

1. Introduction

This handout is written to give users a basic understanding of the university's OpenVMS systems. If you find any errors in this guide or have any comments please send them to csainz@umiami.ir.miami.edu. This guide is also available on the web at <http://www.miami.edu/ir/pubs/manuals/>.

After reading this handout, you should be able to perform the following tasks:

- **Log on** (or **sign on**) to the OpenVMS system
- Execute simple commands
- Create, list, edit, and remove files
- Create and execute programs
- Obtain printouts of your files, including printouts of your program's execution
- **Log off** from the system

This typestyle will be used for things that the computer displays.

This typestyle will be used for things that you must type in.

If you have questions about anything that you have read in this document, please ask one of the User Assistants at the front desk. If you are at home, you can call the Help Line at 284-HELP.

2. Logging On To The OpenVMS System

Making Connections from ISF

In the ISF Computer lab you will see this **menu**:

Welcome to the ISF Computer Lab

- <1> IBM/CMS (VM/CMS)
- <2> GABLES OpenVMS
- <3> Jaguar (Unix)
- <4> Student Integrated System
- <5> Library
- <6> EASY
- <7> SCICS
- <8> Local Prompt

Enter Choice:

If you do not see the above menu, but see this on the screen

Local -011- Session 1 disconnected from xxxxx

Enter To Continue>

Press *[RETURN]* and you should see the menu.

If you see anything else on the screen like VM/ESA or SCICS, etc. Press the *[DO]* key and the *[F14]* key.

From the menu, type the number for OpenVMS and press *[RETURN]*. You should now proceed to the section titled **Identifying Yourself**.

Making Connections via Modem

To use the OpenVMS system through a modem, set your communications program as follows:

Eight (8) data bits
No parity
One (1) stop bit

For up to 14400 baud the number to call to connect is:
284-6010.

For 28800 baud only the number to call to connect is:
284-6700

If you are using a low speed modem your responses will be better if you turn off both buffering and data compression on your modem.

Once you have made a connection, press *[ENTER]* and you should see the following prompt:

Local>

At the Local> prompt, type the letter *c* (for connect) and press the *[RETURN]* key:

Local> *c* *[RETURN]*

If the connection was made, the system will inform you of this with something similar to this message:

Local -010- Session 1 to GABLES on node STIMPY established

You are now connected to the OpenVMS Cluster.

Identifying Yourself

Now, the system will display something similar to the following:

University of Miami (IR) VAX 4000-600 VMS Vx.x-xx

Username:

At the Username: prompt, type in your userid and press the *[RETURN]* key.

If you do not know what your userid is, please get the handout entitled **Userid Processor**. When you find out what your userid is, proceed with this handout.

The system will respond with another prompt:

Password:

You are required to supply a password so that the system knows that you are really who you say you are. Since this is your first time logging into the system, you will enter your default password. Your default password is formed by taking the last four (4) digits of your student number and the first two letters of your last name and putting them together. For example, if your name was John Public and your student number was 123-45-6789, then your default password would be 6789PU.

Enter your default password at the Password: prompt and press the *[RETURN]* key. Notice that what you type is not displayed on the screen. This is so nobody can look over your shoulder and see what your password is. If your username is valid, the system will accept the default password and log you on.

Changing Your Password

At this point, the system will ask you to change your default password. This is because your default password is much too easy for anyone else to figure out.

If at any time you want to change your password, use the command **SET PASSWORD** and follow the process illustrated here.

The system will begin the process with this prompt:

Old password:

Type in your default password (the password you just typed to get logged on) and press the *[RETURN]* key. Again, the system will not display what you are typing. The system will then respond with this prompt:

New password:

You should choose a password that will be hard for others to guess. Do not use your first, middle, maiden, or last names. Do not use names in general. They are easy to remember, and

also easy to guess by others. Do not pick a password that somehow describes yourself. Be obscure, but pick some series of letters and numbers that you can remember. A password is not very good if you make it so hard that even you cannot remember it.

Never write down your password. You may accidentally lose whatever you wrote your password on.

Never give out your password to anyone. It is against University policy to give out your password.

Finally, a password must begin with a letter (it does not matter whether it is upper- or lowercase, the system does not differentiate), and it must be at least six characters long. When you have decided on a password, type it in and press the *[RETURN]* key. The system now displays this prompt:

Verification:

Since you cannot see what you are typing, the system is just making sure that you typed what you wanted to type. If you made a mistake, just press the *[RETURN]* key. The system will tell you that your password was not changed and you must begin the process again. Otherwise, just retype your new password and press the *[RETURN]* key.

