCAI
```

To manage your files, you must be able to identify them in such a way that DCL can locate them. Such identification occurs by means of a VMS file specification:

```
node"username password"::disk:[directory]filename.filetype;version-number
```

For example,

```
ANLCV1::CCnnn:[username.FORTRAN]EXPL.FOR
```

refers to the second version of a file named "EXPL.FOR" residing on disk "CCnnn" ("CCnnn" is a logical name for the device name) of node "ANLCV1" in the subdirectory FORTRAN of the directory "username." The filetype "FOR" by convention identifies this program as a Fortran program.

### Using Directory Manager (DM)

Directory Manager (DM) is a full screen utility that gives users a visual representation of their directory structure and allows users to work with their files and directory structure more easily. Directory Manager allows you to view your directory structure in a tree-like form. You can move up and down your directory tree by using the arrow keys instead of using the **SET DEFAULT** command. It also has single keystroke commands to replace commonly used or lengthy DCL commands. To invoke Directory Manager, enter:

```
$  DM
```

For further information on Directory Manager, enter:

```
$  HELP DM
```

## Using Default Strings

It is rarely necessary to give a full-file specification. The system maintains a default string that it uses to fill in missing fields of a partial file specification. The default string refers to a directory on a disk. The phrase current directory is synonymous with the current value of the default string. Table 7 lists the DCL commands **(SET DEFAULT** and **SHOW DEFAULT)** for adjusting and reviewing the value of the default string as well as several other DCL commands to manipulate files.

For instance, to use the complete file specification with the DCL **COPY** command to copy version four of a file with the name "tst.dat" to version one of another file named "tst2.dat," enter:

```
$  COPY cc999:[username.subdir]tst.dat;4 cc999:[username.subdir]tst2.dat
```

The DCL will store these files on disk "cc999," in the directory of "username," in the subdirectory of "subdir."

If you were to set your default string to be equivalent to the subdirectory "subdir" with the command

```
$  SET DEF [username.subdir]
```

you could then enter a partial file specification:

```
$  COPY tst.dat;4  tst2.dat
```

VMS will reference the current default string to complete the file specification you are referencing in the **COPY** command.

Table 7

**Standard File Manipulation Commands**

COPY duplicates a specified file, as in the command line:

```
$  COPY infile outfile
```

CREATE/DIR creates a subdirectory (you can have seven levels of subdirectories) within the current directory, as in the command line:

```
$  CREATE/DIRECTORY [username.directory-name]
```

DELETE removes a specified file from the current directory, as in the command line:

```
$  DELETE filename.filetype;version
```

DIRECTORY lists files and subdirectories located in the current directory, as in the command line:

```
$  DIRECTORY
```

PURGE eliminates all previous versions of a file (VMS creates a new version of a file whenever it is modified. Users must therefore habitually discard old, unneeded files to free disk space and reduce storage charges), as in the command line:

```
$  PURGE filename.filetype
```

RENAME changes the name of a file, as in the command line:

```
$  RENAME in-filename.filetype out-filename.filetype
```

SET DEFAULT allows you to change the value of the current default string so that you can move from subdirectory to subdirectory in order to access the files in these subdirectories, as in the command line:

```
$  SET DEFAULT [username.READER]
```

SHOW DEFAULT displays the value of the current default string, as in the command line:

```
$  SHOW DEFAULT
```

TYPE displays a specified text file on the screen, as in the command line:

```
$  TYPE filename.filetype
```

Online help on correct syntax and usage is available for these and other VMS commands.

## Using Logical Device Names

We strongly encourage the use of preassigned logical device names in place of actual names of physical devices for device independence. The logical name can be used synonymously with the physical device name in all references to the device. Also, if you refer to the files in your program by logical names and for some reason the files are moved to a different disk, the system manager will change the logical name definitions to make the transition transparent to your commands and procedures.

A logical device name is equivalent to a string. The string may contain a device name and a directory, but it will relate to actual devices such as your terminal or disk. The system manager sets up most logical name definitions.

An important logical name defined for you by the system is "SYS$LOGIN," which contains your root address. When you enter the command

```
$  SHOW LOGICAL SYS$LOGIN
```

the system will give you the "SYS$LOGIN:" string. "SYS$LOGIN" is a logical name set up for you by the system that equates the actual name of the disk drive device on which your files are located. To see which disk drive you are currently assigned to, enter:

```
$  SHOW LOGICAL CCnnn
```

where "nnn" is your divisional cost center.

Another important logical name is "SYS$INPUT." The command line

```
$  SHOW LOGICAL SYS$INPUT
```

will display the value of the logical "SYS$INPUT" as your terminal device name, as specified in the response to the command line:

```
$  SHOW TERMINAL
```

Another common use of logical names occurs in redefining the logical names "SYS$INPUT" and "SYS$OUTPUT." These logical names designate your terminal as your standard input and output device. You can redefine "SYS$OUTPUT" to be equivalent to a file specification with the following command:

```
$  DEFINE SYS$OUTPUT JUNK.DAT
```

Then the output from following DCL commands that would normally go to your terminal will instead go to the specified file "JUNK.DAT." Similarly, you can redefine "SYS$INPUT" prior to running a program so that the program will refer to the specified file for input rather than to your terminal.

To return a logical device name to its default value, enter:

```
$  DEASSIGN SYS$OUTPUT
```

At login time, VMS defines several other logical names that are useful to you: "SYS$LOGIN," "SYS$LOGIN_DEVICE," and "SYS$SCRATCH." To see the assigned equivalence string, enter the DCL command line:

```
$  SHOW LOGICAL/JOB
```

When you need to reference your login device, use the logical name "SYS$LOGIN_DEVICE" instead of the "CCnnn:" format logical name. Your file reference will continue to work if you change cost centers or if your division undergoes a reorganization.

### Using Wildcard Characters

A wildcard character is a symbol that you can use to apply DCL commands to several files at once, instead of having to specify each file in a separate command line. Two wildcard characters--the asterisk and the percent sign--can be used to represent portions of a file specification field. For instance, the command line

```
$   COPY *.DAT *.SAV
```

will copy each file (in your default directory) with a filetype of "DAT" to new files (in your default directory) that have the same filename and a filetype of "SAV."

You can also use the asterisk wildcard character, as in the following example:

```
$   TYPE *19*.DAT
```

This command line will display on your terminal screen the contents of all your files (in your default directory) with a filetype of "DAT" and a filename containing both the string "19" and other characters both before and after that string. The asterisk can also represent the empty string.

The percent sign wildcard character allows you to specify all files containing any single character in the position the percent sign occupies in the file specification. For example, the command line

```
$   TYPE 198%.DAT
```

will display on your terminal screen the contents of all files (in your default directory) having a filetype of "DAT" and a filename of "198" plus any other single alphanumeric character.

A third type of wildcard--the ellipsis (...) is helpful when you cannot remember in which directory you saved a file. Using the ellipsis permits you to search through your subdirectories for that file. For example, the command line

```
$   TYPE [username...]1984.DAT
```

will tell the **TYPE** command to search all your subdirectories (beginning with your root directory) for a file named "1984.DAT." When DCL finds this file, it will write the complete file specification to your terminal screen prior to writing out the contents of the file.

## PROTECTING FILES

VAX/VMS identifies a user of a directory, a subdirectory, or a file as belonging to one of four user categories: system, owner, group, or world. An explanation of each of these categories follows:

- **System (S)** refers to the VMS operating system and its controlling users, the system operators, and the system manager.
- **Owner (O)** refers to the user (usually the creator of the file) to whom the file belongs.

- **Group (G)** refers to a set of users (those that have the same group number or same group identifier in their user identification code) who have access rights to one another's directories and files within those directories (unless protected otherwise). At ANL, all VAX cluster accounts with the same cost code have the same group identifier.

- **World (W)** refers to all users, including the system operators, the system manager, and users both in an owner's group and in any other group.

Each category has default access codes: read (R), write (W), execute (E), and delete (D). The read access code allows you to display the file on your screen, duplicate the file, or send it to a printer. The write access code allows you to modify the file. The execute access code allows you to execute files consisting of programs or procedures. The delete access code allows you to eliminate the file.

For any given file, VMS identifies you as a user in one of the four categories. Your access to that file is based upon the codes assigned to that category. You must also have access to the directory in which that file resides. On the central VAX cluster, the default codes differ for directories and for files:

- For directories, the default access codes for the system and owner categories are read, write, and execute. The default access codes for the group category are read and execute. The default access codes for the world category is execute.

  Directories are created without the delete access code. Consequently, even owners cannot eliminate a directory unless they have manually changed the code to include **DELETE** with the following command:

  ```
  $   SET PROTECTION=(O:RWED) name-of.dir
  ```

- For files, the default access codes for the system and owner categories are read, write, execute, and delete. The group and world categories have no access codes at all. The security requirements of the system dictate the default access rights of users to files. The owner of a file can override these default rights.

Owners can change the protection on their directory, their subdirectories, and/or their files; ultimately, the owners are responsible for protecting their files. The DCL command

```
$   SET PROTECTION=(category:code) filename.filetype
```

will modify the specified file to the desired protection. For example, to allow anyone in your group to read the file, write to the file, and to allow all other users to read and execute the file, the owner of the file would give the group category the read and write codes and the world category the read and execute codes with the DCL command:

```
$   SET PROTECTION=(G:RW,W:RE) LIST.DAT
```

Another example of this DCL command shows the sequence of commands necessary to change the main directory protection to give the world category read and execute access to your root directory:

```
$   SET PROTECTION=(W:RE) [-]username.DIR
```

Invoke the HELP utility for more detailed information on the use of the **SET PROTECTION** DCL command. See "How to Protect a File" in *Introduction to VMS* (AA-LA04A-TE) for more general information.

To give the world read access to all your files and subdirectories, enter:

```
$  SET PROTECTION=(W:R) [-]username.DIR
$  SET PROTECTION=(W:R) [username...]*.*;*
```

*Do not use these commands if any of your files contain login passwords for the VAX 8700 or other systems (e.g., Cray X-MP).* To set the default protections for future files, add the following command line to your "login.com file:"

```
$  SET PROTECTION=(S=RWED,O=RWED,G=RE,W=RE)/DEFAULT
```

## TRANSFERRING FILES

Use the NJE emulator to transfer files to and from the central VAX cluster. Several commands used to transfer files are defined below.[1] Examples and syntax for online NJE HELP are available for transferring files by entering the command:

```
$  HELP @NJE
```

NJE files are received into a subdirectory to your main directory and named either ".NJE" (as on some divisional VAXes) or ".READER" (as on the central VAX cluster nodes).

### From the VAX 8700 to Other Nodes on the Laboratory-Wide NJE Network

To transmit files on the Lab-wide NJE network from the VAX 8700, use the **SEND PUNCH** and **SEND PRINT** commands.

The **SEND PUNCH** command will transmit an 80-character record text file from the VAX 8700 to any other remote Argonne node, BITnet node, or node in a network that has a gateway to BITnet, (in transmission, any tabs in the file will convert to spaces):

```
$  SEND PUNCH filename.filetype nodename username
```

For example, to send a file to a user on the Chemical Technology VAX computer, enter:

```
$  SEND PUNCH MYDATA.DAT ANLCMT username
```

The **SEND PRINT** command will transmit a 132-character record file to any other remote Argonne node or BITnet node:

```
$  SEND PRINT filename.filetype nodename username
```

For example, to send a file called "MYOUTPUT.TXT" through ANLOS to the IBM 3800 laser printer, enter:

```
$  SEND PRINT MYOUTPUT.TXT ANLOS 3800
```

[1] *Guide to VAX/VMS NJE Networking at ANL* (ANL/TM 444) discusses these commands in detail.

The **PUTFILE** command will transmit any type of general file from the VAX 8700 to a destination on the Lab-wide NJE network. The file is sent in binary form by default. You must use the "/TEXT" qualifier to send an ASCII file:

```
$  PUTFILE filename.filetype nodename username
```

(When you send a file, the filetype will change to SFT.) For example, to send a VMS executable image file to an account on the Chemistry Division VAX computer from the central VAX 8700, enter:

```
$  PUTFILE TESTPROG.EXE ANLCHM username
```

NJE will transfer the file to the recipient's account in the READER subdirectory (or, on some divisional VAXes, the NJE subdirectory). VAX users receiving the file would then use the **GETFILE** command to reconstruct the file (again, the default is binary):

```
$  GETFILE [username.READER]TESTPROG.SFT TESTPROG.EXE
```

where "[username.READER]" is the recipient's READER subdirectory, "TESTPROG.SFT" is the filename and filetype of the arriving file, and "TESTPROG.EXE" is the filename and filetype of the file as it will appear in your default directory. You must use the "/TEXT" option if you are sending or receiving an ASCII file.

For more details on using these commands, see *Guide to VAX/VMS NJE Networking at ANL* (ANL/TM 444).

## From CMS to the VAX 8700

Text files sent from CMS to accounts on the VAX 8700 via the NJE network will go into a user's [.READER] subdirectory. (Account Services creates this subdirectory when establishing each new user's account.) To transfer the CMS text file SAMPLE DATA A1 to a VAX 8700 account, enter (in CMS):

```
$  SENDFILE SAMPLE DATA A1 TO username AT ANLCV1 (NEW
```

The "NEW" option generates a NETDATA-coded file that appears in the username [.READER] subdirectory with the filename:

```
[username.READER]SAMPLE.DATA_NETDATA
```

To convert the file to the required VMS text format, use the **GETFILE** command to receive and reconstruct the file:

