VAX BOOK

VAX BOOK

THE VAX BOOK

Joseph E St Sauver

Release 1

February 1990

PREFACE:

This document has numerous system dependent features; to use it at another site, you should obtain a machine readable copy of the Digital Standard Runoff source and modify it appropriately for your system. Obtain a copy via anonymous FTP from DECOY.UOREGON.EDU (128.223.32.19) or contact:


```
Joseph E St Sauver
Assistant Professor/Statistical Programmer
Office of University Computing
235 Computing Center
University of Oregon
Eugene, OR 97403
(503) 346-4394 ext 25
BITNET:          JOE@OREGON
INTERNET:        JOE@OREGON.UOREGON.EDU
```


The supercomputing portion of this guide was developed via a U of Or Affiliate Consulting Grant from the NCSA at the U of Il-Urbana.

To my friends, consultees, bosses, coworkers, teachers and students: thank you. This guide wouldn't have been possible without your patience, encouragement and help.

To my wife Beverly, who uncomplainingly sacrificed many of our evenings and weekends together (and much of her sanity) to my preoccupation with this guide: all my love, always. Thanks for putting up with me so far!

CONTENTS:

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

CONTENTS (continued):

INTRODUCTION

WHAT IS "THE VAX"?

The VAX is the University of Oregon's central academic computer system, a Digital Equipment Corporation (DEC) VAX 8800 running version 5.2 of the VMS operating system. It replaced the DEC 1091 and IBM 4341 that had previously supported centralized research and instructional computing requirements, and is complemented by a DEC VAX 6430 for administrative applications (upgraded from a 6230).

While the VAX is one of the largest members of Digital's family of VAX/VMS computers, the 8800 is usually considered to be a "super minicomputer" or "midrange" machine. Although comparisons of different computers is a difficult task, one common basis for comparison of machines used for scientific tasks is system performance on the LINPACK linear equation benchmarks, done in terms of millions of floating point operations per second (MFLOPS). [For more information, see, generally, MIPS magazine, or J.J. Dongarra, Supercomputing Review, January 1990, pp. 73-79]

MFLOPS FOR LINPACK BENCHMARK (n=100):

System	Speed
IBM PC/XT w. 8087 (MS 3.2)	.007 MFLOPS
IBM PC/AT w. 80287 (MS 3.2)	.009 MFLOPS
Apple Macintosh SE	.01 MFLOPS
Apple Macintosh II	.06 MFLOPS
Apple Macintosh SE/30 (16 MHz)	.09 MFLOPS
NeXT (0.7 beta MACH GCC)	.20 MFLOPS
80386 (20 MHz) w. 80387	.25 MFLOPS
80386 (25 MHz) w. 80387	.33 MFLOPS
80386 (33 MHz) w. 80387	.53 MFLOPS
DEC VAX 8800 (VMS 4.5)	.99 MFLOPS
Sun SPARCstation 1	1.3 MFLOPS
SGI Personal Iris/SGI 4D20	1.5 MFLOPS
DECstation 2100	1.6 MFLOPS
SUN SPARCStation 1 with Weitek	2.3 MFLOPS
DECSystem 3100 with R2010 coproc	3.2 MFLOPS
CRAY Y-MP/832 (8 proc./6 nsec.)	200 MFLOPS

WHAT IS VMS?

VMS 5.2 is the operating system that runs on the OREGON VAX, acting as the fundamental interface between you, the user, and the actual hardware of the VAX.

VMS controls access to the system, manages files, prevents users from interfering with each other, and provides the environment applications software needs to run.

While VMS is more complicated that many microcomputer operating systems (such as MS-DOS), it is in general far **less** complicated than many traditional mainframe operating systems.

I <u>USED</u> TO USE A <BLANK> AT <BLANK>...

Due to the comparatively transitory nature of students and faculty, new users frequently arrive from other schools. In some (happy) cases, the new user's former school ran a VAX/VMS system just like ours. In many other cases, though, what was used "there" is often quite different from what we have "here," and getting settled in can take a bit of adjustment.

Sometimes even longtime "VAX" users find out they are in for some changes.

In particular, it can be a shock to find out that the "VAX" you'd used for several years back at XYZ University ran ULTRIX, a type of UNIX, which is not **at all** the same as the VMS operating system we run on the OREGON "VAX."

While the VMS operating system will allow you to define aliases for VMS commands (so that you **can** make $ DIR into $ ls if you **really** want to), we'd encourage you to give VMS a try before you go to all the trouble of trying to mimic UNIX. You may be pleasantly surprised by VMS!

HOW MUCH DO I NEED TO KNOW?

While every VAX user needs to know at least a few VMS commands, you don't need to learn every command in the this book in order to be able to use the VAX!

You **should** look browse through this book to get a feel for the sort of things the VAX can do for you, but don't try to learn everything all at once. Learn what you need, as you go along.

MORE INFORMATION ABOUT THE VAX

This document covers most of the VAX commands you'll ever need. However, for more detailed information on a VAX command, you can type:

```
$ HELP
```

An online VMS tutorial is also available:

```
$ VMS_TUTOR
```

Finally, a complete set of VAX (and other) manuals is available for inspection in the Computing Center Documents Room, 205 Computing Center.

SHORT NON-CREDIT CLASSES:

The Computing Center periodically offers short non-credit classes designed to help faculty, staff and individual students learn to use the VAX. A listing of the classes offered is available from the upstairs receptionist at the Computing Center.

Note that the rooms used to teach these short classes are generally too small for an entire class or section of students. However, if you're an instructor and would like to have someone come in to help your class learn to use the VAX, contact Joe St Sauver at x6-4394 or send mail to VAX username JOE

GETTING STARTED

ACCOUNTS ON THE VAX:

Before you can do any work on the VAX, you'll need to open an account. The type of account you should open depends upon your circumstances:

"REGULAR" ACCOUNTS: Faculty, staff and graduate teaching fellows (GTFs) should contact their departmental office to make arrangements for VAX access to support their instructional, research, and administrative computing requirements.

"ST" ACCOUNTS: Students who need to use the VAX for a course will be given a temporary "ST" username by their instructor. Instructors should contact their department for authorization to open student accounts, and then contact the Accounting Clerk in Room 107 of the Computing Center to obtain the necessary forms.

"COMMUNICATIONS" ACCOUNTS: Faculty, staff and GTFs who simply need access to the VAX for low volume electronic communications purposes may only need a free VAX "communications" account. These accounts allow access to all VAX facilities and software, but have a $50.00/term charge limit and a 20 CPU second/logon session limit.

"CASH" ACCOUNTS: Finally, note that University-affiliated individuals do have the option of opening a cash-basis account by making a minimum $50.00 deposit with the Accounting Clerk. Your computing charges will then be deducted from that deposit.

However, unaffiliated individuals are NOT currently allowed access to the U of O VAXes (even on a cash basis) due to various software licensing restrictions.

If you have any questions about arranging for an account, contact Connie French at x6-4404, or stop by 107 Computing Center.

LOCATING A TERMINAL OR WORKSTATION:

PERSONAL WORKSTATIONS: Many VAX users use a personal computer or terminal to work with the VAX from their home or office.

Used terminals can often be found for $100 or less, with new VT100-compatible terminals running between $300 and $400. A basic offbrand PC clone microcomputer can usually be purchased for under $600.

Also, faculty, staff, and students enjoy substantial discounts on purchase of new IBM, Zenith and Apple personal computers through the University's Microcomputer Purchase Plan. Macs, for example, start at $978, PS/2's at $862.11 (including shipping to the CC) and Zenith portables at $1199 (plus shipping insurance and handling). Call x6-4402, or visit Room 202 for details on available models and prices.

DEPARTMENTAL WORKSTATION LABS: Some units provide microcomputers and/or terminals connected to the VAX for use by their faculty and students. For instance, the School of Management provides VAX access from a number of terminals in the Chiles Computer Lab. Contact individual departments for details on available resources.

PUBLIC WORKSTATION LABS: Public VAX terminals are available in Computing Center Room 105, and you can also use the microcomputers in the PC Lab (Room 165) to connect to the VAX when the Lab isn't reserved for classes. Free terminal access to the VAX is also possible from four terminals in the EMU Computer Lounge.

Note that in purchasing a terminal you'll want a VT100, VT200, or VT300 series ("ANSI-compatible") terminal or terminal emulator. While you **can** use a TVI 912 or other non-VT100 terminal with the VAX, you won't be able to use any fullscreen software such as EDT.

CONNECTIONS:

Once you have access to a terminal or microcomputer, the next thing you need to get started is a connection to the VAX.

MODEMS: You can dial into the VAX via a 300, 1200 or 2400 baud modem. While this method has the lowest direct user cost, it also has the lowest transmission speed, and sometimes all lines to the VAX may be busy. Routine VAX users will probably want to have a hardwired connection instead.

HARD WIRED CONNECTIONS: In the past, hardwired connections were provided by means of twin-pair "telephone wire" connections to the University's "Gandalf" port contender. Now, however, virtually all new hard wired connections are made via UONet, a campus-wide broadband fiber optic/coax ethernet. UOnet connections come in three forms:

TERMINAL SERVER: The simplest of these three forms simply replaces a Gandalf connection with a UONet terminal server connection. Your terminal or micro remains connected by a comparatively slow (19,200 baud or less) serial RS-232 port using twin pair cable run to a UONet terminal server.

LOCALTALK VIA A KINETICS FASTPATH: VAX users can also be handled via a departmental Localtalk network connected to UONet via a Kinetics Fastpath box. Note that while Localtalk networks are traditionally associated with Macintoshes, PC users can also be accommodated with a PC Localtalk card and Stanford University's TCP-IP or a comparable product.

DIRECT CONNECTION TO UONet: A third alternative is to obtain a direct connection to UONet using an actual ethernet card (such as a 3Com card for a PC).

REMOTE NETWORK CONNECTION: Offsite users can also connect to the VAX via the Internet.

For more information on getting connected, contact Dale Smith or Dave Meyer at x6-4394.

LOGGING ON

LOGGING ON VIA A UONET TERMINAL SERVER:

```
<return> <return>            <-- Wake up UONet
Computing Center Terminal Server CS200-2
UONet>C OREGON                <-- Choose the VAX
<return> <return>      <-- Now wake up the VAX
Username: RJOHNSON     <-- Enter your username
Password: MAGICISLAND  <-- Enter your password
```

LOGGING ON (HAYES COMPATIBLE MODEM):

```
<start your micro's communication program>
ATDT 346-3565          <-- DIAL IN (2400 BAUD)
CONNECT 2400
<break> <return>        <-- Wake up the switch
enter class
OREGON                        <-- Choose the VAX
<return> <return>      <-- Now wake up the VAX
Username: PROGERS      <-- Enter your username
Password: COULDAHADAV8 <-- Enter your password

Your communications program  should be set to:
8 data bits, 1 stop, no parity, Xon/XOff  flow
control;  preferred terminal emulation: VT100.

2400 baud only:                 (503) 346-3565
1200 or 300 baud only:          (503) 346-4391
```

LOGGING ON VIA A GANDALF CONNECTION:

```
<break> <return>        <-- Wake up the switch
enter class
OREGON                        <-- Choose the VAX
<return> <return>      <-- Now wake up the VAX
Username: JSMITH       <-- Enter your username
Password: SOUPISFOOD   <-- Enter your password
```

LOGGING ON VIA TELNET:

```
<Start TELNET on the remote system you're on>
*OPEN OREGON.UOREGON.EDU    <-- (128.223.20.2)
*USER TPETERS           <-- Send your username
*Password: TEETAH       <-- Send your password
```

LOGGING ON VIA DECNET:

```
$ SET HOST OREGON           <-- (OREGON=60.601)
Username: TBOONE
Password: PICKEMUP
```

LOGGING ON FROM THE PC LAB (165 CC):

1) Look for the Novell F:\LOGIN> prompt. If you don't see it, check to make sure the screen brightness is turned up, and the machine is turned on (the PC's power switch is on the right side of the case, near the back). If you still don't see F:\LOGIN> SIMULTANEOUSLY hold down <CTRL> and <ALT> and then hit <DEL> on the key pad. If the PC doesn't reboot and show the F:\LOGIN> prompt after a minute or so, turn it off, wait for about 45 seconds and then turn it back on.

2) Logon to the Novell network as user TELNET:

```
F:\LOGIN> LOGIN TELNET
```

3) Check to make sure the PC's floppy disk is in the disk drive. If it isn't, take it from its envelope on the PC and put it in the drive. You must use **YOUR** machine's disk!

4) Now use the cursor (arrow) keys to highlight TERMINAL EMULATOR. Hit return to select that choice. Move the cursor to TELNET. Hit return to select that choice.

5) Now specify that you want the VAX:

```
OREGON
```

Proceed to logon.

WHEN YOU'RE DONE AND HAVE LOGGED OFF THE VAX: After logging off the VAX, please remember to also get out of the Novell network. To get out of the Novell network, hit an <ESC> to return to the main Novell menu, and then use your arrow keys to highlight LOGOUT. Hit return. Turn the monitor brightness down.

LOGGING ON VIA THE "ONLINE" NETWORK

If you need to get to the Oregon VAX from other parts of the state, you can try:

```
                             "OLD" #  "NEW" #
Eastern Oregon               963-8000 962-8000
Oregon Health Sciences       225-7665 494-7665
Oregon Inst. Technology      883-7007 885-7007
Portland State               229-3101 725-3101
Southern Oregon State        482-5821 552-5821

<return> <return>   <-- wake up ONLINE network
HOST SITE: UO   <-- UO may not echo when typed
CONNECTED
<return> <return>   <-- wake up U of O Gandalf
enter class
OREGON  <-- now logon as you normally would...
```

LOGGING IN NATIONWIDE:

Many faculty and administrative staff routinely travel outside the state, yet need to continue to be able to connect to the VAX. While these individuals can place a toll call back to our modems, a local number would be far more convenient and economical.

We are, therefore, interested in information concerning "guest" TELNET network access (to at least the Oregon VAX) via local numbers for the following metropolitan areas:

Albany, Albuquerque, Anchorage, Atlanta, Baltimore, Boston, Buffalo, Chicago, Cincinnati, Cleveland, Columbus, Dallas, Denver, Detroit, Fairbanks, Hartford, Honolulu, Houston, Indianapolis, Juneau, Kansas City, Las Vegas, Los Angeles, Memphis, Miami, Milwaukee, Minneapolis, New Orleans, New York City, Norfolk, Oklahoma City, Pheonix, Philadelphia, Pittsburgh, Providence, Reno, Rochester, Sacramento, St Louis, St Paul, Salt Lake Cty, San Antonio, San Diego, San Francisco, Santa Fe, Seattle, Spokane, Syracuse, and Washington, DC.

If you can help out, please contact JOE@OREGON or JOE@OREGON.UOREGON.EDU Thanks!

WHY DOES IT TAKE SO LONG TO LOGON?

Users often ask, "Why does it take so long after I enter my username and password before I get the dollar sign prompt?"

Well, it takes time for the system to execute its system-wide initialization routines (as listed in SYS$MANAGER:SYLOGIN.COM), and your own LOGIN.COM also slows things down.

For example, when the system-wide initialization file was timed recently, it was found to use about 4.35 CPU seconds, broken out as follows:

```
SYSTEM-WIDE INITIALIZATION TASK      CPU     %
------------------------------------------------
User Authorized for This Machine?    0.21  4.8
User Budget Allocation Okay?         2.31 53.1
Define Cluster-Wide Commands         0.14  3.2
Define OREGON Specific Commands      0.10  2.3
Run ALIASES to define Mail Nicknames 0.24  5.5
Define EQS Commands                  0.09  2.1
Define MINITAB Commands              0.10  2.3
Define NEWS Commands                 0.07  1.6
Do TCP/IP Setup                      0.14  3.2
Do JNET Setup                        0.25  5.8
Run NOTICE Routine                   0.22  5.1
Run Message of the Day Routine       0.21  4.8
Cleanup                              0.27  6.2
                                     ---- ----
                       SYSTEM TASKS: 4.35 100%
```

A typical LOGIN.COM uses another 1.94 seconds:

```
USER-CONTROLABLE INITIALIZATION      CPU     %
------------------------------------------------
LOGIN.COM (as originally provided)   0.69 35.5
Add SCRATCH to get SYS$SCRATCH space 0.37 19.1
Add @SYS$SYSTEM:UNSUPPORTED_SETUP    0.88 45.4
                                     ---- ----
                         USER TASKS: 1.94 100%
                   ============================
                   TOTAL TIME: 6.29 CPU SECONDS
```

Of course, those 6.29 CPU seconds (or so) will require far more "wall clock" time!

MISCELLANEOUS LOGIN-TIME COMMANDS

TO REPEAT THE "MESSAGE OF THE DAY":

```
$ MOTD
```

MAKING NOTICE QUIT NAGGING YOU:

```
$ NOTICE
<hit returns until you see "No more notices">
```

DECLARING YOUR TERMINAL TYPE:

```
Terminal type [unknown]: VT100
```

If you enter an incorrect terminal type, or miss entering any terminal type whatsoever due to hitting too many returns, use:

```
$ SET TERM/VT100
```

CHANGING YOUR PASSWORD:

```
$ SET PASS
Old password: HIGHSTICK
New password: SLAPSHOT       <-- New pass must
Verification: SLAPSHOT      be >= 6 chars long
```

LOOKING AT YOUR COMPUTING BUDGET:

NOTE: You will **NOT** be able to login if you exceed your computing budget!

```
$ BUX
```

LOGGING OFF THE VAX:

```
$ EOJ                  <-- to log off quickly

$ LO    <-- to log off and see resource usage
```

SETTING UP A LOGIN.COM FILE:

Whenever you login or run a batch job, two command procedures automatically run.

The first is a system wide file called SYS$MANAGER:SYLOGIN.COM. That file defines software symbols and logical names, checks to make sure your account has money, reports on any files in your reader, etc.

After SYS$SYLOGIN finishes, VMS then looks for a file called LOGIN.COM in your own directory.

Each VAX account automatically gets a generic LOGIN.COM (from SYS$SYSTEM:SAMPLE_LOGIN.COM) when the account is initially created. However, as you become increasingly experienced in using the VAX you can modify your LOGIN.COM file to customize your account. For example:

```
$ ED LOGIN.COM
*C
$ IF F$MODE .NES. "INTERACTIVE" THEN GOTO MORE
$
$ ! INTERACTIVE-ONLY STUFF GOES HERE
$ SET TERM/VT100
$ MAIL :== MAIL/EDIT
$ @DISK$USER1:[JOE]NEWS_EDIT.COM
$ DEF/NOLOG/JOB TELNET_ESCAPE "|"
$ DEF/NOLOG/JOB SWING$EDIT EDT
$ DEF/NOLOG/JOB TOM "IN%""FFTAB@ALASKA"""
$ SET TERM/APPLICATION_KEYPAD
$ DEF/KEY/TERM KP1 "SH QUE/ALL SYS$BATCH"
$ DEF/KEY/NOTERM KP2 "SUBM/NOPRIN/KEEP/NOTI "
$
$ MORE:
$ ! STUFF EXECUTED FOR ALL JOBS BELONGS HERE
$ PRINT :== "PRINT/NOTE=""LOCK BOX 100"""
$ DEF MY$DISK 'F$PARSE("SYSLOGIN",,,"DEVICE")'
$ SCRATCH
$ @SYS$SYSTEM:UNSUPPORTED_SETUP.COM
$
$ EXIT
<ctrl>-Z
*EX
```

Note that if you make changes to your LOGIN.COM file, you'll need to say: $ @LOGIN.COM before the changes'll work.

USING VMS HELP:

```
$ HELP                          <-- get into help

$ HELP COPY   <-- get help on the COPY command

$ HELP TYPE *    <-- see TYPE & subtopics, too

$ HELP COPY...   <-- see EVERYTHING under COPY

$ HELP/NOI          <-- suppress the obnoxious
                             intro help screen

$ HELP/OUT=TEMP.LIS RENAME      <-- put RENAME
                  help info into file TEMP.LIS

$ HELP/PRINT RENAME <-- print RENAME help info

$ HELP/NOPAGE        <-- eliminate page breaks
```

NOTE: Some HELP topics are effectively "hidden." You may need to be persistent and/or imaginative to find the information you want. For example:

TOPIC:	HELP COMMAND:
FORTRAN DO STATEMENT:	$ HELP FOR STAT DO
FTP:	$ HELP NET TCP FTP

Other VAX products (such as MAIL and EDT) support their own HELP systems separate from the main VAX HELP; see those separate HELP systems for details on the products.

QUICKDOCS:

Quickdocs are locally prepared help files made available to users as quickly as possible, without the sometimes lengthy formal review and corrections process official Computing Center documents receive. For a list of available Quickdocs, enter:

```
$ DIR SYS$DOCUMENT:
```

For an index to available Quickdocs, use:

```
$ PRINT SYS$DOCUMENT:INDEX.QUICKDOC
```

USING CALENDAR:

CALENDAR is a computerized calendar designed to help you manage your schedule. It can be used to remind you of imminent appointments, and it can even be used to allow others to tell when you'll be available.

To use CALENDAR in its normal mode, use:

```
$ SET TERM/VT100      <-- you must use a VT100!
$ @SYS$SYSTEM:UNSUPPORTED_SETUP
$ CALENDAR                    <-- start CALENDAR
```

SHARING CALENDAR INFORMATION:

To share your CALENDAR data with others, you must allow other users to have read access to your CALENDAR.DAT file. Each user who will need to frequently consult your CALENDAR will probably also want to create a new easy-to-type nickname to use in referring to your CALENDAR:

```
[FROM YOUR ACCOUNT, say, JSMITH on DISK$USER2]
$ CALENDAR  <-- make sure CALENDAR.DAT exists
$ SET PROT=W:E [000000]JSMITH.DIR
$ SET PROT=W:RE SYS$LOGIN:CALENDAR.DAT

[FROM OTHER ACCTS. NEEDING ACC. TO YOUR CAL.]
$ JOHNNY :== CALENDAR/READ_ONLY-
/FILE=DISK$USER2:[JSMITH]CALENDAR.DAT
$ JOHNNY
```

NOTE: Your colleagues will probably want to add the JOHNNY command (or whatever they choose to call the pointer to your CALENDAR) to their LOGIN.COM file so they can automatically use it simply by typing JOHNNY

NOTE: You can have more than one (completely independent) calendars on your account. For example, perhaps JSMITH will use the JOHNNY calendar for public appointments, but maybe he'll have another PRIVATE calendar for sensitive or private appointments:

```
$ PRIVATE:==CALENDAR/FILE=SYS$LOGIN:SECRET.DAT
$ PRIVATE      <-- make sure SECRET.DAT exists
$ SET PROT=(W,G) SYS$LOGIN:SECRET.DAT <-- keep
                 other people from reading it!
```

MANAGING YOUR SESSION:

IMPORTANT SPECIAL CHARACTERS

DELETE	<-- Deletes one character
<ctrl>-U (or <ctrl>-X)	<-- Deletes to beginning of line
<ctrl>-S	<-- Pauses output
<ctrl>-Q	<-- Restarts output
<ctrl>-O	<-- Suppress output
<ctrl>-C	<-- Interrupts what is currently being done
<ctrl>-Y	<-- More severe interrupt - abnormally terminates EDT ["undo" with $ CONTINUE]
<ctrl>-R	<-- Reprints current line
<ctrl>-W	<-- Reprints your whole screen (in EDT)
<ctrl>-Z	<-- Exit character/EOF

USING THE VMS COMMAND LINE EDITOR

<ctrl>-B (or up arrow)	<-- Backup to previously entered command
(down arrow)	<-- Move the other direction through cmd's
<ctrl>-D (or left arrow)	<-- Move cursor 1 char to the left
<ctrl>-F (or right arrow)	<-- Move cursor 1 char to the right
<ctrl>-H (or BACKSPACE)	<-- Move cursor to the beginning of the line
<ctrl>-E	<-- Move cursor to end of line
<ctrl>-J	<-- Delete one word to the left of the cursor
<ctrl>-A	<-- Toggle Overstrike/Insert Mode

INTERRUPTS:

When you interrupt an executable image by typing a control-Y, execution of that program ceases, but not necessarily irrevocably!

After you type a control-Y you can do one of three things:

First of all, if you're <u>really</u> sure you don't want to resume execution of that image, you can make the interruption permanent with the command:

```
$ STOP
```

On the other hand, if you hit control-Y unintentionally, you can immediately resume execution by entering the command:

```
$ CONTINUE
```

If you're not sure if you want to stop or to continue, your best bet is to use the command:

```
$ SPAWN
```

to let you snoop around and make up your mind without interfering with the interrupted image (which you may still yet want to restart). Once you decide what you want to do, enter:

```
$ EOJ
```

to logout of your spawned sub-process, and then enter either STOP or CONTINUE

NOTE: Why is it necessary to SPAWN a process if you need to poke around before deciding if you want to stop or continue a process you've interrupted?

Well, if you interrupt an executable image and then <u>accidentally</u> run any other executable image while you're trying to make up your mind (about continuing), your interrupted image will <u>not</u> be continue-able! This is a particularly troublesome limitation when you realize that many VMS "commands" are actually executable images! Using SPAWN eliminates this potential problem.

UONET Commands:

NOTE: If you connect via the Gandalf but need to get to UONet, select UONET when the Gandalf tells you to "Enter class".

CONNECTING VIA UONET

```
UONet> C OREGON  <-- Connect to Oregon by name
UONet> C 128.223.20.2      <-- Connect by IP #
```

NOTE: Due to security considerations, it is no longer possible to directly connect to non-UOREGON.EDU machines directly from UONet. Connect to a UOREGON.EDU host (such as the VAX 8800) and then TELNET back out to the other system.

WORKING WITH MULTIPLE SESSIONS

You can run up to three sessions simultaneously if you login via UONet. This gives even the poorest of terminals at least some of the multi-session power of true workstations.

```
UONet> C OREGON      <-- connect to session #1
<do some work in session #1>
<ctrl>-]     <-- escape to UONet (repeat twice
                      if using Kermit on a PC)
UONet> C JANUS           <-- start session #2
<do some work in session #2>
<ctrl>-]                  <-- escape to UONet
UONet> RES 1            <-- resume session #1
```

NOTE: Don't forget to logoff off **ALL** the machines you may be connected to and also remember to close UONet with the L command!

DISCONNECTING FROM UONET SESSIONS

```
UONet> DISC 1             <-- drop session #1
UONet> L                <-- drop ALL sessions
```

MISCELLANEOUS UONET COMMANDS

```
UONet> SHOW SES  <-- See what sessions we have
UONet> SHOW DEFAULT   <-- See default settings
UONet> DO STATUS  <-- See if we're up and okay
```

FILES

VAX FILENAMES:

A complete VAX filename has the form:

```
NODE::DISK:[USERNAME.SUBDIR]NAME.EXT;VERSION
```

Normally, most of this information will automatically default to the correct value so you can simply use filenames such as:

```
FOURIER.SAS
FRESHMAN.DAT
APPROX.FOR
```

Under some relatively uncommon circumstances you'll need to provide more information. For example, if you are referring to a file that resides on someone else's account, you'll usually need to explicitly specify the name of the other account, as well as that other account's disk pack name:

```
DISK$USER2:[JSMITH]BREADSALES.DAT
DISK$USER4:[KBROWN.1978]TAX.DAT
```

A NOTE ON GOOD FILE NAMING PRACTICES:

While VAX filenames CAN have 0 to 39 characters before the dot and 0 to 39 characters after the dot, you should try to keep your filenames "reasonable." Generally, this means that you should choose filenames that have 1 to 8 characters before the dot, and 1 to 3 characters after the dot.

You should also use standard file extensions whenever you can. Common extensions include:

```
.DAT    raw data files
.FOR    Fortran source files
.LOG    batch log/error log files
.LIS    output files for printing
.SAS    SAS command files
.SPS    SPSSx command files
.BMD    BMDP command files
.MIN    MINITAB command files
.SYS    SPSSx binary system files
.RNO    Documents with runoff formatting
```

VAX "MAGIC" FILES:

Some files your account may seem to "create:"

CALENDAR.DAT – Your electronic calendar. Created when your use the CALENDAR command.

LOGIN.COM – This file contains commands that are automatically executed every time you login. Each account automatically gets a LOGIN.COM when it is created. You may want to customize yours.

MAIL.MAI – Your electronic mailbox. Created when you first receive a mail message.

MAIL$<some big number> – These contain mail messages too big for your your main mailbox. Don't fool around with them! Let MAIL manage them for you!

NEWSRC. – This file keeps track of the "NEWS" items you've read. Created by NEWS

PERSONAL.LGP – Your personal DECSpell dictionary. Created when you use SPELL

<something>.JOU: – EDT journal files. Automatically created by EDT.

<something>.TJL: – TPU (or EVE) journal files. Automatically created by TPU (or EVE).

SOME VAX FILE NAMING "TRIVIA:"

– You can substitute < and > for [and] in directory names

– You can use a "." for the ";" normally between the extension and version number

– The absolute "earliest edition" of a file can always be referred to as version -32768

VAX/VMS RECORD ATTRIBUTES:

Every VMS file has associated record attributes that control how VMS interprets and manages records in that file.

In general, there are three major categories of file formats you'll encounter under VMS:

o CARRIAGE RETURN CARRIAGE CONTROL ("LIST"): These are "normal" flat ASCII files with a carriage return/line feed sequence at the end of each record. When you create a new file in EDT (or some other text editor) this is the sort of file you get. Virtually all VMS commands (such as EDIT, TYPE, PRINT or DUMP) work fine with this type of file.

o FORTRAN CARRIAGE CONTROL ("LINE PRINTER"): These are the type of file typically created as output from Fortran, SAS, SPSSx and other programs. Fortran carriage control format files use column one of each record to manage spacing of line printer reports. Recognized carriage control characters include:

Col. 1 Code	Interpretation
`<null>`	do no carriage control
+	overprint
`<space>`	single space (normal print)
0	double space (skip line)
1	top-of-form (page eject)

While Fortran carriage control files are intended for printing on a line printer, you can also edit them, although you may find the carriage control characters somewhat distracting.

o "CARRIAGE CONTROL EQUALS NONE" ("BINARY"): These are unformatted files. Examples include .OBJ and .EXE files, as well as files created using unformatted Fortran WRITE statements. These files are not printable or editable because their contents typically are not standard printable ASCII characters.

HOW BIG IS A DISK BLOCK, ANYHOW?

New VAX users often ask "Just how big is a block of VAX disk space, anyhow?" A VAX disk block is 512 characters (or "bytes") long.

A page of double-spaced text is about 3 blocks

A survey or test with 100 respondents scanned by Scanner Services into 275 character records is about 54 blocks

A regular 5.25" PC floppy holds 720 blocks

A regular 3.5" Mac diskette holds 1600 blocks

A typical VAX account has 10,000 blocks

SCRATCH space available for temporary storage: 300,000 blocks are available from any username (subject to demands of other SCRATCH users)

An efficiently written 6250 BPI 2400' tape theoretically holds about 330,000 disk blocks (assuming 800x80 character per tape block, and standard 0.6" interblock gaps).

A VAX disk pack holds about 1,200,000 blocks.

NOTE: Even the smallest of VAX files will be allocated at least three blocks due to disk efficiency considerations.

SPECIAL REQUESTS FOR LOTS OF DISK SPACE:

Users who will routinely need additional disk space can obtain up to 50,000 blocks of permanent disk space by getting authorization for the increased allocation from their departmental computer representative, and then contacting Connie, the accounting clerk.

Under special circumstances, more than 50,000 blocks may be available to accommodate a large project lasting only a couple of months. If you need to make such a special space request, contact the system manager, Rick Millhollin, by sending a mail message to username RICKM.

EDITORS

EDT, EMACS, EVE, VI – WHICH IS "BEST"?

Users often ask "Which VAX editor is best?"

EDT: EDT is the traditional (and default) VAX editor. EDT has both full screen and line editor modes, and is an extremely efficient editor for the knowledgable user, making excellent use of the terminal keypad keys for editing tasks. EDT does have a few drawbacks, however. The worst of these shortcomings are: (a) No way to toggle between insert and overstrike mode, (b) No provision to shift the 80 column "window" to the right or to the left when viewing wide files, (c) Inability to even so much as open files wider than 255 characters, (d) poor pattern matching capabilities, and (e) no support for multiple on-screen windows.

EVE: EVE, on the other hand, overcomes EDT's five worst problems as noted above, but does a pathetic job of using the terminal keypad. Unless you extensively redefine keys, you'll find yourself having to type painfully long commands to do even the simplest of tasks in EVE. Also, if you use NCSA Telnet on PC's, you'll find that NCSA defines the PageUp, PageDown, Home and End keys (on the PC keypad) to accomplish those functions using the keystrokes which are correct for **EDT,** not EVE!

EMACS: EMACS is an important editor for users of non-VT100 terminals since it is the only VAX full screen editor that can be run on non-VT100 terminals. However: (a) EMACS is not officially supported, so if you use it you're "on your own," (b) EMACS commands use hard-to-remember key bindings, and (c) some EMACS commands (like the ^S search command) require special terminal settings to avoid problems.

VI: VI, like EMACS, is an unsupported editor. It is available primarily for Unix users who are already familiar with it.