If everything is correct the system will display the system prompt (also called the **Digital Command Language (DCL) prompt**), a single dollar sign (\$).

If you forget your password, you will need to contact Terry Helmers at 284-3849 to have it reset.

3. File-Related Commands

In order for you to use any computer effectively, you must know how to list, create, and edit files. This section will focus on the commands that are used to accomplish these tasks.

Filenames

Before we get into the commands, you will need to know how the OpenVMS system names files. Filenames on the OpenVMS system follow this general format: *filename.ext*. The *filename* part must begin with a letter and can be a maximum length of 39 characters (which includes the alphanumeric, the underscore, and the dollar sign(\$)). The *ext* (for *extension*) part uses only the first three characters of the extension to determine a file's type. But it can also be up to 39 characters.

At the end of every filename, the system appends a version number. It is separated from the filename by a semicolon (;). The version number is added when changes are made to a file. The latest version of a file has the highest number.

Examples of filenames and some common extensions are listed below:

LOGIN.COM	(COMmand file)
DATA.DAT	(DATa file)
STUFF.TXT	(TeXT file)
PROGRAM.OBJ	(OBJect file)
PROGRAM.EXE	(EXEcutable file)
NUM2STR.LIS	(LISTing file)
JUNE11.PAS	(PAScal program file)
DEG_TO_RAD.C	(C program file)

Listing Files

The **DIRECTORY** command will list all the files that you have.

```
$ DIR
```

If you do not have any files in your directory, the system will respond with an error message:

```
%DIRECT-W-NOFILES, no files found
```

Otherwise, the system might display something like this:

```
Directory USER1:[USERS.991MTH523x.pqrs2z]
LOGIN.COM;7   LOGIN.COM;6   MAIL.MAI;1
Total of 3 files.
```

Creating And Editing Files

Creating and editing a file is accomplished through one command, **EDIT**. To create a new file, type in this command:

```
$ EDIT filename.ext
```

where *filename.ext* is the name of the file you want to create. For our purposes, we will use **NEWFILE.TXT** as the filename for the new file.

You do not need to include the version number when creating or editing a file. The system will use the latest version.

Once you press **[RETURN]**, the system will enter the editor. The editor will then display its prompt and a message telling you that **NEWFILE.TXT** does not already exist:

```
Input file does not exist
[EOB]                      (This marks the end of the file.)
*
```

To begin full-screen editing, you must enter this command at the asterisk prompt:

** c*

This tells the editor that you wish to make changes to your file. Once you press the *[RETURN]* key, the screen will be cleared and all you will see is the end-of-file mark. Now type a line of text like so:

This is line one.

When using the editor, the cursor keys can be used to move around in a document. To leave the editor and create/save the file, do as follows:

[CTRL] [Z] (Hold down *[CTRL]* key then press the letter *[Z]*)

** EXIT*

If you use the **DIR** command now, you should see the filename of the file you just created. To get back into this file, or any other text file, simply use the **EDIT** command and the filename of the file you want to edit:

\$ EDIT NEWFILE.TXT

When you have finished working with a file, you can use the **EXIT** command to save the changes. If, for some reason, you do not wish to save the changes you made, then use this command at the edit command prompt:

** QUIT*

These commands are **NOT** interchangeable. There is an important difference. **EXIT** will save your changes, **QUIT** will not.

4. Executing Some Simple Commands

The system will now allow you to type in commands for it to execute. Here are a few of the most common commands.

Unless specified otherwise, you must press the *[RETURN]* key at the end of a command. For simplicity we will omit this from this point on.

HELP

Use this command to find information on commands. There are two ways to use the **HELP** command. The first is without a command:

\$ HELP

This will enter you into the Help Facility. From here, you get an alphabetical listing of all the various commands and topics that are used on the OpenVMS system.

The second way to use **HELP** is when you know what command you need help with:

\$ HELP SHOW

CONTROL-C

The control c command, [CTRL] and [c], will stop any process that you have started.

DELETE

This command deletes or removes the named file:

\$ DELETE filename.ext;v

You must include the version number of the file you are deleting. If you want to delete all versions of a file, use an asterisk (*) for the version number.

Be Careful with this command. There are ways to get files back if they are accidentally deleted, but it is difficult and time-consuming.

DIRECTORY

This command will list all the files that you have.