```
$  GETFILE /TEXT [username.READER]SAMPLE.DATA_NETDATA SAMPLE.DATA
```

The "/TEXT" option translates EBCDIC characters to ASCII characters. "SAMPLE.DATA_NETDATA" is the filename to be read. "SAMPLE.DATA" is the default name of the file to be created; you need not specify this default name unless you decide to have it renamed.

## From MVS Wylbur to the VAX 8700

You can send your Wylbur active file to the VAX 8700 by using the **DO SENDFILE** procedure. The system will prompt you for an address. To send your active file to your VAX 8700 account, enter:

```
username at ANLCV1
```

This procedure will then submit an IBM MVS batch job with the name "usernamexx," where "username" refers to your MVS account and "xx" is a Wylbur job number. A text file will go to the specified VAX 8700 account through NJE, which will put the file in the ".READER" subdirectory of the account. The name of this file will be "usernamexx.ACTIVE." This "usernamexx.ACTIVE" file will have the appropriate format for VMS. You can edit the file, rename it, remove it, copy it, delete it, or type it to the terminal.

For active files whose records are 80 characters or less, you can use the Wylbur **PUNCH** command:

```
PUNCH UNNUMBERED DEST=ANLCV1.username
```

For active files with up to 132 characters per record (133 if the first character is Fortran carriage control), enter:

```
LIST OFFLINE UNNUMBERED [CC] DEST=ANLCV1.username
```

where "CC" is used if the file contains Fortran carriage control characters in column 1.

## From a DECnet Node to a DECnet Node

To transfer a file from one VMS system to another through DECnet, use the DCL **COPY** command:

```
$ COPY filename1 filename2
```

The specification for a file on the local node is

```
disk:[directory]filename.filetype;version-number
```

and the specification for a file on the remote node is

```
node"username password"::disk:[username.subdir]fname.ftype;version-number
```

where "fname.ftype" is the name of the file. The quoted field is referred to as the access control string. If you have a proxy account on the remote node, you do not have to include your access control string. See "Requesting a DECnet Proxy" in Chapter 2 for more detailed information.

The following is an example of the syntax to copy a file from remote node ANLCMT to the local node through DECnet. The example assumes that the user is currently logged on to the local node and has a proxy account on node ANLCMT from the local node. The user executes the command line:

```
$ COPY ANLCMT::DISK_USER1:[username]name.tst  tst.dat
```

The following example is the opposite of the previous example. The user has logged on to the Chemical Technology Division (CMT) VAX system and wants to copy a file to the VAX 8700. In this example, the user does not have a proxy account on the VAX 8700 for the CMT VAX system; therefore, the user must establish access to the account by using the access control string:

```
$  COPY tst.dat ANLCV1"username password"::cc123:[username]*.*
```

## Via TCP/IP

The Multinet TCP/IP software (described in "Available Networks and Protocols" in Chapter 1) provides FTP (a file transfer program) as a user interface with the ARPAnet standard File Transfer Protocol. You can transfer files between the central VAX 8700 and a remote TCP/IP node by using this program:

1. To invoke this file transfer program, enter:

   ```
   $  ftp
   ```

   You will receive an "FTP>" prompt.

2. To receive a terminal display of available commands, enter:

   ```
   FTP>  HELP
   ```

   or

   ```
   FTP>  ?
   ```

3. Prior to transferring a file from a TCP/IP node to your VAX 8700 account, you must use the FTP command **open node-name** to establish a connection. For example, to transfer a text file from your Cray account, you would enter:

   ```
   FTP>  open xmp.ctd.anl.gov
   ```

   The system will notify you that the connection is open and will return you to the prompt (the system will always return you to FTP following execution of an FTP command).

4. You must then identify yourself by logging on to the remote node (with the **login** command):

   ```
   FTP>  login
   ```

5. The system will respond with:

   ```
   Foreign username:  username
   ```

6. The system will then prompt you for your password (your password will not appear on the terminal screen):

   ```
   Password:
   ```

At this point you have established a connection and identified yourself as the owner of the account. (After you enter your password, the "FTP>" prompt will appear on your terminal screen. You can now issue FTP commands through this connection to a remote node.)

You can also open a connection to a TCP/IP node by using the command line:

```
$  ftp/prompt nodename
```

The system will then require you to logon to that account.

7. You can transfer either ASCII or binary files. ASCII file transfer is the default. To transfer a file to your VAX 8700 account, use the FTP **get** command:

```
FTP>  get remote-filename  local-filename
```

For instance, to transfer your Fortran source code file "prog1.f" from CRAY X-MP/14, enter:

```
FTP>  get prog1.f prog1.for
```

Note that the format of the file specification for the remote file is in VMS form. You can list your remote directory files with the FTP **ls** command to establish the correct file specification:

```
FTP>  ls
```

The system will list all your remote files on the terminal screen with the designated format.

8. To transfer a file from your VAX 8700 account to your remote node account, use the FTP **put** command:

```
FTP>  put local-file  remote-file
```

For example, to transfer your source code file "prog1.for" from the VAX 8700 to node XMP, enter:

```
FTP>  put [username]prog1.for;1 prog1.f
```

Provide a complete file specification (excluding nodename and logical disk drive) when the file is located somewhere other than where your current default is pointing. Supply the filename and filetype when the file is in your current default directory.

9. If you are transferring binary files, enter

```
FTP>  binary
```

prior to issuing the **put** or **get** commands.

10. To logoff the remote node, enter:

```
FTP>  close
```

Execution of this command will return you to FTP.

11. You can return to the DCL level by entering **exit, bye,** or **quit:**

```
FTP>  quit
```

DO NOT MICROFILM
THIS PAGE

# CHAPTER 5

# TAPE MANAGEMENT

CTD maintains a tape library containing both library tapes and personal tapes. You can use these library tapes to store your files. In this case, you are the owner of the data on the tape while CTD retains ownership of the tape. These library tapes are assigned to you upon request until you release them, whereupon they are made available to other users. They remain in the library unless you purchase them. The tapes are identified by an external volume serial number consisting of six digits.

You can add your own tape to the library by bringing it to the Counter in Room A-134, Building 221, for check-in by one of the operators, or you can purchase a tape from the operator at the counter (or by calling extension 2-7681 and requesting a personal tape). The operators will assign your tape an external volume serial number beginning with "U" and followed by five digits. Operations will keep such tapes in the library for two weeks and then return them to you through ANL mail.

In general, tapes are either labeled or non-labeled.[2] Labeled tapes may have ANSI-standard or IBM-standard labels. In general, you should use ANSI-standard labeled tapes on the VAX. The label on an ANSI-standard labeled tape is the internal volume serial number. Initially, the internal and external volume serial numbers are the same for CTD library tapes (however, a user may overwrite the internal label by accident). A non-labeled tape has only an external volume serial number. You can define the internal label when you transfer your files to the tape. ANSI-standard labeled tapes and non-labeled tapes are available to you from the CTD library as scratch tapes. Both labeled and non-labeled tapes are acceptable as personal tapes. Tapes that you purchase as personal reels can be non-labeled tapes. We recommend that you maintain the integrity of the ANSI-standard internal label of your tape. assign an internal label to your tape. The VAX system will check the ANSI-standard label to verify that the correct tape has been mounted.

The TAPELIBRARY utility allows you to obtain reports for tapes owned by a single user or by an entire cost center. It can report on and update the status of tapes in the CTD tape library. Information reported includes dataset name, volume serial number, tape data control block (DCB) information, internal tape label, creation date, expiration date, and date of last access. You can also use the TAPELIBRARY commands to query or update the status of a tape (i.e., to expire a tape when you no longer need it). TAPELIBRARY submits a request to an IBM MVS batch job to process the report. The system will prompt you for the output destination of the report. If you choose the default destination, the system will send a file to your READER directory containing the requested information.

[2] A labeled tape has a volume serial number written as the first record on the tape. The computing system uses this record to ensure correct tape identification, which is essential to protecting tape data from accidental misuse. A non-labeled tape has no volume serial information on the first tape record. Note that all tapes must have at least some kind of record at the beginning of a tape to be mounted on Argonne central computing tape devices (unless a process to prevent label-checking is used). Otherwise, tape runaway occurs. The system looks at the first record to see if the tape is labeled or not.

On request, CTD Computer Operations personnel will mount (on the central VAX 8700 tape drive) ANSI-standard labeled library tapes, non-labeled library tapes, scratch library tapes, and personal reels. The operator will check the requestor's authorization for the external volume serial number on all tapes before mounting them. VMS will verify the internal label on ANSI-labeled tapes.

## BASIC PROCEDURES FOR USING TAPES ON THE CENTRAL VAX 8700

To use tapes on the central VAX 8700, follow these five steps:

1. To gain exclusive access, **ALLOCATE** the tape drive ("$1$MUA0:").
2. **MOUNT** the tape with the "/COMMENT" qualifier which specifies the tape's external label so the operator knows which tape to place on the device.
3. Use the **BACKUP** or **COPY** commands, or run a program to read or write files on the tape.
4. After using the tape, **DISMOUNT** the tape so the operator can remove it from the tape drive.
5. **DEALLOCATE** the tape drive to make it available again for other users.

For example, to create an ANSI-labeled 6250 bits per inch tape in which disk files (in a directory) are saved in a single VMS saveset file on a personal reel with the external-label "U00099," enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT="WRITE, U00999"
$  BACKUP/DENSITY=6250/INIT/REWIND/VERIFY/LABEL=internal-label
_From:  [username.subdirectory-name]*.*
_To:  TAPE:saveset.BCK/LOG
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

where "internal-label" specifies the internal volume label which cannot exceed six characters in length. (If you omit the "/LABEL" qualifier, the internal label will be the first six characters of the saveset name.) The **BACKUP** command will check the integrity of the files copied to the tape when you specify the "/VERIFY" qualifier. The "/LOG" qualifier will write to your screen the names of the files copied to the tape as they are being copied. Note that saveset files can only be restored from this tape to a VAX/VMS system. Do not use this method if you plan to use the tape files on a non-VMS computer system.

### Allocating the Tape Drive

Before you can obtain exclusive access to the VAX 8700 tape drive (which is necessary when using these tapes), you should check the current status of the device by entering

```
$  SHOW DEVICE $1$MUA0:
```

which will give you the message that the tape is either ONLINE, OFFLINE, or ONLINE ALLOCATED. If the device is ONLINE, it is available for use. If the device is OFFLINE, it is unavailable for use. If the device is ONLINE ALLOCATED, it is currently in use. To allocate the tape drive to your session, enter:

```
$  ALLOCATE $1$MUA0: logical-name:
```

where "$1$MUA0" is the VMS tape device name, and "logical-name" is any alias you designate to the tape drive. (For the examples in this chapter, "TAPE" is the alias that is used.)

## Mounting the Tape

After allocating the tape drive, you must have your tape mounted using the **MOUNT** command. To identify your tape to the operator, give the operator the external-volume serial number (in the "/COMMENT" qualifier) and include the internal label, when available, to verify that the proper tape has been mounted.

To mount an ANSI-standard labeled personal tape (for read only purposes), enter:

```
$  MOUNT logical-name internal-label /COMMENT="U99999"
```

To request an ANSI-standard labeled scratch tape, enter:

```
$  MOUNT/FOREIGN logical-name: /COMMENT="WRITE,SCRATCH=AL"
```

For ANSI-labeled scratch tapes, you must use the key words "SCRATCH=AL" instead of an external label and add the "/FOREIGN" qualifier. After entering the **MOUNT** command, a request will be sent to the operator to mount a scratch tape. You will see the message:

```
%MOUNT-I-OPRQST, Please mount device _$1$MUAO: (HSC000)
write, scratch=al
```

Once the tape has been mounted, the MOUNT utility will display the internal label, which is the same as the external-volume serial number. You will see the message:

```
%MOUNT-I-MOUNTED, nnnnnn mounted on _$1$MUAO: (HSC000)
%MOUNT-I-RQSTDON, operator request cancelled - mount completed successfully
```

where "nnnnnn" is the external label of the tape. Note that you will need the external label for future reference whenever you want that specific tape mounted.

To mount an ANSI-standard labeled library tape, enter:

```
$  MOUNT logical-name: internal-label /COMMENT="ring-status, external-label"
```

where "logical-name" is the alias designated in the **ALLOCATE** command, "internal-label" is the ANSI-standard label, and "ring-status" determines read and write privileges. (By default, write protection is in effect. You must enter the **WRITE** if you want to write to the tape.) "External-label" is the external volume serial number. For those tapes that you are going to read or modify, VMS will determine the density from the first record in the volume.

To mount a non-labeled tape, enter:

```
$  MOUNT/FOREIGN logical-name: /COMMENT="ring-status, external-label"
```

The "/FOREIGN" qualifier is used whenever you do not know the internal label of the tape or if the tape is non-labeled.

You may need to include the qualifiers "/BLOCKSIZE=N" and/or "/RECORD=N," in the case of non-labeled tapes so that VMS knows the block and record structure of the tape.

The mounting process may take up to ten minutes because the operator must verify your status as a valid user to mount the tape. If in that time there is no response, call the operator at extension 2-5421 for assistance.

## Writing and Reading from the Tape

After you mount the tape, you can transfer your files to or from the tape by using the appropriate commands.

To transfer VMS files from disk to a system-independent, ASCII-formatted tape, enter:

```
$  COPY infile outfile
```

where "infile" is the VMS file specification and "outfile" is the "logical-name:" referring to the tape drive. To copy from the tape to the disk, enter the same **COPY** command, where "infile" is the "logical-name: filename.filetype" and "outfile" is the VMS file specification. Tapes created by this method may be used on non-VAX/VMS systems.

To exchange tapes with other VMS systems, use the VMS BACKUP utility. Create a new tape with BACKUP by entering:

```
$  BACKUP/INIT/REWIND/LABEL=internal-label/DENSITY=NNNN/VERIFY
_From:  filename.filetype
_To:  logical-name:saveset-name.BCK/LOG
```

where "NNNN" is the tape density in bytes per inch (1600 or 6250), "filename.filetype" is the VMS file being transferred, and "saveset-name" is a user-defined tape filename. The "internal-label" is an ANSI-standard label. If the "internal-label" is unspecified, it will default to the user-supplied "saveset-name." The "/INIT," "/REWIND," and "/VERIFY" qualifiers instruct VMS to initialize the tape, to begin at the beginning of the tape marker, and to verify written files. Extensive online help is available by entering **HELP BACKUP** at the DCL prompt. Note, do not use the **BACKUP** command for tapes that will be read on non-VMS systems. For further information on BACKUP, see the examples that follow later in this chapter.

You can also write and read files directly from a tape using Fortran programs; however, CTD discourages this procedure for long programs because you must have the tape device allocated for the life of your Fortran program. Because CTD has only a single VAX tape drive, other users may need to use the tape drive before your program is complete. CTD encourages you to WRITE your files to a disk and then use COPY or BACKUP to put them on tape. If this is not possible, see the examples at the end of this chapter for writing and reading tapes with a Fortran program.

## Dismounting the Tape

When you are finished transferring files, **DISMOUNT** the tape so that it can be removed from the tape drive by entering:

```
$  DISMOUNT logical-name:
```

## Deallocating the Tape Drive

Before the operator can remove the tape from the drive, you must **DEALLOCATE** the tape drive to make it available to other users. (Simply using the **DISMOUNT** command, will not do this.) Enter:

```
$  DEALLOCATE logical-name:
```

Copying your files from the tape to your disk may require more storage than you have available on your permanent disk. To view the amount of storage available, enter:

```
$  SHOW QUOTA
```

Use scratch space ("SYS$SCRATCH") if you do not have enough permanent storage by setting your default string to "SYS$SCRATCH" (prior to using the **COPY** or **BACKUP** commands), by entering:

```
$  SET DEF SYS$SCRATCH
```

When you use BACKUP, the newly created disk file will have the creation date of the original file on the disk, not the date you restored it. These dates can be important. If you store these files on scratch space, the system may delete them the same day unless you change the file date to the current date (i.e., every night, the system deletes all scratch files more than seven days old). To preserve your files, transfer them to your permanent disk.

To read a system-independent tape to disk, Record Management Services (RMS) requires information about file characteristics. For instance, if the files you are copying are Fortran source code, your blocksize might be 1840 (depending on the transferred file characteristics) and your record length might be 80. The blocksize can be any number you specify as long as that number is divisible by the record length. If you omit the record length, RMS will create a variable-length record formatted file with the maximum record length equal to the blocksize.

If you do not know the file's blocksize, you can dump the tape to your terminal. Use the **MOUNT** command in step two (earlier in this chapter) for a non-labeled, ASCII-formatted tape:

```
$  MOUNT/FOREIGN logical-name: /COMMENT="ring-status, external-label"
```

Use the DCL **DUMP** command to determine the file's blocksize which will be stored at the beginning of the tape's first block:

```
$  DUMP/BLOCKS=(START:1,END:1) logical-name:
```

You can also use the DCL **DUMP** command to determine the file's record size which is also stored at the beginning of the tape's first block.

Issue the **DISMOUNT** command with the "/NOUNLOAD" qualifier (which will also rewind the tape and point to the beginning-of-tape marker) prior to issuing another mount command with a known blocksize and record length qualifier:

```
$  DISMOUNT/NOUNLOAD logical-name:
```

Then enter:

```
$  MOUNT/FOREIGN/BLOCKSIZE=n/RECORD=m logical-name:
```

You can begin copying your files from the tape to the disk once you have determined the blocksize and record length and have issued the appropriate **MOUNT** command. To copy files, issue a **COPY** command for every file:

```
$  COPY logical-name:file1.dat file1.dat
$  COPY logical-name:file2.dat file2.dat
```

To copy the second file from the tape to the disk, you must read the first file from tape to position the tape at the beginning of the second file.

You must **DISMOUNT** and **DEALLOCATE** the tape drive when you have finished.

## EXAMPLES FOR CREATING AND USING TAPES IN VMS

The following section describes other ways to use VMS tapes. Explanations are given for the relevant qualifiers used in each command.

To restore the files from a labeled tape (U00099) when your disk files are stored in a single saveset file, enter:

```
$  ALLOCATE $1$MUA0:  TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT="WRITE, U00099"
$  BACKUP/DENSITY=6250/REWIND/VERIFY/LABEL=internal-label
_From:  TAPE:FILES.BCK
_To:  [username.subdirectory]*.*
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

Use this example to restore the files which were written to tape in the first example (earlier in this chapter).

To list the saveset name stored on a tape (U00099) if you do not know the internal label or the name of the saveset on the tape (information will be written to the file TAPE_DIRECTORY.DAT), enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/OVERRIDE=ID TAPE: /COMMENT="READ, U00099"
$  DIRECTORY/OUTPUT=TAPE_DIRECTORY.DAT TAPE:
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

To list the names in the saveset file on the tape on your screen, enter:

```
$  TYPE TAPE_DIRECTORY.DAT
```

To list the files stored in a saveset file on tape, enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT= "READ, U00099"
$  BACKUP/LIST=DIRECTORY_LIST.DAT/REWIND TAPE:FILES.BCK
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

To list the names of the files in the saveset file on your screen, enter:

```
$  TYPE DIRECTORY_LIST.DAT
```

To extract particular files out of a saveset file on tape (U00099), enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT="READ, U00099"
$  BACKUP/REWIND/SELECT=filename.filetype TAPE:saveset.BCK *.*
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

To relabel (or label for the first time) a tape with ANSI-standard labels, enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT="WRITE, U00099"
$  DISMOUNT/NOUNLOAD TAPE
$  INITIALIZE TAPE: internal-label
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

where the "internal-label" you specify must be six or less characters in length.

To copy files to an ANSI-standard labeled tape that will be read on a non-VMS system, enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT TAPE: internal-label /COMMENT="WRITE, U00099"
$  COPY filename.filetype TAPE:
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

To copy files from an ANSI-standard labeled tape, enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT TAPE: internal-label /COMMENT="READ, U00099"
$  COPY TAPE:filename.filetype filename.filetype
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

To copy files from an ASCII, non-labeled tape that was created on a non-VMS system, enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN/BLOCKSIZE=N /RECORD=M TAPE: /COMMENT="READ, U99999"
$  COPY/LOG TAPE:file1.dat file1.dat
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

When copying files from a non-labeled tape you must specify both a filename for the input file and for the output file you want to put on disk. You must also mount the tape with the correct blocksize and record length (see the **DUMP** command described earlier). The above procedure must be used when copying files from non-labeled tapes which were created on systems other than VMS.

To request an ANSI-standard labeled scratch tape from the library in order to copy disk files to the tape, you must **INITIALIZE** the tape the first time you use it. Initializing the tape will reset the tape internal label, which was previously written on the tape, so that VMS can read it. The new internal label, specified with the **INITIALIZE** command, should be the same as the external label. Enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT="WRITE, SCRATCH=AL"
$  DISMOUNT/NOUNLOAD TAPE:
$  INITIALIZE TAPE: internal-label
$  MOUNT TAPE: internal-label
$  COPY filename.filetype TAPE:
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

After mounting the scratch tape, the system will return an external label to you so that you can remount this tape.

To copy the files from the scratch tape (written in the previous example), enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT TAPE: internal-label /COMMENT="READ, external-label"
$  COPY TAPE:filename.filetype filename.filetype
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

To write data to the tape with a Fortran program, enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT TAPE: internal-label /COMMENT="WRITE, external-label"
$  RUN WRITE.FOR
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

where, the "WRITE.FOR" file could consist of the following code:

```
       PROGRAM WRITE
       OPEN (UNIT=9 FILE 'TAPE:FOR009.DAT', STATUS='NEW')
       DO 10 I=1,100
         WRITE (9,*) I
00010   CONTINUE
       END
```

To read data from the tape with a Fortran program, enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT TAPE: internal-label /COMMENT="READ, external-label"
$  RUN READ.FOR
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

where "READ.FOR" could be the following code:

```
       PROGRAM READ
       OPEN (UNIT=9, FILE='TAPE:FOR009.DAT', STATUS='OLD')
       DO 10 I=1,100
         READ (9,*) J
00010   CONTINUE
       END
```

If you have more data than will fit on one tape, you must initialize the tapes you will need with ANSI-standard labels to match the external labels on the tapes. If the labels do not match, the following code will not work because the operators will not know which tape to put on the drive after the previous tape becomes full. Once you have initialized all the tapes, enter

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT="WRITE, external-label1"
$  BACKUP/DENSITY=6250/VERIFY/REWIND/INIT -
_$ /LABEL=(external-label1, external-label2, external-label3)
_From:   [username...]*.*
_To:  TAPE:saveset.BCK/LOG
$  DISMOUNT TAPE:
$  DEALLOCATE TAPE:
```

to backup your entire account, including all files and subdirectories, on multiple tapes (assuming you have that many files). This code can also be used to backup enormous data files stored in your scratch directory.

If you have an ANSI-standard labeled tape and you have forgotten the internal label name, the system will return the internal tape label to you if you enter:

```
$  ALLOCATE $1$MUA0: TAPE:
$  MOUNT/FOREIGN TAPE: /COMMENT="READ, external-label"
$  DISMOUNT/NOUNLOAD TAPE:
$  MOUNT TAPE: LABEL
```

where "LABEL" is the "internal-label" name just returned to you. You can then use the tape you want and then enter the **DISMOUNT** and **DEALLOCATE** commands when you are finished.

# CHAPTER 6

# PRINTING VMS FILES

This chapter describes how to use the various printing options available to you on the central VAX cluster.

## VMS FILE PRINT CHARACTERISTICS

The VMS Record Management Services (RMS) associates various characteristics with files. You can view the characteristics of a file by entering the DCL command:

```
$  DIR/FULL  filename.filetype
```

One of the characteristics is the record attribute of the file. Four possible attributes exist: carriage return carriage control, Fortran carriage control, print carriage control, and no carriage control. The system passes this information along for file transfer to other computers or devices.

The appropriate record attribute information for the file will pass to the IBM print queue. For instance, the output device interprets a "1" (when it appears as the first character of a record) as a page eject for files with a record attribute of `FORTRAN CARRIAGE CONTROL`. The record attribute designations of `CARRIAGE RETURN CARRIAGE CONTROL`, or `LIST` have a line feed and a carriage return preceding each record. The `PRINT CARRIAGE CONTROL` specification applies to the RMS system Variable with Fixed Control (VFC) record format in which each record has a two-byte header containing carriage control. In general, carriage control is null for this type of file. Form feeds are determined by the specific device symbiont.

The record attributes of files depend upon the program you use to create them. Table 8 shows the default file characteristics for several of the more common creation techniques.

Although record attributes may differ, each one will be handled appropriately. However, when you append files together (each having different record attributes), you will be creating one file and the carriage control characteristics for that file will be the same as for the file to which you are appending the others. For instance, if you want to concatenate your LIS and MAP files into a single file with your program output file (having Fortran Carriage Control for printing on the IBM 3800 laser printer), you could use the commands:

```
$  CONVERT/APPEND filename.MAP filename.LIS
```

and

```
$  CONVERT/APPEND program.OUT filename.LIS
```

The filename of your resulting single file would be "filename.LIS," and you would lose Fortran carriage control from your Fortran output file "program.out." You can avoid losing Fortran carriage control by running your Fortran output file through a tool that will translate your Fortran carriage controls to the comparable carriage controls for a LIST file. Enter the commands:

```
$  @SYS_PUBLIC:CONVTR program.OUT
$  APPEND filename.MAP filename.LIS
$  APPEND  program.TMP001 filename.LIS
$  PRINT /QUEUE=3800 filename.LIS
```

where "CONVTR" will replace any ones and zeroes in the first byte of the record with the appropriate form feeds and carriage returns. Pluses as the first character of the record indicate overstriking of the preceding records. The resultant file will have a filetype of "TMP001."

Once you have converted the record format of your Fortran output file to contain CARRIAGE RETURN CARRIAGE CONTROL print characteristics, you can append all the files (with like print characteristics) into a single file. You can then send that file to a printer and expect properly formatted output.

Most of the time you will be dealing with the LIST type of carriage control. If you find that the system is not interpreting carriage controls as you would expect, call the User Services Consultants at extension 2-5405 for assistance.

Table 8

**Default File Characteristics for Five Common Creation Techniques**

| *How Created* | *Default File Organization* | *Record Format* | *Record Attributes (Print Characteristics)* |
|---|---|---|---|
| EDT, EVE, LSE editors | Sequential | Variable | Carriage return carriage control or List |
| System generated batch log file | Sequential | VFC, 2 byte header | Print file carriage control |
| DCL DEFINE/ASSIGN Command redirecting "SYS$OUTPUT" | Sequential | Variable | Carriage return carriage control or List |
| Fortran program | Sequential | Variable (unless RECORDTYPE specified other) | Fortran carriage control |
| Compiler and Linker LIS and MAP files | Sequential | Variable | Carriage return carriage control or List |

## VAX PRINTER QUEUES

You can send files to a remote printer that is attached to a Local Area Transport (LAT) Ethernet terminal server. Currently, there are several such output devices defined on the central VAX cluster. To view a list of all the VAX printers, including the printers in your area, enter the DCL **SHOW QUEUE** command:

```
$ SHOW QUEUE /DEVICE=(PRINTER,TERMINAL) *
```

The system will display on your terminal all print queues defined on the central VAX cluster. Each queue entry will include the name, status, and description of the queue.

For instance, two currently defined print queues are:

```
ANEL05_LN03_PRINT  (Electronics--LN03)
DSAPS2_QMSPS800_PRINT  (Advanced Photon Source--QMS800--PostScript)
```

To send a file to these types of print queues, enter:

```
$ PRINT/QUEUE=queue-name filename.filetype/options
```

To send an ASCII file to the IBM 3800 laser printer, enter:

```
$ PRINT/QUEUE=3800 filename.filetype
```

Please note that the logical name "SYS$PRINT" (which typically is assigned to a default system output device) is undefined on the central VAX cluster allowing you to define your own default printer. Redefining "SYS$PRINT" allows you to conveniently print within VMS applications (e.g., Mass-11, MAIL, DIRECTORY MANAGER). You can use the **PRINT** command without having to specify a queue name, for instance:

```
$ PRINT/PARAMETER=TEXT filename.filetype
```

To define the logical name "SYS$PRINT," enter:

```
$ DEFINE SYS$PRINT queue-name
```

To have the IBM 3800 laser printer as your default printer, enter:

```
$ DEFINE SYS$PRINT 3800
```

To make one of your printers available from a VAX print queue, contact the Consulting Office (Building 221, Room A-139, extension 2-5405).

## PRINTING FILES THROUGH NJE

You can use the Network Job Entry (NJE) emulator to transfer print files from the VAX 8700 to other systems. The **SEND PRINT** command sends files to other Argonne file transfer network printers. The **PRINTPS** command will send an ASCII file to a PostScript device, and the **LISTPS** command will send a PostScript file to a PostScript device. Please note that when referring to the **PRINTPS** and the **LISTPS** commands, you can also use the **PSPRINT** and the **PSLIST** commands, respectively. Examples and syntax for online NJE HELP are available for transferring files by entering:

```
$ HELP @NJE
```

To send ASCII files to output devices, refer to the following examples:

- To send an ASCII file to a remote node as a print file, enter:

```
$  SEND PRINT filename.filetype node destid [forms]
```

where "node" is the nodename of the remote node where the destid is located (Argonne file transfer network nodenames and BITnet nodenames are valid), "destid" is the device to which the file is being sent, and the optional "forms" is either LINE (for IBM 3800 laser printer lined background) or POSTSCRI (for PostScript-formatted files).

- To send an ASCII file to a printer connected to the High Energy Physics VAX (node ANLHEP), enter:

```
$  SEND PRINT MYPROG.DAT ANLHEP PRINTER
```

- To send an ASCII file to the fiche machine, with or without titles, enter:

```
$  PRINT/PARAMETER=TEXT/QUEUE=FICHE filename.filetype
```

- To send a PostScript-formatted file to a central IBM PostScript printer, use the **LISTPS** command:

```
$  LISTPS filename.filetype node::destid LAND
```

or

```
$  LISTPS filename.filetype node::destid PORT
```

where the optional "node::" is the name of the node to which the printer is connected (the default when "node::" is omitted is ANLOS), "destid" is the PostScript output device, the optional "LAND" or "PORT" defines the page orientation (LANDscape=11" by 8 1/2" and PORTrait=8 1/2" by 11").

- To send a PostScript-formatted file to a PostScript device in Building 221, Room A-120, enter the command:

```
$  LISTPS postscript.file RM113PR2
```

- To send an ASCII file to a PostScript output device, enter:

```
$  PRINTPS fname.ftype node::destid option1=n,option2=n ...
```

where "fname.ftype is the name of the file, the optional "node::" is the name of the node to which the printer is connected (the default when the nodename is omitted is ANLOS), "destid" is the PostScript output device, and the options include "PORT", "LAND", "FONT", "SIZE", "MAR", and "COPIES." "PORT" and "LAND" are orientation selections (8 1/2" by 11" [the default] and 11" by 8 1/2" respectively). "FONT=shortfontname" selects a PostScript font (the default is Courier).[3] "SIZE=n" specifies the fontsize (or pointsize) desired (the default is 10). "MAR=n" specifies a shift to the right of the left margin. "COPIES=n" specifies the number of copies to be printed (the default is one).

[3] To view the available fonts on your terminal screen, enter:

```
HELP PSPRINT OPTIONS FONT AVAIL_FONT
```

To print two copies of an ASCII-formatted file in 13 point Courier on the Apple LaserWriter in Building 221, Room A-120, enter:

```
    $  PRINTPS file.dat RM113PR2 COPIES=2 SIZE=13
```

To view online HELP for the previous tools, enter:

```
    $  HELP @ANLUTIL
```

# CHAPTER 7
# DEVELOPING PROGRAMS IN VMS

VMS provides an efficient environment for developing programs that consists of three basic steps:

1. Create your source code (by using one of the editors).

2. Compile and link your code.

3. Debug your code.

Because VMS compilers for every VMS language generate object modules in the same format, you can use different languages to create routines. You can then link all the modules together with the VMS linker (along with any libraries you might need) to create an executable program (an image). VMS has an Interactive Symbolic Debugger that will aid you in trouble-shooting run-time errors or logic difficulties.

Programming languages available to VMS users for creating programs include Basic, C, Fortran, Macro (the VAX/VMS Assembler), and Pascal. For more information on these languages, see Chapter 9.

## THE DEVELOPMENT PROCESS

To develop a program in VMS, follow these five steps:

1. Create a program in an editor, such as EDT, the Extensible VAX Editor (EVE), or the Language Sensitive Editor (LSE). Write the program in one of the languages mentioned above.

2. Compile the program, using the Basic, C, Fortran, or Pascal compiler or the Macro assembler (depending on the language you chose) to create an object module.

3. Link all external modules to generate an executable program. These may include modules from the mathematical libraries (SLATEC, IMSL, NAG), modules from the Disspla graphics library, and modules written in a different language.

4. Run the program.

5. Debug the program if run-time errors are encountered.

Currently, the central VAX cluster has Basic, C, Fortran, and Pascal compilers available, as well as the Macro assembler. You can invoke these compilers by using the commands **BASIC, CC, FORTRAN, PASCAL,** and **MACRO** respectively. For example, given a file "PROG.FOR" containing Fortran source code, you would issue the following sequence of commands to compile, link, and run the program:

```
$  FORTRAN PROG
$  LINK PROG
$  RUN PROG
```

Note the omission of the filetype in each command line. The Fortran compiler assumes a filetype of FOR and generates an object file with the same filename as the file being compiled and with a filetype of OBJ. The linker assumes a filetype of OBJ and generates an executable image with the filetype EXE. Finally, the **RUN** command assumes a filetype of EXE. Many options (or qualifiers) are available at each step; descriptions of these options are available through the HELP utility.

The VAX/VMS Symbolic Debugger is a source code development tool that allows you to trace and correct bugs in a program that runs with unexpected results or terminates abnormally because of run-time errors. If you use the debugger, you need to compile your source code with the DEBUG and NOOPTIMIZE options. You must use the DEBUG option also in linking the object module. For further information on this debugger, refer to the *VMS Debugger Manual (AA-LA59B-TE).*

## VAX 8700 ARITHMETIC FEATURES

The VAX 8700 has all of the VAX floating-point arithmetic formats (F, D, G, and H) implemented in hardware to achieve the best processing speeds in extended arithmetic calculations. Users who need the flexibility of floating-point arithmetic with an increased exponent range will benefit. The VAX 8700 has hardware for extended floating-point formats called G-floating and H-floating, unlike the older VAX-11/7nn generation computers (which are equipped with F-floating and D-floating hardware but emulate other formats in the software). MicroVAX II computers have hardware for G-floating but not for H-floating.

Table 9 compares the four floating-point formats. Of the two double-precision formats, the advantage of the G format over the D format is the increased exponent range. The increased range simplifies the algorithms required to solve problems where there exists a very wide variation in the magnitude of numbers. The advantages of the H format are the large number of significant digits and the very large exponent range.

Fortran users may select the G-floating format rather than the default D-floating format for double precision arithmetic by coding the G_FLOATING option on the compile directive. For example, to compile the Fortran modules contained in the file "T48AB.FOR" (with G-floating format), enter:

```
$  FORTRAN /G_FLOATING T48AB
```

In Fortran, use of the G_FLOATING option selects the G-floating format for DOUBLE PRECISION, REAL*8, DOUBLE COMPLEX, and COMPLEX*16 declarations and constants.

C language users may use G-floating for values of type DOUBLE by selecting the G_FLOAT option on the compile statement:

```
$  CC /G_FLOAT c_program
```

C users must link the G_FLOAT library (SYS$LIBRARY:VAXCRTLG) as well:

```
$  LINK module,SYS$LIBRARY:VAXCRTLG/lib
```

Pascal users can use G-floating for values of type DOUBLE by selecting the G_FLOATING option as follows:

```
$  PASCAL /G_FLOATING pascal-program
```

BASIC users can use G-floating for floating-point data by selecting the REAL_SIZE option as follows:

```
$  BASIC /REAL_SIZE=GFLOAT basic-program
```

Table 9

**VAX Floating-Point Format Characteristics**

| | *Number of Bits* | | | | *Approximate* | *Approximate* |
|---|---|---|---|---|---|---|
| *Format* | *Exponent* | *Mantissa* | *Sign* | *Total** | *Exponent Range* | *Significant Digits* |
| F | 8 | 24 | 1 | 32 | ±38 | 7 |
| D | 8 | 56 | 1 | 64 | ±38 | 16 |
| G | 11 | 53 | 1 | 64 | ±308 | 15 |
| H | 15 | 113 | 1 | 128 | ±4932 | 33 |

* The mantissa is stored in normalized form with the most significant bit (always a 1) omitted.

## USING MATHEMATICAL AND SCIENTIFIC LIBRARIES

Scientific subroutine libraries available on the VAX 8700 include the International Mathematical and Statistical Libraries (IMSL), the Numerical Algorithms Group (NAG), and the Sandia Laboratory-Albuquerque Technical Library (SLATEC). These libraries contain Fortran-callable routines that you can use by linking them to your program.

### International Mathematical and Statistical Libraries (IMSL)

To link the object module of a program to the IMSL libraries, enter:

```
$  LINK PROGRAM,IMSL/OPT
```

### Numerical Algorithms Group (NAG)

To link the object module of a program to the NAG routines, enter:

```
$   LINK PROGRAM,NAG/OPT
```

If your program uses the G-floating point data type, enter

```
$   LINK PROGRAM,NAGG/OPT
```

to access the NAG routines compiled with G-floating point options.

Online help is available on NAG routines by entering:

```
$   HELP @NAG
```

### Sandia Laboratory-Albuquerque Technical Library (SLATEC).

To link the object module of a program to the SLATEC routines, enter:

```
$   LINK PROGRAM,SLATEC/OPT
```

If your program uses the G-floating point data type, enter

```
$   LINK PROGRAM,SLATECG/OPT
```

to access the SLATEC routines compiled with G-floating point options.

For listings of available documentation on these and other scientific subroutines libraries, see *Recommended Documentation for Computer Users at ANL* (ANL/TM 379).

## USING RUN-TIME LIBRARY ROUTINES

A Run-Time Library of routines is available on VAX systems. The VAX/VMS Run-Time Library allows a program to access VAX system services (e.g, create directories, delete files). This Library is accessible to programs written in any VMS programming language by making the appropriate call to the routine. For more specific information on this Library, see the *VAX/VMS Run-Time Library Routines Reference Manual* (AA-LA76A-TE, AA-LA70A-TE, and AA-LA77A-TE), or enter:

```
$   HELP RTL
```

## THE VAX/VMS SYMBOLIC DEBUGGER

The VAX/VMS Symbolic Debugger permits you to interact with your code while it is executing. You must compile your source code with the "/DEBUG/NOOPTIMIZE" options, as in the following example:

```
$ FORTRAN/DEBUG/NOOPT  filename
```

You must then use the **LINK/DEBUG** option as follows:

```
$ LINK/DEBUG filename
```

When you issue the **RUN** command, the system will invoke the Debugger. If you are using a DEC VTnnn terminal, you may enter