TECO: TECO is another unsupported editor. It is cryptic, unforgiving and difficult to learn to use well. Not recommended.

USING THE EDT FULL SCREEN TEXT EDITOR:

EDT is the traditional VAX/VMS text editor; it can be used in its full screen "flavor" only on a VT100-compatible terminal or micro.

```
$ SET TERM/VT100
$ ED MYFILE.TXT                  <-- Get into EDT
*C                       <-- Change to fullscreen
stuff can now be entered here   <-- Enter text
<ctrl>-Z                    <-- Leave fullscreen
*EX                            <-- Exit from EDT
```

The bare essentials needed for moving around:

```
Cursor:  Use the        Scroll:  <KP5><KP8>
         arrow keys              <KP4><KP8>

Top:      <PF1><KP5>
Bottom:   <PF1><KP4>
```

Searching for stuff:

```
Find backward: <PF1><PF3> tongue <KP5>
Find forward:  <PF1><PF3> tripe <KP4>
```

Deleting stuff:

```
Delete char:         <DELETE>
Delete end-of-line:  <KP4>

Help:   <PF2>
```

NOTE: EDT is always in insert mode; it cannot be toggled between insert and overstrike.

PROBLEMS? If you see C* when you try to go into full screen mode, you have not correctly identified your terminal as a VT100! Type EXIT and then try again after typing:

```
$ SET TERM/VT100
```

See APPENDIX 1: EDT for more information on using EDT, including a way to make the keypad enter numbers instead of EDT commands.

EMACS: A NON-VT100 FULL SCREEN EDITOR

Micro EMACS is a full screen text editor that can be used on virtually any variety of terminal. Note, however, that the Computing Center does NOT officially support this editor, so you should use full screen EDT instead if you are using a VT100 compatible terminal.

```
$ @SYS$SYSTEM:UNSUPPORTED_SETUP
$ DEF/JOB TERM "vt100"  (or "tvi912", etc.)
```

NOTE: the terminal name MUST be lowercase

```
$ EMACS MYFILE.TXT          <-- Get into EMACS
stuff can now be entered here   <-- Enter text
<ESC> Z                    <-- Exit from EMACS
```

Toggling between insert and overstrike:

```
Insert:         ^X ^M OVER <return>
Overstrike:     ^X M OVER <return>
```

The bare essentials needed for moving around:

```
Cursor:     ^P                    Scroll:   ^Z
        ^B      ^F
            ^N                              ^V

Top:      <ESC> <
Bottom:   <ESC> >
```

Searching for stuff:

```
Find stuff forward:    ^S tripe <ESC>
Find stuff backward:   ^R tongue <ESC>
```

Deleting stuff:

```
Delete char:         ^H
Delete end-of-line:  ^K

Help:    <ESC> ?
```

NOTE: Some EMACS commands (such as the ^S search command) require you to turn off flow control before they'll work properly!

See APPENDIX 2: MICRO EMACS for more info.

EDT LINE MODE TEXT EDITOR:

If you don't have a VT100 terminal, you'll probably want to use EMACS. However, some non-VT100 users may prefer EDT line mode:

```
$ ED MYSTUFF.DAT     <-- go into EDT line mode
*I                   <-- go into INSERT mode
          Heree is a line of tecks
          Heree is anuther
          Heree's some more tecks
          <CTRL-Z>   <-- leave INSERT mode
*EX                  <-- exit from EDT
```

Basic EDT line mode commands:

```
*HELP                <-- get help while in EDT

*.                   <-- type current line
*.+1                 <-- type next line
*<return>            <-- type next line
*-1                  <-- type previous line
*2                   <-- type line 2
*1:3                 <-- type lines 1 thru 3
*14;-3               <-- goto line 14;
                         then back 3 lines
*T B                 <--
type beginning of file
*T E                 <-- type end of file
*T W                 <-- type the whole thing

*'anuther            <-- search for "anuther"
*S/anu/ano/2         <-- subst. only on line 2
*S/tecks/text/1:3    <-- subst. on lines 1-3
*S/Heree/Here/W      <--
subst. throughout file

*RES                 <-- renumber the lines

*M 2 TO 1.1          <-- move line 2 to 1.1
*C 1:3 TO 4          <-- copy lines 1:3 to 4

*R                   <-- replace current line
*D                   <-- delete current line
*D 2:3               <-- delete lines 2:3

*WR STUFF.DAT        <-- write STUFF.DAT
*EX                  <-- exit and save file
*QUIT                <-- abandon file--unsaved
```

ANOTHER VT100 EDITOR: EVE

EVE is another VAX text editor; it can only be used on a VT100-compatible terminal or micro.

```
$ SET TERM/VT100
$ EVE MYFILE.TXT                   <-- Get into EVE
stuff can now be entered here   <-- Enter text
<ctrl>-Z                          <-- Leave EVE
```

Toggling between insert and overstrike:

```
<KP ENTER>
```

The bare essentials needed for moving around:

```
Cursor:      <KP5>             Scroll: <KP.>
       <KP1>       <KP3>
             <KP2>                     <KP0>

Top:       <KP4> TOP <RETURN>
Bottom:    <KP4> BOTTOM <RETURN>
```

Searching for stuff:

```
Find stuff:         <PF1> tripe <RETURN>
Change direction:   <PF3>
```

Deleting stuff:

```
Delete char:           <DELETE>
Delete start-of-line:  <CTRL-X>

Help:     <PF2>
```

NOTE: If you see "SYS$INPUT must be a supported terminal" when you try to go into EVE, you have not correctly identified your terminal as a VT100! Exit from EVE and try again after typing: $ SET TERM/VT100

See APPENDIX 3: EVE for more detailed information about using EVE.

VI: A FULL SCREEN EDITOR FOR UNIX FANS

VI is a full-screen "mode-oriented" editor that is always in either insert mode, command mode or last line (:) mode. If you lose track of what mode you're in, hit <ESC>

NOTE: VI uses CaSe SeNsItIvE commands! Also note that VISETUP "breaks" EVE and other TPU-based editors until you log off and back on again.

```
$ @SYS$SYSTEM:UNSUPPORTED_SETUP
$ VISETUP                           <-- Define VI
$ VI SOMEPROG.FOR                 <-- Get into VI
i     <-- Insert text (note "i" is lowercase!)
stuff can now be entered        <-- enter text
<ESC>                       <-- stop inserting text
ZZ                              <-- exit from VI
```

The bare essentials needed for moving around:

```
Cursor:   Use the normal       Scroll:   ^B
          arrow keys                     ^F

Top:      :1 <return>
Bottom:   :$ <return>
```

Searching for stuff:

```
Find stuff forward:    /tripe
Find stuff backward:   ?tongue
```

Deleting stuff:

```
Delete char:          x     <-- lowercase "x"!
Delete end-of-line:   D     <-- uppercase "D"!

Help:       :Help
```

TECO:

TECO is one of the oldest and most powerful editors on the VAX, but it is terse and unforgiving. Don't attempt to use TECO unless you know EXACTLY what you're doing.

```
$ ED/TECO MYPROG.FOR         <-- edit MYPROG.FOR
<ESC><ESC>EX<ESC><ESC>        <-- exit from TECO
```

BASIC VMS COMMANDS:

LOOKING AT YOUR FILE DIRECTORY:

```
$ DIR                  <-- list file names only

$ DIR/SIZE             <-- list names and sizes

$ DIR/FULL          <-- detailed file directory

$ DIR [...]           <-- show all files in any
                   subordinate directories too

$ DIR/SINCE=TODAY  <-- list only today's files

$ DIR/SINCE=1-JAN-1988 <-- list file since ...

$ DIR/SIZE/SEL=SIZ=MIN=100  <-- Any big files?

$ DIR *.FOR            <-- list only .FOR files
```

PRINTING FILES ON THE FAST PRINTERS:

```
$ PRINT KGB.DAT               <-- print KGB.DAT

$ PRINT/JOB=5 CIA.TXT       <-- print 5 copies

$ PRINT/QUE=LP$CC105 GRU.DAT       <-- use the
                       terminal room's printer

$ PRINT/QUE=LP$XEROX NSA.DAT       <-- use the
                           Xerox laser printer

$ PRINT/FORMS=SMALL FBI.DAT       <-- print on
                       8.5x11" burstable white

$ SH QUE/FORMS     <-- other forms can we get?

$ SH QUE/ALL/D=P  <-- what jobs are waiting to
                     print in the print queue?
$ DEL/EN=121 SYS$PRINT  <-- Kill print job 121
                            in queue SYS$PRINT
```

NOTE: Files to be printed should NOT be deleted until they've actually printed! Unlike some other computer systems, under VMS no copy is made of files queued for printing, so the ORIGINAL file to be printed **MUST** still be available when it is finally that file's turn to get printed.

DISPLAYING WHAT FILES CONTAIN:

```
$ TY MYFILE.DAT             <-- type MYFILE.DAT

$ TY/PAGE APPLES.XYZ       <-- type APPLES.XYZ,
               pausing at the end of each page
```

COPYING FILES:

```
$ COPY BLUE.DAT RED.DAT    <-- copy BLUE.DAT to
                                        RED.DAT

$ COPY DISK$USER2:[RMILLER]GLUE.DAT  *.*
                <-- copy GLUE.DAT from RMILLER
                     to your current directory
```

RENAMING FILES:

```
$ RENA BLUE.TXT.* RED.DAT.*          <-- rename
                        BLUE.TXT to be RED.DAT

$ RENA XYZ29.*.*  PAYROLL.*.*   <-- rename all
               XYZ29 files to be PAYROLL files
```

NOTE: You cannot RENAME files from one disk pack to another; use COPY instead.

NOTE: You must have DELETE access to a file to be able to RENAME it; you can use the SET PROTECTION command to give yourself delete access to a file if need be.

SEARCHING FOR TEXT IN YOUR FILES:

```
$ SEA [...] "TRACTORS"     <-- Search all your
                     your files for "TRACTORS"

$ SEA/MATCH=OR [...] "RED","BLUE"   <-- Search
                        all your files for the
                       strings "RED" or "BLUE"

$ SEA/NOHI/F=NONULLS XYZ.TXT ""       <-- Type
               XYZ.DAT using special mnemonics
              for control codes (except nulls)
```

PURGING EXTRA COPIES OF FILES:

```
$ PU AV.SAS              <-- keep 1 copy of AV.SAS

$ PU                       <-- keep only 1 copy of
                             each file in this dir
```

NOTE: By default your account keeps only the two most recent copies of a file; to authorize retention of more than that number of copies, see the commands in SETTING VERSION LIMITS, below.

DELETING UNWANTED FILES:

```
$ DEL PROGRAM1.PAS.7          <-- delete one file

$ DEL PROGRAM1.PAS.*    <-- delete all versions

$ DEL PROGRAM1.*.*      <-- delete all PROGRAM1

$ DEL *.OBJ.*         <-- delete all .OBJ files

$ DEL/CON *.EXE.*      <-- explicitly O.K. each
                     file that's to be deleted
```

NOTE: Think BEFORE you delete! We have nothing like the PC's "Norton Utilities" for the VAX so when a VAX file is deleted it is GONE unless it is on a personal backup tape, or on a system backup tape! [You should be aware that user requests to restore inadvertently deleted files from system backup tapes incur a $25-$50 (real money) charge.]

NOTE: For help in deleting subdirectories, see the SUBDIRECTORIES section, below.

SORTING A FILE BY KEY FIELDS:

```
$ SORT/KEY=(POS:4,SIZE:6) RAW.DAT SORTED.DAT
```

APPENDING ONE FILE TO ANOTHER:

```
$ APPEND/NEW  TOP.DAT,BOTTOM.DAT  STACKED.DAT
```

or...

```
$ COPY TOP.DAT,BOTTOM.DAT STACKED.DAT
```

FINDING DIFFERENCES BETWEEN TWO FILES:

```
$ DIFF FILE1.DAT FILE2.DAT
```

LOOKING AT FILES WITH "DUMP":

DUMP shows you a file in hexadecimal format as well as in ASCII. DUMP is most useful when you want to look at a file that contains non-printable characters.

```
$ DUMP/RECORDS=COUNT:3 FOO.EXE
```

LOOKING AT FILES WITH "MORE":

MORE shows you a file a screenful at a time, and even allows you to backup and to search for (case specific) strings in a file.

```
$ @SYS$SYSTEM:UNSUPPORTED_SETUP.COM
$ MORE BLATHER.LIS           <-- look at a file
<space>                <-- see another screenful
?                     <-- see what else I can do
```

CLEARING THE SCREEN:

```
$ TYPE/PAGE NL:
```

VMS "TRICKS":

GETTING ACCESS TO SCRATCH SPACE:

```
$ SCRATCH        <-- Requests 300000 blocks for
       use until 6:00 AM tomorrow, using names
              such as: SYS$SCRATCH:BANANAS.DAT
```

You can use SCRATCH space two different ways:

In approach one, copy any files you're using to SYS$SCRATCH, make SYS$SCRATCH your default directory, do your work in SYS$SCRATCH, and then copy any modified files back to your permanent area.

For example, assume we want to run HUGEPROG on input MYBIG.DAT, with output to MYOUTPUT.LIS:

```
$ SCRATCH                     <-- Get scratch space
$ COPY MYBIG.DAT  SYS$SCRATCH:  <-- Copy files
$ COPY HUGEPROG.EXE  SYS$SCRATCH:   to scratch
$ SET DEF SYS$SCRATCH       <-- Move to scratch
$ RUN HUGEPROG          <-- Run the big program
$ COPY MYOUTPUT.LIS SYS$LOGIN  <-- Copy output
           back to (permanent) login directory
```

In approach two, you can avoid the hassle of copying files back and forth from SYS$SCRATCH, but you'll need to modify your programs to explicitly point any big files at SYS$SCRATCH:

For example, assume you're writing a large report (called X.LIS) from a Fortran program. You'd change your old open statement from:

```
OPEN (UNIT=10,FILE='X.LIS', ...      to:
```

```
OPEN (UNIT=10,FILE='SYS$SCRATCH:X.LIS', ...
```

NOTE: SCRATCH defines SYS$SCRATCH to point to the temporary area ONLY UNTIL YOU LOG OFF; SYS$SCRATCH then reverts to pointing to your login directory. You'll need to re-issue the SCRATCH command if you logoff and back on.

NOTE: SYS$SCRATCH isn't properly recognized from within FTP. Within FTP you'll need to use filenames like DISK$USER6:[SCRATCH.JSMITH] instead of simply SYS$SCRATCH:

USING SWING TO MOVE THROUGH YOUR DIR:

SWING visually represents your directories and lets you navigate thru them using your cursor.

```
$ @SYS$SYSTEM:UNSUPPORTED_SETUP.COM
$ DEFINE SWING$EDIT EDT
$ SWING
<use your arrow keys and return to choose>
```

NOTE: Use OPTION FILE for a list of files

<u>CAUTION:</u> Exercise EXTREME care in using the Delete option in SWING – since SWING will let you delete entire directory trees as easily as a single file, you can inadvertently eliminate many or all of your files quite easily!

<u>WARNING:</u> Also note that while SWING's Delete command DOES ask you to explicitly type YES to confirm that you actually want to do a deletion, it is NOT necessary to hit a return after typing YES.

Thus, you should NEVER use the command line editor (up-arrow/down-arrow keys or <ctrl>-B) while doing deletions from within SWING! That is, if you roll back to a line that says simply "YES" good old SWING will take that as the okay to start deleting stuff even if you were really trying to roll back to a yet earlier "NO" !!!

CAPTURING OUTPUT FROM YOUR TERMINAL:

```
$ SET HOST OREGON /LOG    <-- logon a 2nd time
$ RUN MYPROG  <-- run your program or whatever
$ EOJ                  <-- ... then logoff #2
$ PRINT SETHOST.LOG         <-- print your log
```

BODYGUARD: NEUTRALIZE HITMAN

HITMAN is an automatically executed system routine that attempts to identify and "hit" (kill) inactive processes. Currently HITMAN is set to terminate processes trees which haven't generated at least 1 BIO/DIO or 1 page fault, or used at least 30 msec CPU time over the following time periods:

```
08:01 - 17:00        Inactive for 9 hours
17:01 - 23:00        Inactive for 2 hours
23:01 - 08:00        Inactive for 1 hour
```

Typically, processes go inactive because a user didn't logoff when through. This can have serious security and financial implications.

However, there may be times when you are willing to assume full responsibility for logging off properly and would like to protect against getting "hit" unexpectedly. Under those circumstances you'll want to neutralize HITMAN.

Because HITMAN won't kill a process that has an active subprocess, HITMAN can be kept away by BODYGUARD (or any other subprocess that periodically generates a small spasm of activity exceeding the specified thresholds).

To run BODYGUARD, use:

```
$ SPAWN/NOWAIT @DISK$USER1:[JOE]BODYGUARD.COM
```

When you logoff, your BODYGUARD subprocess will also automatically be eliminated.

RECALL:

VMS remembers up to the last 20 commands you've executed, and you can recall those commands to re-execute them or to correct a typographical error.

While most people simply use the up and down arrows (or <CTRL-B>) to recall commands, you should be aware that there is also a RECALL command **somewhat** similar to UNIX's history command.

To see the commands currently stored in the recall buffer, use the command:

```
$ RECALL/ALL
```

Recall any particular command from that list either by number or by the first few letters of the command.

For example, let's say you want to recall command 6 from the list shown by RECALL/ALL

```
$ RECA 6
```

Or, maybe you'd like to recall the last directory command you executed. Enter:

```
$ RECA DIR
```

You can also wipe out the commands currently stored in the RECALL buffer by using the command:

```
$ RECALL/ERASE
```

NOTE: Unlike the UNIX history command, the numbers VMS RECALL assigns to commands are perpetually updated... RECALL 1 always returns the command you executed most recently, RECALL 2 retrieves the second most recently executed command, etc.

HOW SHOULD I COMPRESS VMS FILES?

In choosing a file compression scheme, you should consider three factors:

```
(1) How EFFECTIVE is the program? Does it
    generate highly compressed files?
(2) How FAST is the program? Does it take
    forever to compress or uncompress?
(3) How UBIQUITOUS is the program? Does it
    use a widely adopted algorithm?
```

It would probably be best if VMS offered a high performance native "COMPRESS" command, but the only native VMS compression-related routines are the "DCX" routines (described in the VMS Utilities manual). These routines seem to perform poorly when employed.

For example, in attempting to compress a draft of this book (then some 788 blocks long), the Fortran DCX code shown in the Utilities manual actually INCREASED the size of the file to 823 blocks! And, the DCX routines were quite slow!

Two alternatives to the DCX routines are the Liv-Zempel-Welch compression routines in wide use on UNIX systems ("compress"), and the SQUEEZE program widely used on microcomputers.

To give you some idea of the relative performance of the available compression routines, consider the following:

Compression Type	File Size	Compression Time
(uncompressed)	788 blocks	0 CPU seconds
DCX (Fortran)	823 blocks	54.3 CPU seconds
SQUEEZE	450 blocks	32.2 CPU seconds
LZCOMP -X	313 blocks	13.3 CPU seconds
LZCOMP (default)	361 blocks	7.2 CPU seconds

COMPRESSING FILES USING LZW:

The following compression commands use a version of the (UNIX-standard) Lempel-Ziv-Welch high efficiency compression algorithm. [The VMS executables used were obtained via anonymous FTP from KUHUB.CC.UKANS.EDU]

```
$ @DISK$USER1:[JOE.LZW]SETUP.COM
$ LZCOMP -V BIG.DAT BIG.DAT_LZW   <-- compress
$ LZDCMP -V BIG.DAT_LZW BIG.DAT <-- decompress
```

NOTE: To move files to a another system using a UNIX "compress V3.0"-compatible format, use the -X flag with LZDCOMP. Make sure you transfer the resulting compressed file in BINARY mode.

CAUTION: Do **NOT** attempt to use LZCOMP on VAX binary files!

COMPRESSING FILES WITH SQUEEZE:

While you should generally use the LZW routines to handle file compression on the VAX, you can also use the popular "SQUEEZE" format:

```
$ @DISK$USER1:[JOE.SQZ]SETUP.COM
$ SQZ BIG.DAT   <-- squeeze BIG.DAT to BIG.DQT
$ USQZ BIG.DQT           <-- unsqueeze BIG.DAT
```

CAUTION: Do **NOT** attempt to use SQUEEZE on VAX binary files!

WARNING: Always confirm the usability of newly squeezed files before deleting your unsqueezed version. To do so, immediately after squeezing a file, unsqueeze the squeezed version and use the VMS DIFF command to compare the original and squeezed-then-unsqueezed file you've created!

ENCRYPTING TEXT FILES WITH DES:

Antti Luoko's Finnish C-language implementation of the Data Encryption Standard (DES) is now available on the VAX.

Before using this DES implementation, however, be aware that:

(1) To satisfy the DES, the DES algorithm must be implemented in hardware, not software.

(2) The source code for this implementation came from comp.sources.unix via UUNET.UU.NET, as did the source for the required getopt, index and strchr routines.

While it is believed that the DES code and supporting routines perform as advertised, the cryptographic resources required to confirm this aren't available here at Oregon.

Therefore, if you decide to rely on these routines for critical security applications, you do so solely on the basis of your own assessment of their suitability, integrity and reliability.

(3) A truly secure system requires far more than mere encryption of individual files. Information that **REALLY** needs to be SECURE shouldn't be on a shared, networked system (like OREGON) at all!

(4) Lastly, know that it is illegal to export DES code without U.S. government approval.

```
$ DES :== $DISK$USER1:[JOE.DES]DES.EXE
$ DES -E -K FOOBAR XYZ.TXT XYZ.DES <-- encrypt
$ DES -D -K FOOBAR XYZ.DES XYZ.TXT <-- decrypt
```

NOTE: DO NOT forget your DES key. We have no way to "override" the DES to help you recover the encrypted contents of a file! Also, encrypt only plain text files – DO NOT attempt to encrypt binary (.EXE, .OBJ, etc) files... Finally, don't forget to destroy any plain text versions of the original file after encryption.

SET FILE/ENTER:

There are times when you'd like to be able to give a file more than one filename. For example, maybe you're an instructor, and you accidentally misspelled the name of a shared data file in a handout you gave to one of your classes, but spelled the filename correctly in some other handouts. [Assume for now that you can't simply contact your students to correct the error.]

What can you do in such a situation? You could always create a second copy of the file using the misspelled name, but that uses twice the disk space that's actually necessary, and can also cause problems with data concurrency between the two files if you fail to religiously update **both** of the files when you make changes to **either** of them.

One solution to problems of this kind is to use the SET FILE/ENTER command to create a filename "alias." For example:

```
$ SET FILE/ENTER=FOO.DAT THUD.DAT     <-- Make
    "FOO.DAT" another name for file "THUD.DAT"
```

NOTE: The VMS SET FILE/ENTER command does **NOT** work the same way the UNIX ln command does! Under VMS if you DELETE <u>either</u> the original file name or an alias file name, the <u>actual file itself</u> will be deleted, not just the particular file name specified.

If you just want remove one of the names by which a file is known, use SET FILE/REMOVE

```
$ SET FILE/REMOVE FOO.DAT
```

Be sure you don't use SET FILE/REMOVE to remove the LAST name by which a file is known, since this leaves the corpus of the file floating around on the disk pack without any name (!), a situation which can only be rectified by the system manager using ANALYZE/DISK.

Because of these potential perils, Digital recommends you avoid using SET FILE/ENTER and SET FILE/REMOVE under most circumstances.

SOME TEXT FILE TOOLS:

Because there are some text editing tasks that aren't handled very well by any existing DCL commands, a number of additional tools have been written to overcome these deficiencies.

As written, these utilities work with variable length records 2048 characters (or less) in length.

To access the utilities, do:

```
$ @DISK$USER1:[JOE.CHOP]SETUP
```

Add additional blank lines to a file:

```
$ ADDLINES/ADD=7/AFTER=212 ORIG.DAT LONG.DAT
```

Extract only a range of columns from a file:

```
$ COLS/START=6/END=14 WIDE.DAT NARROW.DAT
```

Convert tabs to spaces (five spaces per tab):

```
$ DETABIT TABED.DAT SPACED.DAT
```

Glue two files together side to side:

```
$ GLUE LEFT.DAT RIGHT.DAT JOINED.DAT
```

Write sequential numbers into a file:

```
$ NUMBER/FROM=20/TO=80/STEP=5/DIGITS=4 X.DAT
```

Pad a ragged-right file to a fixed width:

```
$ PAD/WIDTH=85 RAGGED.DAT PADDED.DAT
```

Strip trailing blanks (make the file "ragged right"):

```
$ RAGGED PADDED.DAT RAGGED.DAT
```

Take only specified lines from a file:

```
$ ROWS/FROM=100/TO=130 ALL.DAT SUBSET.DAT
```

MOVING AN ACCOUNT USING BACKUP:

Occasionally, a user will need to move all the files (and the subdirectory structures holding those files) from one account to another. This is quite common, for instance, when a user moves from an old account to a new one at the beginning of the term.

To do this, do the following:

(1) Logon to the old account. For the purpose of this example, we assume the user's old account is called JSMITH

(2) Then use the commands:

```
$ SCRATCH
$ BACKUP/LOG  [JSMITH...]*.*.*  -
SYS$SCRATCH:TEMP.BCK/SAVE
$ SET PROT=W:RE SYS$SCRATCH:TEMP.BCK
```

This will create a temporary disk BACKUP saveset containing the contents of the original account.

(3) Proceed to logoff the old account and then logon to the new account. We assume it is JOHNS for this example, and that you want to put all the files from the old account into a subdirectory tree beginning [JOHNS.OLD_ACCT...]

To do so, use the commands:

```
$ BACKUP/LOG/SELECT=([JSMITH...])  -
DISK$USER6:[SCRATCH.JSMITH]:TEMP.BCK/SAVE  -
[JOHNS.OLD_ACCT...]
```

NOTE: Be **SURE** the files you want are on your new account in good shape before you delete any files from your old account!!!

SETTING VERSION LIMITS:

```
$ SET DIR/VER=300 [000000]JDOE.DIR    <-- JDOE
           keeps up to 300 copies of each file
         for any new files created from now on

$ SET FIL/VER=300 [...]*.*        <-- keeps up
           to 300 copies of any existing files
```

LOOKING AT YOUR ACCOUNT:

```
$ SH PROC/ALL         <-- Look at your process

$ SH QUOTA            <-- How much disk space?

<ctrl>-T               <-- Show process status
```

MONITORING GENERAL SYSTEM USAGE:

```
$ MONI SYS         <-- watch system as a whole

$ MONI PROC/TOPC         <-- watch CPU loading
```

CONTINUOUSLY MONITORING A JOB:

(1) Locate the the PID of the job you want to monitor. Note that this must usually be one of YOUR OWN jobs!

```
$ SH USERS/FULL JSMITH
```

(2) Assume the the PID in this case is 21003A24. Monitor that job with:

```
$ SH PROC/CONT/ID=21003A24
```

(3) When done monitoring, quit with a <ctrl>-Z

DCL COMMAND PROCEDURES:

WHAT IS DCL?

Digital Command Language ("DCL") allows you to combine VAX/VMS commands (commands that you'd normally enter at the dollar sign prompt) into "programs" or "command files." Conditional and unconditional branch statements, subprograms, file input and output support, substitutable parameters and lexical functions are all available.

However, because DCL is interpreted (rather than compiled) and is therefore relatively slow, it is used most frequently to create little utilities or shells. DCL often is used to execute a common sequence of commands (such as a compile/link/go job) or to serve as a front end to third party software.

NOTE: For more information on DCL, see the **DCL Dictionary,** and the **Guide to Using DCL and Command Procedures on VAX/VMS** [both VMS manuals], as well as "VAX/VMS: Writing Real Programs in DCL," a Digital Press book written by Paul C. Anagnostopoulos.

EXECUTING A DCL COMMAND PROCEDURE:

```
$ ED GO.COM          <-- Make the command file
*C
$ FOR 'P1'             <-- compile the program
$ LINK 'P1'               <-- link the program
$ DEFINE SYS$INPUT SYS$COMMAND  <-- take input
       from the terminal rather than this file
$ RUN 'P1'                 <-- run the program
<ctrl>-Z
*EX

$ @GO HW1      <-- Invoke the procedure with @

$ DOIT :== @DISK$USER2:[JSMITH]GO.COM      <--
                    Define GO.COM to be "DOIT"

$ DOIT HW1     <-- Run GO.COM (as DOIT) on HW1

$ SET VERIFY    <-- Watch the commands execute
$ DOIT HW1                 <-- Run the program
$ SET NOVERIFY        <-- Disable verification
```

LOGICAL NAMES:

Logical names are used most frequently on the U of O VAX to provide user-friendly "nicknames" for disk packs and canned software areas. Thus, for example, rather than having to deal with the physical disk pack called:

```
$1$DUA5:
```

users may more conveniently refer to that pack as:

```
DISK$USER2:
```

Moreover, not only are such logical names somewhat easier to remember, they also isolate the user from system-wide reconfigurations and should therefore ALWAYS be used instead of actual "physical device" names.

To define a logical name, use the DEFINE command:

```
$ DEFINE EZTOTYPE BIGHAIRYNAMETHATSAREALPAIN
```

The logical name (or "nickname") you define will last until you log off or deassign the logical name with the DEASSIGN command:

```
$ DEASSIGN EZTOTYPE
```

To see the logical names that **you** currently have defined, use:

```
$ SH LOGICAL/PROC/JOB
```

To see **all** (non-concealed) logical names, use:

```
$ SH LOGICAL/ALL
```

To avoid obnoxious advisory messages whenever you change the value of a logical name, use the /NOLOG qualifier on the DEFINE command.

Also note that when you do a MOUNT you're creating a logical name pointing at the tape:

$ MOUNT/FOR/COM="blah blah" MU: 555123 <u>MYTAPE:</u>

SYMBOLS:

The most common use for symbols on the VAX is to define custom commands. Thus, for instance, if you always print on small 8.5x11" burstable white forms, you might decide to redefine your print symbol with the command:

```
$ PRINT :== PRINT/FORM=SMALL
```

You can also define symbols that point at executable images you've created. Thus, for example, if you had a program called XYZ.EXE, you could define a symbol to execute it with:

```
$ XYZ :== RUN DISK$USER4:[JDOE]XYZ.EXE
$ XYZ       <-- run that program by typing XYZ
```

Note, though, that if you want to be able to pass information to the program from the command line (i.e., pass a string such as the name of a file to the program at the time you execute the command), you must define a "foreign" symbol:

```
$ XYZ :== $DISK$USER4:[JDOE]XYZ.EXE <-- define
               command to be executed; note FULL
          PATH and "extra" leading dollar sign
$ XYZ BLAHBLAH     <-- program XYZ can now get
                    the string BLAHBLAH with the
                  LIB$GET_FOREIGN system service
```

To see the symbols **you** currently have defined, use:

```
$ SHOW SYMBOL/ALL
```

To see all system wide symbols currently defined, use:

```
$ SHOW SYMBOL/ALL/GLOBAL
```

Occasionally users become confused about when to use logical names and when to use symbols. DEFINE a logical name when that name may need to be **iteratively translated.** On the other hand, assign a symbol with :== when that symbol will **not** need iterative translation (for example, when you create a new command).

USING THE COMMAND DEFINITION UTILITY:

An alternative to using symbols is the CDU:

```
$ EVE PIGGY.CLD   <-- Build Command Definition
DEFINE VERB PIGGY
  IMAGE "DISK$USER1:[JOE]PIGGY.EXE"
  QUALIFIER WITHDRAWL,NONNEGATABLE
  QUALIFIER DEPOSIT,NONNEGATABLE
  QUALIFIER BALANCE
  PARAMETER P1,VALUE(DEFAULT="0")
  DISALLOW ANY2(WITHDRAWL,DEPOSIT,BALANCE)
<CTRL-Z>
$ EVE PIGGY.FOR      <-- Build Fortran Program
<tab> integer cli$present
<tab> character*12 amt
<tab> print *,'PERSONAL PIGGY'
<tab> open (unit=10,file='PIGGY.DAT',
<tab> 1 status='unknown')
<tab> read (10,'(i12)',end=100) ibal
<tab> rewind (10)
<tab> goto 200
100 <tab> ibal=0
<tab> goto 300
200 <tab> call cli$get_value('P1',amt,len)
<tab> read (amt(1:len),'(i12)') iamt
<tab> if (cli$present('WITHDRAWL')) then
<tab>   ibal=ibal-min(ibal,(iabs(iamt)))
<tab> else if (cli$present('DEPOSIT')) then
<tab>   ibal=ibal+iabs(iamt)
<tab> else if (cli$present('BALANCE')) then
<tab> else
<tab>    print *,'USE:   $ PIGGY/BALANCE'
<tab>    print *,'       $ PIGGY/DEPOSIT nn'
<tab>    print *,'       $ PIGGY/WITHDRAWL nn'
<tab> endif
300 <tab> print *,'BALANCE: ',ibal,' CENTS'
<tab> write (10,'(I12)') ibal
<tab> end
<CTRL-Z>
$ FOR PIGGY         <-- Compile personal piggy
$ LINK PIGGY           <-- Link personal piggy
$ SET COMMAND PIGGY     <-- Define the command
$ PIGGY                        <-- Make a bank
$ PIGGY/DEPOSIT 125         <-- Make a deposit
$ @SYS$SYSTEM:UNSUPPORTED_SETUP <-- Setup Verb
$ VERB PIGGY  <-- Scrutinize the Piggy Command
```

PROCESSES:

PROCESSES:

Processes are the fundamental unit of activity on the VAX. Whenever you login, submit a batch job, spawn a subprocess or run an image, you create a process with a unique process id (PID).