\$ *DIRECTORY*

or

\$ *DIR*

If you do not have any files in your directory, the system will respond with an error message:

%DIRECT-W-NOFILES, no files found

Otherwise, the system might display something like this:

```
Directory USER1:[USERS.991MTH523x.pqrs2z]
LOGIN.COM;7  LOGIN.COM;6  MAIL.MAI;1
Total of 3 files.
```

PRINT

To make a printout of your files type the following:

\$ *PRINT filename.ext*

You should get a message from the system telling you that the file is now printing or is waiting to be printed. For picking up print-outs see section 6.

PURGE

This command will delete all but the most recent versions of your files. It is useful when the system complains that your quota has been exceeded.

\$ *PURGE*

If you only want to purge the old versions of a specific file, then include the filename with the command:

\$ *PURGE OLDFILE.PAS*

Be Careful with this command. There are ways to get files back if they are accidentally deleted, but it is difficult and time-consuming.
--

SHOW

This command tells the system to display information. You tell it what to show:

```
$ SHOW USERS
```

This example will give a listing of all the users currently using the OpenVMS system.

```
$ SHOW QUOTA
```

This example will show a statement from the system telling how much storage space, or quota, has been assigned to you and how much is in use.

SET

This command allows you to change various aspects of the system. For example, you are able to set the prompt with this command:

```
$ SET PROMPT="My Prompt "
```

If you do this correctly, the dollar sign prompt will be replaced by whatever you put between the quotes.

TYPE

This command displays a text file on the screen. Never use this command on a non-text file, since this will probably lock up your terminal.

```
$ TYPE/PAGE login.com
```

The /PAGE option displays one screen at a time, allowing the user to read a file that is larger than one screen.

5. Compiling And Executing Programs

More than likely, you will be using the system to create and execute programs. The creation of program files, called **source files**, use the exact same process as shown in Creating And Editing Files, in Section 3. The only difference is that you must use the correct extension. The correct extension depends on what kind of program you are writing. If you are writing a Pascal program, the extension must be .PAS. If it is a C program the extension must be .C.

When you have completed entering in your source file, you will need to use a compiler. A compiler is a program that checks your source file for syntax and creates what is called an object file.

Compiling A Program

To compile a Pascal source file called `program.pas` type the following:

```
$ PASCAL PROGRAM
```

The compiler will look for a file called **PROGRAM.PAS**. This is why it is important to use the correct extension in your filenames. Here is a short list of other compiler commands, a sample filename with the correct extension for each, and what languages they are used for:

FORTTRAN	NUMBERS.FOR	FORTTRAN
MACRO	JOKER.MAR	Assembler
BASIC	ORANGES.BAS	BASIC
CC	SILLY.C	C

If your source file passes inspection by the compiler, then it will create an object file using your filename and **OBJ** as the extension. In this example, the object file would be called **PROGRAM.OBJ**. You can then use the *LINK* command to link the object file and create an executable that can be run:

```
$ LINK PROGRAM
```

The linker will look for a file called **PROGRAM.OBJ**, which the compiler created for you.

DO NOT use the file extension when linking your object file. If you do, this will produce a stream of error messages. Pressing the [CTRL] and [C] keys together several times should stop the process. Also, be careful not to link your source file! Nothing will happen to your file, but a string of errors will occur. Follow the same steps to stop the process as used above.

When the linker is finished, it will create an executable file which you can run. The filename will be your filename with an **EXE** extension. In this example, the file created would be called **PROGRAM.EXE**. To run your program, use the *RUN* command:

```
$ RUN PROGRAM
```

The /LIST Qualifier

When you compile your program, you may wish the system to produce a **listing file**, or a file which contains your source file and detailed information about its compilation (or the errors produced by compilation). Use the */LIST* qualifier with any of the compiler commands:

```
$ PASCAL/LIST PROGRAM
```

or

```
$ FORTRAN/LIST NUMBERS
```

This will create a file with your filename and an extension of **LIS**. In this example, the file would be called **PROGRAM.LIS**. You can then use the *TYPE* command to examine the file:

```
$ TYPE PROGRAM.LIS
```

or

```
$ TYPE NUMBERS.LIS
```

Linking C Object Files

The C linker requires that you provide a library file of the functions that it needs to create an executable file. First create a file called **CLIB.OPT** as follows:

```
$ EDIT CLIB.OPT
```

```
Input file does not exist
```

```
[EOB]
```

```
* c
```

```
{the screen is cleared}
```

```
sys$library:vaxcrtl.exe/share
```

```
[CTRL] [Z]
```

```
* EXIT
```