```
DBG> SET MODE SCREEN
```

which allows you to use the debugger in full-screen mode. Unless you redefine the logical name "SYS$INPUT" to some filename, the Debugger will look to you for input. You can then use the Debugger commands to **STEP** through your program's execution, **EXAMINE** variables, and, if necessary, **DEPOSIT** different values. This tool is extremely helpful in debugging your code. Enter **QUIT** to exit from the Debugger. Help is available while in the Debugger by entering **HELP** at the "DBG>" prompt. For further information, refer to the *VMS Debugger Manual* (AA-LA59A-TE).

# CHAPTER 8

# USING VMS BATCH, COMMAND PROCEDURES, AND SUBPROCESSES

This chapter describes the use of DCL command procedures and their use in running batch jobs.

## VMS BATCH

Using VMS batch as opposed to performing strictly interactive computing may be preferable to you, because your processing charges are less, and you receive a log of your session. We recommend the batch mode of operation for:

- Non-interactive tasks that derive no benefit from the interactive capabilities of timesharing systems.
- Long jobs whose resource requirements lead to unreasonably long terminal sessions and degrade the service of other terminal users.
- Routine, repetitive, or periodic housekeeping tasks.
- Jobs whose results are not needed immediately.
- Costly jobs that can take advantage of the lower rates charged for batch.

When executing a batch job, your "login.com" file is executed first and your process is left in your root directory. If the files you need are in a subdirectory, you must issue a **SET DEFAULT** command in your batch procedure to put you in the correct subdirectory for executing your program.

To submit batch jobs to the central VAX 8700, enter:

```
$   SUBMIT/QUEUE=queue-name/CPUTIME=time filename.filetype
```

where "queue-name" specifies your choice of queues (see Table 10), "time" allows you to override the default CPU time limit up to the maximum for the queue, and "filename.filetype" is a command file containing DCL commands controlling your batch job.

To obtain detailed information on the available options for the **SUBMIT** command, enter:

```
$   HELP SUBMIT
```

Table 10

**Central VAX 8700 Batch Queues**

| *Name* | *CPU Default* | *CPU Limit*† | *When Started* |
|---|---|---|---|
| Short_W_Batch (Default "SYS$BATCH")‡ | 5 min. | 5 min. | Always |
| W_Batch | 15 min. | 60 min. | Always |
| X_Batch | 15 min. | 60 min. | 7:00 p.m - 7:00 a.m. |
| Y_Batch | 15 min. | 60 min. | Sat. 7:00 a.m. - Mon. 7:00 a.m. |
| Special_X_Batch | 15 min. | no limit | Sat. 7:00 a.m. - Mon. 7:00 a.m. |
| Special_Y_Batch | 15 min. | no limit | Sat. 7:00 a.m. - Mon. 7:00 a.m. Low priority, when little or no other work remains |

The X_BATCH queue will automatically start processing the submitted jobs on workday evenings at 7:00 p.m. and will stop at 7:00 a.m. the following morning. Any jobs still in the queue at the time the queue is stopped will be delayed until the queue is started again. The X_BATCH queue is inactive whenever the Y_BATCH queue is active.

The Y_BATCH and SPECIAL_X_BATCH queues automatically start processing the submitted jobs on Saturday morning at 7:00 a.m. The queues are stopped the following Monday morning at 7:00 a.m. Any jobs still in the queues at the time the queues are stopped will be delayed until the queues are started again. These batch queues are started during regular ANL holidays. Jobs submitted to the SPECIAL_X_BATCH queues are on a very low priority and are processed when the CPU has little or no other work.

The SPECIAL queues require operator intervention to start processing long jobs and jobs that require large memory. The operators will start jobs in the SPECIAL_X_BATCH and SPECIAL_Y_BATCH queues whose CPU time limits are less than three and eight hours respectively without requiring you to inform them beforehand to expect such jobs. Contact the consultants on extension 2-5405 or the operators on extension 2-5421 for SPECIAL queue jobs that exceed these limits to allow the operators to ensure that long-running jobs are not terminated by scheduled maintenance activities.

† The VAX 8700 batch queue CPU limit ranges from five minutes to an unlimited amount of time. However, note that should your job abort because of a system failure or an error committed by the Operations staff, you are entitled to a refund of *no more than fifteen minutes of CPU time.* Note also that Account Services does not process refunds of less than fifty dollars.

‡ Jobs submitted without a queue name are entered into the "SYS$BATCH" queue, which for the VAX 8700 is equivalent to SHORT_W_BATCH. If you wish, you may override this definition by defining another existing queue to be equivalent to "SYS$BATCH."

An example of the SUBMIT command is:

```
$  SUBMIT/QUEUE=X_BATCH SAMPLE
```

DCL will look for a filename "sample.com" to submit to the "X_BATCH" queue. The CPU time limit will be the default for the "X_BATCH" queue.

To obtain information on the state of all batch jobs in the system, enter:

```
$  SHOW QUEUE/ALL/BATCH
```

The VAX 8700 has several batch job queues available, differing in charge rates, availability, and characteristics (e.g., CPU time, job limits, and scheduling priorities).

To see specific batch characteristics on your screen, enter the DCL command:

```
$   SHOW QUEUE/BATCH/FULL
```

When a batch job is submitted, it is assigned an entry number which can be used to reference it. The "SHOW QUEUE/ALL/BATCH" command will show you the entry number. This number is necessary should you decide to abort the job. For example, to delete an entry from a batch queue, enter:

```
$  DELETE/ENTRY=entry-number
```

To abort a batch job that is executing, enter:

```
$  STOP/ENTRY=entry-number
```

Do not modify or purge the command file if you have already submitted a job and it has not yet started. The queue entry contains the complete file specification, and the file specification includes the version number. A way to achieve flexibility in modifying a command file at any time is to submit another command file that invokes the command file that contains the DCL for the real job.

## COMMAND PROCEDURES

A command procedure is a user-created file containing DCL commands (e.g., your "login.com" file is a command procedure that the system automatically executes for you whenever you logon). Command procedures can be executed interactively or in batch sessions.

### Creating Command Procedures

Some of the basic rules for creating command procedures are:

1. Use any of the available editors to create a command procedure. Assign the file a name with the filetype of COM as you would ordinarily to create a file.

2. You must preface each DCL command with the dollar sign.

3. The comment designator is a dollar sign followed immediately by an exclamation mark; the DCL command processor ignores all characters to the right of the exclamation mark.

4. Data lines do not begin with a dollar sign.

5. A hyphen designates continuation to the next line. Commands may continue over multiple lines.

6. Parameter values are assigned to symbols "P1" through "P8."

7. DCL will substitute values for symbols, where the symbol is surrounded by apostrophes.

8. Lexical functions and logical names are valid.

9. You can invoke command procedures from within another command procedure.

10. You can suppress unwanted system messages from within command procedures by entering the command line

```
$  DEFINE SYS$OUTPUT NL:
```

in your command procedure.

For more detailed instructions on command procedures, refer to *Guide to Using VMS Command Procedures* (AA-LA11A-TE).

The following is an example of a command procedure. It compiles Fortran code, with or without options, and then links that code, with or without options:

```
$!  The following SET NOON command inhibits error checking.  The
$!  programmer can check system symbols $STATUS and $SEVERITY
$!  (Odd values are TRUE and Even values are FALSE).
$!
$  SET NOON
$! Value of parameter one (P1) is being assigned to FILE.
$! If no parameter is provided, the system will prompt the
$! user for one.
$ FILE = P1
$ IF FILE.EQS. "" THEN INQUIRE FILE -
     "Name of Fortran source"
$! Ask user if user wants the code to be compiled with the
$! DEBUG option.  Then compile code.
$ INQUIRE ANSW "DEBUG OPTION? (Y/N)"
$ IF ANSW .EQS. "Y" THEN -
     OPT = "/LIST/DEBUG/NOOPTIMIZE"
$ FOR 'FILE''OPT'
$! Compile success or failure is saved in symbol "$STATUS."
$! If compile is unsuccessful, go to label "FIX."
$ IF .NOT. $STATUS THEN GOTO FIX
$! Compiled successfully.  Change options for "DEBUG" to
$! correct linker options.
$ IF ANSW .EQS. "Y" THEN -
     OPT = "/MAP/CROSS_REFERENCE/DEBUG"
$ LINK 'FILE''OPT'
$ EXIT
$!
$! Compilation errors found.  Begin editing source code.
$ FIX:
$ DEFINE SYS$INPUT SYS$COMMAND
$ EDIT 'FILE'.FOR
```

### Executing Command Procedures

Once you have written a command procedure, you can initiate execution of the file interactively by using the "at" sign (@). You may spawn a subprocess to execute the command procedure by using the **SPAWN** command, or you may execute the command procedure in a batch queue by using the **SUBMIT** command.

The "at" sign will execute the commands in the file from within your current process:

```
$  @command-proc-name [parameter-1] [parameter-2] . . . [parameter-8]
```

where "command-proc-name" is the name of a file with a filetype of COM (the optional "parameter-1" through optional "parameter-8" would follow). You can enter this statement interactively at the DCL prompt, or it can be in a command procedure itself.

The **SPAWN** command will create a subprocess from which the command procedure will execute (see "Processes and Subprocesses" later in this Chapter for more details):

```
        $  SPAWN @command-proc-name
```

where "command-proc-name" is the name of a file with a filetype of "COM." You can enter this statement interactively at the DCL level, or it can be in a command procedure.

The **SUBMIT** command will create a new process. Your "LOGIN.COM" procedure will execute in the new process which will have its own symbols, logicals, quotas, etc. The new process will be in batch mode with a lower priority (see the beginning of this chapter for more details):

```
$  SUBMIT [/PARAMETERS=(parameter-1,. . .parameter-8)] command-proc-name
```

The "command-proc-name" is the name of a file with a filetype of COM. The optional PARAMETER qualifier is used to specify the parameters to be used by your command procedure. Having to use the optional PARAMETER qualifier (to pass parameters to a batch command procedure) is different from interactively invoking a command procedure in which the parameters are positional on the command line. You can enter this statement interactively at the DCL level, or it can be in a command procedure.

## PROCESSES AND SUBPROCESSES

VMS recognizes any session as a process--the term used to describe your interaction with the operating system. Any time you begin a session (open a new process), you are in either interactive or batch mode. When you submit a command procedure to a batch queue, you are initiating a process in batch mode in which the queue manager starts your job. All batch queues other than system queues process jobs at a lower dispatching priority than interactive queues.

Your initial login process, whether interactive or batch, is called the parent process and can have up to ten subprocesses, called children. A child process shares the parent's quotas, symbols, priorities, logical names, default CLI, default disk, and default directory. The "login.com" file does not execute when you create the child process. Use the DCL **SPAWN** command to create a subprocess:

```
        $  SPAWN [command-string]
```

where the optional "command-string" can be either a DCL command or a command procedure. The parent process hibernates until the child process has completed its task, unless you instruct DCL (with the NOWAIT qualifier) not to suspend the parent process. If you use the **SPAWN/NOWAIT** command and logoff your parent process before the child process has completed its task, the child process will terminate. Subprocesses cannot exist without an existing parent process.

Spawning a subprocess can be useful if you want to continue issuing DCL commands while not destroying your current process. For instance, you could suspend your process by using ^Y and then **SPAWN** a subprocess to do other work. You could then return to the parent process and issue the DCL **CONTINUE** command to pick up where you left off. How you return to the parent process depends upon the command you used to create the subprocess. If you issued the **SPAWN** command *without* a command-string, you must use the DCL **LOGOUT** command. If you issued the **SPAWN** command with the command-string equal to some DCL command (e.g., **SPAWN MAIL**) the system will return you to your parent process when the task is complete, (e.g., when you **EXIT** MAIL).

You can **SPAWN** a subprocess to execute a command procedure by entering:

```
$   SPAWN @command-proc-name
```

where the "command-string" is an "at" sign (@) followed by a command procedure name. This command line will create a child process, execute the commands contained in the command procedure, and return control to the parent process (which will be hibernating). To continue working in the parent process while the child process is open, enter:

```
$   SPAWN/NOWAIT @command-proc-name
```

where "NOWAIT" is used as a qualifier, remember that each process shares the same logical names ("SYS$INPUT" and "SYS$OUTPUT"). A command procedure executing in a subprocess that writes to your "SYS$OUTPUT" will appear on your terminal while you are working in your parent process.

# CHAPTER 9

# USING AVAILABLE SOFTWARE

Table 11 lists software available on the central VAX 8700, and following sections describe much of this software.

Table 11

**Available Software**

| | |
|---|---|
| * ANSYS | * Ingres |
| Basic | Kermit |
| C | Language Sensitive Editor (LSE) |
| * CERNLIB | * LaTeX Text Formatter |
| Code Management System (CMS) | Macro Assembler |
| Codebook | * MACSYMA (with LISP processor, EMACS editor) |
| Common Data Dictionary (CDD) | * Mass-11 Word Processor |
| * Cuechart (setup Tellagraf) | SLADOC |
| Data Connection | Module Management System (MMS) |
| Datatrieve | * NAG Mathematical Library |
| DECalc-Plus | Pascal |
| DECShell | * PME |
| Directory Manager | * Prolog (with EMACS editor) |
| * Disspla | REPCON |
| Disspla-GKS | * SAS |
| SLATEC Mathematical Library | SAS/Graph |
| EDT Text Editor | TDMS Forms Management |
| EVE Text Editor | * Tellagraf |
| FILCON | TeX Text Formatter |
| Forms Management System | * Toolpack |
| Fortran | VAX Sort/Merge Utility |
| * GNUEMACS | VAX/VMS Symbolic Debugger |
| IMSL Mathematical Library | * WYLVAX |

* Use the **SETUP** command.

To view a list of products whose symbols and logicals are not included in the global symbol table, use the **SETUP LIST** command.

## THE ANSYS ENGINEERING ANALYSIS PROGRAM

The ANSYS program (developed by Swanson Analysis Systems) has many engineering applications. It contains many routines that will solve engineering problems by the finite element method. You can use it on the central VAX 8700 both interactively and in batch mode. The default graphics device type for this program is a VT241. Use this program by first setting up the appropriate symbols:

```
$  SETUP ANSYS
```

To execute the program, enter:

```
$  ANSYS
/int
```

where "/int" indicates interactive use. To terminate the program, enter (at the "BEGIN-INP=" prompt):

```
 /EOF
```

For further information on ANSYS, see the *ANSYS Engineering Analysis System User Manual,* Volumes I and II.

## THE BASIC LANGUAGE

To invoke the Basic compiler, enter:

```
$  BASIC filename
```

at the DCL prompt. The compiler will look for a filetype of ".BAS" by default. Enter **HELP BASIC** at the DCL prompt to display the available compiler options.

Use the "/DEBUG" qualifier when you need to run the VMS Symbolic Debugger program on your code. The "/NOOPTIMIZE" qualifier is not applicable to Basic. You then need to link your object module with or without the "/DEBUG" qualifier, depending on what you have done at the compile step.

## THE C LANGUAGE COMPILER

You must access the C library of routines prior to linking your C object code, by entering:

```
$  DEFINE LNK$LIBRARY SYS$LIBRARY:VAXCRTL
```

You may need to define additional library definitions, depending on the needs of your source code. Two other C libraries you may need are "SYS$LIBRARY:VAXCRTLG" for G-floating point data type routines and "SYS$LIBRARY:VAXCCURSE" for the Curses Screen Management functions and macros (the VAX C Screen Management package). Review the online HELP utility for more information on link libraries:

```
$  HELP CC LINK_LIBRARIES
```

To invoke the C compiler and generate object modules for each file, enter:

```
$  CC filename1, filename2
```

By default the compiler will look for a filetype of "C."

To generate one object module through the C compiler, enter:

```
$  CC filename1 + filename2
```

Use the "/DEBUG/NOOPTIMIZE" qualifiers with the compiler when you need to run the VMS Symbolic Debugger. You can then link your object module with any library by using the **LINK** command (at the DCL prompt) with or without the "/DEBUG" qualifier, depending on what you did during the compile.

## CODEBOOK, CUECHART, DISSPLA, AND TELLAGRAF

Refer to "Using Graphics in VAX/VMS" in Chapter 10 for more details.

## THE CODE MANAGEMENT SYSTEM (CMS)

The VAX/DEC Code Management System (CMS) maintains large systems of online code by storing routines in a library. CMS systematically keeps track of user modifications to the code and records all user access. Online information is available by entering:

```
$  HELP CMS
```

More detailed information is available in *Guide to VAX DEC/Code Management System* (AI-KL03A-TE).

## COMMON DATA DICTIONARY (CDD)

The VAX/DEC Common Data Dictionary (CDD) is used to store shareable data definitions in a hierarchical data base. VAX language compilers and VAX information layered products, like Datatrieve, have access to these definitions. Refer to DEC's reference manual on CDD for more detailed information or enter:

```
$  HELP CDD
```

## DATATRIEVE

Datatrieve is an interactive VAX tool used to create databases. To invoke Datatrieve, enter:

```
$  DTR
```

You will then receive the "DTR>" prompt. Enter **EXIT** at the "DTR>" prompt to return to DCL.

Two online tutorials for Datatrieve are available. *Datatrieve for Users* is available by entering:

```
$  RUN DTRCAI
```

*Datatrieve for Programmers* is available by entering:

```
$  RUN DTRPCAI
```

Users should refer to the DEC Datatrieve user guides and reference manuals for detailed information.

## THE DECALC-PLUS SPREADSHEET

DECalc-Plus is a powerful VAX spreadsheet package that is functionally comparable to most personal computer spreadsheet packages; with one exception, large files load much faster with DECalc-Plus. DECalc-Plus users also have the capability of combining DECalc-Plus with user-written programs and using the spreadsheet for either input, output, or both.

Enter **HELP CALC** to receive online HELP at the DCL level. Refer to *VAX Decalc-PLUS V3.0* for more detailed information.

To invoke DECalc-plus, enter:

```
$  CALC
```

In DECalc, you must preface all commands with:

```
/
```

To return to the DCL level, enter:

```
/E
```

Online context-sensitive HELP is available in DECalc by pressing the PF2 key.

DECalc creates a subdirectory called "DECALC_GRIDS" for you in which to store your spreadsheet files.

## DECSHELL

The Bourne shell is available in most of the other versions and implementations of the Unix operating system, including Cray UNICOS. The DECShell can help you if you have programs that use certain features of the shell environment, if you are already familiar with Unix tools, or if you need a Unix environment to test Bourne shell scripts (procedures) that you are developing for the Cray X-MP/14 UNICOS system.

CTD acquired the DECShell as part of the Digital VAXset package, which includes VAX C, the VAX Code Management System (VAX CMS), and the VAX Module Management System (VAX MMS). The Bourne shell environment that the DECShell product implements is an alternate to the Digital Command Language (DCL) environment. However, DCL is accessible from the DECShell and vice versa.

The standard Bourne shell features are available in the DECShell product. Some of the best known features for shell script programming are standard files (stdin, stdout, stderr), input-output redirection, pipes, and structured constructs (if...then...else, for, while, until, break, etc). In addition, the DECShell product includes many Unix commands. An abbreviated list of the DECShell commands and a short description of each appears in Table 12.

The DECShell also includes the awk, lex, m4, and yacc tools. A complete list of commands and programs appears in the online documentation (see "DECShell Help").

Table 12

**Sample List of DECShell Unix Commands**

| *Name* | *Description* |
|---|---|
| alias | enables environment variables to be used as command synonyms |
| cat | concatenates and displays files |
| cd | changes working directory |
| chmod | changes mode of file (i.e., protection bits) |
| cp | copies file(s) |
| diff | compares files and displays differences |
| echo | echoes arguments |
| ed | invokes standard Unix text editor |
| exec | executes a command and exits |
| export | passes environment variables to programs and sub-shells |
| find | finds files |
| grep | searches file for pattern |
| join | joins files according to specified relations |
| logout | terminates an interactive terminal session |
| ls | lists contents of directory |
| mkdir | creates a directory |
| mv | renames files and directories |
| pr | formats and prints file |
| pwd | prints working directory name |
| read | assigns the values of the next line read to the specified variable |
| rm | removes or deletes files |
| rmdir | removes a directory |
| set | sets shell flags or displays current environment variables |
| sh | executes shell command script or creates a shell subprocess |
| sort | sorts or merges files |
| tar | archives tapes |
| tee | fits pipes |
| tr | transliterates characters |

## Invoking the Unix Bourne Shell

The Bourne shell serves as a standard VMS command line interpreter (CLI) and executes as an alternate to the Digital Command Language CLI. Depending on your individual needs, you may invoke the Bourne shell in several ways, including (1) at login, (2) as the CLI on the batch job SUBMIT command, (3) as your default CLI, and (4) in a DCL subprocess.

You can invoke the Bourne shell as your CLI at login time when you respond to the username prompt as follows:

```
Username: username/CLI=SHELL
```

After you respond to the password prompt that follows, you will receive the default "%" prompt and may enter Bourne shell commands.

In batch jobs you can enter the CLI option on the DCL submit command as follows:

```
$  SUBMIT/CLI=SHELL <shell script>
```

Enter additional options on the **SUBMIT** command as needed. Note that **SUBMIT** is a DCL command that you execute either from the DCL environment or from the Bourne shell environment.

To select the Bourne shell as your CLI, you may request that the CLI for your account be changed from DCL to the Bourne shell. Contact Account Services at extension 2-5425 or come to Building 221, Room A-147, to request the change. After the change, you will not need to specify the CLI at login or in batch jobs unless you wish to invoke DCL instead.

If you normally use DCL but occasionally wish to execute Bourne shell commands or programs, you may create a VMS subprocess that invokes the Bourne shell. Use the DCL **SPAWN** command as follows:

```
$  SPAWN/CLI=SHELL
```

You will then receive the "%" prompt, and you can execute Unix (Bourne shell) commands. Return to the DCL environment by entering **LOGOUT** or pressing Ctrl-Z to terminate the subprocess.

## Access to DCL

The **DCL** command in the Bourne shell environment permits execution of DCL commands

```
%  dcl <dcl-command line>
```

or with no argument creates a new DCL shell subprocess:

```
%  dcl
$  ...
```

You may avoid entering the string **dcl** prior to each DCL command by entering the shell `set -d` command. After this, if the shell does not recognize a command, it will search the DCL tables and execute the corresponding commands. The `set +d` command reverses this behavior if needed.

### The Unix File System

The DECShell product gives you a Unix-like view of the VAX/VMS file system. Unix and VMS file systems are very similar, each being hierarchical (tree-structured). Unix commands manipulate files by using Unix file system syntax. Although Unix has no version-number concept, the DECShell product commands and tools view the version number as part of the file name; the period replaces the semicolon as a separator between the filetype and version number. You can access the same files either by VMS (DCL) commands or by Unix (Bourne shell) commands. The specification of a VMS file in both DCL and Unix syntaxes follows:

DCL:

```
disk:[username.subdir...]nam.typ;ver
```

Unix:

```
/disk/dir/subdir.../nam.typ.ver
```

where "dir" is a top-level directory name, "subdir" is a subdirectory name, and "nam," "typ," and "ver" are the filename, filetype, and version number portions of the filename. The familiar Unix command to change the working directory (**cd**) enables you to reset your default disk and directory and then specify files by just their filename and type as in the DCL environment.

### DECShell Help

The VMS HELP utility provides online documentation for the DECShell product. You can obtain help on a Bourne shell command by entering (in DECShell):

```
% dcl help shell [shell feature]
```

Or by entering (in VMS):

```
$ help shell [shell feature]
```

For further Unix information, refer to *A Practical Guide to Unix System V* (0-8053-8915-6). For complete documentation of the DECShell product, refer to *Guide to VAX DEC/Shell* (AI-AU32B-TE). These documents are available at the Document Distribution Counter (Building 221, Room A-134) or through the mail (by calling extension 2-5405 and requesting a copy).

Terminal keys that you use to control your VAX session or to recall and edit your command lines serve the same function for the DECShell CLI as they do for all VAX CLIs.

## THE FORMS MANAGEMENT SYSTEM (FMS)

The DEC/VAX Forms Management System (FMS) is a VMS forms management utility. Online HELP is available for FMS by entering **HELP FMS** at the DCL prompt. You can find detailed information in the *VAX Forms Management System Utilities Reference Manual.*

## THE FORTRAN LANGUAGE COMPILER

To invoke the VMS Fortran compiler, enter:

```
$   FORTRAN filename
```

By default, the compiler will look for a filename with a filetype of ".FOR."

If you wish to view all the available compiler options, enter:

```
$   HELP FORTRAN
```

Use the "/DEBUG/NOOPTIMIZE" qualifiers when you want to use the VMS Symbolic Debugger and the "/LIST" qualifier when you want to produce a printable output of the program and error messages.

After invoking the compiler, you can then link the resulting object module(s) with any library by using the DCL **LINK** command. If you compiled with the Debugger, you should also use the "/DEBUG" qualifier. For further information, refer to Chapter 7 in this document or the *VMS Debugger Manual* (AA-LA59A-TE).

## THE KERMIT FILE TRANSFER PROGRAM

Kermit is a file transfer program. You can access online information about Kermit at the DCL prompt by entering:

```
$   HELP KERMIT
```

More detailed information appears in the *Kermit User Guide*

To invoke Kermit, enter:

```
$   KERMIT
```

You will receive the "Kermit-32" prompt.

To learn more about available Kermit commands, enter (at the "Kermit-32" prompt):

```
Kermit-32>  HELP
```

To return to the DCL level, enter:

```
Kermit-32>  EXIT
```

## THE LANGUAGE SENSITIVE EDITOR (LSE)

The VAX Language Sensitive Editor (LSE) is an editor that has the additional capability of providing you with language-specific templates that reduce your keystrokes during source code development and modification. You can create your own LSE templates and modify the provided templates. Multiple windows and buffers are available. You can also compile, review, and correct your syntax errors from within the editor. EVE and EDT keypads are available to you in LSE.

To invoke LSE, enter:

```
$  LSE fname.ftype
```

LSE will key in on the given filetype to determine which templates should be provided for you. For instance, the filetype of ".FOR" will dictate Fortran templates.

The display on your terminal will appear in full screen mode. To toggle between your display and the LSE command line (the "LSE>" prompt), press Ctrl-Z. Help is available at the "LSE>" prompt and by pressing the PF2 key from within your window.

To return to DCL and simultaneously save your file, enter:

```
LSE>  EXIT
```

If you wish to exit but not save your file, enter **QUIT** instead.

An LSE tutorial is available on the central VAX 8700. To invoke this tutorial, enter:

```
$  RUN LSECAI
```

You must use a VT100-compatible terminal and set your terminal device type to VT100 prior to invoking LSE. You should also refer to DEC's LSE user's guides and reference manuals for detailed information.

## LATEX AND TEX TEXT FORMATTERS

LaTeX and TeX are formatting languages available on the central VAX 8700 computer. TeX was developed by Donald E. Knuth of Stanford University to create technical documents containing complex mathematical equations. Figure 4 contains an example of text and equations produced by the TeX program.[4]

Although most users consider the generation of mathematical expressions to be the forte of TeX, TeX can also produce high-quality text and tables. For more information, see *The TeXbook* and the *LaTeX User Guide and Reference Manual.*

To use LaTeX or TeX, you must create a file with a filetype of ".tex" that contains LaTeX or TeX commands. Then, at the DCL level, enter