INTERACTIVE PROCESSES: The average VAX users is most familiar with using an interactive process: i.e., logging on and doing work in a conversational mode "at the dollar sign." However, there are some more advanced ways you can use VAX processes to improve your productivity.

BATCH PROCESSES: For example, you can submit a big project as a batch job to be run non-interactively during periods of low use, or in a discount batch queue. To run a batch job, create a .COM file containing the commands you'd like to execute, and then launch that command file with the SUBMIT command. Running batch jobs is discussed in more detail in the next panel.

SUBPROCESSES: Note that sometimes you don't want to bother building a batch job command file, but you also don't want to tie up your terminal while you compile a large program or wait for a tape to be mounted. In such a case you should think about spawning a subprocess to handle those tasks in the background while leaving you free to edit some other file or read your mail, say. Note that unlike a batch job, you don't need to build a command file to use subprocesses; another important distinction is that subprocesses die when their parent process terminates, while batch jobs continue to execute in the background.

DETACHED PROCESSES: Another type of background process is the detached process. A detached process is created when you use the RUN/DETACH command to execute an image. Detached processes continue to execute to completion even if the originating process terminates. [Detached processes are infrequently used by the average user.]

WORKING WITH THE BATCH WORLD:

Batch is particularly convenient when you're working with large jobs, or with jobs that you'd like to run late at night, or with jobs you'd like to run in a discount batch queue.

```
$ ED MYJOB.COM              <-- create a com file
*C
$ FORTRAN SAMPLERUN            <-- batch job will
$ LINK SAMPLERUN                compile, link and
$ RUN SAMPLERUN                run a fortran prog
23.5 34.1 27.6 5 8                  <-- some data
<control>-Z                  <-- end the com file
*EX

$ SUBMIT/NOPRINT/KEEP/NOT MYJOB.COM           <--
     submit the batch job to the default queue

$ SUBM/NOPRIN/KE/NOT/QUE=LOW$BATCH MYJOB   <--
          submit MYJOB.COM to the cheapo queue

$ SUBM/NOPRIN/KE/NOT/AFTER="15-JUN-1989:3:00"-
/QUE=LOW$BATCH MYJOB.COM  <-- submit MYJOB.COM
          to the cheapo queue to execute after
               3 AM on the 15th of June, 1989.
```

Output from your batch job will go into a file in your login directory called MYJOB.LOG

```
$ SH QUE/ALL/FULL  <-- show default que values

$ SH SYS/B          <-- look at all batch jobs

$ SH QUE/ALL/BATCH   <-- Spot your job entry #
$ SH SYS/B     <-- Use entry # to spot its PID
$ SH PROC/CONT/ID=4EB6    <-- Monitor PID 4EB6
<CTRL-C>                <-- Interrupt monitoring

$ SH QUE/ALL/BATCH         <-- show batch jobs
$ DEL/ENTRY=2756 SYS$BATCH    <-- delete a job
```

NOTE: When a batch job begins it ALWAYS starts in your "root" or login directory, even if you launched the job from a subdirectory. Therefore, remember to $ SET DEF if needed!

Also note that a batch job may not always run immediately since the queues usually limit the number of simultaneously executing batch jobs.

SPAWNING A (NOWAIT) SUBPROCESS

```
$ SPAWN/NOWAIT FOR MYPROG          <-- Spawn a
                  subprocess to compile MYPROG

$ SH SYS/S  <-- See how subprocesses are doing
```

NOTE: The command following SPAWN/NOWAIT should be one you could **type at the dollar sign** – that is, unlike a batch job, don't just name a .COM file, and unlike a detached process, don't just name a .EXE file. Specify a complete **COMMAND** you could legitimately type at the dollar sign! (This **can** include the commands RUN FILENAME.EXE or @FILENAME.COM of course!)

USING SPAWN AND ATTACH:

Sometimes you may want to have two subprocesses active alternately. For instance, you might want to have one subprocess sitting in one subdir, and another subprocess in a second subdir, allowing you to easily pop back and forth. You can do this with the following commands:

```
$ SET PROC/NAME=ONE  <-- Name original session
$ SPAWN/PROC=TWO             <-- Make session #2
$ ATTACH ONE    <-- Return to original session
$ ATTACH TWO          <-- Return to session #2
$ SH PROC/SUB  <-- What subprocesses do I have
$ STOP TWO                  <-- Stop session two
```

RUNNING DETACHED PROCESSES:

When you want an executable image to continue to run even after the creating process has terminated, create a detached process:

```
$ RUN/DETACH/PROC="MYJOB" BIGPROG.EXE
$ SH SYS/ALL                <-- Look for "D" jobs
```

NOTE: The file you specify to run as a detached image should be a .EXE file

---

FILE PROTECTIONS AND ACL'S:

PROTECTIONS: CONTROLLING ACCESS TO FILES

General access to your files is controlled by a protection scheme that defines four levels of potential access for four categories of users:

THE FOUR CATEGORIES OF USERS:

o WORLD (W) – Everybody

o GROUP (G) – At the U of O, all accounts from your department are part of your "group" (note that this includes **student** accounts requested by instructors in your department).

o OWNER (O) – This is whoever OWNS a given file. The owner of a FILE may or may not be the same as the owner of the ACCOUNT that the file "lives" under. Usually, the account used to make or copy a file is the actual "owner" of that file.

o SYSTEM (S) – This is the system manager, and VMS itself. Generally speaking, forget this category even exists. The protections you set for this category are essentially irrelevant since privileged users can give themselves access to ANY file if they are so inclined; note, however, that you CAN mess up access to your mailbox and other files if you miss set the protections for this group.

THE FOUR FILE ACCESS LEVELS:

o READ (R) – able to read (or copy) the contents of a file

o WRITE (W) – able to write (or modify) the contents of a file

o EXECUTE (E) – able to execute (or "run") the file, but not able to read, write, or delete it

o DELETE (D) – able to delete (remove) the file altogether

Generally speaking, NEVER give GROUP or WORLD users WRITE or DELETE access to your files.

DIRECTORY PROTECTIONS:

In addition to being able to protect individual files, general protections also play an important role in controlling access to your directory and subdirectories.

Generally speaking, a user must have (at least) execute access to your main directory (and to any intervening subdirectories) to be able to access any file.

Note, however, that with **only** execute access, other users won't be able to read your directory to learn the **names** of your files. Therefore, if you want colleagues to be able to access a particular file, you'll need to provide them with the specific file names they'll need since they won't be able to search your directory themselves.

If you DO want to allow other users to be able to read your directory, put EXECUTE **and** READ protections on your directory. They can then do a directory of your account just as they'd do a directory of their own area. Note, also, that users **MUST** have READ access to your directory to be able to do a "wildcard" copy of files from your account.

CAUTION: Don't rely on denying read access to your directory to protect (otherwise accessible) sensitive files from scrutiny. There are many different ways people can detect or guess the names of your files, thereby circumventing the security of your files. If you don't want people to be able to get at a file, take all WORLD and GROUP protections off the **FILE** – DON'T depend on having only EXECUTE protections on your **DIRECTORY!**

Lastly, **NEVER** give WORLD or GROUP either WRITE or DELETE access to any directory!

DEFAULT PROTECTIONS:

New versions of existing files generally inherit the protections of the previous version of the file; brand new files receive whatever default file protections you've established. By default, here at the U of Oregon, new files are created with (W,G,O:RWED,S:RWED) protections.

SETTING GENERAL FILE PROTECTIONS:

```
$ DIR/PROT    <-- See current file protections

$ SET PROT=W:RE *.*.*  <-- let anyone read any
                 of your current files, if you
                      tell them the file names

$ SET PROT=W:RE CRAY.DAT.* <-- let anyone read
                 CRAY.DAT, if its name's known

$ SET PROT=(G,W) CDC.DAT.* <-- protect CDC.DAT
                     from being read by anyone

$ DIR/PROT [000000]JSMITH.DIR  <-- See current
                         directory protections

$ SET PROT=W:RE [000000]WGROAT.DIR  <-- allows
           anyone to browse WGROAT's filenames

$ SET PROT=(W:E,G:E) [000000]WGROAT.DIR    <--
         prevents anyone from getting his file
           names but allows access to files if
              the file names are already known

$ SET PROT=(W,G) [000000]WGROAT.DIR  <-- keeps
         anyone from getting at any of his files
      even if they already know the file names
```

DEFAULT PROTECTIONS FOR NEW FILES:

```
$ SHOW PROT      <-- see current default prots

$ SET PROT=W:RE/DEFAULT <-- as a default, give
                           read permission for
                         anything created from
                    now on during this session
```

SPECIFIC FILE PERMISSIONS:

While general file protections work fine for most users, there are some instances where you need to grant special access privileges to only one user (or a group of users). In such cases, you'll want to use Access Control Lists (ACLs) instead of general file protections.

While ACLs allow a bewildering array of attributes to be established for a given file, the most common ACL-related commands a user would use are shown below:

```
$ DIR/ACL PROG.FOR  <-- See PROG.FOR's current
                  Access Control List (if any)

$ SET ACL/ACL=(ID=FNIX,ACC=R) PROG.FOR     <--
            Give FNIX "R"ead perms to PROG.FOR

$ SET ACL/ACL=(ID=RBRITE,ACC=CONTROL) GOLD.DAT
               <-- Gives RBRITE all the access
              enjoyed by the actual file owner

$ SET ACL/ACL=(ID=[BA_DSC,*],ACC=R) INVERT.EXE
           <-- Gives all members of the BA_DSC
               group read access to INVERT.EXE

$ SET ACL/DEL PROG.FOR     <-- Remove all ACLs
        except those with the PROTECTED option
```

NOTE: ACC=NONE may not eliminate access by a given person to a particular file; some users have multiple accounts and you may miss one or more of their IDs.

ACLs that pertain to a particular individual should be put ahead of more general ACLs since the first ACL that's matched terminates ACL evaluation.

Overuse of ACLs impact system performance.

For information on current group codes, see: DISK$USER1:[JOE]GROUPCODES.TXT

For information on particular group membership use the WHO command. Example: $ WHO BA_DSC

SUBDIRECTORIES:

ORGANIZING YOUR SUBDIRECTORIES:

Interest in subdirectories typically parallels file accretion: lots of files equals lots of interest!

When you do find yourself with a lot of files and an urge to get organized by putting those files into subdirectories, keep in mind the following considerations:

FLAT or TALL SUBDIRECTORY STRUCTURE? A flat subdirectory structure puts your subdirectories immediately under your top-level directory, while a tall subdirectory structure is more hierarchical with subdirectories under subdirectories (up to a maximum of eight levels deep). Flat structures are conceptually simpler to work with than tall structures, but tall structures impose more organization upon your files than flat structures do. Most people use at most a couple of levels of subdirectories.

ORGANIZE BY PROJECT, LANGUAGE, OR ? There are many different ways you can organize your subdirectories.

Some people prefer to organize their subdirectories by project:

```
[.DISSERTATION] [.JASA_ARTICLE] [.CLASS]
```

while others prefer to organize their subdirectories on the basis of the tool used:

```
[.FORTRAN]  [.MACRO]  [.SAS]  [.SPSSX]
```

while yet others prefer to organize on the basis of the type of file:

```
[.DAT] [.DCL] [.EXE] [.SRC]
```

and some people even organize their files chronologically:

```
[.WINTER_88] [.SPRING_89] [.SUMMER_89]
```

DIRECTORY NAMES: In a nutshell, keep them short and sweet since you'll be typing them a lot. Use [.BANJO] not [.INSTR_121110110231]

WORKING WITH SUBDIRECTORIES:

```
$ CREATE/DIR [CHALL.ONIONS]      <-- create an
                            ONIONS subdirectory
                           below CHALL's top dir

$ CREATE/DIR [.PICKLES]   <-- create a PICKLES
                          subdirectory below the
                               current directory

$ SH DEF                           <-- where am I?

$ DIR *.DIR                       <-- any subdirs?

$ SET DEF [.PICKLES]         <-- move down into
                             PICKLES subdirectory

$ SET DEF [-]          <-- move back one level
                         from the current subdir

$ SET DEF [-.OLIVES]      <-- move back 1 level
                        from your current subdir
                           then move into OLIVES

$ SET DEF SYS$LOGIN       <-- move back to your
                             main (top) directory

$ DEL [.NUT]*.*.*     <-- to delete a dir: (1)
$ SET PROT=O:D NUT.DIR delete all files in the
$ DEL NUT.DIR.1              directory, (2) give
                        yourself perms to delete
                    the dir, THEN (3) delete it

$ RENA *.FOR.* [JDOE.ANT]    <-- move all .FOR
                 files from the current dir to
                             the ANT subdirectory

$ RENAME [JDOE.SPIDER]*.DAT.* []  <-- move all
                    .DAT files from the SPIDER
                     subdir to the current dir
```

NOTE: When moving files from one directory to another, always explicitly rename ALL VERSIONS! For example, specify ABC.FOR.* rather than simply ABC.FOR – if you omit that final asterisk, you will end up moving only the most recent version of the file, with earlier versions still at their "old" location to potentially confuse you!

REFERRING TO DIRS AND SUBDIRS:

```
[JSMITH]        <-- JSMITH's top level directory

[KBROWN.XYZ]       <-- KBROWN's XYZ subdirectory
[.QRPS]         <-- the QRPS subdirectory under
                        the current directory
[ST60001.HWK.PROJ1]       <-- ST60001's PROJ1
                          subsubdir under HWK

[]              <-- the current dir (or subdir)
[-]             <-- the directory above this one
[-.FLY]         <-- the FLY subdir "next to" the
                          current directory

[...]            <-- all dirs from here on down

[000000]         <-- the top-level directory on
                    a disk; account directories
                     are created under [000000]

SYS$LOGIN:       <-- directory you start out in
SYS$SCRATCH:     <-- your login directory UNTIL
                you type $ SCRATCH, then points
                    to a temporary scratch area
```

NOTE: Much of the confusion associated with subdirectories comes from implicit substitution of parts of filenames. Ex:

```
$ RENA XYZ.FOR [.MARCH]
```

is really equivalent to saying:

```
$ RENA XYZ.FOR [JSMITH.MARCH]XYZ.FOR
```

Default values are conveniently substituted in for "missing" elements (current dir, file name and file extension in this case).

Wildcard characters are also often used:

```
*    means any 0 or more chars at this spot
%    means exactly any one char at this spot
```

For example, to move any files with three letter filenames beginning in "X" to subdirectory MYXSTUFF, you'd use:

```
$ RENA X%%.*.*  [.MYXSTUFF]
```

ROOTED DIRECTORIES:

From time to time while you're using the VAX, you may see what looks like an "odd" directory specification with a form something like:

```
DISK$BLAH:[FOO.][WHIZZO]
```

This isn't a mistake! What you're seeing is a "rooted directory." Most often, rooted directories are used to define a nickname from which a whole collection of subdirectories can "take root."

For example, you might do something like:

```
$ DEFINE MY_SCORES DISK$USER2:[JDOE.TESTS.]
```

You can then create subdirectories under that "root" with commands such as:

```
$ CREATE/DIR MY_SCORES:[SAT_SCORES]
```

Files in that directory would then have file names that look like:

```
MY_SCORES:[SAT_SCORES]JAN.DAT
MY_SCORES:[SAT_SCORES]SUMMARY.DAT
```

You can, of course, use the "long form" name instead:

```
DISK$USER2:[JDOE.TESTS.][SAT_SCORES]JAN.DAT
```

but that really defeats the purpose of hiding all that stuff as a rooted directory, eh?

NOTE: Since logical names go away when you log off, you'll want to be sure to add a copy of any DEFINE commands (that set up nicknames for rooted directories) to your LOGIN.COM file!

COMMUNICATING:

YOUR ELECTRONIC MAIL ADDRESS:

If your VAX 8800 username is JSMITH, then use:

```
local address:       JSMITH
BITNET:              JSMITH@OREGON
INTERNET:            JSMITH@OREGON.UOREGON.EDU
DECNET:              OREGON::JSMITH
```

SENDING MAIL TO OTHER ADDRESSES:

To send mail to a local user of the VAX 8800, simply use his username:

```
TJONES
```

If the user is on a machine DECNet-ed to the 8800 use NODE::USERNAME to reach him:

```
OPTIC1::BFRANK
```

If you're sending BITNET or INTERNET mail, enclose the address in IN%" "

```
IN%"ESKPIE@ALASKA"
IN%"RSNOOTY@GUANO.HARVARD.EDU"
```

If you're sending UUCP mail to ...!node!userid use:

```
IN%"userid@node.uucp"
```

Some other mailer hints:

– Case can matter: Some machines have case sensitive userids, so watch capitalization.

– Specify explicit message routings (if needed) using percent signs. Mail to IN%"ZIGGY%XYZ%JFK@LMN" is first sent to site LMN, which strips its address (@LMN) and changes the next % to an @ before sending the message on to JFK, where the process iterates.

– If you have mail that bounces, you can check with the University of Oregon Computing Center Postmaster (username RHALLER) for help getting your mail through.

THE BASICS OF USING MAIL:

```
$ MAIL                          <-- get into MAIL

MAIL> DIR         <-- What messages do I have?

MAIL> <return>                 <-- Read a message

MAIL> R                    <-- Reply to a message
To:   NMILLER
Subj: OUR ARTICLE
How's IS it coming?    <-- Hit <return> at the
<ctrl>-Z                        end of each line!

MAIL> PRINT <-- Print mess. when you exit MAIL

MAIL> 4                    <-- Re-read message #4
MAIL> EXTRACT TEMP.LIS  <-- Put it in TEMP.LIS
MAIL> EX                           <-- Leave MAIL
$ PRINT TEMP.LIS          <-- Print the message

MAIL> DELETE 3            <-- Delete message #3

MAIL> S              <-- Now send RDOE a message
To:   RDOE
Subj: TODAYS MEETING
I can't make it today. Sorry, Pete.
<ctrl>-Z

MAIL> S  <-- Now send a message on the network
To:   IN%"FFGGN@ALASKA"
Subj: Your monograph on the High Arctic Vole
Can you send me a review copy of this piece?
<ctrl>-Z

MAIL> S DISCORDIAN.DAT      <-- Send this file
To:  JSMITH
Subj: Here is the file you asked for last week

MAIL> READ/NEW            <-- read new messages
                received while already in MAIL

MAIL> SELECT MAIL  <-- goto old message folder
MAIL> SELECT NEWMAIL      <-- goto new messages

MAIL> HELP                 <-- Get help on MAIL

MAIL> EXIT                  <-- Get out of MAIL
```

SOME MORE ADVANCED MAIL FEATURES:

```
MAIL> REPLY/EXTRACT      <-- Include a copy of
                   their message in your reply

MAIL> S
To:    @CLASS.DIS     <-- Send mail to each ID
Subj:  Assignment 2      in the file CLASS.DIS
Assignment 2 is now due at 5 PM this Friday.
<ctrl>-Z

MAIL> 7                   <-- Review message 7
MAIL> F                      <-- Forward it...
To:   AQUINLAN              <-- to AQUINLAN...
Subj: Handle this!       <-- with instructions
MAIL> D                  <-- then delete it...

MAIL> 5                   <-- Review message 5
MAIL> MOVE CLASS          <-- Move it to CLASS
Y                        <-- OK the new folder

MAIL> DIR/FOLDER   <-- What folders do I have?
MAIL> SELECT COMMITTEE    <-- Choose COMMITTEE

MAIL> DELETE 3:8  <-- Delete messages 3 thru 8

MAIL> COMPRESS   <-- Compress existing mailbox
MAIL> EX                   <-- Get out of mail
$ DELETE MAIL.OLD.*     <-- Delete old mailbox

MAIL> SET MAIL [.MB]   <-- Hide mail$... files

MAIL> SET FORWARD JJONES   <-- Forward my mail
MAIL> SET FORWARD IN%"""WAWA@XYZ.HAWAII.EDU"""

MAIL> SET COPY SEND,FORWARD,REPLY   <-- Keep a
           carbon of every message you produce
MAIL> SET P="TOM BLACK"  <-- Identify yourself
 (permanently) with this info on every message

$ MAIL :== MAIL/EDIT     <-- make mail use EDT

$ WHO SMITH   <-- Find username for all SMITHS
$ SET BROAD=NOUSER1        <-- Stop local SENDs
$ SET BROAD=NOUSER2      <-- Stop foreign SENDs
$ SET BROAD=NOUSER3   <-- Stop router messages
$ SET TERM/NOBROAD      <-- Stop ALL SEND msgs
```

MAIL KEYPAD MAPPING:

```
-----------------------------------------
|PF1      |PF2      |PF3      |PF4      |
|         |   HELP  | EXT/MAIL|  ERASE  |
|   gold  |         |         |         |
|         | dir/fold| extract | sel mail|
|         |         |         |         |
-----------------------------------------
|7        |8        |9        |-        |
|   SEND  |  REPLY  | FORWARD | READ/NEW|
|         |         |         |         |
| send/edi|rep/ed/ex|frwd/edit| SHOW NEW|
|         |         |         |         |
-----------------------------------------
|4        |5        |6        |,        |
| CURRENT |  FIRST  |  LAST   | DIR/NEW |
|         |         |         |         |
|curr/edit|firs/edit|last/edit| dir mail|
|         |         |         |         |
-----------------------------------------
|1        |2        |3        |ENTER    |
|  BACK   |  PRINT  |   DIR   |         |
|         |         |         |         |
|back/edit|pr/p/noti|dir/9999 |         |
|         |         |         |  ENTER  |
-------------------------------|         |
|0                  |.        |  select |
|       NEXT        |   FILE  |         |
|                   |         |         |
|     next/edit     |  delete |         |
|                   |         |         |
-----------------------------------------
```

FINDING A PERSON'S MAIL USERNAME:

While you can search for local usernames using the $ WHO command, users frequently ask "How do I find non-OREGON e-mail address?"

By far the best way to initially get a user's e-mail address is to ask the person for it – in person, on the telephone or by U.S. mail. [And help avoid creating this kind of communication problem yourself: add your complete e-mail address to your business cards, and routinely include it in professional articles and correspondence!]

If it is absolutely impossible for you to ask a user for his or her username directly, you can also ask the remote node's postmaster to help you identify a user's username. Note that a remote node's postmaster doesn't HAVE to do this for you, but will usually do so as an accomodation if you ask politely. [If you need help identifying the name of the network nodes which exist at a given site, see the panel "FINDING ADDRESSES" later in this book.]

If the user you're trying to track down has an account on an Internet node, you can send mail to the special username "postmaster" (at the remote node) for assistance. For example:

```
IN%"postmaster@foouvms.thud.edu"
```

If your user is on Bitnet instead of the Internet, you may need to use the special userid "POSTMAST" instead of "postmaster" and for some nodes even that may not work. To check to see the username of the postmaster of node PSUORVM, for example, use:

```
$ SEND NETSERV@UCBCMSA "GET NODENTRY PSUORVM"
```

INSTEAD OF ASKING THE POSTMASTER:

Sometimes you can avoid asking a remote postmaster for help finding a user's username:

(1) An Internet WHOIS server may know him:

```
$ WHOIS -H STANFORD.EDU NORMAN
```

There is currently no comprehensive list of WHOIS servers, so the only way to know if a WHOIS server is available is to try it.

(2) A BITNET LISTSERV server may know him:

```
$ SEND LISTSERV@PSUORVM "/WHOIS NORMAN"
```

You'll need to pick the right LISTSERVer, which may not be a trivial task.

(3) Or, try some other BITNET-based servers:

```
$ SEND NETSERV@BITNIC "UDS FIND :NAME NORMAN"
```

[See DISK$USER1:[JOE]NAMESERV.TXT for a list of BITNET-based name servers to try]

(4) Or, an Internet "enhanced" finger server may know about your user:

```
$ FINGER NORMAN@SKINNER.CS.UOREGON.EDU
```

As with WHOIS, there is no comprehensive list of enhanced finger servers.

(5) Finally, NYSERNet's White Pages Pilot Project may know about your user:

```
$ TELNET WP.NYSER.NET
login: fred              <-- must be lowercase
fred> whois -org *       <-- Is his org there?
fred> whois surname "norman"  <-- look for him
fred> quit
```

SENDING MAIL TO OTHER NETWORKS:

COMPUSERVE EASYPLEX: Assume you want to send mail to a Compuserve EasyPlex user such as id 70009,1484 You'd use the VAX mail address:

`IN%"70009.1484@compuserve.Com"`

Notice that the comma normally seperating the two-number CompuSeve ID must be converted to a period. Also be aware that EasyPlex messages are limited to a maximum of 50,000 characters.

COMPUSERVE INFOPLEX: Assume you want to send mail to a Compuserve InfoPlex user such as user xyzco:john You'd then use the VAX mail address:

`IN%"john@xyzco.Compuserve.Com"`

An InfoPlex user with a departmental address, such as xyzco:sales:jane would have a VAX mail address of:

`IN%"jane@sales.xyzco.Compuserve.Com"`

MCI MAIL: You can send VAX mail to an MCI Mail user with any of the three following address formats:

```
IN%"accountname@mcimail.Com"
IN%"mci_id@mcimail.Com"
IN%"full_user_name@mcimail.Com"
```

FIDONET MAIL: To send mail to John Smith at Fido node 1:129/87.2, use the VAX address:

`IN%"John.Smith@p2.f87.n129.z1.fidonet.org"`

JANET MAIL: JANET is the most common British network, but JANET addresses are "reversed" from American usage. Therefore, if you see a JANET address that looks like:

`SPLAT@UK.AC.BLAH.FOO`

you'd need "un-reverse" it and add IN%" ":

`IN%"SPLAT@FOO.BLAH.AC.UK"`

SOME MAIL CAUTIONS:

"DISGUISED" MAIL: Current network mail transfer conventions make it possible for a technically knowledgable user to send mail impersonating anyone else they choose ("spoofed mail") or to send mail with no apparent sender whatsoever ("anonymous mail"). For this reason the "From:" field of a mail message should **NOT** be relied upon when authorship of a message is critical. Obtain confirmation by some other means for any suspicious or particularly important message.

[While there have not generally been problems with disguised mail so far, increasing use of mail for critical academic and administrative communications makes it important that you be aware of this limitation.]

MAIL "RECEPTION" PROBLEMS: Also be aware that some individuals do not routinely logon and read electronic mail messages they've received; other individuals delegate responsibility for reading electronic mail to a GTF, secretrary or administrative assistant.

Moreover, you should realize that while most electronic mail gets to its destination in minutes (or at most hours), some topologically remote sites (particularly UUCP-based sites) may not receive mail you've sent for several days.

Therefore, time-critical messages, or messages with sensitive contents should probably **NOT** be sent by electronic mail unless you can confirm that the intended recipient, and only the intended recipient, actually reads his own mail on a routine basis.

THE QUOTE BUG (NOW FIXED): The infamous quote bug (i.e., unmatched double quote marks in a network mail address resulting in the speedy exhaustion of user resources) has been fixed. Now, if you omit one of the required double quote marks from a network address, you'll simply get an error message.

SETTING UP MAIL NICKNAMES: ALIASES

While you can continue to set up mail "nicknames" using logical names created one-at-a-time with the DEFINE command, the University of Oregon now also supports a more sophisticated mail nickname facility called ALIASES:

```
$ ED SYS$LOGIN:ALIASES.DAT
*C
# Make up some local nicknames
billy           wjgraham
lisa            st61234
# Make nicknames for some network addresses
# (note that you can omit the normal IN%" "):
todd            tjs@gomwortz.pershtec.edu
tom             tstorr@alaska
# Make up some "distribution list" aliases
# (NOTE: you MUST include IN%" " for any
# network addresses that are part of your
# lists! Also note that @filename, if used,
# must come first in your list, AND notice
# that you cannot put spaces in between ids!):
coworkers       jjones,IN%"u1234@brownvm"
phdstudents     bbecker,schu,llmarsh,steveg
myclass         @sys$login:dsc335.dis,bbecker
<ctrl>-Z
*EX
$ ALIASES   <-- (runs automatically at login!)
```

For more information, see:

```
$ HELP UO_Software ALIASES
```

FAXING FROM THE VAX?

We currently have NO ability to send or receive FAXes from the VAX, but note that University of Oregon departments **can** send and receive FAXes by bringing documents to room 3 Johnson Hall.

Inbound FAX: (503) 346-3127 Include a cover page clearly identifying the intended U of O recipient!

USING U-MAIL:

U-Mail is a locally written utility that makes it possible for anyone with a VAX account to compose and send memos to any person with an entry in the campus phone directory.

U-Mail addresses are formed by replacing spaces between first and last names and middle initials with periods. Omit all special characters except periods and dashes. Drop nicknames enclosed in double quotes and titles such as MD or Jr. Capitalization is disregarded.

For example:

Campus Directory Name	U-Mail Name
Public, John Quincy	john.q.public
O'Grady, Mary	mary.ogrady
Fox, J. Pierpont	j.pierpont.fox
Finch-Hatton, Denys	denys.finch-hatton
Welby, Marcus, MD	marcus.welby
Ruth, George "Babe" Herman	george.h.ruth

To initiate a letter to Mary O'Grady, for example, you'd type:

```
$ MAIL
MAIL> S
To: mary.ogrady
Subject: Your enthographic information
Enter your message below. Press CTRL/Z...
```

Offsite users can send U of O U-Mail by using addresses like:

```
mary.ogrady@oregon.uoregon.edu
```

By default, U-Mail is printed with a cover page containing the recipient's name and campus address, and then turned over to Campus Mail for delivery via intracampus mail.

If you want your U-Mail sent directly to your normal VAX mail area or to UF, contact Rich Haller at username RHALLER or x6-4394 to make the necessary arrangements.

USING SEND TO SEND INSTANT MESSAGES:

"SEND" messages appear immediately on your recipient's terminal screen, regardless of what else they may be doing. Use SEND judiciously – many people become startled or irritated when their concentration is interrupted by an unexpected message.

```
$ SEND KROE "Where did you put the car keys?"

$ SEND FXHELP@ALASKA "Is your MAILER dead?"

$ SEND RWILL
    RWILL: Your coffee pot was smoking so I
    RWILL: threw it into the courtyard.... Tom
    RWILL: <return>

$ SET BROAD=NOUSER1     <-- Disable local SEND
$ SET BROAD=NOUSER2    <-- Disable remote SEND

$ FINGER                 <-- Find out who's on
$ FINGER @ALASKA       <-- ... even in Alaska!
$ SEND @HDETUD1 CPQ NAMES    <-- Use CPQ NAMES
                        for non-Internet sites
```

USING PHONE:

PHONE lets you conduct a truly interactive conversation (right down to seeing every "typo" that takes place). PHONE even accommodates "conference calls" with up to 6 parties "online" at once. Note, however, that because PHONE messages have to be handled in "real time" (rather than when they may be conveniently dealt with), MAIL is generally preferred to PHONE for all but the most urgent of messages.

```
$ PHONE               <-- (Use only on VT100's)

% DIR                          <-- See who's on

% TLILLY                       <-- Phone TLILLY

% ANS                         <-- Answer a call

<ctrl>-Z                            <-- Hang up
<ctrl>-Z                  <-- Get out of Phone
```

USING SEND TO SEND FILES:

NOTE: SEND/FILE works only to send files to other BITNET sites! Use MAIL or FTP to move files to users with INTERNET addresses!

To send files to another VAX/VMS account:

```
$ SEND/FILE/VMSDUMP FRIZ.DAT JBENNY   <-- send
                   the file FRIZ.DAT to JBENNY

$ SEND/FILE/VMSDUMP *.* TSTEVENS  <-- send ALL
                 files in this dir to TSTEVENS

$ SEND/FILE/VMSDUMP *.FOR FXABC@ALASKA     <--
                  send all .FOR files to FXABC
                 at Alaska-Fairbanks (a BITNET
                 site running a VAX under VMS)
```

To send files to non-VAX accounts:

```
$ SEND/FILE FROZ.DAT 557@OREGON1      <-- send
                   the file FROZ.DAT to 557 on
                    our IBM 4341 (running CMS)
```

RECEIVE-ING FILES YOU'VE BEEN SENT:

```
$ RECEIVE        <-- See what's in your reader
Files received for JSMITH
Source file          Class     Node User
DUMPIT.SAS;2         PUN N   OREGON LBETTS
RECEIVE> EX                  <-- leave RECEIVE

$ RECEIVE FRY.DAT     <-- receive only FRY.DAT

$ RECEIVE *      <-- receive ALL waiting files

$ RECEIVE/NODELETE  BOIL.DAT   <-- receive the
                 file BOIL.DAT, but don't pull
                            it from the reader

$ RECEIVE
RECEIVE> TYPE STUFF.DAT   <-- look at a reader
RECEIVE> DELETE STUFF.DAT.1      file and then
RECEIVE> EX                          delete it

$ RECEIVE
RECEIVE> HELP      <-- request help in RECEIVE
RECEIVE> EX                  <-- leave RECEIVE
```

SUBSCRIBING TO A LIST USING LISTSERV:

LISTSERVers are automatic mail processors (or "reflectors" that support online electronic discussions organized around a common theme such as ham radio, anthropology, IBM-compatible personal computers, or any of hundreds of other things.