Now, whenever you want to link your C object file (for this example we have a file called **HELLO.OBJ**), you will type in this command:

```
$ LINK HELLO, CLIB/OPT
```

Note that the command is **CLIB** (slash) **OPT** and not **CLIB** (dot) **OPT** as is in the name of the file

C Command-Line Arguments

The C language provides a method that can be used to allow a user to pass from the DCL prompt arguments that can be used in their program. For example, if a program was written that counted the number of characters in a string, a function that asks the user to type in the string might be included:

```
$ RUN COUNTSTRING
```

```
Please enter the string to be counted here: CHARCOUNT
```

```
The number of characters is 8.
```

By using the command-line argument feature, the user can enter the string at the same time the program is run:

```
$ COUNTSTRING CHARCOUNT
```

The number of characters is 8.

There is no **RUN** command preceding the program name. The **RUN** command does not allow you to pass arguments to your program. Therefore you create a **symbol** (something which represents some other value) to allow you to pass arguments to the program.

Setting up

In order to use the command-line arguments with a C program, the program's main function must be declared as follows:

```
main (argc,argv)
int   argc;
char  *argv[]
{{function implementation goes here}}
```

The parameter **argc** is the argument count (starting count at 1), and **argv** is the character array containing the argument (starting at array element 0). Next a DCL prompt that will represent the directory path of the executable file needs to be assigned. If the program's executable file is called **HELLO.EXE**, then the symbol assignment would be as follows:

```
$ HELLO == "$ USER1:[USERS.903MTH517D.USERID]HELLO.EXE"
```

In making this symbol assignment, you do not include the version number. The vax will assume that you mean to use the most recent version of the file.

The directory path that will be used will probably be different than the one shown in the example. The system will show the directory path with this command:

```
$ DIRECTORY filename.EXE
```

where *filename.EXE* is the name of the executable file you need to make a symbol for. The directory path will be listed after the word **DIRECTORY**. If you wanted to find the directory path of **HELLO.EXE** then type the following command:

```
$ DIRECTORY HELLO.EXE
```

The system will respond with something similar to this:

```
Directory USER1:[USERS.903MTH517D.USERID]
HELLO.EXE;1
Total of 1 file.
```

The string of characters after the word **DIRECTORY** is the directory path that is used when making a symbol. When you are finished with the set-up, arguments can be passed from the DCL prompt to the program. Continuing with the example, the command is as follows:

\$HELLO arguments-1 arguments-2 ... arguments-n

where **arguments-1** ... are the list of arguments you want to pass to the program.

6. Printing Out Your Files

General Printing

To make a printout of your files, type:

```
$PRINT filename.ext
```

You should get a message from the system telling you that the file is now printing or is waiting to be printed.

Only try to print text files, all other files will produce unexpected results.

The User Assistants at the front desk of the ISF will pick up the system printouts every fifteen minutes. You will need to go to them to collect your printout. If you decide to pick up your printout later, it will be filed at the front desk alphabetically by your username for 48 hours.

Printing Your Program's Execution

When you have completed your program to the point that you have created an executable file, you may need to make a printout of your program while it runs. In order to do this, you will need to create a **batch job**. A batch job is a command file that contains a series of commands to be executed with the results sent to either another file or to the printer.

For this example, let's say you have a Pascal program that asks for a last name and then outputs the name backwards. The program's filename is **BACKWARDS.PAS**. You have already compiled it and linked it and it works flawlessly. Now you want to create a batch job so that you will get a printout of the listing file, a printout of the user's input, and a printout of your program's execution.

First, you must create a file containing the user's input. You will call this file **DATA.DAT**. The data in this file must be entered exactly as if a user were entering it into the program. If your program asks for both first and last names in that order, then your data file would have to contain first and last names **in the order the program expects to receive them**. If you do not put the data into the file correctly, your program will not execute properly, or, may not execute at all.

The program is going to ask for three names, each name followed by the *[RETURN]* key. The process for creating the **DATA.DAT** file is as follows:

```
$EDIT DATA.DAT
```

```
Input file does not exit  
[EOB]  
* c
```

{the screen is cleared}

CARLOS GROSS
JULIE CLAUS
CHRISTOPHER BROCCOLI
[CTRL] [Z] (Hold down [CTRL] then press [Z])
* **EXIT**

Now that the data file has been created, you can now make the command file that will be the heart of the batch job.

A command file is nothing more than a file that uses DCL as its language instead of Pascal or C (DCL is the language that you have been using at the dollar sign prompt). You cannot compile or link a command file.