```
$  SETUP TEX
```

to create the TeX environment (this statement creates the environment for both TeX and LaTeX).

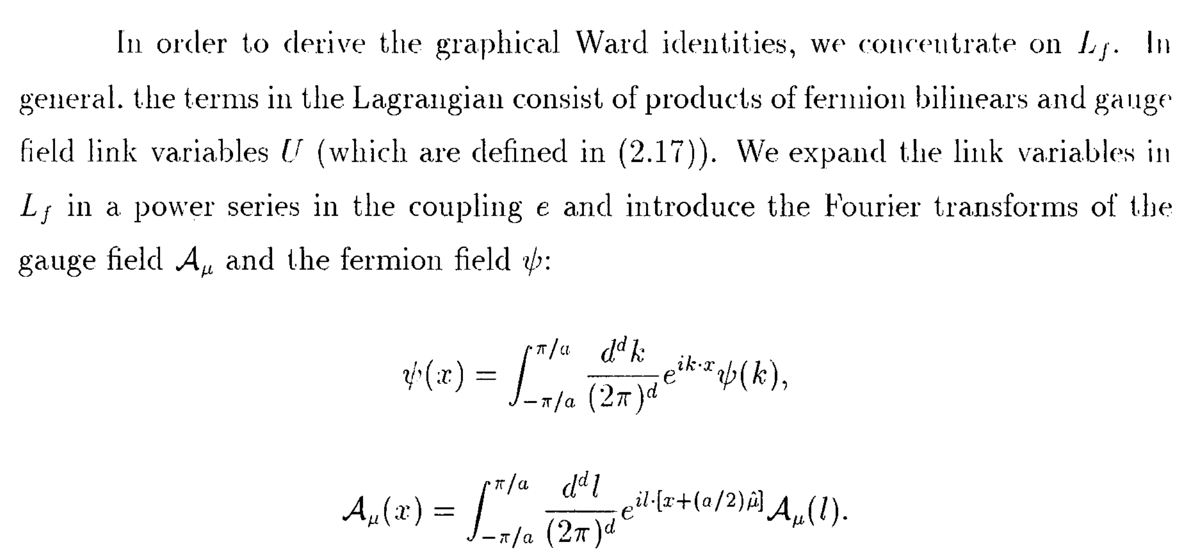
To invoke LaTeX or TeX (the default filetype is "TEX"), enter:

```
$  LATEX filename.tex
```

or

```
$  TEX filename.tex
```

[4] The example in Figure 4 is from Geoffrey T. Bodwin (High Energy Physics), *The Equivalence of Dirac-Kahler and Staggered Lattice Fermions in Two Dimensions* (ANL-HEP-PR-87-112), pp. 19-20.



In order to derive the graphical Ward identities, we concentrate on $L_f$. In general, the terms in the Lagrangian consist of products of fermion bilinears and gauge field link variables $U$ (which are defined in (2.17)). We expand the link variables in $L_f$ in a power series in the coupling $e$ and introduce the Fourier transforms of the gauge field $\mathcal{A}_\mu$ and the fermion field $\psi$:

$$\psi(x) = \int_{-\pi/a}^{\pi/a} \frac{d^d k}{(2\pi)^d} e^{ik\cdot x}\psi(k),$$

$$\mathcal{A}_\mu(x) = \int_{-\pi/a}^{\pi/a} \frac{d^d l}{(2\pi)^d} e^{il\cdot[x+(a/2)\hat{\mu}]}\mathcal{A}_\mu(l).$$

Figure 4: **Example of Text and Equations Produced by TeX**

This process may require you to respond to TeX messages. In some cases a carriage return will suffice. Refer to the reference manuals for the appropriate action. When the program returns you to the DCL prompt, you will find that the TeX program has created additional files in your current directory with filetypes of ".aux," ".lis," and ".dvi." The ".aux" file is used by TeX, and the ".lis" file contains a listing of the messages you received when you invoked TeX or LaTeX. TeX creates a ".dvi" file as the standard TeX output file (which is device-independent). To create a file for a PostScript printer (which is currently the only printer type available on the central VAX 8700 for TeX) from the ".dvi" file, enter:

```
    $  DVIALW filename
```

The default filetype for this command is "dvi."

The DVIALW program has optional parameters ("d" is a digit reflecting the appropriate unit):

| | |
|---|---|
| -cd | Select the number of copies. The default is one. |
| -md | Select font magnification. The default is 1500. Other valid magnifications are 913 for a smaller font and 1800 for a larger font. If the number you specify is unavailable, the program will substitute the closest size available. |
| -xdin | Select the width of the left margin. By default, your page margin will be calculated from any TeX or LaTeX settings and magnification parameter. This option will override all others if you use it. The number can be negative (e.g., -x-.3in, where ".3in" is three tenths of an inch) to shift your margin to the left. |

**-ydin** Select your top margin. By default, your page margins will be calculated from any TeX or LaTeX settings and magnification parameter. This option will override all margin settings if you use it. The number can be negative (e.g., -y-1.0in, where 1.0in = one inch) to shift your top margin up.

An example with these options is:

```
$  DVIALW -c5 -m946 -x.2in -y.2in sample
```

This PostScript device driver will create a file with a filetype of "dvi-alw." The font magnification is reduced to 946, both margins (top and left) are shifted two tenths of an inch, and five copies of the file will print. This file contains all the PostScript commands necessary to print your file on a PostScript device. You can send this file to any PostScript output device, as in the following example:

```
$  SEND PRINT filename.DVI-ALW ANLOS RM111PR1 POSTSCRI
```

where "RM111PR1" refers to the typesetter in Graphic Arts and "POSTSCRI" designates the file as a PostScript file. You can also send this file to any ANLOS PostScript output device with the **LISTPS** command. The **LISTPS** command designates the file as a PostScript file, as in the following example:

```
$  LISTPS filename.DVI-ALW ANLOS::RM113PR2
```

where "RM113PR2" refers to the public CTD Apple LaserWriter (Building 221, Room A-113).

To preview TeX documents before printing, you can use the DVITOVDU command. Choose option 5 of the LaTeX command to create a file that the previewer software can use.

```
$  DVITOVDU filename.DVI
```

## MACRO ASSEMBLER

VAX MACRO is an assembly language for programming VAX computers using the VMS operating system. Source programs written in VAX MACRO are translated into object (or binary) code by the VAX MACRO assembler, which produces an object module and optionally, a listing file.

## MACSYMA

Macsyma is used for performing symbolic and numerical mathematical manipulations. This program was developed by the Mathlab Group of the MIT Laboratory for Computer Science. You can use Macsyma to differentiate, integrate, take limits, solve systems of linear or polynomial equations, factor polynomials, expand functions in Laurent or Taylor series, solve differential equations, compute Poisson series, and manipulate matrices and tensors.

To invoke Macsyma, you must first assign the appropriate symbols and system logical names by entering:

```
$  SETUP MACSYMA
```

Then enter:

```
$  MACSYMA
```

To exit from Macsyma, enter:

```
    (Cn)  quit();
```

You can find detailed information on how to use this package from the Macsyma Reference Manuals available at the Document Distribution Counter (Building 221, Room A-134).

## THE MASS-11 WORD PROCESSOR

Mass-11 is a powerful VAX word processor that is menu-driven and easy to use. Mass-11 has the typical word processing features and you can review a document on your screen prior to sending it to a printer. To add the Mass-11 symbols to your process, enter:

```
    $  SETUP MASS11
```

Invoke Mass-11 by entering:

```
    $  MASS11
```

Mass-11 will look for a "DEFAULTS.DAT" file in the directory you are in and will need to create one the first time you invoke the word processor from that directory. The "DEFAULTS.DAT" file is a Mass-11 required file, so do not delete it. However, if you do delete it, Mass-11 will recreate it.

You will then see the Mass-11 main menu. The *first* time you invoke the program, you must create a folder from within Mass-11 for its internal use. To do so, first type **WP** for Word Processing from the main menu, then type **UT** for Utilities Menu from the word processing menu, and finally type **BF** for Build Document Folder which will prompt you for the name of the document folder to be built. You can then begin adding documents to this folder. Select **WP** from the main menu, then select **CR** to create a document for this folder.

Press the PF1 key and type **?** to receive online HELP. You can page through the topics by using the up- and down-arrow keys to select the topic, and then type enter to display the HELP on that topic.

Press the PF1 key and type **X** to save your document. Press the PF1 key, type **Q** , and then type **Y** , to return to the word processing menu without saving your document.

Using Mass-11 on the central VAX 8700 permits several options in printing your document. You can create a printable file (an ASCII file) or a PostScript-formatted file. You can send these files to a printer by issuing the appropriate command at the DCL prompt. *Output devices in your building can be installed in Mass-11 for you so that you can print directly to them by using the QP option from the Mass-11 word processing menu.* Call User Services at extension 2-5405 to make the necessary arrangements. You can then select one of the Mass-11 **PRINT** commands in the word processing menu to send your files to your printer.

Use the following procedure to send your document to an ASCII printer:

1. Create your document.

2. Select the UT option on the word processing menu.

3. Select CM option (Connect Mass-11 to ASCII) on the utilities menu.

4. Press Ctrl-R to put you at a level in which you can enter DCL commands. You can enter the following command to send your document to an ASCII printer:

```
MASS11>  SEND PRINT filename.filetype node device-name
```

5. To return to Mass-11, enter:

```
MASS11>  LOGOUT
```

Use the following procedure to send your document to a PostScript-compatible printer:

1. Follow steps 1-3 above.

2. Select the DM option from the word-processing menu, select the PR option from the defaults menu, and change your printer-ID to POST.

3. Follow steps 4 and 5 above. They are different only in the commands you use from the "MASS11>" prompt:

```
MASS11>  LISTPS filename.filetype node::device-name
```

Or (for any other PostScript output device):

```
MASS11>  SEND PRINT filename.filetype node device-name POSTSCRI
```

For more detailed information, see *Mass-11 Word Processing VAX/VMS Mass-11 EDT Editor* (Version 8).

## THE MODULE MANAGEMENT SYSTEM (MMS)

DEC's Module Management System (MMS) is a program that allows you to update systematically any file that depends upon other files that may selectively need recompilation, relinking, etc. Using input that establishes the dependencies of each module, MMS compares revision dates to determine which operations (compile, link) are required upon the modules. The result is complete updating with minimum processing. The function of MMS is similar to the Unix MAKE utility. Online information is available by entering:

```
$  HELP MMS
```

For detailed information on how to use MMS, see *Guide to VAX DEC/Module Management System* (AA-P119C-TE).

## THE PASCAL LANGUAGE COMPILER

To invoke the VMS Pascal compiler, enter:

```
$  PASCAL filename
```

The default filetype for Pascal source code is "PAS." To use the VMS Symbolic Debugger, you must use the "/DEBUG/NOOPTIMIZE" qualifiers when you compile your code. You can then link the generated object module(s) with libraries by using the **LINK** command. You must use the "/DEBUG" qualifier with the linker if you used the Debugger qualifiers when you compiled your code. To view the available compiler options, enter:

```
$  HELP PASCAL
```

## SAS AND SAS/GRAPH

The Statistical Analysis System (SAS) is a software system for data analysis, providing tools for information storage and retrieval, data modification and programming, report writing, statistical analysis and file handling. SAS/Graph software provides an intelligent system for producing information and presentation graphics. Output devices include graphics terminals, printers, plotters and graphics metafiles which you may send to any of the hardcopy graphics devices provided by CTD.

To use the SAS or SAS/Graph software, first initialize your VMS environment by entering:

```
$  SETUP SAS
```

For online help on SAS and SAS/Graph, enter:

```
$  HELP @SAS
```

For further information on SAS/GRAPH, see Chapter 10. See *Recommended Documentation for Computer Users at ANL* (ANL/TM 379) for a listing of SAS and SAS/Graph user guides and reference manuals available to you at the Document Distribution Counter.

## SLADOC

SLADOC is a documentation program that retrieves information about the SLATEC scientific subroutine library. This retrieval is initiated via the **SELECT** command, which specifies a desired module, a list of keywords, or a category. Within SLATEC, a particular module might be the name of a subroutine, such as VINT and a keyword might be EIGENANALYSIS. Figure 5 provides an example of a SLADOC session.

The SLADOC program is easy to use. You can then choose to obtain a complete table of contents by entering **CONTENTS** at the "MODULE:" prompt or you can obtain a listing of SLATEC CATEGORIES by entering **CATEGORIES** at the "MODULE:" prompt. These listings are helpful in determining how SLATEC routines are grouped. Keywords are acceptable as well (see the example in Figure 5), if you know what you are looking for. By default, the system will display information on your terminal screen unless you specify the name of the file as in the example in Figure 5. That example specifies "SOMEFILE.NAME" as the name of the file.

```
To invoke SLADOC, enter
    $ SLADOC

Online help is available for assistance.

  To initiate a retrieval,                        enter     SELECT
  For help on SELECTing,                          enter     ? SELECT
  For a list of available COMMANDS,               enter     ? LIST_COMMANDS
  For more info on getting help,                  enter     ? HELP
  To display the INTRO screen,                    enter     ? INTRO
  For help on any prompt in question,             enter     ?
  To return to the SLADOC "COMMAND = " prompt,    enter     <control/c>
  To exit SLADOC,                                 enter     EXIT

Enter SELECT, COMMENT, EXIT, or ? for help. COMMAND = SELECT
                                            COMMAND   SELECT is accepted.