To see what list topics currently exist, look at the current list-of-lists, available in both the Documents Room and the Consulting Office. [We obtain a new copy whenever an updated version is released via anonymous FTP from NIC.DDN.MIL] You can also print your own copy (110+ pp.!) with the command:

```
$ PRINT DISK$USER1:[JOE.CARDS]LIST_INFO.LIS
```

Let's assume that after taking a look at the list-of-lists, you've decided that you'd like to subscribe to the Info-IBMPC list, a list focussing on technical aspects of IBM PC use. Looking at the list-of-lists, you'll see that you join the Info-IBMPC list by issuing the command:

```
$ SEND LISTSERV@POLYGRAF "SUB IBMPC-L TOM DOE"
```

(replace TOM DOE with your real name).

To stop receiving the Info-IBMPC list, you'd enter:

```
$ SEND LISTSERV@POLYGRAF "SIGNOFF IBMPC-L"
```

You'd then be removed from the Info-IBMPC list.

Unfortunately, not all lists are so sophisticated. Some lists, such as the one covering Software Engineering, have yet to be automated and require human intervention in administrivia such as adding and deleting list members. The correct address for requesting such administrative action, and any other information required will always be shown in the current list-of-lists for any non-automated list.

GETTING A LIST OF ONLINE SERVERS:

For a rather comprehensive list of online mail or message-based servers try:

```
$ SEND BITSERV@UCBCMSA "GET BITNET SERVERS"
```

PACKAGING FILES FOR MAIL:

MAIL was really created to handle relatively small amounts of straight ascii text – either messages, or small "flat" data files. Problems begin to arise in many cases when you attempt to use MAIL to send anything else, such as large-ish data files, or binary format files.

So what do you do when you have big files, or files with unprintable characters? Well, the preferred approach is to use FTP to move the file (if it is going to another Internet machine) or to use SEND/FILE/VMSDUMP (if the file isn't TOO big and it is going to another VAX/VMS system that also runs JNET software).

Unfortunately, sometimes the ONLY way you have to move files to another user is by MAIL. For example, perhaps your user is on Compuserve or Fidonet – MAIL is about the only thing you can use to send files to such individuals.

The accepted solution to such problems (when MAIL is the only tool available for moving a file) is to package the file into a form suitable for mail – either by converting unprintable binary characters to printable ones, or by chopping up overly large files into manageable pieces, or both.

The most two most common tools for handling these tasks are VMS_SHAR, and UUENCODE (possibly with the assistance of CHOP).

[**NOTE:** See the Miscellaneous Microcomputer Topics Section later in this book for information on converting common microcomputer files to a MAIL-able plain ASCII format!]

VMS_SHAR:

VMS_SHAR takes one (or more) input text files, and automatically repackages them into one (or more) mailable, check-summed, self-unpacking files each no larger than 31 blocks (i.e., 16000 bytes).

NOTE: Files to be processed by VMS_SHAR can **NOT** include tab characters! Remove tabs using DETABIT (for plain text files) or FPRETTY/NOLIST (for FORTRAN programs written using DEC's special tab formatting)!

To make and ship a VMS_SHAR file, do:

```
$ VMS_SHAR :== @DISK$USER1:[JOE.SHAR]VMS_SHAR
$ VMS_SHAR RAWA.DAT,RAWB.DAT MYSHAR.COM
$ MAIL
MAIL> S MYSHAR.COM
To:   IN%"JSMITH@FOOUVMS"
Subj: MYSHAR.COM
MAIL> EX
```

If you receive a VMS_SHAR file as a mail message:

```
$ MAIL
MAIL> 1                  <-- pull up the message
MAIL> EXTRACT/NOHEADER MYSHAR.COM
MAIL> EXIT
<edit the file to pull out leading garbage>
$ @MYSHAR.COM
```

This will unpack the packaged files. Make SURE you know (and trust) the people you accept files from, and carefully inspect any files you receive before executing them!!! Know your sender and understand what the files you receive do before you execute them!!!

NOTE: If you want to create archive file "chunks" bigger than 31 blocks, copy VMS_SHAR.COM from its current home to your own account, change MAX_PART_SIZE globally in that COM file from 31 to whatever size is desired, and redefine the VMS_SHAR symbol to point at your copy instead of the original.

UUENCODE/UUDECODE:

UUENCODE converts executable files to printable, seven-bit ASCII characters that can be sent via e-mail. UUENCODED files are typically about 35% bigger than the original non-UUENCODED file.

The following example demonstrates how you'd ship a copy of a VMS executable called BIG.EXE to another VAX/VMS system via MAIL:

ENCODE AN OUTBOUND EXE FILE:
```
$ @DISK$USER1:[JOE.UUENCODE]SETUP.COM
$ UUENCODE BIG.EXE BIG.UUE  <-- encode the EXE
$ CHOP BIG.UUE 15K HEADER <-- make 15KB chunks
$ MAIL
MAIL> S BIG.UUE1
To:   IN%"JSMITH@FOOUVMS"
Subj: BIG.UUE1 uuencoded file
MAIL> S BIG.UUE2
To:   IN%"JSMITH@FOOUVMS"
Subj: BIG.UUE2 uuencoded file
<continue for each "chunk" of the archive>
MAIL> EXIT
```

If you receive a UUENCODEd executable file, you'll need to UUDECODE it. Make SURE you know and TRUST whomever you get an executable from since it is virtually impossible to inspect an EXE file to verify a priori that it does ONLY what it claims it does!

DECODE INBOUND UUENCODED EXE FILES:
```
$ @DISK$USER1:[JOE.UUENCODE]SETUP.COM
$ MAIL
MAIL> 1       <-- "read" the uuencoded message
MAIL> EXTRACT/NOHEADER BIG.UUE1
MAIL> <continue till all chunks are extracted>
MAIL> EXIT
<edit files to pull out leading garbage, and
also make sure chunk numbers match filenames;
note that the order files are sent may not be
the same order in which files are received!>
$ APPEND/NEW BIG.UUE1,BIG.UUE2 BIG.UUE
$ UUDECODE BIG.UUE BIG.XMO <-- for binary file
$ XMODTOEXE BIG.XMO BIG.EXE  <-- make the .EXE
```

NOTE: For more information on using UUENCODE, see the TXT files in DISK$USER1:[JOE.UUENCODE]

NETWORK BASICS:

THE INTERNET:

The Internet is an international electronic network that links participating government agencies, educational institutions and commercial entities. It is estimated that the Internet currently has over 60,000 seperate hosts ranging in size from PCs to the largest of Crays.

The backbone of the Internet is comprised of a series of NSF-affiliated regional networks, and represents the current incarnation of what used to be known as ARPANet. Our campus broadband network (UONet) is part of our regional network (NWNet), which is part of NSFNet and thus the Internet.

Users of the Internet can send mail messages, get and put files via FTP (file transfer protocol), and even login to remote systems just as if they were physically present at the remote site.

For a partial list of some major Internet sites, see: ANONYMOUS_DIR:HOSTS.INTERNET

University of Oregon UOnet Internet hosts are listed in: ANONYMOUS_DIR:HOSTS.UO

OTHER NETWORKS WE'RE CONNECTED TO:

BITNET: "Because Its Time" Network. Bitnet is a national and international electronic network that supports electronic mail and file transfer traffic on a store-and-forward basis: mail hops from host to host, being sent onward as resources become available at each machine. Note that BITNET-only sites can **not** do remote login or use FTP to handle file transfer. For a list of BITNET sites, see: ANONYMOUS_DIR:HOSTS.BITNET

DECNET: DECNET interconnects some of the VAXes here at the U of O. For a list of current hosts, see ANONYMOUS_DIR:HOSTS.DECNET

WHICH NET AM I WORKING WITH?

A common question is "Which network am I using? I got a mail message from someone in Washington, and I can't tell if it is a BITNET message or what!"

Well, first of all, many times it won't really matter to you which network you're using if all you're doing is replying to some mail. However, if you really DO need to know which network a person is using, sometimes their address will provide a clue. For instance:

ADDRESS EMPLOYED:	NETWORK USED:
node::user	DECNet
user@site	BITNET (usually)
user@site.subdom.dom	Internet (usually)
siteb!sitea!user@sitec	UUCP (gatewayed)

Of course, if an address explicitly ends with .BITNET or .UUCP, that's what it is.

NOTE: At times it may be difficult or impossible to tell whether a site is on BITNET or the Internet due to two factors:

(a) BITNET is adopting "Internet-like" domain-style addresses, and so what looks like an Internet address may actually be a "new style" BITNET address

(b) Local Internet sites may give you an abbreviated (non-fully qualified address) and this abbreviated address may thus look like a "short" eight character BITNET address.

In truly tough cases when you really need to know what network a user is on, you may need to explicitly ask that user for the information.

We regret any inconvenience or confusion this unavoidable situation may cause.

COMPARING NETS: WHAT WORKS WHERE?

Many users are confused about what can been done via the various networks we're connected to... Here's a concise summary:

SERVICE	INTERNET	BITNET	DECNET
Electronic Mail	Yes	Yes	Yes
TELNET (Remote Login)	Yes	No	No
SET HOST	No	No	Yes
SEND (On Screen Msgs)	No	Yes	No
PHONE	No	No	Yes
FTP (File Transfers)	Yes	No	No
SEND/FILE	No	Yes	No
VMS Inter-Node COPY	No	No	Yes
FINGER (See Who's On)	Yes	No	No
CPQ NAMES	No	Yes	No
PHONE DIR node::	No	No	Yes
VMS Inter-Node PRINT	No	No	Yes
lpd Inter-Site Print	Yes	No	No
Shared VMS Files	No	No	Yes
Shared nfs Files	Yes	no	No

NOTE: lpd print sharing and nfs file sharing aren't currently generally available under VMS.

FINDING INTERNET ADDRESSES:

Each site on the Internet is identified by an Internet Protocol number and address. For instance, the University of Oregon's VAX 8800 system's are:

```
OREGON.UOREGON.EDU or  }
128.223.20.2 or        } = VAX 8800
128.223.32.18          }
```

A question that naturally arises is "How do I find out **some other system's** Internet address?" There are a couple of different approaches you can try. The easiest way is to use the VAX SEARCH command to look for that site:

```
$ SEARCH ANONYMOUS_DIR:*.* "HARVARD"
```

You can also use the NSLOOKUP program. For example, assume you wanted to know the names of machines in the HARVARD.EDU domain. You could use the commands:

```
$ NSLOOKUP
> SET TIMEOUT=100
> SERVER HARVARD.EDU
> LS HARVARD.EDU
> <CTRL-Z>
```

A third alternative is to TELNET to NIC.DDN.MIL and then use the WHOIS command to look for a match.

Finally, for really tough net addresses you may want to try mailing an inquiry to the INFO-NETS list. See the LIST-OF-LISTS for more information on getting started with INFO-NETS. Note that INFO-NETS is **NOT** the right place to try to get the **username** of a particular user at a remote site! (See, instead, the earlier panel on FINDING A PERSON'S MAIL USERNAME)

USING TELNET:

TELNET allows you to directly login to a remote computer system just as if you were at that site.

For example, let's assume that you want to logon to the University of Alaska's Statewide Assembly Bulletin Board account (this account was chosen because anyone is allowed to login to it without needing to know a password):

```
$ TELNET ACAD3.FAI.ALASKA.EDU      <-- Alaska!
Login: SYSGABB
<try Alaska's Bulletin Board if you want to>
<logoff the Bulletin Board account when done>
telnet>QUIT                      <-- Leave TELNET
```

TN3270:

TN3270 is designed to provide software emulation of IBM 3270 terminals. This emulation is required for access to some IBM-based sites that don't run Series/1 or 7171 ASCII terminal controlers.

For example, try TN3270 on Rice's library:

```
$ DEF TERM "vt100"          <-- Note lowercase!
$ TN3270 LIBRARY.RICE.EDU     <-- Note TN3270!
<wait for initial screen>
<hit tab twice to get to COMMAND ===> )
DIAL LIBRIS           <-- connect to the catalog
<CRTL>-Z           <-- clear the screen to start
```

TN3270 Keymapping Information for VT100:

```
PF1-PF9:     <KP 1>      to  <KP 9>
PF10-PF19:   <PF1> <KP0> to  <PF1> <KP9>
PF20-PF29:   <PF2> <KP0> to  <PF2> <KP9>
PA1-PA3:     <CTRL>-P 1  to  <CTRL>-P 3
CLEAR:       <CTRL>-Z

                    <CTRL>-K
ARROWS:     <CTRL>-H    <CTRL>-L
                    <CTRL>-J
```

FILE TRANSFER PROTOCOL:

USING FTP:

FTP (File Transfer Protocol) is a facility that will let you use the VAX to retrieve files from remote computer systems.

To get at files on a remote system via FTP, you need a username on the remote system, or the remote system needs to have created a special anonymous "guest" account.

One popular system that has anonymous FTP is the Kermit communication package support machine at Columbia in New York City. To log onto the Kermit machine at Columbia, enter:

```
$ FTP WATSUN.CC.COLUMBIA.EDU     <-- Columbia!
*LOGIN anonymous         <-- Login as anonymous
                          ... note lowercase!!!
*Password: JSMITH        <-- Send your username
```

At that point you're ready retrieve a file:

```
*CD kermit          <-- Move into the kermit dir
*DIR               <-- What files are out there?
*GET read.me                     <-- Get a file
*CLOSE                  <-- Close the connection
*QUIT               <-- Exit FTP -- You're done!
```

The file read.me is now in your VAX directory!

To just type a file to your screen, use:

```
*GET filename TT:    <-- "get" to the terminal
```

To get binary format files:

```
*BINARY                      <-- set binary mode
*QUOTE "TYPE L 8"   <-- demand eight bit words
*GET filename                   <-- get the file
```

When you're done transfering binaries, use:

```
*ASCII
```

to change back to text mode.

MORE FTP:

To get multiple files with one command use:

```
*MGET XYZ.*
```

This initiates retrieval of all files called XYZ.anything in the current remote directory. Reply with a **lowercase** y to accept each file when asked. If you want to just accept everything without the lowercase y's being needed, say: NOINTERACTIVE <RETURN> before issuing the MGET

To get a marker each time FTP transfers a packet (so you can tell that your transfer is actually taking place) say HASH <RETURN> before entering your GET or PUT command...

Automating your FTP login process: If you routinely FTP to a particular host, you may want to create a file called FTPLOGIN. in your root directory to "semiautomate" this process. Each line in this file should contain the IP address of the system you're connecting to, your username on that system, a dash (representing your password) and an optional account number (if used). **NEVER** replace the dash representing your password with a REAL password!

WHEN DO I "GET" AND WHEN DO I "PUT"?

Realize that VAX FTPing can be done with the VAX acting as either a CLIENT or as a SERVER.

If you start FTP on OREGON, and thereby connect from it to the other remote system, OREGON will be running an FTP CLIENT. You'll GET files from the other system to OREGON (or, conversely, you'll PUT them from OREGON to the other system).

If you start FTP on the other system and connect from it back to OREGON, OREGON will be running an FTP SERVER. You'll PUT files from the other system to OREGON (or, conversely, you'll GET them on the other system from OREGON).

NATIONWIDE LIST OF ANONYMOUS FTP SITES:

```
$ PRINT DISK$USER1:[JOE.CARDS]FTP.SITES
```

NOTE: To obtain the absolutely most recent edition of this file, retrieve (via anonymous FTP, of course!) the file ftp.list.Z from the pub/unix-misc directory on ssyx.ucsc.edu and then UNCOMPRESS it

---

SOME POPULAR FTP SITES:

New anonymous FTP users often ask "What are the most popular anonymous FTP sites?" Five of the most popular have to be:

WSMR-SIMTEL20.ARMY.MIL: This is the classic/authoritative PC (and other) software archive.

SUMEX-AIM.STANFORD.EDU: SUMEX is the holy grail of Macintosh software archives.

WUARCHIVE.WUSTL.EDU: WUARCHIVE provides an extensive PC software collection (including many of the files from SIMTEL20); WUARCHIVE also has taken the progressive step of making their collection NFS mountable from other Unix systems.

UUNET.UU.NET: This is a privately-owned, public-access Unix system run on the East Coast. Network access (even from here at Oregon) is consistently fast, the archives are well organized and holdings are extensive. Highly recommended.

GATEKEEPER.DEC.COM: The maintainers of GATEKEEPER themselves said it well: "The ... archive is full of nearly every bit of useful public domain software which has ever been made available on the Internet..." Well run. An excellent resource.

GETTING SIMTEL20 FILES VIA FTP:

To retrieve files from SIMTEL20 via FTP, begin by logging onto SIMTEL20 and identifying the files you want to retrieve:

```
$ FTP WSMR-SIMTEL20.ARMY.MIL         <-- Logon
ANONYMOUS
<enter your username (or whatever) here>
*DIR               <-- Look at the default dir
*GET SIMTEL20-MSDOS.DIRLIST  TT:   <-- look at
                       MSDOS directory listing
*CD PD1:<MSDOS.SURFMODL>         <-- change to
                      subdir (ignore password)
*DIR               <-- Look at SURFMODL subdir
```

Note that most SIMTEL20 files are compressed to minimize file space used, and to reduce the size of the files you retrieve. To properly receive .ARC and similar binary files enter:

```
*BINARY      <-- announce the files are binary
*QUOTE "TYPE L 8"     <-- set word length to 8
```

You can then retrieve the files you want. For example, for SURFMODL we'd use:

```
*NOINTERACTIVE
*MGET SURFMODL20*.ARC    <-- get several files
```

You can then close the connection:

```
*CLOSE
```

and download your files to your PC:

```
$ KERMIT                   <-- Start VAX Kermit
KERMIT-32> SET FILE TYPE BINARY   <-- Set mode
KERMIT-32> SEND *.ARC     <-- Initiate sending
<CTRL>-] C             <-- Escape to PC Kermit
KERMIT-MS> RECEIVE C:\    <-- Accept the files
```

Now uncompress the files. (See the next panel for details).

NOTE: It may be difficult or impossible to FTP to SIMTEL20 during the day – plan on working on SIMTEL20 at night or on weekends.

UNCOMPRESSING ARC FILES:

You can uncompress files on the VAX or PC, however it is probably most efficient to do it on the PC after you've downloaded them, thereby simultaneously reducing your download time and avoiding VAX CPU decompression charges.

DECOMPRESSING ARC FILES ON THE PC: To uncompress .ARC files on your PC, use Phil Katz's PK361.EXE program. If you don't already have a copy, you can get a self-unpacking one by anonymous FTP from DECOY.UOREGON.EDU Don't forget to retrieve it in BINARY mode! After you retrieve it, put it in a directory by itself and type:

```
PK361
```

PK361 will then uncompress itself. Once you've gotten and run that file, you can then use the command:

```
PKUNPAK FOO.ARC
```

to uncompress all files in FOO.ARC

DECOMPRESSING ARC FILES ON THE VAX:

To decompress ARC files on the VAX, use the commands:

```
$ SCRATCH              <-- request scratch space

$ SWEEP :== $DISK$USER1:[JOE.SWEEP]SWEEP   <--
                        define the SWEEP command

$ SWEEP FOO.ARC     <-- start SWEEPing FOO.ARC
E                            <-- Extract files
9999                      <-- choose all files

$ COPY SYS$SCRATCH:*.*  SYS$LOGIN:*.*      <--
          move files from scratch to perm disk
```

Note that it will be important to download any executable files in BINARY mode once you've gotten them decompressed.

UNCOMPRESSING TAR-Z FILES:

Files created on (or intended for) UNIX-based systems are often stored in compressed TAR-Z format.

The easiest way to decompress/de-TAR those files is to use a UNIX system.

One UNIX system at the Computing Center is DECOY, a NeXT workstation running MACH, (a flavor of BSD 4.3 Unix). For information on getting an account on DECOY, send a VAX Mail message to username JOE (JOE@OREGON) from your OREGON VAX account. You'll get a mail message back when your account has been created.

After you get your DECOY username and password, immediately telnet to decoy and change your password; you can then proceed to do your file decompression work:

```
$ telnet decoy      <-- Go to DECOY.UOREGON.EDU
login: jsmith     <-- Enter your DECOY username
Password:     <-- Enter initial DECOY password

1 % passwd                  <-- Change password
2 % ftp oregon                     <-- Goto VAX
login: jsmith   <-- Enter your OREGON username
Password:       <-- Enter your OREGON password
*binary     <-- Set binary file transfer mode
*get xyz.tar-Z  <-- Send the file from the VAX
*quit                              <-- Exit FTP

3 % mv xyz.tar-Z xyz.tar.Z     <-- Rename file
                    so the file is something.Z
4 % uncompress xyz.tar       <-- Uncompress it
5 % tar -xvf xyz.tar                <-- Untar it
6 % ls -l           <-- Look at your directory

7 % ftp oregon                     <-- Goto VAX
login: jsmith   <-- Enter your OREGON username
Password:       <-- Enter your OREGON password
*put xyz.dat       <-- Put the file to the VAX

*quit                              <-- Exit FTP
8 % logout                    <-- logout DECOY
```

Remember that UNIX is CaSe SeNsItIvE!

UNIX SHAR FILES:

In addition to seeing tar.Z files, another common way to "package" files on a UNIX system is the UNIX shar file. If you need to work with shar files, you'll need to request an account on DECOY as explained previously.

Once you get your DECOY account and have moved your shar file there, use vi (or another editor) to insure that there isn't any "trash" at the top of the file (such as mail message headers, informational notes, etc.) – generally speaking, the first "real" line of a shar file should be ! /bin/sh

You can then unshar the file by saying:

```
% sh sharfilename
```

The file(s) in the shar archive will then be unloading into your current directory.

OTHER PC COMPRESSION SCHEMES:

While Phil Katz's PKPAK/PKUNPAK has been the "standard" PC file compression utility (using the .ARC file extension), you may also encounter a number of other schemes. Identify them by their distinctive file extensions. PC file decompression programs for a number of them can be retrieved via anonymous FTP from DECOY.UOREGON.EDU (in binary mode):

```
.ARC -- Use PK361.EXE (self-unpacking)
.ZIP -- Use PKZ092.EXE (self-unpacking)
.ZOO -- Use ZOO201.EXE (self-unpacking)
.LZS -- Use LARC333.EXE (self-unpacking)
.LBR -- Use LU.ARC (unpack with PKUNPAK)
.?Q? -- Use SQ.EXE and USQ.EXE (ready to go)
```

MACINTOSH COMPRESSION SCHEMES:

Most Mac users employ StuffIt and/or BinHex, both of which are available from the Mac BCS CD-ROM disk in Room 207 Computing Center.

NEWS:

WHAT IS NEWS?

USENET NEWS is a unique information sharing resource quite distinct from mail (and from mailing lists or LISTSERVers).

The best way to understand NEWS is to consider what happens when you send mail to a "conventional" mailing list for re-distribution: essentially, an individual copy of your mail message is re-sent to every list subscriber.

This means that if a given site has "n" subscribers to a particular list, that site will receive (and the network will carry) "n" mail messages every time someone posts something to the list.

NEWS uses a different philosophy to disseminating information. NEWS directly attacks the question, "Why should a site (potentially) receive a dozen identical copies of a given message? Why not provide a single central copy of the message, and let everybody simply look at that local 'master' copy, instead?"

This may sound a lot like the microcomputer bulletin boards you may be familiar with, and it is quite similar, but NEWS has a unique twist. Messages posted via NEWS are generally transparently "propagated" to other NEWS sites across the United States (and sometimes overseas).

Because of the diverse nature of individual NEWS users interests, postings within NEWS are organized hierarchically by subject. Local NEWS administrators determine which newsgroups are carried, and sometimes a local NEWS administrator mmay elect not to carry a particular group that is of little interest to local users. On the other hand, a site may also create "local-only" newsgroups such as the "UO." newgroup hierarchy here at the University of Oregon; because those newsgroups deal with strictly local topics, they are not propagated to other NEWS sites.

WHAT'S BETTER: NEWS OR MAILING LISTS?

Many users want to know what's "better" – NEWS or conventional mailing lists. In fact, each has advantages and disadvantages.

In some cases, you may not have a choice. A given topic may only be available as either a mailing list or as a newsgroup, but not both. However, in some cases, you'll have a choice between joining a mailing list or following a NEWS newsgroup. When you have that choice, consider the following points:

When you subscribe to a regular mailing list, message traffic from that list goes to your mail box along with all other mail you receive. Some people find this convenient, enjoying the fact that they can "read everything from within mail." Other people find the high volume of mail from some lists may make them more prone toward overlooking important personal mail messages.

You should also know that reading NEWS on the VAX can be expensive due to the large amount of CPU time the NEWS reader uses. In fact, reading NEWS is essentially impracticable from "communications only" accounts due to their 20 CPU second/session limit!

Moreover, NEWS messages aren't kept around forever – messages are deleted on a rolling basis, usually after a couple of days. Thus, if you're gone for a couple of weeks of vacation, you'll probably miss some postings of interest. On the other hand, mail messages from a "conventional" mailing list will continue to pile up while you're gone, and you may return to find yourself facing a hundred (or more) unread messages from a mailing list.

Finally, if you're just trying to decide what newsgroups or mailing lists you'd like to follow, it is a lot easier to read a few messages from a bunch of different newsgroups than it is to get added (and then often deleted) from a bunch of manually-maintained mailing lists.

READING USENET AND OTHER NEWS:

A number of USENET and other newsgroups are accessible to VAX users via a full screen (VT100) news reader. Invoke NEWS by saying:

```
$ @DISK$USER1:[JOE]NEWS_EDIT       <-- use EDT!
$ NEWS                            <-- Start NEWS
```

Use your cursor keys to move down to a topic of interest, then hit return to read the messages available under that topic. (You can also type a number to move to that number)

```
SOME COMMON NEWS COMMANDS

NEWSGROUPS             <-- See newsgroup listing
DIR                    <-- See message directory
EXIT                              <-- Leave NEWS
PRINT                         <-- Print a message
EXTRACT              <-- Put a message in a file
FORWARD               <-- Send a message in mail
REPLY               <-- Reply only to originator
POST               <-- Post a message NATIONWIDE
FOLLOWUP                  <-- Comment NATIONWIDE
REGISTER COMP.SYS.NEXT      <-- Join that group
DEREGISTER COMP.SYS.NEXT   <-- Remove yourself
```

For more information on using NEWS, go into news and read the BEGIN news group.

NOTE: Before you post something to a NEWS newsgroup, be aware that it will be posted to all other NEWS sites **nationwide.** Think carefully before you make a posting!

Just to give you an idea of the breadth of the substantive topics available, you'll find newsgroups on everything from a discussion of the VMS operating system (COMP.OS.VMS) to a discussion of the Chinese political system (SOC.CULTURE.CHINA) to news about FUSION (SCI.PHYSICS.FUSION) to hot job tips (MISC.JOBS.OFFERED) to a general potpourri in the MISC.MISC and JUNK newsgroups.

Some newsgroups you may be familiar with from other NEWS sites may not be present in the Oregon NEWS newsgroup listings.

NEWS KEYPAD MAPPING:

```
-----------------------------------------
|PF1      |PF2      |PF3      |PF4      |
|         |         |         |SKIP/NEWS|
|   gold  |  HELP   |   SKIP  |         |
|         |         |         |skip/foll|
|         |         |         |         |
-----------------------------------------
|7        |8        |9        |-        |
|   DIR   | DIR/NEW | DIR/REG | DIR/ALL |
|         |         |         |         |
|  print  | extract |extra/all|extra/app|
|         |         |         |         |
-----------------------------------------
|4        |5        |6        |,        |
|         |    UP   |         |         |
|   DIR   |         | DIR/REG | DIR/ALL |
|         |   top   |         |         |
|         |         |         |         |
-----------------------------------------
|1        |2        |3        |ENTER    |
|  READ   |  DOWN   |READ/NEXT|         |
|         |         |         |         |
|read/head| bottom  |read/next|         |
|         |         |    /head|         |
------------------------------|         |
|0                  |.        |         |
|     READ/NEW      |READ/NEW/|         |
|                   | FOLLOWUP|         |
|  read/new/header  |read/new/|         |
|                   |foll/head|         |
-----------------------------------------
```

USING THE GNUS NETWORK NEWS READER:

While the VAX-based ANU NEWS news reader will be used by most OREGON users use to read NEWS, users with access to a UNIX workstation running GNU EMACS can also read NEWS using the GNUS NNTP-based news reader.

The biggest advantage of GNUS (over ANU NEWS) is that it can be "pointed" at virtually any cooperating NEWS host, thereby making it possible to read some newsgroups carried on other hosts but not currently available on OREGON

We assume your .EMACS file looks like:

```
(setq load-path
(cons
(expand-file-name "<your system's gnus dir>")
load-path))
(setq gnus-nntp-service 119)
(setq gnus-nntp-server "<NNTP srvr address>")
(setq gnus-your-domain "uoregon.edu")
(setq gnus-default-article-saver
   'gnus-Subject-save-in-file)
(setq gnus-novice-user "yes")
(autoload 'gnus "gnus" "Read network news." t)
(autoload 'gnus-post-news "gnus"
   "Post a news." t)
(put 'eval-expression 'disabled nil)
```

STARTING GNUS:

```
% emacs <RETURN>             <-- Launch gnu emacs
<ESC><ESC> (gnus) <RETURN>     <-- Launch gnus
```

SUBSCRIBING/UNSUBSCRIBING TO GROUPS:

```
[if in art. buff. hit = to goto subject buff.]
[if in sub. buff. hit q to goto newsgrp buff.]
L          <-- request full list of newsgroups
[use arrows to move to group to be changed]
u      <-- unsubscribe/subscribe to that group
l   <-- display list of subscribed groups only
```

EXITING GNUS:

```
[if in art. buff. hit = to goto subject buff.]
[if in sub. buff. hit q to goto newsgrp buff.]
q              <-- quit gnus from newsgrp buff
y    <-- confirm you really want to exit GNUS
<CTRL-x><CTRL-c>       <-- exit from gnu emacs
```

GNUS NEWS READER (CONTINUED):

READING ARTICLES IN A NEWSGROUP:

```
[if in art. buff. hit = to goto subject buff.]
[if in sub. buff. hit q to goto newsgrp buff.]
[use arrows to move to the group to be read]
=                     <-- select this newsgroup
[use arrows to move to article to be read]
<SPACE>                   <-- read this article
<SPACE>             <-- scroll forward by pages
<DEL>              <-- scroll backward by pages
=                 <-- return to list of articles
q               <-- return to list of newsgroups
```

MAKING A POSTING:

```
[if in art. buff. hit = to goto subject buff.]
[if in sub. buff. hit q to goto newsgrp buff.]
[goto grp. you'll be posting to using arrows]
=                         <-- select that group
a        <-- add a posting to the group OR...
f     <-- followup (reply) to the group OR...
r             <-- reply only to article author
[enter meaningful subject line] <RETURN>
<RETURN>       <-- accept default distribution
[enter your message text]
<CTRL-c><CTRL-c>                  <-- POST it!
<CTRL-x> k        <-- OR... abort the posting!
<CTRL-x> 0   <-- (if needed, zap post. buffer)
```

SAVING AN ARTICLE TO A FLAT FILE:

```
[assume you're reading the art. to be saved]
o                       <-- do output to a file
foo.txt <return>        <-- provide a file name
=                         <-- select that group
```

SORTING YOUR NEWSGROUPS:

```
% sort .newsrc >xyz; cp xyz .newsrc; rm xyz
```

SOME OTHER GNUS COMMANDS:

```
BUFFER? COMMAND?  DOES ----------------------
any     <CTRL-l>  Redraw screen (l=el, not 1!)
any     <CTRL-g>  Abort current action
newsgrp C         Catchup (mark all as "read")
newsgrp <CTRL-v>  Forward one page in listing
newsgrp <ESC>-v   Backward one page in listing
newsgrp U grp <RET>  Subscribe to group "grp"
newsgrp j grp <RET>  Jump to the group "grp"
subject J num <RET>  Jump to the article "num"
```

GENERAL INFORMATION ONLINE SYSTEMS

ALASKA'S GENERAL ASSEMBLY:

SYSGABB offers information about the University of Alaska's General Assembly.

```
$ TELNET ACAD3.FAI.ALASKA.EDU
Login: SYSGABB
```

CASE WESTERN RESERVE'S FREENET SYSTEM:

FreeNet is an "experiment in community computing" that's best understood by trying it.

Resources on FreeNet are rather uniquely organized according to a "town square" metaphor. Thus, for instance, you might select "The Arts Building" to see information about the literary arts.

While much of the information on FreeNet oriented toward local Cleveland activities, FreeNet does have copies of the U.S. Constitution and other important historical documents online, among other unusual features.

```
$ TELNET CWNS5.INS.CWRU.EDU
login: fnguest            <-- must be lowercase
```

University of Oregon users who'd like a copy of the FreeNet registration form can pick one up from Joe St Sauver if they'd rather not have to wait while one is sent from Cleveland.