In this file, you must include all the commands you wish to be executed. In order to tell the system that these are DCL commands and not just pieces of data, you must precede each line with a dollar sign. For your example, the file should be entered as follows:

\$ EDIT RUN2.COM

Input file does not exist
[EOB]
* **c**

{the screen is cleared}

\$ PASCAL/LIST BACKWARDS
\$ TYPE BACKWARDS.PAS
\$ LINK BACKWARDS
\$ TYPE DATA.DAT
\$ DEFINE/USER_MODE SYS\$INPUT DATA.DAT
\$ RUN BACKWARDS

[CTRL] [Z] (Hold down [CTRL] key then press [Z])

* **EXIT**

If some of the commands used here are unfamiliar, use the Help Facility to get more information.

To execute a batch job, it must be submitted to a special **queue** for processing. A queue is a place in the OpenVMS system where jobs are kept until the processor is ready to accept them. To submit the batch job, use this command:

\$ SUBMIT/NOTIFY/NOPRINT/QUEUE=FAST RUN2

The **/NOTIFY** qualifier tells the system to inform you when the batch job has been completed. The **/QUEUE** qualifier tells the system to submit the batch job into the queue called **FAST**. The **/NOPRINT** qualifier tells the system to save the results into a file rather than send it to the printer. This is done so that if some error is generated, you will not waste page after page of paper for a printout that you are not going to use anyway.

If there are no problems, the system will issue a message telling you that the job has completed execution. A file called **RUN2.LOG** will be created. This file will contain the results of the batch job. You can examine it using the *TYPE* command if you wish. To print it out, just type:

```
$ PRINT RUN2.LOG
```

The file will be sent to the printer. If the system does have a problem with the batch job, it will tell you that the job was terminated with an error. The file **RUN2.LOG** will still be created, and you can use the *TYPE* command in order to discover what the error is since it will be included in the file. You will need to correct whatever the error is and re-submit the job.

7. Accessing E-Mail

Electronic Mail allows users to send messages to other users within the university community and around the world. When you log onto the OpenVMS system, you may hear your terminal beep and have something like this message displayed:

```
You have 3 new Mail messages.
```

This means that you have three mail messages that you have not read.

About Hostnames And Usernames

To be able to send mail to a person at another site, you will have to know that person's username and hostname. The username identifies the person within the system the same way your username identifies you in the OpenVMS system. The hostname identifies which system you are in. For example, the host name designated for the OpenVMS system is "umiami.ir.miami.edu." The internet address combines the username and hostname to identify both the user and the system. The standard format for the internet address is:

```
username@host.dept.company.domain
```

In the OpenVMS system the internet address must be enclosed in the string `IN%''`.

```
MAIL> SEND  
To: IN%"username@host.dept.company.domain"
```

When sending mail to someone on the same system as you, just type the userid.

About Mail Folders

Learning to use folders in mail can be helpful. When you receive mail it is put into a folder called **NEWMAIL**, after it has been read the computer moves it automatically to a folder called **MAIL**. If you delete a message it is moved to a folder named **WASTEBASKET**. If you want to retrieve a message that you have deleted you can get it from the **WASTEBASKET** folder. When you exit mail the **WASTEBASKET** folder is automatically deleted. You can also create folders to

store messages and move messages from folder to folder with the move command, (see Basic Mail Commands). To see a list of your mail folders, type the following:

```
MAIL>DIR/FOLDER
```

If after seeing the list of folders you want to select a specific folder, type the following:

```
MAIL>SEL foldername
```

This will allow you to read the messages in that folder as you would with new mail. If at any time you would like to delete a folder just delete all of the messages in the folder and the folder will be deleted automatically.

Getting Into Mail

To enter the Mail Facility, just type in this command at the dollar-sign (DCL) prompt:

```
$MAIL
```

and press the [RETURN] key. The system will respond with this prompt:

```
MAIL>
```

You are now inside the Mail Facility. If you have any unread Mail messages, the system will also display the number of these unread items.

Basic Mail Commands

HELP

This command gives information on Mail commands. If you know the command that you need help with just type the command following the HELP.

```
MAIL>HELP EXTRACT or MAIL>HELP
```

COMPRESS

This command makes your mail file smaller. To use this command type the following:

```
MAIL>COMPRESS
```

If no errors were returned, exit mail and delete MAIL.OLD.

```
$DEL MAIL.OLD;*
```

If you do not **delete** the old file you will actually have less space than before the compression, since you would have the compressed file MAIL.MAI and the old uncompressed file MAIL.OLD.