The following is an example of a SLADOC session:

MODULE     :
KEYWORDS   :  EIGENANALYSIS
KEYWORDS  2:
CATEGORY   :

A SELECTion option is below:

KEYWORDS   = *EIGENANALYSIS*

    Now looking at SLATEC. ...  Be patient. ...
I just read   706 index records and found     1 matching modules.

Current output file= TT:
Output options      =    /NOFormfeed /Header /Lines=ALL
  Enter [ret> for default file and options,
  or enter complete file name (TT: for terminal)
  and-or /options.  Your entry?_ SOMEFILE.NAME
New output file is below: CC999:[B12345]SOMEFILE.NAME;1
Output options      =    /NOFormfeed /Header /Lines=ALL
Now writing the      1 matching modules to
  file-- CC999:[B12345]SOMEFILE.NAME;1
To stop output and return to COMMAND prompt enter control/c.     ...
Enter SELECT, COMMENT, EXIT, or ? for help.  COMMAND = EXIT
                                             COMMAND   EXIT  is accepted.

         Thank you very much.  Good-bye.
```

Figure 5: **Using the SLADOC Documentation Program**

## THE TERMINAL DATA MANAGEMENT SYSTEM (TDMS)

Use the VAX Terminal Data Management System (TDMS) to design terminal display forms for use in an interactive environment. Online information is available by entering:

```
    $  HELP TDMS
```

Refer to the *VAX TDMS Forms Manual* (AA-GS13B-TE) for detailed information on using TDMS.

# CHAPTER 10

# USING GRAPHICS IN VAX/VMS

Several packages for producing high-quality graphics are available on the central VAX 8700 computer. Cuechart, Tellagraf, Disspla (all from Computer Associates), and SAS/Graph (from the SAS Institute, Inc.) offer a wide range of flexibility and functionality.[5] Brief descriptions of these programs and their use on the central VAX 8700 follow. For further information on available documentation, refer to the "Graphics" subsection in Chapter 3 of *Recommended Documentation for Computer Users at ANL* (ANL/TM 379).

## CUECHART

Cuechart is a menu-driven graphics package with pre-designed but customizable standard charts, including bar, line, area, pie, and word charts. You can create tables as well. Note that while Cuechart builds the chart descriptions, Tellagraf (described later) is used to actually create the charts. Optionally, you can add Tellagraf commands to your file to enhance the chart. To invoke Cuechart at the DCL prompt, enter:

```
$ CUECHART
```

You will need to select the type of output device you would like to use to print your chart. To view a list of available output devices, at the "Please enter a CHART-ID or a command . . ." prompt, enter:

```
DEVICE LIST
```

To select an output device, enter:

```
DEVICE device-type
```

Use the DEVICE POP option to create a device-independent and a computer-independent graphics metafile. A device type of POP will create a file named "META.DAT" and store it in your current default directory. You can send "META.DAT" to any graphics output device with the **HARDCOPY** command at the DCL "$" prompt. The system will also create files for you named "TAGPRO.DAT" (if it doesn't already exist) and "CUEPRO.DAT" and store them in your current directory. These files will contain your device selections. You should then enter the Chart-Id that you want to create. Once you have provided Cuechart with all of the necessary data, enter the **DRAW** command. (If you enter a **DRAW** command and "CUEPRO.DAT" doesn't exist, Cuechart will use "TAGPRO.DAT" to find your device specification. If neither file exists, Cuechart will tell you to use the **DEVICE** command.)

[5] Datatrieve (from the Digital Equipment Corporation) is a VAX data management language that also has some graphics capabilities (see "Datatrieve" in Chapter 9).

Cuechart creates two other files: "CUESAV.DAT," which contain the Tellagraf commands you just entered (you can edit this file if you need to enhance it) and "CUECHO.DAT," which contains a log of your Cuechart session.

Cuechart will select layouts, colors, and font styles to accommodate your selected device, but you can modify these defaults. The layout of your chart controls the resolution and orientation of your chart. Colors and fonts are self-explanatory. For a list of available options, enter:

```
LAYOUT HELP
COLORS HELP
FONTS HELP
```

See the *Cuechart User's Guide* (2.0) for detailed information on using Cuechart.

## CODEBOOK

CA-Disspla Codebook is a library of interactive tools that are helpful for generating basic CA-Disspla programs. See *CA-Disspla Codebook's User Guide* for more information on Codebook and its capabilities.

## TELLAGRAF

Tellagraf is a versatile, flexible, and conversational computer graphics system that produces publication-quality charts and graphs. The available Tellagraf options are shaded fonts, the postprocessor, Data Connection (File Connection, Report Connection, External Program Connection, and Decision Support Connection), Tables, and Pinpoint. You may execute these program products by entering the commands:

```
TELLAGRAF (for Tellagraf)
FILCON (for File Connection)
REPCON (for Report Connection)
```

You can reach the Decision Support Connection by entering:

```
Enter:  CALC DATA
```

All the Computer Associates terminal drivers are available to you. To take advantage of Argonne graphics hardcopy devices, choose the device driver POP or CGM and send the resulting metafile ("META.DAT" or CGMBOUT.DAT) with the **HARDCOPY** command over to MVS batch for postprocessing.

See the *Tellagraf User's Guide, Tellagraf Pocket Guide, Tellagraf/Pinpoint User's Manual,* and *Data Connection User's Guide* for further information on using Tellagraf.

## DISSPLA

Disspla is a graphics package of Fortran-callable routines. The available Disspla options are shaded fonts, mapping, page layout (tabletting), postprocessor, business features, contouring, dynamics, and the Graphics Kernel Standard (GKS). You may link your programs to either of the graphics libraries as follows:

```
$  LINK your-program, DISLIB/OPT   (for the Disspla library)
```

or

```
$  LINK your-program, GKSLIB/OPT   (for the GKS library)
```

Again, all the Computer Associates terminal drivers are available to you. In addition, two types of metafile drivers are available to create files which you can postprocess with the **HARDCOPY** command to obtain hardcopy output. See "Creating Graphics Metafiles in VAX/VMS" below for more information. For more informaton on Disspla itself, see the *Disspla Pocket Manual*, the *Disspla User's Manual*, and the *Supplement to CA-Disspla* (ANL/TM 467).

## SAS/GRAPH

SAS/Graph allows you to (1) analyze data and display results graphically on graphics terminals and (2) produce quality graphics on hardcopy devices (e.g., film plotters, pinplotters, electrostatic plotters, and PostScript devices). See *Recommended Documentation for Computer Users at ANL* (ANL/TM 379) for a listing of SAS/Graph user guides and reference manuals available to you at the Document Distribution Counter.

To invoke SAS/Graph at the DCL prompt, enter:

```
$  SETUP SAS
```

You will need to select the type of output device you would like to use to plot your chart. To specify the graphics output device at the SAS "?" prompt, enter:

```
?  GOPTIONS DEVICE=device;
```

Valid output devices are listed in Table A3.1 of Appendix A in the *SAS/GRAPH User's Guide*. If you neglect to specify a graphics output device, the system will prompt you to enter a device name.

With SAS/Graph you can create a POP or CGM metafile for later postprocessing. To specify POP or CGM as your output device and invoke SAS/Graph with the DEVICE=POP or DEVICE=CGM parameter, enter:

```
?  SAS/DEVICE=POP filename
```

## CREATING GRAPHICS METAFILES IN VAX/VMS

Several programs are available on the VAX 8700 for creating graphics metafiles, which are both device-independent and machine-independent. The following paragraphs explain how to create a metafile by using the respective Computer Associates graphics packages and SAS/Graph. Currently, you can create two different types of graphics metafiles with Cuechart, Tellagraf, and Disspla: computer graphics metafiles (CGM) and picture metafiles (POP). The CGM is an ANSI-standard metafile that both Computer Associates software and other vendor software can take as input and produce output. The POP metafile, though driver-independent and machine-independent, is compatible only with Computer Associates software.

In Cuechart, you can establish your output device as a metafile by entering (at the "Enter CHART-ID" prompt):

```
    DEVICE CGMTAG    (for CGM metafile creation)
```

or

```
    DEVICE POP       (for POP metafile creation)
```

In Tellagraf, when the profile prompter prompts you, choose CGMTAG for your DEVICE to create CGM metafiles; choose POP to create POP (or picture) metafiles. Using the CGMTAG driver in either Cuechart or Tellagraf results in the creation of the CGM metafile "CGMBOUT.DAT." Using the POP driver results in the creation of the POP metafile "META.DAT."

With Disspla, you can create the binary CGM metafile by nominating the CGM binary driver:

```
    CALL CGMBO ('',0,0)
```

Although we recommend using the binary metafile format, other CGM metafile formats are available (see the *Disspla User's Guide* for more information). To create a POP (or picture) metafile, choose the COMPRS driver:

```
    CALL COMPRS
```

The name of the POP metafile that Disspla creates is "POPFIL.DAT."

Once you have created the metafile, you can view it (or plot it with a locally attached device) with the postprocessing utility CGMPOP (for CGM metafiles) or ANYPOP (for POP metafiles). See "Postprocessing Metafiles" below for examples of how to use these two utilities.

To receive hardcopy output from a metafile (either the CGM or POP type), enter:

```
    $ HARDCOPY
```

Then answer the prompts.

## POSTPROCESSING METAFILES

Postprocessing a metafile involves taking a metafile, translating it to a system-dependent and device-dependent file, and plotting it to the device upon which the file is dependent. You can postprocess your metafile in a number of ways. ANYPOP is a program that will accept as input a POP file named "META.DAT" and plot it to your terminal (e.g., to such terminals as the VT240, REGIS, Tektronix, and Hewlett-Packard). You can invoke this program with the command:

```
    $ ANYPOP
```

Then follow the prompts. For example, to plot (on a VT240 terminal screen) a metafile that you created with Tellagraf:

1. Enter:

```
    $ ANYPOP
```

2. Select the appropriate device by entering (for REGIS), the number "3":

```
    3
```

3. At the "Enter model" prompt, select the corresponding number for the VT240:

```
3
```

4. Enter your monitor type as either monochrome (0) or color (1):

```
0
```

5. Enter your postprocessor directives ("DRAW=1" is the default).

You can postprocess a Computer Graphics Metafile (CGM) by entering:

```
$ CGMPOP
```

The system will look for a file named "CGMBIN.DAT." Follow the prompts.

To send your metafile to a plotting hardcopy device, use the **HARDCOPY** command. In interactive mode, you will receive full prompting when you enter:

```
$ HARDCOPY
```

If you do not want full prompting, enter:

```
$ HARDCOPY metafilename.filetype graphics-driver [option1] [option2]
```

Use this command syntax in the batch mode as well.

Online help is available for HARDCOPY by entering:

```
$ HELP HARDCOPY
```

For assistance in using ANYPOP and CGMPOP, call the User Services consultant at extension 2-5405.

# APPENDIX A

# COMPUTING AND TELECOMMUNICATIONS DIVISION POLICY FOR USER ACCOUNTS

## USER RESPONSIBILITIES

Argonne's computing services are available solely for the purpose of carrying out Laboratory work (including authorized work of other agencies). Laboratory work consists primarily of assigned technical and management work, but it also includes professional development undertaken with the knowledge and approval of your supervisor. Laboratory computers are not to be used for any other purposes, such as games or personal, social, fraternal, or private business.

Certain sensitive data are stored on the computer. No one may access or attempt to gain access to such data unless specifically authorized to do so. Should you receive misdirected output that is not sensitive, simply return it to the place where you obtained it. However, when you believe that misdirected output contains sensitive information, please forward the output to the Computer Protection Program Manager.

## PASSWORDS

The Computing and Telecommunications Division provides services only to authorized account holders. To assure that users of services are who they claim to be, we have implemented validation methods that require passwords to logon to interactive systems or to submit batch jobs. We require that you not share or divulge your password. Please see the User Services Consultants for alternatives to sharing passwords. If you believe your password has been compromised, you should change it immediately. Additionally, you should inform the Computer Protection Program Manager at extension 2-7151 when (a) you believe our security mechanisms are not working properly, (b) you suspect that someone is deliberately trying to break into the system, or (c) you have access to sensitive information.

## CHARGES

The Computing and Telecommunications Division charges for the use of computer resources to recover the costs of providing services. Charges are incurred for all services, including interactive computing, batch computing, graphics, and disk and tape storage. A list of current rates may be obtained at the Document Distribution Counter, Building 221, Room A-134, or you may order one by calling User Services at extension 2-5405.

All computing charges are charged to your computer account, which consists of a three-digit cost center code assigned to your division cost center and an eight-digit activity code representing a particular activity or project. To use the computing services you must complete a "Computer Use Authorization Request" and establish a userid and password. When you are working on more than one project, you may set up a computer use authorization for each project and charge your work to an appropriate project account. An account alias will identify the account to be charged for batch jobs or interactive sessions. When you begin a session, you must use an account alias to indicate which project account is to be charged. When you intend to use more than one account, please fill out a separate form for each account identifying the services you need.

Depending upon the category of the user's organization, outside users are charged either an add-on percentage for general administrative expense or a combination of general administrative overhead expense, Laboratory computing equipment depreciation, general depreciation, and DOE administrative expense. Refer to the *Computing and Telecommunications Rate Sheet* for current surcharges to non-ANL users.

To receive an itemized list of your computing charges for a given month, enter:

```
$  HELP USERDETAIL
```

To receive an itemized list of your cost centers computing charges for a given month, enter:

```
$  HELP COSTCSUM
```

# INDEX