DELAWARE'S OCEAN NETWORK INFO CENTER:

OCEANIC provides information of interest to those involved in oceanographic work. TEK4010 output cam be sent for some map data if your terminal or terminal emulator can display it.

```
$ TELNET DELOCN.UDEL.EDU
Username: INFO
```

MARYLAND'S AIM SYSTEM:

The University of Maryland's AIM system provides information about the University of Maryland that would be of interest to those considering studying at Maryland or visiting College Park.

```
$ TELNET AIM.UMD.EDU
```

MICHIGAN GEOGRAPHIC NAME SERVER:

This resource provides online info about known geographic locations. Among the information provided is population, altitude, latitude and longitude, zip code and area code.

```
$ TELNET MARTINI.EECS.UMICH.EDU 3000 <-- NOTE:
         You must explicitly specify port 3000
```

PENN STATE'S BULLETIN BOARD:

Much like Maryland's AIM system, Penn State's EBB system provides current information about activities at Penn State.

```
$ TELNET CAC3270.PSU.EDU
VT100 <RETURN>
<RETURN>        <-- Just hit <RETURN> at login
EBB <RETURN>
```

PENN STATE'S PENpages:

A database of agricultural information maintained by the PA Department of Agriculture is available via:

```
$ TELNET PSUPEN.PSU.EDU
Username: PNOTPA
```

PRINCETON'S PNN SYSTEM:

PNN (Princeton News Network) is an excellent example of how a computing center can support computing aimed at non-technical users. For instance, PNN provides the current Princeton weather information, train time schedules, a quote of the day, sports info, police blotter reports, etc.) in as intuitive a manner as possible. [See Howard Strauss's article "University Wide General Interest On Line Information Systems that Work – And that You Can Afford," in ACM SIGUCCS User Services Conference XVII Proceedings (1989)]

```
$ DEFINE TERM "vt100"
$ TN3270 PUCC.PRINCETON.EDU
<CTRL-Z>
PNN
```

TIME SERVERS:

The National Bureau of Standards (now "NIST") offers a high precision Automated Computer Time Service (ACTS) from its Boulder, CO, Time and Frequency Division.

BY MODEM: The normal access route to the ACTS service is by modem, dialing into 303-494-4774 at 1200 or 300 baud. Without special software, accuracy of the time signal over land-lines is about 0.1 second at 1200 baud, although use of special delay-correcting software at 300 baud can reduce this error to about 1 millisecond.

For information on obtaining a copy of the delay-correcting software for modem access to ACTS, contact Office of Standard Reference Materials, NIST, Gaithersburg, MD 20899. A 5 1/4" PC diskette with the ACTS delay-correcting software (including documentation, executables and C language sources) currently costs $35.00 [Or, try anonymous FTP to WSMR-SIMTEL20.ARMY.MIL – look in PD1:<MSDOS.SYSUTL> for NBSCOM12.ARC and NBSTIME.ARC]

FROM THE NETWORK: Internet users can also get time information [on a test/trial basis] by using the the "rtime" protocol server on INDIA.COLORADO.EDU

Note that INDIA.COLORADO.EDU's clock is anchored to ACTS, and reportedly is accurate to some tens of milliseconds when accessed from comparatively adjacent Internet hosts.

[If you don't have an rtime client available, you can try TELNETing to port 13 on that machine for ASCII-format time information.]

BY RADIO: Users with shortwave radios can obtain high precision continuous government time broadcasts by tuning to 5.000, 10.000, or 15.000 MHz in the high frequency bands.

BY (VOICE) PHONE: You can also access the Naval Observatory's master clock voice time report by calling: 202-653-1800

UNIDATA'S DENVER WEATHER FEED:

This application illustrates how sites can make selected bits of information (such as the current Denver weather) available to a client base.

```
$ FINGER WEATHER@UNIDATA.UCAR.EDU
```

Unfortunately, Eugene and other more locally pertinent weather information isn't currently available online. However, you can dial 503-484-1200 for a KUGN [voice] weather report, or you can monitor 162.400 MHz with a scanner to receive official NOAA weather info.

[If you're travelling you may want to check (in this order) 162.550, 162.475, 162.425, 162.450, 162.500, or 162.525 MHz if you can't get the weather info you need on 162.400! You can also request a brochure entitled "NOAA Weather Radio" that describes all 380 Weather Radio stations nationwide from the National Weather Service (Attn: W/OM11), NOAA, Silver Spring, MD 20910.]

WEBSTER: DICTIONARY AND THESAURUS:

The Webster Server provides access to NeXT's Webster online dictionary and thesaurus.

```
$ TELNET DECOY.UOREGON.EDU 2627      <-- NOTE:
         You must explicitly specify port 2627

D cannon                     <-- define CANNON
EN ski  <-- Find words starting SKI [anything]
I D     <-- Dictionary entries only (default)
I T                <-- Thesaurus entries only
I DF      <-- See ANY dictionary matches only
HELP                       <-- get online help
EX                      <-- get out of Webster
```

NOTE: Due to the relatively primitive interface being used, you will NOT be able to backspace to correct typos – simply re-enter the command instead.

"MAGIC" TELNET PORTS:

Because a number of users have asked why it is necessary to specifically ask for port 2627 when using the online dictionary available on DECOY, the following brief discussion of special TELNET ports is offered.

By consensus, a number of special TELNET ports are reserved on TCP/IP hosts to provide certain special services (usually as defined in RFC's available via anonymous FTP from NIC.DDN.MIL).

On our VAX, these special ports are documented in:

```
TWG$COMMON:[NETDIST.ETC]SERVICES.
```

while on most UNIX hosts these special ports are listed in the /etc/services file.

Although these special ports form the basis for some fundamental network tasks (such as file transfer, remote interactive login, network Usenet NEWS propagation, and electronic mail message transfer), they also are often used to provide some rather esoteric services. For example, one such service does nothing but echo, character for character, everything you type:

```
$ DEFINE TELNET_ESCAPE "|"
$ TELNET DECOY 7
Blah foo thud
|Q                              <-- close and quit
```

In addition to special port numbers reserved network wide for specific services, system administrators have the ability to create additional locally assigned port numbers. This is what was done to establish the dictionary server maintained on DECOY.

Generally speaking, however, the "magic" port numbers need not be known or remembered because easy-to-remember commands are created to hide the hard-to-remember port numbers for commonly used services.

ONLINE LIBRARY CATALOGS:

ACCESSING THE LIBRARY'S JANUS SYSTEM:

You can search the holdings of the University of Oregon Library by TELNETing to the Library's online catalog, JANUS.

NOTE: At this time public JANUS access is limited to eight simultaneous sessions, so you may encounter problems getting connected (you'll be told that your login is "denied" if too many other public users are currently connected). If you encounter this problem, you may want to contact Rick Gates, the JANUS system librarian/manager, at (503) 346-3049 (username: RGATES@OREGON) to let him know you're having trouble getting connected.

NOTE: You must be on a VT100, Wyse, or TVI925 terminal (or terminal emulator) to be able to use JANUS!

```
$ TELNET JANUS2.UOREGON.EDU   <-- get on JANUS

login: janus    <-- "janus" must be lowercase!
<return> <return> <return>   <-- wake up JANUS

What kind of terminal are you using?
v        <-- specify VT100 (must be lowercase)
y             <-- confirm your terminal choice
```

JANUS is menu driven from there on out

For help getting going in JANUS, consider taking a look at the Information screen (which you can select from the main Janus menu).

One last hint: if you need to "back out" of a bad choice (such as requesting an author search when you wanted to do a subject search), hit the <ESC> key!

NOTES ON DISTANT LIBRARY CATALOG ACCESS

WHY ONLINE LIBRARY CATALOGS? In working on a dissertation or major article, there may be times when you'd like to access research library resources other than those available at the University of Oregon. In the old days, this might have meant travelling to Berkeley or L.A., but now many major university library catalogs are available online. You can search these remote sites as easily as you can search JANUS.

BORROWING BOOKS YOU FIND: If you do find a particular book you'd like to borrow from a remote site, however, you will still need to deal with the Inter-Library Loan (ILL) people at the University Library to make the necessary arrangements. Online library catalogs are a way for you to **search** for books, not a way to directly **borrow** them!

ONLINE RESOURCES: Note that some libraries **do** also offer online resources that you **can** use directly, though, such as encyclopedias or collections of technical documents, but be aware that the library catalogs described below are **NOT** the same as propriety commercial databases such as those offered by DIALOG or NEXIS.

PATIENCE, PLEASE: Be aware that access to distant online library catalogs may at times require some patience since response times increase as the network gets heavily loaded. Response is usually quite good at night or on weekends, and worst in the late afternoon.

THE TRICKS OF THE TRADE: In the following panels, detailed information on getting started or getting disconnected from a particular remote library is only shown when doing so tends to be "complicated." Most libraries use self-explanatory menus, but sometimes getting started or getting disconnected can require some special tricks. Generally speaking, the libraries that have the **shortest** descriptions are the ones **easiest** to use.

ALASKA'S GNOSIS SYSTEM:

NOTE: GNOSIS access is available via TYMNET only! If you're using a modem you can directly dial (503) 343-0044 for Eugene public (2400/1200/300 baud) TYMNET access.

```
Enter terminal identifier: A
Please login: GNOSIS-USA
AKNet: Please login: GNOSIS-USA
AKNet: Password: 466747-872
<RETURN><RETURN>
:HELLO JSMITH,USER.GNOSIS

TO LOGOFF:
/QUIT
<CTRL>-Z
```

BOSTON U'S TOMUS ONLINE CATALOG:

```
$ TELNET LIBRARY.BU.EDU      <-- Connect to BU
```

BROWN U'S ONLINE CATALOG: JOSIAH

```
$ DEF TERM "vt100"      <-- must be lowercase!
$ TN3270 128.148.128.41       <-- note TN3270!
<wait for initial screen>
COMP                    <-- select the library
JOSIAH                            <-- Username
JOSIAH                            <-- Password
```

NOTE: If you appear to get stuck at logoff (after using the LOGOFF option provided), use:

```
<ctrl>-G <ctrl>-C
quit
```

CALIFORNIA-BERKELEY'S CATALOG: GLADIS

```
$ TELNET 128.32.159.19
```

CALIFORNIA SYSTEM'S CATALOG: MELVYL

NOTE: MELVYL requires that your terminal or PC to be set for LOCAL ECHO or HALF DUPLEX. If you run MELVYL without properly reseting your terminal, you won't see anything you type in! For example:

```
C:> KERMIT                        <-- Start Up Kermit
<...logon as you normally would...>
<ctrl>-] C                       <-- Escape to Kermit
Kermit-MS> SET LOCAL ON            <-- Set Local Echo
Kermit-MS> C                            <-- Reconnect

$ TELNET 31.1.0.1           <-- Connect to MELVYL
     (alternative IP numbers: MELVYL.UCOP.EDU,
          31.0.0.11, 31.0.0.13, and 31.1.0.11)
TERMINAL? VT100     <-- declare your term type
```

CARNEGIE MELLON UNIVERSITY LIBRARY:

```
$ DEF TERM "vt100"      <-- MUST be lowercase!
$ TN3270 VMA.CC.CMU.EDU 1   <-- MUST specify 1
```

UNIVERSITY OF CHICAGO LIBRARY:

```
$ TELNET OLORIN.UCHICAGO.EDU
enter class  lib48
```

COLORADO RESEARCH LIBRARIES: CARL

```
$ TELNET PAC.CARL.ORG
```

COLUMBIA'S ONLINE CATALOG: CLIO

```
$ TELNET 128.59.40.200     <-- connect to CLIO
<return>   <-- wake up the terminal controller
VT-100 <return> <-- declare your terminal type
<return>                      <-- wake CLIO up
```

```
Use <tab> to move around.
```

Make sure you exit completely from CLIO when done by choosing option 2 on the quit screen.

CONNECTICUT/TRINITY/WESLEYAN SYSTEM:

```
$ TELNET TS1.WESLEYAN.EDU  <-- get on term srv
LIBRARY             <-- connect to the library
```

```
LOGOFF:  % <RETURN> EXIT <RETURN>
```

CORNELL'S ONLINE CATALOG:

```
$ DEF TERM "vt100"      <-- must be lowercase!
$ TN3270 CORNELLC.CIT.CORNELL.EDU
<hit a <RETURN> at userid/password screen>
LIBRARY
```

DARTMOUTH'S LIBRARY CATALOG:

```
$ TELNET LIB.DARTMOUTH.EDU
```

DELAWARE'S SYSTEM: DELCAT

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET DELCAT.UDEL.EDU      <-- 128.175.13.6

TO LOGOFF:
|Q                   <-- vbar Q quits + closes
```

EMORY UNIVERSITY'S ONLINE CATALOG:

```
$ DEF TERM "vt100"      <-- must be lowercase!
$ TN3270 EMUVM1.CC.EMORY.EDU
<RETURN>        <-- hit <RETURN> to get CP READ
DIAL VTAM
LIB
<PF1>        <-- get started using the catalog
```

EUGENE, OR PUBLIC LIBRARY CATALOG:

NOTE: The Eugene, OR, Public Library is available only via dialin (i.e., by modem).

Also note that while the Eugene Public Library purportedly offers 2400 baud dialins, only 1200 and 300 baud access has been confirmed. Communication settings are the same as for the U of O VAX.

```
ATDT 687-5198    <-- dial the Eugene Pub. Lib.
<return>                  <-- wake up the system
1        <-- declare terminal type as a VT100

TO LOGOFF:
EXIT                            <-- exit OPAC
```

FLORIDA'S LIBRARY SYSTEM:

```
$ DEF TERM "vt100"      <-- must be lowercase!
$ TN3270 NERVM.NERDC.UFL.EDU  <-- note TN3270!
<wait for initial screen>
<return>
DIAL VTAM                           <-- start VTAM
NERLUIS            <-- at NERDC VTAM IS ACTIVE
ENTER SELECTION:   1
GROUP-ID:          FCL    <-- these may change
OPERATOR:          AAYA      in the future for
PASSWORD:          AAYA     tracking purposes
LUIS                 <-- at SIGNON IS COMPLETE

TO LOGOFF:
#$#$
LOGOFF
<ctrl>-C
QUIT
```

NOTE: If you get a "protected field" error when entering LUIS above, hit a <tab> and try it again.

HARVARD UNIVERSITY'S HOLLIS SYSTEM:

NOTE: You'll probably find other online libraries far easier to access unless you **really** need to use HOLLIS.

```
$ DEF TELNET_ESCAPE "|"     <-- vbar is escape
$ TELNET FASNET.HARVARD.EDU    <-- goto FASNET
<RETURN>                      <-- wake up FASNET
FASNET                         <-- select FASNET
destination: HOLLIS            <-- select HOLLIS
<CTRL>-E <RETURN>          <-- wake up the modem
*D                                  <-- dial ...
59500                          the HOLLIS number
<RETURN>      <-- wake up the terminal control
VT100               <-- declare your term type
<RETURN><RETURN>              <-- wake up HOLLIS

TO LOGOFF:
|Q                   <-- vbar Q quits + closes
```

ILLINOIS CHAMPAIGN/URBANA'S SYSTEM:

```
$ DEF TELNET_ESCAPE "|"
$ TELNET GARCON.CSO.UIUC.EDU 620     <-- NOTE:
          You must explicitly request port 620

TO LOGOFF:
|Q                   <-- vbar Q quits + closes
```

ILLINOIS CHICAGO'S SYSTEM:

NOTE: You must use TN3270 to access the Illinois machine successfully.

```
$ DEF TERM "vt100"      <-- must be lowercase!

$ TN3270 UICVM.UIC.EDU        <-- Note TN3270!
<wait for initial screen>
<RETURN>
DIAL PVM
<move cursor under the S of "S NOTIS">
<RETURN>

TO LOGOFF:
####
<CTRL-P> 1
```

NOTE: <CTRL-Z> is screen clear if needed

JOHNS HOPKINS UNIVERSITY ONLINE CATALOG:

```
$ DEF TELNET_ESCAPE "|"
$ TELNET 128.220.2.33   <-- Goto Johns Hopkins
<return><return>      <-- wake up their switch
DIAL: janus  <-- This is THEIR Janus, not ours
<return>
ENTER TERMINAL TYPE: vt100    <-- declare term
<return>

TO LOGOFF:
|Q                 <-- veebar Q quits + closes
```

KANSAS'S ONLINE CATALOG:

```
$ TELNET KUHUB.CC.UKANS.EDU  <-- Get on Kansas
Username: RELAY                        <-- "Logon"
System? .LC                <-- Note the "dot"!
VT100K                       <-- Note the "K"!
OCAT <RETURN>
<ESC> <ESC> OCAT <RETURN>
```

LEHIGH'S ONLINE CATALOG:

```
$ DEF TELNET_ESCAPE "|"
$ TELNET 128.180.2.13          <-- Goto Lehigh

TO LOGOFF:
|Q                  <-- veebar Q quits + closes
```

MARYLAND'S ONLINE CATALOG: UMCAT

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET UMCAT.UMD.EDU
CAT <return>
<return> <return>

TO LOGOFF:
|Q                    <-- vbar Q quits + closes
```

MICHIGAN'S ONLINE CATALOG: MIRLYN

```
$ DEF TELNET_ESCAPE "|"
$ TELNET CTS.MERIT.EDU     <-- (or 35.1.1.6 or
                                 MERIT.MSU.EDU)
Host? MIRLYN          <-- choose MIRLYN (note:
                        other libraries are also
                     available; enter ? instead)

TO LOGOFF:
|Q                  <-- veebar Q quits + closes
```

MINNESOTA'S ONLINE CATALOG: LUMINA

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET LUMINA.LIB.UMN.EDU
2 <return>                   <-- "I'm a VT100"
LUMINA <RETURN>
A <RETURN>

TO LOGOFF:
|Q                   <-- vbar Q quits + closes
```

USING THE NEW MEXICO STATE LIBRARY:

```
$ TELNET LIBRARY.NMSU.EDU
<return>
HELLO USER.LIBR01        <-- that's "zero one"
```

USING THE UNIV. OF NEW MEXICO'S LIBRARY:

```
$ TELNET BOOTES.UNM.EDU
Username: STUDENT1       <-- STUDENT1-STUDENT6
Terminal Type: VT100
3                            <-- UNM Libraries
1                          <-- General Library
```

SUNY-BINGHAMTON'S ONLINE CATALOG:

```
$ DEF TERM "vt100"      <-- must be lowercase!
$ TN3270 BINGVMC.CC.BINGHAMTON.EDU
<wait for initial screen>
COMMAND ===> DIAL VTAM
ELIXIR

TO LOGOFF:
<CTRL>-Z                <-- try this way first
OFF
UNDIAL

<CTRL>-C        <-- otherwise escape to TN3270
tn3270> Q                    <-- and then quit
```

NEW YORK UNIVERSITY'S CATALOG: BOBCAT:

```
$ DEF TERM "vt100"      <-- must be lowercase!
$ TN3270 BOBCAT.NYU.EDU      <-- note TN3270!
<return>

TO LOGOFF:
<CTRL>-C
```

UNIV. OF NORTH CAROLINA'S CATALOG:

```
$ TELNET 128.109.157.150         <-- NC switch
# lib                     <-- must be lowercase!
```

NORTHWESTERN UNIVERSITY'S SYSTEM:

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET PACX.ACNS.NWU.EDU    <-- 129.105.49.2
<return><return>      <-- wake up their switch
enter class  LUIS              <-- choose LUIS
VT100                <-- declare terminal type

|Q               <-- veebar Q quits and closes
```

NOTRE DAME'S ONLINE CATALOG: IRISHMVS

```
$ TELNET IRISHMVS.CC.ND.EDU     <-- Notre Dame

ENTER COMMAND OR 'HELP':  LIBRARY
```

OHIO-STATE'S ONLINE CATALOG:

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET LCS.US.OHIO-STATE.EDU    <-- Goto OSU
0          <-- choose term type (not echoed!)
HELP          <-- you'll get an error 1st time
HELP         <-- second time it will work okay

TO LOGOFF:
|Q                   <-- vbar Q quits + closes
```

OREGON STATE'S ONLINE CATALOG: OASIS

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET SYTEK.UCS.ORST.EDU  <-- goto term srv
destination: OASIS      <-- choose the library
Logon Please: OASIS         <-- logon to OASIS

TO LOGOFF:
|Q                    <-- vbar Q quits + closes
```

PENN STATE'S ONLINE CATALOG:

```
$ TELNET LIAS.PSU.EDU        <-- 128.118.25.13
<return> <return>
SELECTION? <return>
```

PENN'S ONLINE CATALOG: PENNLIN

```
$ TELNET 128.91.4.10       <-- connect to PENN

<return><return>
DIAL: PENNLIN           <-- choose the library
<return><return>
VT100           <-- declare your terminal type
```

PRINCETON'S ONLINE LIBRARY SYSTEM:

```
$ TELNET PUCABLE.PRINCETON.EDU
# CALL 500
```

PURDUE'S THOR ONLINE CATALOG:

```
$ DEF TERM "vt100"      <-- must be lowercase!
$ TN3270 LIB.CC.PURDUE.EDU    <-- note TN3270!
```

RENSSELAER'S INFOTRAC SYSTEM:

```
$ TELNET INFOTRAC.RPI.EDU   <-- Connect to RPI
```

RICE UNIVERSITY'S CATALOG: LIBRIS

```
$ DEF TERM "vt100"          <-- Note lowercase!
$ TN3270 LIBRARY.RICE.EDU     <-- Note TN3270!
<wait for initial screen>
<hit tab twice to get to COMMAND ===> )
DIAL LIBRIS          <-- connect to the catalog
<CRTL>-Z          <-- clear the screen to start

TO LOGOFF:
<CTRL>-C
tn3270> Q
```

USING TENNESSEE'S ONLINE CATALOG

```
$ TELNET DCA.UTK.EDU          <-- 128.169.200.68
LIBRARY
```

NOTE: Network access to TN from Oregon may be unreliable.

USING TEXAS A&M'S LIBRARY SYSTEM:

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET 128.194.15.40   <-- conn to term srvr
LIBRARY                 <-- choose the library
VT100              <-- declare terminal type
NOTIS                        <-- choose NOTIS

ALTERNATE ACCESS PROCEDURE:
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET VENUS.TAMU.EDU        <-- 128.194.4.1
Usernaem: VTAM
NOTIS

TO LOGOFF:
|Q                   <-- vbar q quits + closes
```

USING TEXAS-ARLINGTON'S CATALOG:

```
$ DEF TELNET_ESCAPE "|"
$ TELNET ADMIN.ARL.UTEXAS.EDU
VT100                <-- Declare terminal type
NOTIS         <-- Choose the library from VTAM
LUUT               <-- Choose LUUT within CICS

TO LOGOFF:
|Q
```

USING TEXAS-AUSTIN'S ONLINE CATALOG:

NOTE: You **MUST** use the **KEYPAD ENTER** key, not the normal return key to enter UTCAT commands once you're logged on!

```
$ DEF TELNET_ESCAPE "|"   <-- make vbar escape
$ TELNET UTAUS.CC.UTEXAS.EDU  <-- Go to Austin
Username: UTCAT
1 <KEYPAD ENTER>
<KEYPAD ENTER>
```

USING TEXAS-DALLAS'S ONLINE CATALOG:

```
$ TELNET IBM.UTDALLAS.EDU   <-- Connect to UTD
VT100                <-- Declare your terminal
LIBRARY                 <-- Choose the library
```

USING VIRGINIA TECH'S ONLINE CATALOG:

```
$ TELNET 128.173.1.4   <-- connect to Va. Tech

CALL vtls           <-- connect to the library
```

WASHINGTON UNIVERSITY-ST. LOUIS'S SYSTEM:

```
$ DEF TELNET_ESCAPE "|"
$ TELNET WUGATE.WUSTL.EDU
login: LUIS                 <-- Login to server
1                         <-- Choose WU Library
VT100                <-- Declare terminal type
```

USING WATERLOO'S ONLINE CATALOG:

Waterloo catalog access is via TYMNET only. If you're using a modem you can directly dial (503) 343-0044 for Eugene public (2400/1200/300 baud) TYMNET access.

```
Enter terminal identifier: A
Please login: DPAC;302033500196
```

USING WYOMING'S ONLINE CATALOG:

```
$ DEF TELNET_ESCAPE "|"
$ TELNET LIBRARY.UWYO.EDU    <-- go to Wyoming
<wait for connection to be made>
02                        <-- declare your term
<CTRL-C>                  <-- redraw the screen
<wait for search form to appear>

TO LOGOFF:
|Q
```

*** OTHER LIBRARIES ***:

If you have information on connecting to other university libraries via the Internet please send mail to JOE@OREGON.UOREGON.EDU with details. Thanks!

FORTRAN:

RUNNING A SMALL FORTRAN PROGRAM:

```
$ ED STATS.FOR              <-- Make the program
*C
<tab> program stats
<tab> implicit none
<tab> real x, sumx, sumxx, minx, maxx
<tab> real xbar, ssdevx, stddevx
<tab> integer n
<tab> data sumx/0.0/, sumxx/0.0/,
<tab> 1 minx/99999.9/, maxx/-99999.9/, n/0/
<tab> open (unit=10, file='stuff.dat',
<tab> 1 status='old', readonly)
<tab> open (unit=20, file='summary.dat',
<tab> 1 status='unknown',
<tab> 2 carriagecontrol='list')
100 <tab> read (10, '(f)', end=200) x
<tab> sumx = sumx + x
<tab> sumxx = sumxx + (x**2)
<tab> n = n + 1
<tab> minx = min (minx, x)
<tab> maxx = max (maxx, x)
<tab> goto 100
200 <tab> xbar = sumx / float (n)
<tab> ssdevx=sumxx - (sumx**2 / float(n))
<tab> stddevx = sqrt (ssdevx / float(n-1))
<tab> write (20, '(''  N  SIGMA X      '',
<tab> 1 ''MAX      MIN     MEAN    STDEV'')')
<tab> write (20, '(i3,5(2x,f7.2))')
<tab> 1 n,sumx,maxx,minx,xbar,stddevx
<tab> close (10)
<tab> close (20)
<tab> end
<ctrl>-Z
*EX

$ FOR/LIST STATS.FOR            <-- Compile prog
$ PRINT STATS.LIS             <-- Print listing
$ LINK STATS.OBJ              <-- Link program
$ RUN STATS.EXE                <-- Run program
$ PRINT SUMMARY.DAT           <-- Print output
```

NOTE: In additional to all standard statments, VAX Fortran programmers here at Oregon can also use routines from the IMSL Math/Stat libraries, DISSPLA Graphics libraries, VAX/VMS System Services, and VAX/VMS Run-Time Libraries (including SMG and symmetric multiprocessing routines). See the following pages for examples of using these routines.

CALLING VMS SYSTEM SERVICES:

Many esoteric bits of information are available via VMS system service calls.

For example, assume you need to retrieve the username of a person running a program. You could get that information with the following:

```
$ ED USETEST.FOR
*C
<tab> PROGRAM TESTIT
<tab> CHARACTER*15 USRNAME
<other code goes here>
<tab> CALL WHOAMI(USRNAME)
<tab> PRINT *,'THIS PROGRAM RUN BY ',USRNAME
<other code goes here>
<tab> END

<tab> SUBROUTINE WHOAMI(USRNAME)
<tab> INTEGER*4 STATUS
<tab> INTEGER*2 W_LEN1,W_CODE1
<tab> INTEGER*4 L_ADDR1,L_LENADDR1
<tab> EXTERNAL SYS$GETJPI
<tab> INCLUDE '($JPIDEF)/LIST'
<tab> CHARACTER*15 USRNAME
<tab> COMMON /ITEMLIST/
<tab> 1 W_LEN1,W_CODE1,L_ADDR1,L_LENADDR1
<tab> DATA W_LEN1/15/
<tab> DATA W_CODE1/JPI$_USERNAME/
<tab> DATA L_LENADDR1/0/
<tab> L_ADDR1=%LOC(USRNAME)
<tab> STATUS=SYS$GETJPI(,,,W_LEN1,,,)
<tab> RETURN
<tab> END
<CTRL>-Z
*EX
$ FOR USETEST
$ LINK USETEST
$ RUN USETEST
```

For online details about each of the VAX system services, see HELP SYSTEM_SERVICES

You'll also want to look at:

o **Introduction to VAX/VMS System Routines** (Volume 8A in the orange documentation set)

o **VAX/VMS System Services Reference Manual** (Volume 8D in the orange documentation set)

A FULL SCREEN SMG EXAMPLE:

```
<tab> INCLUDE '($SMGDEF)/LIST'
<tab> INCLUDE '($TRMDEF)/LIST'
<tab> CHARACTER NAME*30,ADDRESS*30,MASK*4,OK*1
<tab> INTEGER I,ISTAT,P,K,ITERM,D,MD
<tab> EQUIVALENCE (MASK,MD)
<tab> MD=IBSET(MD,2)+IBSET(MD,9)+IBSET(MD,13)
<tab> CALL SMG$CREATE_PASTEBOARD(P)
<tab> CALL SMG$CREATE_VIRTUAL_KEYBOARD(K,,,,0)
<tab> DO WHILE ((OK.NE.'D').AND.(OK.NE.'d'))
<tab> OK=' '
<tab> CALL SMG$CREATE_VIRTUAL_DISPLAY(8,52,
<tab> 1 D,SMG$M_BORDER)
<tab> CALL SMG$PASTE_VIRTUAL_DISPLAY(D,P,2,2)
<tab> CALL SMG$PUT_CHARS(D,'NAME:',2,2)
<tab> CALL SMG$PUT_CHARS(D,'ADDRESS:',4,2)
<tab> CALL SMG$PUT_CHARS(D,'OK? N/Y/D:',7,2)
<tab> CALL SMG$DRAW_RECTANGLE(D,1,17,3,50)
<tab> CALL SMG$DRAW_RECTANGLE(D,3,17,5,50)
<tab> CALL SMG$DRAW_RECTANGLE(D,6,17,8,21)
100 <tab> CALL SMG$SET_CURSOR_ABS(D,2,19)
<tab> CALL SMG$READ_STRING(K,NAME,,30,
<tab> 1 TRM$M_TM_NORECALL,,MASK,,ITERM,D)
<tab> IF (ITERM.EQ.2) GOTO 100
200 <tab> CALL SMG$SET_CURSOR_ABS(D,4,19)
<tab> CALL SMG$READ_STRING(K,ADDRESS1,,30,
<tab> 1 TRM$M_TM_NORECALL,,MASK,,ITERM,D)
<tab> IF (ITERM.EQ.2) GOTO 100
300 <tab> CALL SMG$SET_CURSOR_ABS(D,7,19)
<tab> CALL SMG$READ_STRING(K,OK,,1,
<tab> 1 TRM$M_TM_NORECALL,,MASK,,ITERM,D)
<tab> IF (ITERM.EQ.2) GOTO 200
<tab> IF ((OK.EQ.'Y').OR.(OK.EQ.'y')) GOTO 400
<tab> IF ((OK.EQ.'N').OR.(OK.EQ.'n')) GOTO 100
<tab> IF ((OK.EQ.'D').OR.(OK.EQ.'d')) GOTO 400
<tab> GOTO 300
400 <tab> CALL SMG$SET_CURSOR_ABS(D,2,19)
<tab> CALL SMG$READ_FROM_DISPLAY(D,NAME)
<tab> CALL SMG$SET_CURSOR_ABS(D,4,19)
<tab> CALL SMG$READ_FROM_DISPLAY(D,ADDRESS)
<tab> OPEN (10,FILE='XY.DAT',STATUS='UNKNOWN',
<tab> 1ACCESS='APPEND',CARRIAGECONTROL='LIST')
<tab> WRITE (10,'(A30/A30/)') NAME,ADDRESS
<tab> CLOSE (10)
<tab> CALL SMG$DELETE_VIRTUAL_DISPLAY(D)
<tab> ENDDO
99999 <tab> CALL SMG$ERASE_PASTEBOARD(P)
<tab> END
```

DOING SINGLE CHARACTER I/O:

While VMS Fortran has excellent facilities for doing record oriented I/O, doing input from Fortran on the VAX normally requires that the user hit a carriage return after each line.

Sometimes, however, you want to be able to read or write a single character from Fortran without having to wait for a carriage return or other line terminator, and without having anything show up on the screen. For example, you might want to say something like: "Hit ANY character to continue...." after output:

```
<tab> include 'disk$user1:[joe]chario.for'
<tab> character*1 dummy
<tab> call ttinit()
<tab> print *,'Blah, blah, blah.'
<tab> print *,'Hit ANY key to continue....'
<tab> call qiowget(dummy)
<tab> print *,'Resuming execution!'
<tab> end
```

Or, as another example, perhaps you're writing an automated testing program, and you want a user to respond with an A, B, C, D, or E without having to hit a return. Try:

```
<tab> include 'disk$user1:[joe]chario.for'
<tab> character*1 choice
<tab> call ttinit()
<tab> write (*,'(''$Choose A,B,C,D or E: '')')
<tab> call qiowget(choice)
<tab> call putc(choice)
<tab> print *,'Now doing (whatever)'
<tab> end
```

NOTE: To see how you can even handle multicharacter single key presses (such as KP8 or PF1), print a copy of the "include" file used in the preceding examples.

For detailed information on doing "fancy" I/O of this kind, see <u>Fortran Tools for VAX/VMS and MS-DOS</u> by R. K. Jones and T. Crabtree, John Wiley and Sons, New York (1988).

TIMING FORTRAN CODE:

To time your Fortran code, include the following subroutine in your program:

```
<tab> SUBROUTINE GETTIM(CPU)
<tab> INTEGER*2 W_LEN1,W_CODE1
<tab> INTEGER*4 L_ADDR1,L_LENADDR1
<tab> EXTERNAL SYS$GETJPIW
<tab> INCLUDE '($JPIDEF)'
<tab> INTEGER*4 CPUTIME
<tab> REAL*4 CPU
<tab> COMMON /ITEMLIST/
<tab> 1 W_LEN1,W_CODE1,L_ADDR1,L_LENADDR1
<tab> DATA W_LEN1/4/
<tab> DATA W_CODE1/JPI$_CPUTIM/
<tab> DATA L_LENADDR1/0/
<tab> L_ADDR1=%LOC(CPUTIME)
<tab> CALL SYS$GETJPIW(,,,W_LEN1,,,)
<tab> CPU=FLOAT(CPUTIME)/100.
<tab> RETURN
<tab> END
```

Then, to figure the elapsed CPU time used, do:

```
<tab> CALL GETTIM(START)
<code to be timed goes here>
<tab> CALL GETTIM(END)
<tab> TIME=END-START
<tab> PRINT *,'CPU TIME USED=',CPU,' SECONDS'
```

To see how to make a CPU time "profile" see:

```
DISK$USER1:[JOE.CARDS]TIMERTEST.FOR
```

SPAWNING VMS COMMANDS FROM FORTRAN:

Use the LIB$SPAWN system service to execute VMS commands from within a Fortran program:

```
$ ED STUFF.FOR
*C
<tab> call lib$spawn('SH TIME')
<tab> end
<CTRL-Z>
*EX
$ FOR STUFF.FOR                    <-- Compile It
$ LINK STUFF.OBJ                      <-- Link It
$ RUN STUFF.EXE                        <-- Run It
```

USING AN OBJECT MODULE LIBRARY:

```
$ FOR ALP.FOR,FOO.FOR <-- Compile source files
$ LIB/CREATE MYLIB.OLB ALP.OBJ,FOO.OBJ     <--
     create MYLIB.OLB, inserting ALP and FOO

$ FOR SOMEMAIN.FOR   <-- Compile the main prog
$ LINK SOMEMAIN.OBJ,MYLIB.OLB/L <-- Link MYLIB

$ FOR WOW.FOR   <-- Compile another file later
$ LIB/INSERT MYLIB.OLB WOW.OBJ     <-- Add WOW

$ FOR ALP.FOR <-- Compile a (modified) ALP.FOR
$ LIB/REPLACE MYLIB.OLB ALP.OBJ <--Replace ALP

$ LIB/DELETE MYLIB.OLB ALP.OBJ <-- Zap ALP.OBJ
```

RETRIEVING COMMAND LINE ARGUMENTS:

If you need to be able to pass information to a program from the command line, you should use LIB$GET_FOREIGN, defining the program as a foreign command. A common example of an occaision when you'd like to be able to do this sort of thing is when you're writing a new command and need to be able to retrieve a filename (as an argument to the command) from the command line:

```
$ EVE MYCOM.FOR
<tab> CHARACTER*80 STUFF
<tab> INTEGER ILENGTH
<tab> IRET=LIB$GET_FOREIGN(STUFF,,ILENGTH,)
<tab> OPEN (UNIT=10,FILE=STUFF(1:ILENGTH),
<tab> 1 STATUS='UNKNOWN')
<tab> ! other code goes here
<tab> END
<CTRL-Z>

$ FOR MYCOM             <-- compile the program
$ LIN MYCOM                <-- link the program
$ MYCOM :== $DISK$USER4:[JDOE]MYCOM.EXE    <--
            define the command to be executed;
            note FULL PATH and "extra" leading
            dollar sign
$ MYCOM TEST.DAT <-- you DON'T say "RUN" here!
```

NOTE: The string LIB$GET_FOREIGN returns is always uppercased!

LOWERCASING A STRING:

There are times when you'd like to convert a Fortran string from uppercase to lowercase. Unfortunately, there is no specific lowercasing routine equivalent to STR$UPCASE. However, you can get the job done with STR$TRANSLATE:

```
<tab> INTEGER STR$TRANSLATE,LEN,IRET
<tab> CHARACTER INFN*80,OUTFN*80,FROM*26,TO*26
<tab> DATA FROM/'ABCDEFGHIJKLMNOPQRSTUVWXYZ'/
<tab> DATA TO  /'abcdefghijklmnopqrstuvwxyz'/
<tab> INFN='THIS IS a weird TEST'
<tab> IRET=STR$TRANSLATE(OUTFN,INFN,TO,FROM)
<tab> PRINT *,OUTFN
<tab> END
```

CONVERTING NUMBERS READ AS CHARACTERS:

Due to various circumstances, at times you may end up with a series of digits (that is, a number) stored in character form. In order to make it possible to do arithmetic on that number, you'll need to convert the string into numeric form with code such as:

```
<tab> INTEGER IVALUE
<tab> CHARACTER*7 MYSTRING
<some other code here>
<tab> MYSTRING='3435672'   <-- start as chars!
<tab> READ (MYSTRING,'(I7)') IVALUE
<tab> IVALUE=IVALUE+1950   <-- its an int now!
<code continues>
<tab> END
```

NOTE: To convert from a numeric value to a character expression, change the READ to a WRITE.

NUMERICAL RECIPES:

Users needing excellent VAX/VMS Fortran numerical routines should see Numerical Recipes by William H. Press, et. al., Cambridge University Press, Cambridge (1987).

VMS'S PARALLEL PROCESSING FEATURES:

If you do order-independent matrix operations within do loops, you can use the VAX's symmetric multiprocessing capabilities. For instance:

```
$ ED TEST1.FOR
*C
<tab> integer i,j,n(10000)
cpar$ private i            <-- i is non shared
cpar$ do_parallel 1000 <-- do 1000 per subproc
<tab> do i=1,10000
<tab>    n(i)=n(i)+i+10
<tab> end do
<tab> print *,n(1),n(10000)  <-- or whatever
<tab> end
<ctrl-Z>
*EX

$ FOR/PARA TEST1     <-- use parallel routines
$ LINK TEST1                      <-- link it
$ DEFINE FOR$PROCESSES 4    <-- use 4 subprocs
$ RUN TEST1                        <-- run it
```

NOTE: You generally cannot vectorize do loops that contain input/output statements, do loops that conntain other do loops, or do loops that contain order-of-execution-critical code.

PRETTY-PRINTING FORTRAN PROGRAMS:

FPRETTY will reformat your Fortran code in a pleasing and easy-to-read format.

```
$ @SYS$SYSTEM:UNSUPPORTED_SETUP.COM
$ FPRETTY ABC.FOR <-- Yields formatted ABC.LIS
$ FPRETTY/NOLIST ABC.FOR <-- Reformats ABC.FOR
```

GENERATING FORTRAN FLOWCHARTS:

```
$ INDEX :== $DISK$USER1:[JOE.INDEX]INDEX.EXE
$ INDEX/FLOW=NOMOVE TEST.FOR
$ PRINT TEST.LIS
```

WRITING PORTABLE FORTRAN CODE:

If you're writing Fortran code that has even a **CHANCE** of needing to be move to any other machine, you'll want to take notice of the following points in preparing your code:

(1) Write your Fortran statements in standard fixed column format; avoid using the VMS tab delimited format since many machines won't recognize it.

(2) Use upper case characters only, except as needed within character literals.

(3) Use variable names that are six characters or less in length.

(4) Use a C in column one to code a comment rather than using the VAX exclamation point comment statment extension.

(5) Avoid use of the VAX's extended set of data types. For example, use LOGICAL, INTEGER, REAL, DOUBLE PRECISION or CHARACTER*len but shun BYTE, INTEGER*2, REAL*16, etc.

(6) Avoid the VAX's unnumbered DO / END DO construct; use the conventional numbered DO / CONTINUE construct instead.

(7) Avoid using "fancy formats" in input and output operations; particularly loathsome are run-time formats and variable length formats.

(8) Avoid use of any VAX Fortran language element printed in blue in the VAX Fortran manual: all of these elements are extensions and are prohibited to you; likewise, avoid use of VAX system services, or run time library calls or function references.

(9) Finally, check your program for potential portability problems by compiling it with the command:

```
$ FOR/CHECK=ALL/WARN=ALL/STAND=ALL
```

prior to trying to physically move it to another system.

USING THE DEBUGGER:

```
$ ED BUG.PREFS    <-- Our debugger preferences
*C                       file -- automatically
SET MODULE/ALL         searched for and executed
SET MODE SCREEN       by the debugger at startup
<CTRL-Z>
*EX

$ DEFINE DBG$INIT DISK$USER2:[ABDOWN]BUG.PREFS

$ FOR/NOOPT/DEBUG HTR.FOR      <-- Must include
$ LINK/DEBUG HTR.OBJ         /DEBUG qualifiers!
$ RUN HTR.EXE

DBG> STEP              <-- Execute one statement

DBG> GO                 <-- Run the program till
                               next breakpoint

DBG> EX JTOT                 <-- Examine var JTOT

DBG> EX X(4):X(7)  <-- Examine part of array X

DBG> EX REPORT\DEDUCT      <-- Examine variable
                       DEDUCT in module REPORT

DBG> SET WATCH IJY   <-- Watch for IJY changes

DBG> SET BREAK/CALL          <-- Pause at CALLs

DBG> SET BREAK/BRANCH    <-- Pause at BRANCHes

DBG> SET BREAK %LINE 14   <-- Pause at line 14

DBG> SET BREAK/LINE DO (EXAMINE PAY; GO)  <--
  Break after each line; look at PAY; continue

DBG> SET B AVER WHEN (I.EQ.0) DO (EXAM JJ; GO)

DBG> EXTRACT/ALL LOG.TXT    <-- log the screen

DBG> EXTRACT/ALL/APPEND LOG.TXT  <-- append it

DBG> HELP KEYPAD DEFAULT     <-- get some help

DBG> <CTRL-Z>       <-- exit from the debugger
```

OTHER PROGRAMMING LANGUAGES:

RUNNING A SMALL C PROGRAM:

```
$ ED MEANS.C                <-- Make the program
*C
#include stdio.h
#include math.h
main()
{double x,xbar,ssdevx,stddevx,
    n=0.0, sumx=0.0, sumxx=0.0,
    minx=99999.9, maxx=(-99999.9);
 int end;
 FILE *fl1,*fl2;
 fl1 = fopen("stuff.dat","r");
 fl2 = fopen("summary.dat","w");
 end=fscanf(fl1,"%f",&x);
 while (end != -1)
   {sumx+=x;
    sumxx+=(x*x);
    n++;
    if (x < minx) minx=x;
    if (x > maxx) maxx=x;
    end=fscanf(fl1,"%f",&x); }
 xbar=sumx/n;
 ssdevx=sumxx-((sumx*sumx)/n);
 stddevx=sqrt(ssdevx/(n-1.0));
 fprintf(fl2,"  N SIGMA X     MAX   ");
 fprintf(fl2,"  MIN    MEAN   STDEV\n");
 fprintf(fl2,
   "%3.0f %7.2f %7.2f %7.2f %7.2f %7.2f\n",
   n, sumx, maxx, minx, xbar, stddevx);
 fclose(fl1);
 fclose(fl2); }
<ctrl>-Z
*EX
```

NOTE: The following libraries must be defined before C programs will properly link!

```
$ DEF LNK$LIBRARY SYS$LIBRARY:VAXCCURSE.OLB
$ DEF LNK$LIBRARY_1 SYS$LIBRARY:VAXCRTLG.OLB
$ DEF LNK$LIBRARY_2 SYS$LIBRARY:VAXCRTL.OLB
```

```
$ CC/LIST MEANS.C               <-- Compile prog
$ PRINT MEANS.LIS              <-- Print listing
$ LINK MEANS.OBJ                <-- Link program
$ RUN MEANS.EXE                  <-- Run program
$ PRINT SUMMARY.DAT             <-- Print output
```

RUNNING A SMALL COBOL PROGRAM:

```
$ ED COUNTKITES.COB       <-- Make the program
*C
IDENTIFICATION DIVISION.
PROGRAM-ID. SAMPLE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  OUTPUT-REKORD.
    05 I      PIC 9(2) VALUE IS 1.
    05 FILLER PIC X(6) VALUE IS " KITES".
PROCEDURE DIVISION.
MAINLINE_ROUTINE.
    PERFORM PARA_ONE VARYING I FROM
        1 BY 1 UNTIL I = 11. STOP RUN.
PARA_ONE. DISPLAY OUTPUT-REKORD.
END PROGRAM SAMPLE.
<ctrl>-Z
*EX

$ COB/LIST COUNTKITES.FOR    <-- Compile prog.
$ LINK COUNTKITES.OBJ         <-- Link program
$ RUN COUNTKITES               <-- Run program
$ PRINT COUNTKITES.LIS       <-- Print listing
                                    if desired
```

NOTE If you need to open a file in VAX COBOL, try using statements like the following:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT INPUT-FILE ASSIGN TO "FOO.DAT".
```

RUNNING A SMALL PASCAL PROGRAM:

```
$ ED STATS.PAS
*C
program stats(infile,outfile);
  var x, xbar, ssdevx, stddevx,
        sumx, sumxx, minx, maxx : real;
      n : integer;
      infile, outfile: text;
  begin
    open (infile, file_name:='stuff.dat',
       history:=readonly);
    open (outfile, file_name:='summary.dat');
    sumx := 0.0;
    sumxx := 0.0;
    minx := 99999.9;
    maxx := -99999.9;
    n := 0;
    reset(infile);
    while not eof(infile) do begin
      readln (infile, x);
      sumx := sumx + x;
      sumxx := sumxx + (x**2);
      n := n + 1;
      minx := min (minx, x);
      maxx := max (maxx, x);
    end; {while}
    xbar := sumx / n;
    ssdevx := sumxx - (sumx**2 / n);
    stddevx := sqrt (ssdevx / (n-1));
    rewrite(outfile);
    writeln (outfile, '  n  sigma x      ',
      'max      min     mean      stdev');
    writeln (outfile,n:3,'  ',sumx:7:2,'  ',
      maxx:7:2,'  ',minx:7:2,'  ',xbar:7:2,
      '  ',stddevx:9:2);
    close (infile);
    close (outfile);
  end.
<CTRL-Z>
*EX

$ PASCAL/LIST STATS.PAS       <-- Compile prog
$ LINK STATS.OBJ              <-- Link program
$ RUN STATS.EXE                <-- Run program
$ PRINT STATS.LIS            <-- Print listing
                                  if desired
```

RUNNING A SMALL MACRO PROGRAM:

```
$ ED COUNTKITES.MAR
*C
        .TITLE  SAMPLE MACRO PROGRAM
        ;
        ; DATA SECTION FOLLOWS ***************
        .PSECT  MYDATA,NOEXE,WRT,LONG
I:      .LONG   1        ; LOOP STARTS AT 1
LIMIT:  .LONG   10       ; AND ENDS AT 10
        ;
OUTBUF: .BYTE   32       ; TENS BEGINS AS SPACE
        .BYTE   32       ; ONES BEGINS AS SPACE
        .ASCII  / KITES/  ; ADD'L OUTPUT TEXT
        .BYTE   13,10    ; END WITH A CR/LF
        ;
        .ALIGN  LONG
O_FAB:  $FAB    FNM=<TT>,FAC=PUT
O_RAB:  $RAB    FAB=O_FAB,RBF=OUTBUF,USZ=10
        ;
        ; EXECUTABLE CODE FOLLOWS ************
        .PSECT  MYPROGRAM,EXE,REL,LONG
        .ENTRY  SAMPLE,0
        $OPEN   FAB=O_FAB
        $CONNECT RAB=O_RAB
        MOVW    #10,O_RAB+RAB$W_RSZ ; 10 CHARS
        ;
LOOP:   MOVL    I,R2
        CMPL    I,#10 ; LOOP VALUE EQ 10?
        BNEQU   JUMP        ; ... NO! GOTO JUMP
        MOVB    #49,OUTBUF ; ... YES!  SET THE
        MOVB    #0,R2       ; ... TEN AND ONES!
        ;
JUMP:   ADDB    #48,R2       ; ASCII=BINARY+48
        MOVB    R2,OUTBUF+1 ; MOVE TO OUTBUF
        $PUT    RAB=O_RAB   ; NOW DO THE $PUT
        AOBLEQ  LIMIT,I,LOOP; LOOP UNTIL DONE
        ;
        $EXIT_S
        .END    SAMPLE
<CTRL>-Z
*EX

$ MACRO/LIST COUNTKITES.MAR <-- Assemble prog.
$ LINK COUNTKITES.OBJ     <-- Link the program
$ RUN COUNTKITES.EXE       <-- Run the program
$ PRINT COUNTKITES.LIS       <-- Print listing
                                  if desired
```

RUNNING PROLOG PROGRAMS:

Prolog is a non-procedural, fourth-generation language widely used for "artificial intelligence" work.

The following example illustrates how PROLOG can be used to recursively solve the famous "Towers of Hanoi" problem.

```
     = | =          |          |
    == | ==         |          |
   === | ===        |          |
----------------------------------------
```

[You may recall that in the Tower of Hanoi problem, your task is to move "n" discs from the leftmost peg to the rightmost peg, without ever allowing a larger disc to be placed on top of a smaller disc on any peg.]

```
$ ED HANOI.PRO
*C
/* tower of hanoi (after borland's prog 55) */

hanoi(N) :- move(N,'left','middle','right').

move(1,A,_,C) :- inform(A,C),!.
move(N,A,B,C) :- N1 is N-1,
                 move(N1,A,C,B),
                 inform(A,C),
                 move(N1,B,A,C).

inform(L1,L2) :- write('Move a disk from '),
                 write(L1),
                 write(' to '),
                 write(L2),
                 nl.

:- hanoi(3).
:- halt.
<CTRL>-Z
*EX

$ PROLOG
consult('HANOI.PRO').
```

NOTE: Pay particular attention to capitalization and punctuation when typing in the above example!

RUNNING A SMALL COMMON LISP PROGRAM:

Allegro Common Lisp is another popular artificial intelligence language. To use it here at the U of O, request an account on DECOY, a NeXT workstation. For information on getting an account on DECOY, send a VAX Mail message to username JOE (JOE@OREGON) from your OREGON VAX account.

After you get your DECOY username, immediately telnet to DECOY and change your password; you can then use Common Lisp.

```
$ telnet decoy     <-- Go to DECOY.UOREGON.EDU
login: jsmith    <-- Enter your DECOY username
Password:     <-- Enter initial DECOY password
1 % passwd  <-- Change pass if you haven't yet
2 % vi hanoi.Cl
i
; Tower-of-Hanoi in Lisp (after PH Winston)

(defun tower-of-hanoi (n)
 (transfer 'a 'b 'c n)
 (setq tower-of-hanoi ()))

(defun move-disk (from to)
 (print (list 'move 'disk 'from from 'to to)))

(defun transfer (from to spare n)
 (cond ((= n 1) (move-disk from to))
   (t (list (transfer from spare to (- n 1))
     (move-disk from to)
     (transfer spare to from (- n 1))))))

(tower-of-hanoi 3)
<esc>
:wq
3 % cl <hanoi.Cl
4 % <ctrl>-d
```

GRAPHICS:

GRAPHIC OUTPUT OPTIONS:

TERMINALS:

PC users can do Tektronix (monochrome) 4010/4014 emulation via MS-Kermit version 2.32/A or 3.0. That product is free, and will run on PCs with virtually any type of PC video adaptor, including CGA, EGA, VGA, and Hercules compatible monographics adaptors.

Note that if you are directly connected via a 3Com card to UONet, (free) NCSA Telnet for the PC also provides Tektronix 4010/4014 support.

Macintosh users have similar terminal emulation packages available for their graphic output requirements.

The GIGI (VT125/VT241/REGIS compatible) 4-color graphics terminal formerly available for public use in Room 105 is, unfortunately, no longer available.

A personally owned 8-color Tektronix 4027 may also be available on an as-arranged basis; contact Joe St Sauver (x6-4394) for more information.

Finally, the National Center for Supercomputing Activities has developed a standard raster file format that can be used to store and display 8-bit (256-color) or 24-bit (million-color) graphics output on PCs, Macs or Sun workstations. Obtain viewing software for these machines via anonymous FTP from ZAPHOD.NCSA.UIUC.EDU

HARDCOPY:

Hardcopy output is available via the Center's spooled Zeta-8 pen plotter.

You can also generate Postscript output and download it to a Laserwriter. A Mac and Laserwriter dedicated to laser printing is available for use in 207 Computing Center at a cost of $0.20/page]

MAKING A DISSPLA GRAPH:

```
$ ED GRAPH00.FOR          <-- Make the program
*C
<tab> REAL X(10), Y(10)
<tab> DATA X/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
<tab> DATA Y/2.,3.,5.,5.,2.,4.,2.,1.,6.,8./
<tab> CALL COMPRS
<tab> CALL AREA2D(7.0,9.0)
<tab> CALL HEADIN('A SHORT TITLE',100,1.0,1)
<tab> CALL XNAME('X AXIS LABEL',100)
<tab> CALL YNAME('Y AXIS LABEL',100)
<tab> CALL GRAF(1.,'SCALE',10.,1.,'SCALE',8.)
<tab> CALL CURVE(X,Y,10,1)
<tab> CALL ENDPL(0)
<tab> CALL DONEPL
<tab> END
*EX
$ FOR GRAPH00.FOR          <-- Compile program
$ DISSLINK GRAPH00.OBJ   <-- Link Disspla V.11
$ RUN GRAPH00.EXE          <-- Run the program
$ DISSPOP              <-- Start postprocessor
ZETA                  <-- Generate ZETA output
MODI=1-END(HWROT=MOVIE)*DRAW=1-END  <-- Rotate
<RETURN>                  <-- Execute commands
$ PLOT ZETA.GSF           <-- Plot ZETA output
```

NOTE: If you are using any MYxxx routines (such as MYPIE) or if you need more than one million longwords of blank common, you'll need to use non-shareable DISSPLA. Substitute DISSNOSHARE for DISSLINK above and you should be all set.

NOTE: If you need to send output directly to a graphics device replace CALL COMPRS with the appropriate device nomination call. Details on available drivers can be found in directory DISSPLA$DISK:[DISSPLA.DIS11.DOC]

USING DISSPLA CODEBOOK:

NOTE: Use CODEBOOK when you need assistance in preparing a "skeleton" of your DISSPLA graph.

```
$ DISSCODEBOOK
```

NOTE: See the Codebook Manual for more info

USING DISSPLA GKS GRAPHIC ROUTINES:

NOTE: New graphics applications should be developed using DISSPLA rather than GKS; use DISSPLA GKS only when working with existing code containing embedded GKS calls.

```
$ ED SAMPLE.FOR
<tab> INTEGER SEGMENT_STORAGE
<tab> REAL YEAR(10),GNP(10)
<tab> COMMON SEGMENT_STORAGE(5000)
<tab> DATA GNP /91.,76.,58.,56.,65.,
<tab> 1         72.,83.,91.,85.,91./
<tab> DATA YEAR /1929.,1930.,1931.,1932.,
<tab> 1 1933.,1934.,1935.,1936.,1937.,1938./
<tab> ! OPEN GKS & SEND ERRORS TO FOR010.DAT
<tab> CALL GOPKS(10,5000)
<tab> ! SEND OUTPUT TO POPFIL.DAT
<tab> CALL GOPWK(1,1,102)
<tab> ! ACTIVATE THE WORKSTATION
<tab> CALL GACWK(1)
<tab> ! SETUP THE WINDOW LIMITS
<tab> CALL GSWN(1,1928.,1939.,50.,100.)
<tab> ! SET THE VIEWPORT TO THE WHOLE WINDOW
<tab> CALL GSVP(1,0.,1.,0.,1.)
<tab> ! CHOOSE THE VIEWPORT VIEW JUST DEFINED
<tab> CALL GSELNT(1)
<tab> ! SET THE POLYLINE INDEX TO SOLID LINE
<tab> CALL GSPLI(1)
<tab> ! DRAW THE POLYLINE
<tab> CALL GPL(10,YEAR,GNP)
<tab> ! DEACTIVATE THE WORKSTATION
<tab> CALL GDAWK(1)
<tab> ! CLOSE THE WORKSTATION & GKS
<tab> CALL GCLWK(1)
<tab> CALL GCLKS
<tab> END
<CTRL>-Z
*EX

$ FOR SAMPLE.FOR
$ DISSGKS SAMPLE.OBJ
$ RUN SAMPLE.EXE
$ DISSPOP                  <-- Start postprocessor
ZETA                      <-- Generate ZETA output
MODI=1-END(HWROT=MOVIE)*DRAW=1-END  <-- Rotate
<RETURN>                     <-- Execute commands
$ PLOT ZETA.GSF              <-- Plot ZETA output
```

USING THE ZETA FPS ROUTINES:

NOTE: New graphics applications should be developed using DISSPLA rather than the ZETA fundamental plotting subroutines; use these routines only when working with existing code containing embedded ZETA FPS calls.

```
$ ED SAMPZET.FOR
*C
<tab> OPEN (UNIT=9,FILE='ZETA.GSF',
<tab> 1 STATUS='UNKNOWN',
<tab> 2 CARRIAGECONTROL='LIST')
<tab> CALL PLOTS(53,IDUMB,-9)
<tab> CALL PLOT(4.,4.,2)
<tab> CALL PLOT(4.,0.,3)
<tab> CALL PLOT(0.,4.,2)
<tab> CALL PLOT(0.0,0.0,999)
<tab> END
<CTRL-Z>
*EX

$ FOR SAMPZET
$ LINK SAMPZET,SYS$LIBRARY:ZETA_FPS.OLB/L
$ RUN SAMPZET
$ PLOT ZETA.GSF
```

SAS/GRAPH:

```
$ ED ALASKA.SAS
*C
options ls=72 nocenter;
title 'Tourists Coming to Alaska (by State)';
libname mapfile 'sas$maps:';
filename grafout 'zeta.gsf';
data jundat;
  length stname $ 25;
  infile 'stateht.dat';
  input state count stname &$;
goptions device=zeta887 nodisplay
  gsfmode=replace gsfname=grafout;
proc gmap data=jundat map=mapfile.us;
  id state;
  prism count/nolegend;
<CTRL-Z>
*EX
$ SAS ALASKA
$ PRINT ALASKA.SAS+ALASKA.LOG
$ PLOT ZETA.GSF
```

WRITING POSTSCRIPT FILES FROM SAS:

This section explains how to write Postscript files from SAS for display on Apple LaserWriters or a NeXT megapixel display.

```
$ ED SAMPLE.SAS
*C
DATA;          /* MAKE A SMALL BOGUS DATA SET */
  DO X1=0 TO 20; Y1=25 + 3*X1; OUTPUT; END;
FILENAME MYGSF 'TEST1.PS';
GOPTIONS DEVICE=PS300     /* MAKE 300 DPI PS */
  NODISPLAY      /* SUPPRESS TERMINAL OUTPUT */
  GSFNAME=MYGSF      /* NAME THE OUTPUT FILE */
  GSFMODE=REPLACE    /* OVERWRITE OLD GSF'S */
  GSFLEN=132     /* OTHER VALUES CAN CHOKE! */
  HANDSHAKE=NONE  /* HANDSHAKING NOT NEEDED */
  CHARACTERS      /* USE HARDWARE CHARS...  */
  CHARTYPE=9     /* FROM THE HELVETICA FONT */
  PENMOUNTS=255   /* ALLOW 255 DITH. COLORS */
  HSIZE=10 VSIZE=7.5   /* SET GRAPH DIMENS. */
  FTITLE=NONE /* MAKE TITLE USE PROPER FONT */
  CSYMBOL=BLACK /* AVOID HALFTONING PROBLEM */
  GPROLOG='252150532D41646F62652D312E30'X;
TITLE 'SAMPLE LINE GRAPH';
SYMBOL1 I=JOIN V=NONE;
PROC GPLOT;    /* DO THE HOKEY LITTLE GRAPH */
   PLOT Y1*X1 /OVERLAY;
<CTRL-Z>
*EX
$ SAS SAMPLE
$ PRINT SAMPLE.SAS+SAMPLE.LOG
```

NOTE: Your SAS/Graph postscript output can then be found in TEST1.PS, which can be downloaded via FTP (in normal ASCII mode). Display your output on an Apple Laserwriter by using Macintosh SendPS 1.2.1 (or use the PC command LWPRINT under LocalTalk PC). You can also look at your output on a NeXT using the YAP app OR the Preview app. [The GPROLOG string fixes problems that had been noted previously when attempting to use the Preview App to view SAS/Graph PostScript output.]

For more information on using the SAS/Graph postscript driver, see SAS Technical Report P-178

WRITING POSTSCRIPT FILES FROM DISSPLA:

This section explains how to write Postscript files from DISSPLA for display on Apple LaserWriters or a NeXT megapixel display.

```
$ ED TRIAL.FOR
*C
<tab> IMPLICIT NONE
<tab> REAL X(10), Y(10)
<tab> DATA X/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
<tab> DATA Y/2.,3.,5.,5.,2.,4.,2.,1.,6.,8./
<tab> CALL SETUPPS
<tab> CALL PAGE(7.99,10.78)
<tab> CALL AREA2D(6.0,7.0)
<tab> CALL GRAF(1.,'SCALE',10.,1.,'SCALE',8.)
<tab> CALL CURVE(X,Y,10,0)
<tab> CALL ENDPL(0)
<tab> CALL DONEPL
<tab> END
<tab> SUBROUTINE SETUPPS
<tab> INTEGER IBUF(16)
<tab> IBUF(1)=5       ! SEND OUTPUT TO SPOOLER
<tab> CALL IOMGR(IBUF,-102)
<tab> IBUF(1)=80      ! SET BUFFER TO 80 CHARS
<tab> CALL IOMGR(IBUF,-101)
<tab> IBUF(1)=1       ! LEAVE THESE MAGIC
<tab> IBUF(2)=1       ! VALUES ALONE IF YOU
<tab> IBUF(3)=1       ! WANT A GOOD PS FILE!
<tab> CALL IOMGR(IBUF,-111)
<tab> IBUF(1)=4       ! "GEN UNIQUE FILENAMES"
<tab> CALL IOMGR(IBUF,-104)
<tab> CALL PSCRPT(-1.0,-1.0,-1.0)
<tab> RETURN
<tab> END
<ctrl>-Z
*EX
$ FOR TRIAL
$ DISSLINK TRIAL
$ RUN TRIAL
```

NOTE: Your postscript's in STD00001.DAT (with the next output going to STD00002.DAT, etc.). Download these files as regular ASCII text files and display them as described for SAS/Graph Postscript output on the preceding panel.

NCSA SDS/HDF RASTER GRAPHICS:

A long-standing problem for University of Oregon users has been a lack of mainframe color graphics output support: color graphics terminals have traditionally been rather rare here.

NCSA's HDF routines change all that, offering color output, and even support for animation.

NCSA has produced (and released to the public domain) a series of raster graphic routines centered around a common self-describing data file format called known as the HDF ("Hierarchical Data Format").

NCSA HDF library routines allow users to generate images on supercomputers, midrange systems, workstations, and even microcomputers using a common series of calls from Fortran or C.

Once you've generated your images, you'll then typically download them to a color microcomputer and image using free NCSA microcomputer visualization programs.

Currently supported microcomputer visualization platforms include the Mac II with 256-color display, and PCs with EGA/VGA/NO9 video display adaptors.

NCSA uses two main types of graphics files: "RIS8" raster image set 8-bit hierarchical data format files, and "SDS" floating point scientific data sets.

HDF RIS8 files are displayed on the Mac II using NCSAImage, and on the PC using PCShow. SDS files are displayed on the Mac II (only, for now) using DataScope. [N.B. PCShow users can use FPTOHDF to convert SDS files into RIS8 files].

Oregon users can obtain copies of NCSAImage, PCShow, or DataScope via anonymous FTP from ZAPHOD.NCSA.UIUC.EDU, or you can contact Joe St Sauver for a local copy (bring along a disk!).