DELETE

This command is used to delete messages:

```
MAIL> DELETE message-number
```

If you want to delete all of your messages, type in this command:

```
MAIL> DELETE/ALL
```

DIRECTORY and SELECT

This command will list all the messages stored in a Mail folder. You can see a listing of all the folders you have with this command:

```
MAIL> DIRECTORY/FOLDER
```

In order to see all the messages stored in a folder, you will first need to SELECT the folder:

```
MAIL> SELECT foldername
```

After you have selected a folder, you can then use the DIRECTORY command to list all of the messages:

```
MAIL> DIRECTORY
```

The current mail folder is displayed in the upper right-hand corner of the screen whenever you are reading a message or when you use the **DIRECTORY** command.

EXTRACT

This command is used to extract a file that a user has sent to you. To extract a file enter the following command:

```
MAIL> EXTRACT/NOHEADER filename.ext
```

The /NOHEADER qualifier tells mail to remove the address header from the file.

MOVE

This command is used to move messages from one folder to another or to create a new mail folder:

```
MAIL> MOVE foldername
```

To use this command you must be reading the message you want to move.

READ

To begin reading your mail, type in this command:

```
MAIL> READ messagenumber
```

After reading your first message, you can just use the *[RETURN]* key to begin reading other messages. If someone sends you mail while you are inside the Mail Facility, use this command to read the new mail:

```
MAIL> READ/NEW
```

REPLY

The *reply* command will allow you to send a response to the sender of a message. To do so type this command:

```
MAIL> REPLY
```

SEND

The *send* command is used to send files or a mail message. To send a file use the following command:

```
MAIL> SEND filename.txt
```

To send a mail message enter the following command:

```
MAIL> SEND
```

The difference between sending a file and a mail message is that when a file is being sent the computer will ask for an address and a subject but will not allow a message to be entered. If you decide that you do not want to send the message or file press *[CTRL-C]* (the control and c keys at the same time).

You can also use the following commands:

```
MAIL> SEND/ED OR SEND/CC
```

The */ED* calls the editor so you can type your message, (see editing files). The */CC* option allows you to send a carbon copy of the message to another user.

Customizing Mail

Here are a few options that a user can set to suit their needs. If you want a personal message added to the top of all of your messages, type in the following at the MAIL> prompt:

```
MAIL> SET PERSONAL_NAME "your message here"
```

To remove this message, type the following:

```
MAIL> SET NOPERSONAL_NAME
```

If you want Mail to prompt you for the usernames and addresses that you want to send a Carbon Copy of your message or file to, type in this command:

```
MAIL>SET CC_PROMPT
```

To prevent Mail from prompting you for this information, type the following:

```
MAIL>SET NOCC_PROMPT
```

To have Mail automatically make a copy of any mail that you send or reply to and save it to your mail file, type in this command:

```
MAIL>SET COPY_SELF SEND,REPLY
```

To disable this feature, type in the following:

```
MAIL>SET COPY_SELF NOSEND,NOREPLY
```

If you see an option and are unsure about it or want to know some more, type in *HELP SET* at the MAIL> prompt.

Leaving Mail

When you are finished using the Mail Facility, type in this command:

```
MAIL>EXIT
```

The system will then leave the Mail Facility and present you with the DCL prompt.

8. NEWS

News is an electronic bulletin board utility that spans thousands of systems. It offers users a means to read information on topics of interest to the user, as well as to post information or questions about specific topics.

News is invoked by typing **NEWS** at the DCL prompt.

```
$ NEWS
```

On first entering news, you are presented with a screen showing the topics (or newsgroups) which are currently available.

To read the titles of the news item which have been sent to news for a given newsgroup, position the cursor next to the newsgroup and press *[RETURN]*. The screen will then display a directory of all current items held in the selected newsgroup. To read the text of an item which has been sent to news, position the cursor next to the item and type *[RETURN]*. Once the item has been displayed, typing another *[RETURN]* will display in a sequential fashion each of the notes received under newsgroup or typing the command *DIR/ALL* will again display the list of all newsgroups. Exiting completely from news is accomplished by typing *[CTRL]* or by typing the command *EXIT*.

In case you need more help, NEWS has a *HELP* command.

9. Logging Off

When you have completed all your work on the system, there is one final command you will need to enter:

\$ LOGOFF

The system will log you off and the Local> prompt will return. At the Local> prompt type:

local> *lo*

to log off the university's computer system.

For further assistance see a User Assistant at the front desk or call 284-HELP.