GENERATING AN HDF SDS-FORMAT FILE:

Beginning HDF user will want to obtain a copy of the manual NCSA HDF Calling Interfaces and Utilities, Version 3.0, November 1989. However, the following example illustrates the basic approach needed to create an SDS file:

```
$ @DISK$USER1:[HDF.NEW]SETUP.COM
$ EVE TEST.FOR
<tab> implicit none
<tab> real z1(100,100), z2(100,100)
<tab> integer dim(2),iret,i,j
<tab> integer DFSDputdata,DFSDadddata
<tab> dim(1)=100
<tab> dim(2)=100
<tab> ! generate some bogus data
<tab> do j=1,dim(2)
<tab>   do i=1,dim(1)
<tab>     z1(i,j)=float(i)**2-sqrt(float(j))
<tab>     z2(i,j)=float(i)**3-(float(j))**.333
<tab>   enddo
<tab> enddo
<tab> ! put the first z-array to the file
<tab> iret=DFSDputdata('abc.sds',2,dim,z)
<tab> ! add the next z-array to the sds file
<tab> iret=DFSDadddata('abc.sds',2,dim,z)
<tab> end
<CTRL-Z>
$ FOR TEST             <-- compile the program
$ HDFLINK TEST <-- link with the HDF libraries
$ RUN TEST                 <-- run the program
$ FIXATR /rfm=fixed:512 ABC.SDS    <-- fix the
               SDS file attributes for FTP-ing
[binary ftp ABC.SDS to Mac II/DataScope OR...]

$ FPTOHDF ABC.SDS -o ABC.HDF -r -p blue.pal
      <-- convert the .SDS file to a .HDF file
$ FIXATR /rfm=fixed:512 ABC.HDF    <-- fix the
               HDF file attributes for FTP-ing
[binary ftp ABC.HDF to PC/PCShow or Mac/Image]
```

NOTE: Please make the followng correction to any printed FPTOHDF documentation:

```
Changes to FPTOHDF command line arguments:

  Substitute -f for -F
  Substitute -r for -I
```

GENERATING AN HDF RIS8-FORMAT FILE:

```
$ EVE TEST2.FOR
<tab> implicit none
<tab> integer i,j,k,l,testtot
<tab> character*1 pal(768)
<tab> character*1 pic(20,20)
<tab> byte buff(20,20)
<tab> equivalence (pic(1,1),buff(1,1))
<tab> ! read in a pre-existing palette
<tab> call dfpgetpal('fire.pal',pal)
<tab> ! set that palette
<tab> call dfr8setpalette(pal)
<tab> ! make some bogus data (4 bars 5x20)
<tab> do l=1,4
<tab>   do k=1,5
<tab>     j=(l-1)*5+k
<tab>     do i=1,20
<tab>       buff(i,j)=l
<tab>     enddo
<tab>   enddo
<tab> enddo
<tab> call dfr8putimage('tst.hdf',pic,20,20,0)
<tab> ! make some bogus stripes (3 stripes)
<tab> do j=1,20
<tab>   do i=1,20
<tab>     testtot=i+j
<tab>     if (testtot.le.10) then
<tab>       buff(i,j)=1
<tab>     else if (testtot.le.25) then
<tab>       buff(i,j)=2
<tab>     else
<tab>       buff(i,j)=3
<tab>     endif
<tab>   enddo
<tab> enddo
<tab> call dfr8addimage('tst.hdf',pic,20,20,0)
<tab> end
<CTRL-Z>
$ @DISK$USER1:[HDF.NEW]SETUP
$ FOR TEST2
$ HDFLINK TEST2
$ RUN TEST2
$ FIXATR /rfm=fixed:512 TST.HDF    <-- fix the
               HDF file attributes for FTP-ing
[binary ftp TST.HDF to PC/PCShow or Mac/Image]
```

NOTE: For palettes, see DISK$USER1:[HDF.PAL]

NEOVISUALS:

We will be receiving a VAX version of NeoVisuals software from the SAS Austin, Texas office as soon as the VAX/VMS production release of that software becomes available (i.e., sometime during the Spring of 1990).

When available, Neovisuals will provide photo-realistic graphic output, including polygon modelling and b-spline modelling, spline editing, rendering, surface "texturing," fluid animation, and many other advanced graphics features.

If you're interested in running Neovisuals on other dedicated graphics systems (such as Silicon Graphics Irises) here at the University of Oregon, contact SAS Institute's Austin, TX office at 512-258-5171 for more information on highly favorable educational license arrangements.

STATISTICAL PACKAGES:

ENTERING YOUR DATA:

PUNCHING YOUR OWN DATA: If you punch your own data on a PC, enter it using a spreadsheet such as 1-2-3, with one column per variable and one row per case. If you punch your data on the VAX, you'll probably use a normal text editor such as EDT or EMACS.

PROFESSIONAL KEYPUNCHING: There also are private keypunchers who'll punch data to a PC disk file which can then be uploaded to the VAX.

OPTICAL SCANNING: If your data can be entered on preprinted five or ten answer mark sense scan forms, scanning your data to tape is an option. Pick up pre-printed mark sense bubble sheets ($0.05/form), complete them, and return them for scanning to tape (an additional $0.05/form). See the downstairs receptionist in the Computing Center for more information.

OTHER DATA ENTRY NOTES:

FREE VS FIXED FORMAT: Some data entry approaches virtually **always** yield a fixed format file (optically scanned bubble sheets, for instance, and professionally keyed files). However, if you punch your own data, you'll need to decide between free and fixed format input. In free format, one or more spaces seperate each data item from the next. Fixed format requires that you consistently enter the data for a particular item in the same location for every case. Free format is easier to enter than fixed format, but harder to check for errors.

MISSING DATA: If you're using fixed format to enter your data, use blanks to indicate missing data; note that you must physically put in spaces wherever there's missing data (even at the end of line) since the VAX doesn't automatically pad short lines with spaces. If you're using free format, enter one dot per field to represent a missing value to SAS or SPSSx; don't just skip a field altogether or SAS/SPSSx will "lose its place."

NEVER USE TABS: Use spaces instead!

RUNNING A SAMPLE SAS PROGRAM:

```
$ ED SURVEY.SAS
*C
options ls=72 nocenter;
libname mystuff '[]';  libsearch mystuff;
title 'survey of fastfood preferences';
proc format library=mystuff;
   value foodform 1='hamburgers'
      2='hot dogs' 3='pizza';
   value ageform  1='under 17'
      2='18-22' 3='23-30'
      4='31-59' 5='60 and up';
data mystuff.foodfile;
   infile 'food.dat';
   input casenum 1-3 v1 5 v2 7-8 v3 $ 10-20;
   label casenum = 'case number'
     v1 = 'favorite fast food' v2 = 'age'
     v3 = 'name';
   if 0<=v2<=17 then v2=1;
   else if 18<=v2<=22 then v2=2;
   else if 23<=v2<=30 then v2=3;
   else if 31<=v2<=59 then v2=4;
   else if 60<=v2 then v2=5;
   format v1 foodform. v2 ageform.;
proc freq;
 tables (v1 v2)/all;
<ctrl>-Z
*EX
$ SAS SURVEY
$ PRINT SURVEY.SAS+SURVEY.LOG+SURVEY.LIS
```

NOTE: To do further runs with the permanent SAS data set created in the above run, use SAS code such as:

```
$ ED SURVEY2.SAS
*C
options ls=72 nocenter;
libname mystuff '[]';  libsearch mystuff;
title1 'survey of fastfood preferences';
data;
  set mystuff.foodfile;
proc freq;
  tables v1*v2/all;
<ctrl>-Z
*EX
$ SAS SURVEY2
$ PRINT SURVEY2.SAS+SURVEY2.LOG+SURVEY2.LIS
```

RUNNING A SAMPLE SPSSx PROGRAM:

```
$ ED SURVEY.SPS
*C
SET WIDTH=80
TITLE 'SURVEY OF FAST FOOD PREFERENCES'
DATA LIST FILE='FOOD.DAT'/
 CASENUM 1-3 V1 5 V2 7-8 V3 10-20 (A)
VAR LABELS
  CASENUM 'CASE NUMBER'/
  V1 'FAVORITE FAST FOOD'/
  V2 'AGE'/
  V3 'NAME'/
RECODE
  V2 (1 THRU 17=1)
     (18 THRU 22=2)
     (23 THRU 30=3)
     (31 THRU 59=4)
     (60 THRU 99=5)/
VALUE LABELS
  V1 1 'HAMBURGERS'
     2 'HOT DOGS'
     3 'PIZZA'/
  V5 1 'UNDER 18'
     2 '18-22'
     3 '23-30'
     4 '31-59'
     5 '60 AND UP'/
SAVE OUTFILE='SURVEY.SYS'
FREQUENCIES GENERAL=V1 V2
  /STATISTICS=ALL
<ctrl>-Z
*EX
$ SPSSX/OUT SURVEY
$ PRINT SURVEY.SPS+SURVEY.LIS
```

NOTE: To do further runs with the SPSSx system file created in the above run, use code such as the following:

```
$ ED SURVEY2.SPS
*C
SET WIDTH=80
TITLE 'SURVEY OF FASTFOOD PREFERENCES'
GET FILE='SURVEY.SYS'
CROSSTABS TABLES=V1 BY V2
  /STATISTICS=ALL
<CTRL-Z>
*EX
$ SPSSX/OUT SURVEY2
$ PRINT SURVEY2.SPS+SURVEY2.LIS
```

RUNNING A SAMPLE BMDP PROGRAM:

```
$ ED SURVEY.BMD
*C
/ PROBLEM TITLE IS 'FAST FOOD SURVEY'.
/ INPUT VARIABLES = 3.
  FILE IS 'FOOD.DAT'.
  FORMAT IS '(F3.0,1X,F1.0,1X,F2.0)'.
/ VARIABLES NAMES ARE CASENUM,FAVEFOOD,AGE.
  USE FAVEFOOD, AGE.
/ GROUP CODE(AGE)=17.5,22.5,30.5,59.5.
  NAME(AGE)='0-17','18-22','23-30','31-59',
    '60-99'.
/ SAVE CODE IS LUVLY.
  FILE IS 'MY.LIB'.
  NEW.
  CONTENT IS DATA.
/ END.
<ctrl>-Z
*EX
$ BMDP 2D SURVEY.BMD SURVEY.LIS
<ctrl>-Z
$ PRINT SURVEY.BMD+SURVEY.LIS
```

NOTE: To do further runs with the BMDP save file created in the above run, use code such as:

```
$ ED SURVEY2.BMD
*C
/ PROBLEM TITLE IS 'FAST FOOD SURVEY'.
/ CODE IS LUVLY.
  FILE IS 'MY.LIB'.
  CONTENT IS DATA.
/ TABLE ROW=AGE.
  COL=FAVEFOOD.
/ PRINT PERC=ROW,COL,TOT.
/ STAT ALL.
/ END.
<ctrl>-Z
*EX
$ BMDP 4F SURVEY2.BMD SURVEY2.LIS
<ctrl>-Z   <-- we don't want to run more procs
$ PRINT SURVEY2.BMD+SURVEY2.LIS
```

NOTE: BMDP users should be aware that we do also have the BMDP Data Manager.

RUNNING A SMALL MINITAB JOB:

```
$ ED DEMO.MIN
*C
OUTFILE 'TEMP.LIS'
OH=0
READ 'RAWDATA.DAT' C1 C2 C3
SAVE 'MYSHEET'
TABLE C1 C2 C3
STOP
<ctrl>-Z
*EX
$ MINITAB
MTB> EXECUTE 'DEMO.MIN'
$ PRINT DEMO.MIN+TEMP.LIS
```

RUNNING A SMALL SCA PROGRAM:

```
$ ED TRIAL.SCA
*C
==SAMPLE
PROFILE REVIEW.
INPUT V1, V2, V3. FILE IS 'RAWDATA.DAT'.
BSAVE V1, V2, V3. DATASET IS MYSAMPLE. NEW. @
  FILE IS 'MYSCA.BIN'.
DESCRIBE V1, V2, V3.
<ctrl>-Z
*EX
$ SCA TEMP.LIS
CALL SAMPLE. FILE IS 'TRIAL.SCA'.
$ PRINT TRIAL.SCA+SCAOUTP.DAT
```

RUNNING A SMALL IMSL PROGRAM:

```
$ ED EXAMPLE.FOR
*C
C<tab>GEN AND PRINT 5 UNIFORM RANDOM DEVIATES
<tab> REAL ZRAY(5)
<tab> CALL RNSET(11133551)    <-- Set the seed
<tab> CALL RNUN(5,ZRAY)    <-- Get the numbers
<tab> PRINT *,ZRAY       <-- Print the numbers
<tab> END
<ctrl>-Z
*EX
$ FOR EXAMPLE.FOR           <-- Compile program
$ IMSL EXAMPLE.OBJ             <-- Link program
$ RUN EXAMPLE.EXE               <-- Run program
```

RUNNING A SMALL TS PROGRAM:

```
$ ED WA.BJ  <-- MUST use .BJ (for Box-Jenkins)
*C
101.2237
<etc...>
<CTRL-Z>
*EX
$ @SYS$SYSTEM:UNSUPPORTED_SETUP
$ TS
Program? IDENT
TITLE? SAMPLE RUN
FILE NAME? WA
SMPL? <CTRL-E> 1 100
LPT OR DSK? DSK
Name for output? TEMP
Wide or narrow? NARROW
LOG? NO
STANDARD RUN? YES
OPTION? \
Program? $$
$ PRINT TEMP.DTP
```

SAS/IML: INTERACTIVE MATRIX LANGUAGE:

```
$ ED REGEX.SAS
*C
options ls=72 nocenter;
data example;
  input x1 x2 y @@;
  const=1;
  cards;
  4 2 64  4 4 73  4 2 61  4 4 76  6 2 72
  6 4 80  6 2 71  6 4 83  8 2 83  8 4 89
  8 2 86  8 4 93 10 2 88 10 4 95 10 2 94
  ;
proc iml;
  print "SAS/IML Matrix Regression Example";
  reset print;
  use example;
  read all var {x1 x2 const} into X;
  read all var {y} into Y;
  bhat = inv(x`*x)*(x`*y);
<CTRL-Z>
*EX
$ SAS REGEXAMP
$ PRINT REGEX.SAS+REGEX.LOG+REGEX.LIS
```

USING LISREL:

```
$ ED LIZ.SPS            <-- Matrix input example
*C
INPUT PROGRAM            <-- dummy input program
NUMERIC DUMMY             <-- is required if not
END FILE                 <-- reading in raw data
END INPUT PROGRAM        <-- via SPSSx DATA LIST
USERPROC NAME=LISREL             <-- call LISREL
SAMPLE FACTOR RUN         <-- title for this run
DATAPARAMETERS NI=6 NO=556 MA=KM
LA
'EVALSELF' 'PEVALPNT' 'PEVALTCH'
'PEVALFND' 'EDUCASPN' 'COLLPLAN'
KM SY
 1.00
 .73 1.00
 .70 .68 1.00
 .58 .61 .57 1.00
 .46 .43 .40 .37 1.00
 .56 .52 .48 .41 .72 1.00
MODEL NX=6 NK=2 PH=ST
LK
'ABILITY' 'ASPIRATN'
FREE LX(1,1) LX(2,1) LX(3,1) C
LX(4,1) LX(5,2) LX(6,2)
OUTPUT SE TV RS VA MI FS
END USER                    <-- exit from LISREL
<ctrl-Z>
*EX
$ SPSSX/OUT LIZ          <-- run the commandfile
$ PRINT LIZ.SPS+LIZ.LIS     <-- print the output
$ ED LIZ2.SPS               <-- example number 2
*C
DATA LIST FREE FILE='REG.DAT'/
   Y X1 X2 X3
USERPROC NAME=LISREL
SAMPLE REGRESSION EXAMPLE
DATAPARAMETERS NI=4
NA
'Y' 'X1' 'X2' 'X3'
RA
MODEL NY=1 NX=3
OU TO ULS SS
END USER
<CTRL-Z>
*EX
$ SPSSX/OUT LIZ2
$ PRINT LIZ2.SPS+LIZ2.LIS
```

USING EQS:

```
$ EVE E.EQS         <-- EQS example number one
/TITLE
 SAMPLE FACTOR RUN
/SPECIFICATIONS
 CASES=556; VARIABLES=6;
/LABELS
 V1=EVALSELF; V2=PEVALPNT; V3=PEVALTCH;
 V4=PEVALFND; V5=EDUCASPN; V6=COLLPLAN;
 F1=ABILITY; F2=ASPIRATN;
/EQUATIONS
 V1=2*F1 + E1; V2=2*F1 + E2; V3=2*F1 + E3;
 V4=2*F1 + E4; V5=2*F2 + E5; V6=2*F2 + E6;
/VARIANCES
 F1 TO F2 = 1.0; E1 TO E6 = 10*;
/COVARIANCES
 F2,F1=10*;
/MATRIX
 1.00
 .73 1.00
 .70 .68 1.00
 .58 .61 .57 1.00
 .46 .43 .40 .37 1.00
 .56 .52 .48 .41 .72 1.00
/END
<CTRL-Z>
$ EQS IN=E.EQS DAT=E.EQS OUT=E.LIS LEN=25000
$ PRINT E.EQS+E.LIS

$ EVE T.EQS        <-- EQS example number two
/TIT
 SAMPLE REGRESSION RUN
/SPE
 VAR=4; MA=RAW; ME=LS; CASES=16;
/LAB
 V1=Y1; V2=X1; V3=X2; V4=X3;
/EQU
 V1=1*V2 + 1*V3 + 1*V4 + E1;
/VAR
 V2 TO V4=1*; E1=1*;
/COV
 V2,V3=1*; V2,V4=1*; V3,V4=1*;
/END
<CTRL-Z>
$ EQS IN=T.EQS DAT=T.DAT OUT=T.LIS LEN=25000
$ PRINT T.EQS+T.LIS
```

RUNNING SAS IN DISPLAY MANAGER MODE:

```
$ SAS/FSD=VT100

SAS:       VT100: EDITOR:   LOG:      OUTPUT:
-------------------------------------------
F1         F1     HELP      HELP      HELP
F2         F2     PRINT     PRINT     PRINT
F3         F3     SUBMIT    END       END
F4         KP7    RECALL              COMMAND
F5         KP8    RFIND     RFIND     RFIND
F6         KP9    RCHANGE
F7         KP4    BACKWARD  BACKWARD  BACKWARD
F8         KP5    FORWARD   FORWARD   FORWARD
F9         KP6    SPLIT     SPLIT     SPLIT
F10        KP1    LEFT      LEFT      LEFT
F11        KP2    RIGHT     RIGHT     RIGHT
F12        KP3    CURSOR    CURSOR    CURSOR
PRIOR FLD  KP 0
NEXT FLD   KP PERIOD
ENTER      KP ENTER
REFRESH    KP COMMA
HOME       KP MINUS
DELETE     DELETE
ERASE EOF  LINE FEED (CONTROL-J)
INSERT     BACKSPACE (CONTROL-DELETE)
ARROWS     ARROWS

COMMANDS TO ENTER AT THE ====> PROMPT:
CAPS OFF     <-- don't change case!
BAC n        <-- go backward n lines
FOR n        <-- go forward n lines
TOP          <-- go to top of file
BOT          <-- go to the bottom of file
RIGHT n      <-- shift right n columns
LEFT n       <-- shift left n columns
OUTPUT       <-- go to OUTPUT screen
LOG          <-- go to LOG screen
PROGRAM      <-- go to the editor screen
INCLUDE 'filename'  <-- bring in 'filename'
SAVE 'filename'     <-- save this program
SUBMIT       <-- execute my program
BYE          <-- get out of SAS completely

SOME COMMANDS TO ENTER ON THE LINE NUMBERS:
B,A        <-- Set target BEFORE or AFTER
C,M,D      <-- COPY, MOVE or DELETE this line
CC,MM,DD   <-- COPY, MOVE or DELETE this range
```

RUNNING A LITTLE LIMDEP PROGRAM:

LIMDEP is a program providing support for a number of more sophisticated econometric techniques including PROBIT/LOGIT/TOBIT and multinomial logit models, GLS, simultaneous equation models and survival models.

```
$ ED SAMPLE.LIM
*C
TITLE; SAMPLE LIMDEP OLS REGRESSION RUN$
READ; NREC=87; NVAR=2; NAMES=Y,X; FILE=E1.DAT$
CRMODEL ; LHS=Y; RHS=X, ONE$
STOP$
<CTRL-Z>
*EX

$ @SYS$SYSTEM:UNSUPPORTED_SETUP
$ LIMDEP SAMPLE
$ PRINT SAMPLE.LIM+SAMPLE.LIS
```

RATS Econometric Package:

RATS is another popular econometrics package:

```
$ ED HOTFOOD.RATS
*C
ALLOCATE 0 25
OPEN DATA SPICE.DAT
DATA(ORG=OBS,UNIT=DATA,FORMAT=FREE, $
  MISSING=-999.0) 1 25 $
  SPCYNESS REDPEP BLACKPEP CURRY
LINREG SPCYNESS
# CONSTANT REDPEP BLACKPEP CURRY
<CTRL>-Z
*EX

$ RATS HOTFOOD.RATS
$ PRINT HOTFOOD.RATS+HOTFOOD.LIS
```

S (Statistical Analysis Package):

The S statistical analysis package is **NOT** available on the VAX 8800, but it **is** available on one of the Math Department's machines. Contact Prof Charles Wright for more info.

OPERATIONS RESEARCH PRODUCTS:

DECALC:

DECalc is an electronic spreadsheet program that runs on the VAX under VMS. While it is **NOT** a Lotus 1-2-3 or Excel clone, it does do most of the things you'd normally like a spreadsheet to do, and it works fine for entering statistical data. To run DECalc (from a VT100-style terminal only) enter:

```
$ CALC/NOPASS/M=A    <-- must specify /M=A for
     algebraic evaluation of math expressions!
```

CELLS:

```
-- The grid has calls labeled A1 to ZZ9999
-- Make cell references absolute by adding one
   or more dollar sign:  $B7  C$12  $D$5
-- Specify cell ranges using a colon:  F5:H9
-- Scientific ("E-notation") values are okay
```

SOME COMMONLY USED DECALC FUNCTIONS:

```
* / + - ^                  Basic math operators

ABS(C12)              Absolute value of cell C12
ROU(2.438,2)        Rounds (for a value of 2.44)
ROU(6789,-3)        Rounds (for a value of 7000)
TRUNC(2.438,2)         Truncates (yielding 2.43)
TRUNC(6789,-3)         Truncates (yielding 6000)

AVE(A1:A10,B1:B20)           Mean of those cells
COU(L14:L29)                Count of those cells
STDEV(C40:C50)        Std Dev (N-1) of the cells
SUM(K9:K20)           Takes the sum of the cells

LS(SLOPE,D1:D10,E1:E10)        D is indep, E dep
LS(CONSTANT,D1:D10,E1:E10)       Intercept value
LS(FIT,D1:D10,E1:E10)                  R-squared

SQRT(D8)                   Takes the square root
COS(.567)                 Takes cosine (radians)
ACOS(1.10)           Takes inverse cosine (rads)
EXP(C3)                Takes natural exponential
LN(T5)                          Take natural log
ALOG(Y3)           Takes antilog base 10 of (Y3)
LOG(X3)                        Takes log base 10

IF(A7<0,0,A7)           Return 0 if A7 is < zero
TAB(125,A1:A10,B1:B10)  Returns the value of B
              where the corresponding A is 125
```

DECALC (continued):

COMMANDS:

```
-- All DECalc commands are prefix with a slash
   or backslash (/ or \)
-- Select commands by typing the first letter
   of the command and then hitting return, or
   use the cursor to highlight, then <RETURN>
-- Use <DELETE> to "back out" of a command
```

KEYS:

```
<PF2>           Help
<PF3>           Edit Cell
<CTRL>-W        Redraw Screen
<PF1><PF1>      Toggle "Privacy" Shield

<PF4><KP2>      Scroll Down One Screen
<PF4><KP8>      Scroll Up One Screen
<PF4><KP6>      Scroll Right One Screen
<PF4><KP4>      Scroll Left One Screen
```

CHANGING DEFAULT MOVEMENT:

```
/ G M T R                <-- move row-to-row
/ G M T C          <-- move column-to-column
```

SAVING YOUR SPREADSHEET:

```
/ S S MYHOURS <RET>
```

RESTORING YOUR SPREADSHEET:

```
/ S R INCMSTMT <RET>
```

PRINTING TO A FILE IN "FREE" FORMAT:

```
/ P T A1:E14 <RET> TEMP2.DAT <RET> , R
```

READING A FLAT FILE IN "FREE" FORMAT:

```
| A1 <RET> / S T TEMPA.DAT <RET> , R
```

EXITING DECALC:

```
/ E
```

NOTE: For more info, use the <PF2> key in DECalc, or see the DECalc Reference Manual.

USING LINDO:

LINEAR PROGRAMMING EXAMPLE:

```
$ ED BIKE.LIN
*C
MAXIMIZE 50 BIKES + 100 TRIKES + 30 UNIS
ST
2 BIKES + 3 TRIKES + UNIS <= 200
BIKES + TRIKES - 2 UNIS >= 0
2.5 BIKES + 4.0 TRIKES + 0.5 UNIS <= 240
END
DIVE 'TEMP.LIS'
LOOK ALL
GO
YES
QUIT
<CTRL-Z>
*EX
$ LINDO
: TAKE BIKE.LIN
$ PRINT BIKE.LIN+TEMP.LIS
```

INTEGER PROGRAMMING EXAMPLE:

```
$ ED CAPITAL.LIN
*C
MAX 90 X1 + 40 X2 + 10 X3 + 37 X4
ST
15X1 + 10X2 + 10X3 + 15X4 <= 40
20X1 + 15X2 +        10X4 <= 50
20X1 + 20X2 +        10X4 <= 40
15X1 +  5X2 +  4X3 + 10X4 <= 35
END
INTEGER X1
INTEGER X2
INTEGER X3
INTEGER X4
DIVE 'TEMP.LIS'
LOOK ALL
GO
YES
QUIT
<CTRL-Z>
*EX
$ LINDO
: TAKE CAPITAL.LIN
$ PRINT CAPITAL.LIN+TEMP.LIS
```

NOTE: Use **INTEGER** for 0/1 variables, but use **GIN** for general integer variables.

USING THE SLAM II SIMULATION LANGUAGE:

NETWORK SIMULATION EXAMPLE:

```
$ ED EXAMPLE1.SLAM
*C
GEN, STSAUVER, SAMPLE, 23/5/1988, ,,,,,, 80;
LIMITS, 1, 1, 100;
NETWORK;
      CREATE,EXPON(1.25),,1;
Q     QUE(1);
      ACT(4)/1,RNORM(5.,1.);
      COLCT,INT(1),TIME IN SYSTEM,11/4./2.;
      TERM;
      END;
INIT,0,480;
FIN;
<CTRL-Z>
*EX
$ SLAM EXAMPLE1
$ PRINT EXAMPLE1.SLAM+EXAMPLE1.LIS
```

DISCRETE SIMULATION EXAMPLE:

In addition to the above "pure" network type of simulation, SLAM II can also handle discrete network simulations made by calling SLAM from Fortran. For example:

```
$ COPY SLAM$DISK:[SLAM]DBANK.SLAM  *.*
$ COPY SLAM$DISK:[SLAM]DBANK.FOR  *.*
$ FOR DBANK.FOR        <-- Compile the Fortran
$ SLAMLINK DBANK.OBJ  <-- Link the Object File
$ SLAM DBANK DBANK     <-- Run DBANK.SLAM with
                                    DBANK.EXE
$ PRINT DBANK.LIS        <-- Print the results
```

CONTINUOUS SIMULATION EXAMPLE:

SLAM II can also do continuous simulations. For example:

```
$ COPY SLAM$DISK:[SLAM]CONTIN.SLAM  *.*
$ COPY SLAM$DISK:[SLAM]CONTIN.FOR *.*
$ FOR CONTIN           <-- Compile the Fortran
$ SLAMLINK CONTIN         <-- Link the Fortran
$ SLAM CONTIN CONTIN                <-- Run It
```

MATHEMATICA:

Mathematica is a symbolic/numerical math package. To use Mathematica here at the U of O, request an account on DECOY, a NeXT workstation. For information on getting an account on DECOY, send a VAX Mail message to username JOE (JOE@OREGON) from your OREGON VAX account.

```
$ telnet decoy     <-- Go to DECOY.UOREGON.EDU
login: jsmith    <-- Enter your DECOY username
Password:     <-- Enter initial DECOY password
1 % passwd  <-- Change pass if you haven't yet
2 % math              <-- start up mathematica
In[1]:= ?*            <-- Get Mathematica help
In[1]:= 5.45^7
In[2]:= 180!
In[3]:= FactorInteger [24260]
In[4]:= Integrate [4x + 3x^2 + 2x^3, x]
In[5]:= D [19 Log[4x] + Sqrt[2x+x^3], x]
In[6]:= Factor [8x^6 + 2x^3]
In[7]:= (x+y)(x-y)(y^2)(x^3)
In[8]:= Expand [%]

In[9]:=  NRoots [3x^2 - 17 == 0, x]
In[10]:= NIntegrate [Sin[x], {x,0.0,Pi}]

In[11]:= mymat={{3,1,2},{5,4,5},{-1,-.7,-1}}
In[12]:= Transpose[mymat]
In[13]:= Inverse[mymat]
In[14]:= Det[mymat]
In[15]:= Eigenvalues[mymat]
In[16]:= Eigenvectors[mymat]

In[17]:= ReadList["time.dat",Number]
In[18]:= Fourier[%]

In[19]:= <<Tek.m         <-- Gen Tek4010 Output
In[20]:= Plot[Sin[x],{x,0,2Pi}]

In[21]:= $DisplayFunction=Display["foo.g",#] &
In[22]:= Plot3D[Sin[x y],{x,0,3},{y,0,3}]

In[23]:= <ctrl>-d         <-- exit Mathematica

3 % psfix <foo.g >foo.eps <-- fixup postscript
4 % <ctrl>-d                    <-- logout DECOY
```

USING CALCULATOR: AN ONLINE CALCULATOR

CALCULATOR will generally evaluate any statement that would be legitimate in Fortran (including statements invoking mathematical functions such as LOG, COS, etc).

Be careful when using CALCULATOR – integers and real numbers are handled differently, which can dramatically effect the results you obtain. Generally, always specify a decimal point as part of a number.

```
$ CALCULATOR :== $DISK$USER1:[JOE.CALC]CALC

$ CALCULATOR      <-- NOTE: Do NOT rename this
             command CALC or you won't be able
                to use the DECalc spreadsheet!

CALC> 20.78/4.3           <-- do a calculation

CALC> 13/3       <-- NOTE!! Integer truncation
                        takes place unless you
CALC> 13./3.            include decimal points

CALC> 43.7**3.6      <-- can also use ! for **

CALC> (23.7+10.2)/4.5      <-- use parentheses

CALC> %+2.555    <-- %="last value calculated"

CALC> SIN(.786)    <-- evaluate trig functions

CALC> ALOG(3.45)    <-- take natural logarithm

CALC> EXP(%)        <-- do exponentiation, too

CALC> *OCTAL Z   <-- variable z displays octal
CALC> Z=29              <-- Z gets 29 (base 10)

CALC> *B 16    <-- set the default base to hex
CALC> 2E+1C        <-- do a calculation in hex

CALC> ?     <-- see brief summary of commands

CALC> <CTRL-Z>       <-- get out of CALCULATOR
```

For more information on using CALCULATOR:

```
$ PRINT DISK$USER1:[JOE.CALC]CALCDOC.MEM
```

DATABASE PACKAGES:

USING THE ACCENT R DATABASE PACKAGE:

```
DEFINE A NEW BOOK BIBLIOGRAPHY DATABASE.......
$ ACCENT
*CREATE DBL BIBLIO
*DEFINE SD BOOK.DESC
--10 BOOK.TITLE,C,20
--20 LAST.NAME,C,15
--30 FIRST.NAME,C,10
--40 PUBLISHER,C,10
--50 CITY,C,10
--60 YEAR,I,4
--70 PAGES,I,3
--SAVE
*CREATE DS MYBIB SD BOOK.DESC
*QUIT


BUILD THE NEW BOOK BIBLIOGRAPHY ..............
$ ACCENT
*USE DBL BIBLIO
*USE DS MYBIB
*ENTER FROM MYRAW.DAT <-- comma delimited data
*QUIT


ADDING RECORDS MANUALLY ......................
$ ACCENT
*USE DBL BIBLIO
*USE DS MYBIB
*ENTER WITH PROMPT
BOOK.TITLE: War and Peace
LAST.NAME:
<continue entering information>
BOOK.TITLE: ***  <-- magic "let me stop" code!
*QUIT


RUN REPORTS ON THE DATABASE ..................
$ ACCENT
*USE DBL BIBLIO
*USE DS MYBIB
*SORT ON YEAR
*EXTRACT IF YEAR >= 1977 SHOW ALL ON TEMP.LIS
*QUIT

$ PRINT TEMP.LIS
```

OTHER DATABASE MANAGEMENT PACKAGES:

Most VAX administrative applications are now being written in Oracle. However, due to Oracle's use of centralized data access controls, and due to the need to insure a secure operating environment for administrative applications running under Oracle, only specially authorized administrative users are currently able to use Oracle on DONALD, the administrative VAX 6430.

However, it is anticipated that Oracle will shortly be made available on the academic VAX 8800. At that time, academic users will be able to begin using Oracle for their applications.

CONTENT ANALYSIS PROGRAMS:

USING TEXTPACK:

Raw text files for analysis in TEXTPACK should be prepared according to the follows rules:

Begin each chapter (or other major section) with a line having only a vee-bar (|) and the number of that major section. Begin each paragraph (or minor section) with a line having only a plus sign (+) and the number of that minor section. Note that individual sentences are automatically numbered.

Use all upper or all lower case for the raw text, unless you explicitly want to distinguish words according to their case. Seperate all punctuation (periods, colons, tick marks, etc) from text by at least one space.

For example, assume you've made a file called DS.SOL and want to generate word frequencies:

```
$ ED DEMO.TP
*C
$ TP :== DISK$USER1:[TEXTPACK.EXE]
$ RUN 'TP'SENTENCE
OPTION PRE1='|',PRE2='+',PHRA,TEXT=(1,80),*
FNIN='DS.SOL',FNSE='DS.SEN'
$ RUN 'TP'LISTSPLT
OPTION SPLIT,FNSE='DS.SEN',FNSP='DS.SPLIT'
$ SORT/KEY=(POS:24,SIZ:16) DS.SPLIT DS.SPLSRT
$ RUN 'TP'FREQ
OPTION TTR,FNSP='DS.SPLSRT',FNFR='DS.FREQ'
$ RUN 'TP'FRELIST
OPTION FORM=DINA4,FNFR='DS.FREQ'
<CTRL-Z>
*EX
```

Note that you must actually enter the dollar signs in the above file of commands

```
$ @DEMO.TP/OUT=TEMP.LIS
$ PRINT DEMO.TP+TEMP.LIS
```

To obtain a copy of the TEXTPACK manual, use:

```
$ PRINT DISK$USER1:[TEXTPACK.DOC]DOC2.LIS
```

TEXT PROCESSING:

USING RUNOFF TO FORMAT A TERM PAPER:

```
 $ ED MYPAPER.RNO      <-- Create your document
 *C                    <-- Change to fullscreen
 .NO FLAGS ACCEPT     <-- These magic lines are
 .FLAGS BOLD               always there and will
 .FLAGS LOWERCASE ~         not (usually) change
 .S-35                    <-- Begin 35 lines up
 .C                    <-- Center the next line
 ^&ELDORADO MINING COMPANY:~&    <-- Underline
 .S1                           <-- Skip a line
 .C
 ^&They Never Found Their Pot of Gold~&
 .S-6                      <-- Begin 6 lines up
 .LM+40                   <-- Left edge over 40
 JAMES A. SMITH
 .BR                       <-- Start a new line
 999-12-3456
 .BR
 DSC 335 -- ASSIGNMENT #4
 .LM-40                 <-- Restore left margin
 .SPACING 2          <-- Double space the paper
 .NONUMBER     <-- No page numbering on page 1
 .PAGE                    <-- Pop to a new page
 ^*Introduction~*                <-- Bold this
 .S1
 .NUMBER 2   <-- Turn on numbering for p.2-end
 The Eldorado Mining Company [1] <... etc ...>
 .FN                              <-- Footnote
 .REPEAT 15"-"         <-- Footnote seperator
 .BR
 1 Eldorado was incorporated in Uruguay.
 .EFN                         <-- End footnote
 .S1
 These costs included:
 .S1
 ^*TABLE 1: COSTS OF PURCHASING ELDORADO~*
 .S1
 .LIT                      <-- "take literally"
 Land Costs:           $1,234,567.00
 Water Rights:           $200,000.00
 .EL                          <-- "end literal"
 .S1
 <ctrl>-Z
 *EX
$ RUNOFF/RIGHT=5 MYPAPER.RNO
$ PRINT/FORMS=SMALL MYPAPER.MEM
```

USING THE XEROX LASER PRINTER:

In order to use special features of the Xerox laser printer currently connected to the VAX, you'll need to include some special commands in your document. Every document should begin with the lines:

```
=UDK={    <--  Set the special Xerox code to {
```

Thereafter, you can give the Xerox special instructions as follows:

```
Here is some {uunderlined{w text
Here is some {bbolded{p text
Here is some {hsuperscripted{s text
Here is some {lsubscripted{s text
```

(NOTE: subscripts use a small ELL, not the number one).

To use different fonts, use:

```
Some text in {0The Titan Iso Font{0
Some text in {1The SymbolC Font{0
Some text in {2The Kosmos12 Font{0
Some text in {3The Kosmos14 Font{0
Some text in {4The Titan Italics Font{0
```

You can also draw horizontal and vertical lines on the Xerox 2700. Line coordinates are expressed in "dots" (there are 300 dots per inch), starting from the lower left hand corner of the page.

To draw lines use:

```
{xOVER,UP,LENGTH,THICKNESS(LE)  <-- along x
{yOVER,UP,LENGTH,THICKNESS(LE)  <-- along y
```

For example:

```
{x300,300,2700,10(LE)   or
{y300,300,300,10(LE)
```

To queue a job to the Xerox laser printer, use:

```
$ PRINT/QUE=LP$XEROX MYDOC.TXT
```

USING SPELL:

```
$ SPELL MYPAPER.TXT          <-- Spell check it
$ SPELL/RUNOFF SOMETHING.RNO  <-- Spell check,
                  ignoring any RUNOFF commands
```

SENDING TEXT FILES TO PRINTING:

The University of Oregon Printing Department accepts files of keystrokes (for their typesetting equipment) from the VAX. According to a document they produced entitled "How To Connect Word Processors to the Typesetting Equipment at the UO Printing Department," the process is as follows:

(1) Complete an Order and Printing Data Sheet and receive a job number (such as N9999) from the Printing Department.

(2) Upload your file using Kermit as shown under "Miscellaneous Microcomputer Topics" later in this book. Note that the file you upload must be a plain ASCII file – it **MUST NOT** have underlining, bolding or automatic page numbering.

(3) Rename the file so that it has the job number the Printing Department assigned:

```
$ RENAME file.ext N9999.JOB
```

(4) Now send the file to the Printing Dept.:

```
$ SEND/FILE/VMSDUMP N9999.JOB PRTING
```

(5) Tell Printing that the file has been sent, and send them a hard copy of the text including all font numbers used in embedded commands.

MAKING GUMMED LABELS:

```
$ SET TERM/VT100
$ ED ADDRESSES.DAT  <-- make address data file
*C
President Lidia Workhard
Prestiguous College
Somewhere, Ohio 55431
            <-- add blank lines (if necessary)
               to make address five lines long
Dr. James Q. Public
Department of Biostatistics
Dedicated University
1492 Columbus Way
Baton Rouge, LA 12345
<etc>
<CTRL-Z>
*EX

$ ED LABELS.SAS        <-- make label program
data one;
infile 'addresses.dat' lrecl=33 recfm=f;
input line1 $char33./
      line2 $char33./
      line3 $char33./
      line4 $char33./
      line5 $char33.;

filename out 'labels.out';
proc forms dd= out
    align=20 width=33 lines=5
    skip=1 nacross=3 between=3 indent=1;
    line 1 line1;
    line 2 line2;
    line 3 line3;
    line 4 line4;
    line 5 line5;
<CTRL-Z>
*EX

$ SAS LABELS      <-- generate the labels file
$ PRINT LABELS.LOG  <-- print the error log...
$ PRINT/FORM=GUMMED LABELS.OUT  <-- and labels
```

BANNERS (LINE PRINTER-GENERATED SIGNS):

TYPE 1: DENSE BLOCK, ROMAN/GOTHIC, LONG-WISE

```
$ BAN1 :== @DISK$USER1:[JOE.BANNER]BANNER.COM
$ BAN1
R                         <-- Choose Roman Font
Test Message                  <-- Enter Message
Y                   <-- Ask to Print It Now ...
<return>                   <-- On SYS$PRINT ...
<return>       <-- On Default Forms (Greenbar)
```

TYPE 2: SINGLE CHAR, GOTHIC ONLY, LONG-WISE

```
$ BAN2 :== $DISK$USER1:[JOE.GOTHIC]BANNER.EXE
$ BAN2
Test Message
<return>
N
$ PRINT/NOFEED GOTHIC.OUT
```

TYPE 3: SINGLE CHAR, PLAIN FONT, MOVIE-STYLE

```
$ BAN3 :== $DISK$USER1:[JOE.SMALL]BANNER
$ DEFINE/USER SYS$OUTPUT BANNER.LIS
$ BAN3 -c# Test Message
$ PRINT/NOFEED BANNER.LIS
```

LINE PRINTER CALENDARS:

```
$ EVE CAL.SAS
options ls=81 nodate nonumber;
title ' '; proc format; value calform .=' ';
data; input day : date7. bogus @@;
  format bogus calform.; cards;
  1jan89 .  31dec89 .
proc calendar fill header=large;
  id day;
<ctrl>-Z
$ SAS CAL
$ PRINT/FORM=SMALL CAL.LIS
```

NOTE: See /pub/pictures on PLAINS.NODAK.EDU (via anonymous FTP) if you want to add graphic enhancements to these basic SAS calendars.

MAKING A PERMUTED TERM INDEX WITH SAS:

```
$ ED PERMU.SAS
*C
data sometext;
  retain line 1;  length word $30;
  infile 'SAMPLE.TXT' missover;
  input word : @@; bogus=0;
  if word=' ' then do; line+1; input; delete;
  end; realline=lag5(line);
data bogus; length word $30; do i=1 to 5;
  word=' '; bogus=1; output; end;
data combined;
  retain oldline;
  length left $30 right $30 center $30;
  set sometext bogus;
  if (bogus=1) then realline=oldline;
  else if (realline=.) then realline=1;
  oldline=realline; plus5=word;
  plus4=lag(plus5); plus3=lag(plus4);
  plus2=lag(plus3); plus1=lag(plus2);
  curword=lag(plus1); minus1=lag(curword);
  minus2=lag(minus1); minus3=lag(minus2);
  minus4=lag(minus3); minus5=lag(minus4);
  left=trim(minus5)||' '||trim(minus4)||' '||
    trim(minus3)||' '||trim(minus2)||' '||
    trim(minus1);
  center=upcase(trim(curword));
  right=trim(plus1)||' '||trim(plus2)||' '||
    trim(plus3)||' '||trim(plus4)||' '||
    trim(plus5);
proc sort; by center;
data _null_; set; file 'MYINDEX.TXT' noprint;
  firstpt=trim(left(left));
  lengleft=30-length(firstpt)+7;
  startrt=30+length(center)+10;
  if (center^=' '); page=int(realline/60)+1;
  put @1 page 5.0 @lengleft firstpt
    @38 center @startrt right;
<CTRL-Z>
*EX
$ SAS PERMU
$ PRINT MYINDEX.TXT
```

NOTE: This example assumes input is called SAMPLE.TXT, with a fixed 60 line input page length. It also assumes you want a 5 word before and after contextual format.

READABILITY LEVEL CHECKER?

We don't currently have a readability level checker equivalent to that available in the UNIX Writer's Workbench.

SCANNING TEXT AND IMAGES

The Computing Center doesn't currently offer any way of scanning text or images to disk, although a number of commercial firms in Eugene do.

MICROFICHE:

The University does not currently provide computer output microfiche (COM) service.

However, Oregon Micrographics of Eugene does offer the ability to produce microfiche from hard copy originals, including 8.5x11" and 11x14" computer printouts.

To give you a ballpark estimate of the price for this sort of thing, in December of 1989 Oregon Micrographics quoted a price of $45.00 (i.e., their minimum job charge) for producing a microfiche master of a 50 page report, with fiche duplicates running an additional $0.35/ fiche. Contact Oregon Micrographics directly for more information and current prices.

TEX:

TEX/LATEX: A WAY TO TYPESET EQUATIONS

TeX is a system for typesetting text and math equations. To use TeX here at the U of O, request an account on DECOY, a NeXT computer. For information on getting an account on DECOY, send a VAX Mail message to username JOE (JOE@OREGON) from your OREGON VAX account.

```
$ telnet decoy     <-- Go to DECOY.UOREGON.EDU
login: jsmith    <-- Enter your DECOY username
Password:     <-- Enter initial DECOY password
1 % passwd            <-- Change your password
2 % vi demo.tex               <-- Make TeX file
i
%&plain         % use basic "plain tex" format
The EOQ\footnote{$^1$}{Economic Order
Quantity} formula is widely used by inventory
managers to calculate the optimal number of
‘widgets’ to order based on the objective of
minimizing total ordering and carrying costs.

Mathematically we can write the gradual
delivery (model II) EOQ formula as:
$$EOQ =
\sqrt{{{2DS}
\over C}{\left({p}\over {p-d}
\right)}}$$
\end
<esc> :wq
3 % latex demo       <-- run latex on demo.tex
4 % dvips demo  <-- make demo.ps from demo.dvi
5 % <ctrl>-d                 <-- logout decoy
```

You can then download (in BINARY mode) the Postscript code in demo.ps and print it on a Laserwriter by using SendPS 1.2.1 from a Mac or by using LWPRINT from an Appletalk-ed PC.

NOTE: NeXT TeX Postscript transferred in ASCII mode will develop bizarre kerning.

Depending upon what files have already been sent to the Laserwriter to "prep" it, your output may generate a VMerror. If so, reset the Laserwriter by sending it:

```
systemdict /quit get cvx exec
```

WRITING A DISSERTATION USING TEX:

A TeX version of the sample pages described in the most recent ("grey cover") Oregon Thesis and Dissertation manual are available on DECOY by using the command:

```
% cp  /cc/texlib/uothesis.tex  .
```

While these pages have been reviewed by the Oregon Graduate School Editor and are believed to be formatted acceptably, you are urged to reconfirm the continued acceptability of some sample pages of **your own** manuscript in a timely manner. The raw TeX code is richly commented with references to the Graduate School Manual to help you identify the rationale for the various formatting choices employed.

PRODUCING A RESUME USING TEX

For an example of how to produce a resume using LaTex (**NOTE:** this is LaTeX, not regular TeX!) use the command:

```
% cp /cc/texlib/resume1.tex  .
```

USING TEX TO TYPESET LETTERHEAD:

For an example showing how you can use TeX to typeset letterhead stationary, enter the DECOY command:

```
% cp /cc/texlib/letterhead1.tex  .
```

USING TEX TO TYPESET BUSINESS CARDS:

You can also use TeX to typeset businesscards. See a sample of this with the command:

```
% cp /cc/texlib/cards1.tex  .
```

TAPES:

TAPE OVERVIEW:

Before you can use a tape on the VAX, it must be assigned a number and checked into the Computing Center's tape library by the Center's tape librarian. Go to the downstairs receptionist at the Computing Center to drop off your tape and fill out the required form.

When you check in your tape, your tape will be assigned a number, and you'll need to pick a tape password. You'll need that password in order to be able to do any work with that tape on the VAX. [Note that the tape password you choose should be **different** from your general account password.]

BACKUP, COPY, "FLAT" TAPES – WHICH IS RIGHT?

Beginning users are often confused over how they should work with tapes – should they use BACKUP, or COPY, or something else for a given task?

Essentially, use COPY format tapes when you have only a few files that need to be put on tape, or when ease of retrieval is of paramount performance.

Use BACKUP format tapes when you want to save a large number of files (or files with subdirectory information intact); for example, BACKUP is ideally suited for use in making a backup copy of an entire account. BACKUP also puts more data on a given reel than COPY does.

On the other hand, if you are going to be importing and exporting files to non-VAX machines, don't use either COPY or BACKUP format tapes – use either flat ASCII tape files written with CONVERT or flat EBCDIC tapes files written with the TAPES command instead. BACKUP and COPY format tapes contain VAX-specific file formatting that could make them difficult to read on IBM or CDC machines, for example.

USING VAX COPY FORMAT TAPES

You should use VAX COPY format to put files onto a tape when you have only a few files that need to be put on tape, or when ease of retrieval is of paramount importance.

Save CHIPS.DAT, and DIP.DAT to a NEW and unitialized tape (i.e., one with NOTHING which needs to be saved on it already):

```
$ MOUNT/FOR/COM="REEL:123456 WE PW:DODO"-
/DEN=6250  MU:  123456  MYTAPE:
$ LABEL MYTAPE: 123456
$ DISMOUNT MYTAPE:
$ MOUNT/COM="REEL:123456 WE PW:DODO" MU: -
123456 MYTAPE:
$ COPY/LOG CHIPS.DAT MYTAPE:
$ COPY/LOG DIP.DAT MYTAPE:
$ DIR MYTAPE:
$ DISMOUNT MYTAPE:
```

Save V8.DAT, and SNAPPYTOM.DAT to a tape which already has some files to be kept on it:

```
$ MOUNT/COM="REEL:997755 WE PW:EEKOID"  -
MU:  997755  MYTAPE:
$ COPY/LOG V8.DAT MYTAPE:
$ COPY/LOG SNAPPY_TOM.DAT MYTAPE:
$ DIR MYTAPE:
$ DISMOUNT MYTAPE:
```

See what's on a copy format tape:

```
$ MOUNT/COM="REEL:188176 PW:CUPA"  -
MU:  188176  MYTAPE:
$ DIR MYTAPE:
$ DISMOUNT MYTAPE:
```

Retrieve DIP.DAT, and SALSA.DAT:

```
$ MOUNT/COM="REEL:234122 PW:JUPI"  -
MU:  234122  NUTHER:
$ COPY/LOG NUTHER:DIP.DAT    []
$ COPY/LOG NUTHER:SALSA.DAT  []
$ DISMOUNT NUTHER:
```

NOTE: After you've labeled a COPY format tape, **NEVER** remount that tape /FOREIGN – it is easy to accidentally overwrite files already on the tape if you do so.

WORKING WITH BACKUP FORMAT TAPES:

Use BACKUP when you need prophylactic copies of files as a safegaurd against loss.

Save all files to a NEW tape in BACKUP format. NOTE that labelling and/or BACKUP/REWINDing the tape will DESTROY anyting currently on it!

```
$ MOUNT/FOR/COM="REEL:561127 WE PW:PLUMS"-
/DEN=6250  MU:  561127  MYTAPE:
$ SET DEF SYS$LOGIN
$ BACKUP/LOG/LABEL=561127/REWIND -
[...]*.*.*  MYTAPE:28AUG1888.BCK
$ DISMOUNT MYTAPE:
```

Add all files from the current account to an OLD backup tape that has already been initialized:

```
$ MOUNT/FOR/COM="REEL:561127 WE PW:PLUMS" -
MU: 561127 MYTAPE:
$ SET DEF SYS$LOGIN
$ BACKUP/LOG/LABEL=561127 -
[...]*.*.*  MYTAPE:30AUG1888.BCK
$ DISMOUNT MYTAPE:
```

Find out what files are on a backup tape:

```
$ MOUNT/FOR/COM="REEL:562212 PW:TENNIS" -
MU: 562212 MYTAPE:
$ SET MAGTAPE/REWIND MYTAPE:
$ BACKUP/LIST MYTAPE:*.*
$ DISMOUNT MYTAPE:
```

Restore only a couple of files from tape:

```
$ MOUNT/FOR/COM="REEL:208765 PW:GHANA" -
MU: 208765 MYTAPE:
$ BACKUP/LOG MYTAPE:2SEP1988.BCK/SELECT=-
(ELITE.DAT,BOURG.DAT) []
$ DISMOUNT MYTAPE:
```

Restore an entire saveset from tape to the same account it was made from:

```
$ MOUNT/FOR/COM="REEL:207168 PW:HIROHITO" -
MU: 207168 MYTAPE:
$ BACKUP/LOG MYTAPE:30OCT1977.BCK/SELECT=-
([JSMITH...])  [JSMITH...]
$ DISMOUNT MYTAPE:
```

BEST CHOICES FOR TAPES FROM OFF SITE:

```
o RECORDING DENSITY:
  Preferred -- 6250 or 1600 BPI
  Unacceptable -- any other density
o CHARACTER SETS:
  Preferred -- ASCII
  Sometimes Acceptable(*) -- EBCDIC
  Unacceptable -- vendor specific character
  sets (such as HP ASCII, GFRIC ASCII, etc).
o LABELS:
  Preferred -- unlabelled
  Acceptable -- labelled
o LOGICAL RECORD LENGTH (BYTES/RECORD):
  Preferred -- 80 byte fixed length records
  Minimum -- 1 byte fixed length records
  Recommended Maximum -- 255 bytes
  ANSI Standard Maximum -- 2048 bytes
  Absolute VAX Maximum -- 65,534 bytes
  Variable length records -- from 4 to 9999
    bytes (including 4 byte rec. control word)
o BLOCKING:
  Preferred -- unblocked, or blocked at 2048
   (or less) bytes/block
  Unacceptable -- blocks larger than 65,534
o RECORD FORMAT:
  Preferred -- fixed length records
  Acceptable -- ANSI standard variable length
  Unacceptable -- variable length EBCDIC
o PHYSICAL REEL FORMAT:
  Preferred -- standard 2400 foot 11" reels
  Acceptable -- 1200 foot or 600 foot reels
  Unacceptable -- other formats such as TK50,
    DECtape, 3480 or DC data cartridges, etc.
o TRACKS:
  Preferred -- 9 track
  Unacceptable -- 7 track
o PARITY:
  Preferred -- Odd parity
  Unacceptable -- anything else
o SPECIAL CHARACTERISTICS:
  Preferred -- VAX/VMS Backup or Copy Format,
    or a flat plain file format
  Unacceptable -- any other vendor specific
    format such as IBM CMS TAPE LOAD/DUMP
    format. System specific package-prepared
    files are also generally unacceptable
    (e.g., non-VMS, non-export SAS data sets).
```

READING FLAT ASCII TAPES WITH CONVERT:

Use CONVERT when you need to read or write out an ASCII file in a specific format (most typically because the file is coming from, or going to, some other system). Assume you've received a tape with flat 300 character ASCII records:

```
$ ED/FDL F300           <-- Make a new FDL file
<return>                         <-- Proceed to
I                      <-- invoke an FDL script
S                    <-- for a sequential file,
0                    <-- unlimited num of recs,
F                             <-- fixed length,
Y                    <-- let rec's span blocks,
300                 <-- records 300 chars long,
<return>                    <-- (immaterial)
<return>                    <-- (immaterial)
<return>              <-- car. ret. car. ctrl.
EX                   <-- save the new fdl file!
```

```
$ MOUNT/FOR/COM="REEL:241121 PW:EDGAR" -
MU: 241121 MYTAPE:
$ CONV/FDL=F300 MYTAPE: BIGFILE.DAT
$ DISMOUNT MYTAPE:
```

You can also use CONVERT to write out flat tapes for export to other sites. For example, assume we have a new tape on which we want to write out a 300 character flat ASCII file (as described in the above FDL file), padding short records with spaces and truncating any long ones:

```
$ MOUNT/FOR/COM="REEL IS 54121 WE:BEESWAX"-
/DEN=6250  MU:  54121  MYTAPE:
$ LABEL MYTAPE: 54121
$ CONV/FDL=F300/PAD=%X20/TRUNC -
BASKET.DAT MYTAPE:
$ DISMOUNT MYTAPE:
```

Note that while you'd like to be able to write two tape marks at the end of the tape to signal a logical end-of-tape condition, the SET MAGTAPE/END command that you **should** be able to use won't work due to a known bug!

WORKING WITH SCANNER (EBCDIC) TAPES:

Read an option 1 ("scan to tape") tape from Scanner Services (or many other EBCDIC tapes):

```
$ @SYS$SYSTEM:UNSUPPORTED_SETUP.COM

$ MOUNT/FOR/COM="REEL:565132 PW:REACTOR" -
MU: 565132 MYTAPE:
$ TAPES MYTAPE:/OUT=SCANNER.DAT/TOASC/REWIND-
/FILE=1:1/RECLEN=275
$ DISMOUNT MYTAPE:
```

COLUMNS	CONTENTS (LENGTH)
10-13	Case number (1, 2, 3, ...)
26-45	Name (20)
46	Sex
55-64	ID Num (10)
65-70	Spec Code (6)
71-270	Questions 1-200 5-blit-form (200)
71-190	Questions 1-120 10-blit-form (120)

NOTE: The following Fortran code can be used to break the 275 character-long scanner records into 80 character chunks that can be edited using EDT or other editors.

```
$ ed chunkit.for
*C
<tab> implicit none
<tab> character*275 rekord
<tab> open (unit=3,file='scanner.dat',
<tab> 1 status='old',readonly)
<tab> open (unit=4,file='newscan.dat',
<tab> 1 status='unknown',
<tab> 2 carriagecontrol='list')
15 <tab> read (3,'(a275)',end=999) rekord
<tab> write (4,'(a80)') rekord(1:80)
<tab> write (4,'(a80)') rekord(81:160)
<tab> write (4,'(a80)') rekord(161:240)
<tab> write (4,'(a35)') rekord(241:275)
<tab> goto 15
999 <tab> end
<CTRL-Z>
*EX

$ for chunkit.for
$ link chunkit.obj
$ run chunkit.exe
```

SET MAGTAPE, DUMP AND SIMILAR THINGS:

In addition to the "standard" tape access procedures shown in the preceding panels, sometimes you need additional tape manipulation tools. The following commands all assume you've mounted your tape /FOREIGN, with a logical name of MYTAPE:

REWINDING TAPES:

```
$ SET MAGTAPE/REWIND MYTAPE:
```

POSITIONING TAPES:

```
$ SET MAGTAPE/SKIP=FILE:3 MYTAPE:
$ SET MAGTAPE/SKIP=END_OF_TAPE MYTAPE:
```

DUMPING INFORMATION FROM A TAPE:

```
$ DUMP/BLOCK=COUNT:2 MYTAPE:
$ DUMP/BLOCK=COUNT:10/OUT=TEMP.LIS MYTAPE:
```

COPYING TAPE TO TAPE:

The TAPES command can be used to directly copy one tape to another, provided that the tapes have only "flat" files on them

NOTE: This procedure will **NOT** work for BACKUP format tapes!

```
$ MOUNT/FOR/COM="REEL IS 888111 WE:ALEX"  -
MU:  888111  MYTAPE2:    <-- mount target tape
$ LABEL MYTAPE2: 888111  <-- label target tape
$ DISMOUNT MYTAPE2:   <-- dismount target tape

$ MOUNT/FOR/COM="REEL IS 888111 WE:ALEX"  -
MU:  888111  MYTAPE2:  <-- remount target tape
$ MOUNT/FOR/COM="REEL IS 123456 PW:BOB"  -
MU:  123456  MYTAPE1:  <-- mount original tape
$ TAPES MYTAPE1:/OUT=MYTAPE2:      <-- copy it
$ DISMOUNT MYTAPE1: <-- dismount original tape
$ DISMOUNT MYTAPE2:   <-- dismount target tape
```

SOME TAPE TIPS:

– The password you use on your tape should NOT be the same password you use to login! If you've mistakenly given one or more of your tapes your login password, use the $ SET PASS command to change your login password to something new!

– If the data you're writing to tape is valuable, don't make just one copy of it. Tape drives have been known to occaisionally damage tapes, and computer tapes are subject to normal wear and tear just like anything else. Fifteen bucks for a second tape is cheap insurance.

– If you're mailing a tape, pack it securely, protecting it against wetness, crushing and dropping. Be sure to label the box as containing magnetic media and note that it should be kept away from magnetic fields and strong X-ray sources. Did you remember to include documentation with the tape explaining what's on it, how the tape was written, and who can be contacted if technical problems arise?

– If you're flying with a magnetic tape, hand carry the tape and request the security guards to "hand inspect" the tape at the security checkpoint. They will attempt to tell you that it is safe to send the tape through their X-ray machine (and it very well may be), but I wouldn't rely on their assurances. If need be, ask to talk to a supervisor. Be sure to allow extra time in case of security delays.

Also note that export of certain software is strictly controlled, and you may experience customs delays if your destination is overseas.

CD READERS?

Even though an increasing number of datasets are now being distributed on CD Discs, we currently have no ability to read CD's on the VAX. Sorry!

USING MS-KERMIT WITH THE VAX:

USING MS KERMIT AS A TERMINAL EMULATOR:

MS Kermit is the most commonly used PC terminal emulation package at the University of Oregon. It is available free, and provides good VT100 and Tektronix 4010 (monochrome graphics) emulation.

Obtain a copy of the current version of MS-Kermit by bringing a blank floppy disk to 235 Computing Center. We'll swap disks with you, giving you one with a copy of Kermit on it in exchange for your blank.

In order to properly configure your new copy of Kermit we need to know three things: (1) we need to know the baud rate of your modem or hard wire connection (i.e., 1200 baud, 2400 baud, or whatever); (2) we need to know which communication port your PC uses, either COM1 or COM2; (3) finally, we need to know whether you'll be using Kermit on a hard disk system or a floppy disk system.

The Kermit you'll receive will be configured for your machine, and will use the JOE.INI keypad mapping scheme (see below). You must have <NUMLOCK> **ON** for the JOE.INI mapping to work properly!

Start MS Kermit by saying:

```
C:\> KERMIT
```

When the status bar appears at the bottom of your screen, you can use the modem commands shown on the next panel to connect to the VAX. [If you're directly connected via UONet, just hit a couple of returns; if you're directly connected via the Gandalf, hold down the control key and hit a couple of breaks, then hit return a couple of times.]

```
To quit Kermit, use:  <ctrl>-] C QUIT <return>
```

[N.B. While Kermit 3.0 is now available for the PC, we'll probably be staying with 2.32/A for the immediate future.]

SOME HAYES-COMPATIBLE MODEM COMMANDS:

Hayes compatible modems use "AT" commands to dial and execute various modem commands. Enter these modem commands from the keyboard once you've started Kermit and have the status bar on the bottom of your screen.

```
AT                          <-- Modem should reply
                    "OK" if in local mode with
                    verbose comm codes enabled

ATDT 555-1234                  <-- Dial with tones
ATDP 555-5678                 <-- Dial with pulses
A/           <-- repeat last command (redial)

<pause> +++ <pause>    <-- escape to the modem

ATH                                   <-- hang up
ATZ        <-- reset modem (also hangs you up)


ATL0    <-- set speaker to lowest volume (off)
ATL3         <-- set speaker to highest volume

ATM0                      <-- speaker always off
ATM1       <-- speaker on until Carrier Detect
ATM2                       <-- speaker always on
ATM3 <-- speaker on while dialing and after CD

ATE1     <-- echo all characters in local mode
ATQ0                       <-- send result codes
ATV1         <-- display result codes as words
ATX4          <-- select extended result codes

AT&V     <-- view stored configuration profile
ATT         <-- select tone dialing by default
AT&Z0=0 346-3565     <-- store phone number #0
ATS0=0                  <-- disable auto answer
AT&W0          <-- write nonvolatile config #0
ATDS=0         <-- dial using stored config #0
```

NOTE: Enter AT commands all uppercase. In the preceding commands 0 is the number zero, not the letter O, and 1 is the number one, not the letter I.

Some modems may not implement all of the AT commands shown – check your modem manual for details if you encounter problems.

KERMIT 2.32/A JOE.INI KEYPAD MAPPING

NOTE: YOU **MUST** HAVE <NUMLOCK> ON!

VT100 KEY	KERMIT KEY
DELETE	BACKARROW
BACKSPACE	<shift> BACKARROW
CURSOR KEYS	<shift> KEYPAD ARROWS
KEYPAD 0-9	KEYPAD 0-9
KEYPAD DASH	KEYPAD DASH
KEYPAD COMMA	KEYPAD STAR or <PRT SCRN>
KEYPAD ENTER	KEYPAD PLUS
KEYPAD DOT	KEYPAD DOT
PF1 - PF4	F1 - F4

```
Leave TEK 4010 mode:          <alt>-<DASH>

Screen rollback:              <shift>-<PG UP>
Screen rollforward:           <shift>-<PG DOWN>
Beginning of screen buffer:   <shift>-<HOME>
End of screen buffer:         <shift>-<END>

Print Screen:                 <shift>-<PRT SCRN>
Toggle Printer On/Off:        <ctrl>-<PRT SCRN>

Go to MS-Kermit mode:         <ctrl>-] C
Reconnect:                    C

Dump scr to KERMIT.SCN        <ctrl>-] F
Log to KERMIT.LOG             <ctrl>-] C LOG SES
Pause logging                 <ctrl>-] Q
Resume logging                <ctrl>-] R
Terminate logging             <ctrl>-] C CLO SES

Push to DOS (EXIT returns)    <ctrl>-] P
Send Long Break Signal        <ctrl>-] L
Toggle Mode Line Display      <ctrl>-] M
Kermit Help Screen            ALT-H
Kermit Settings Screen        ALT-S

<ctrl>-] C  SET TERM ?    <-- see term options
<ctrl>-] C  SET TERM KEYCLICK ON
<ctrl>-] C  SET TERM COLOR 31  <-- red letters
<ctrl>-] C  SET TERM COLOR 37      <-- white "
<ctrl>-] C  SET TERM COLOR 41  <-- red bckgrnd
<ctrl>-] C  SET TERM COLOR 40      <-- black "
```

UPLOADING TEXT FROM A PC USING KERMIT:

```
A:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT     <-- get the VAX ready to receive
Kermit-32> RECEIVE
<ctrl>-] C       <-- have the PC send the file
Kermit-MS> SEND B:KUKES.DAT
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q               <-- exit VMS Kermit
```

DOWNLOADING TEXT TO A PC USING KERMIT:

```
A:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT         <-- get the VAX ready to send
Kermit-32> SEND SUDS.DAT
<ctrl>-] C       <-- tell the PC to receive it
Kermit-MS> RECEIVE B:\
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q               <-- exit VMS Kermit
```

UPLOADING BINARY FILES USING KERMIT:

```
A:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT     <-- get the VAX ready to receive
Kermit-32> SET FILE TYPE BINARY
Kermit-32> RECEIVE
<ctrl>-] C       <-- have the PC send the file
Kermit-MS> SEND B:KUKES.DAT
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q               <-- exit VMS Kermit
```

DOWNLOADING BINARY FILES USING KERMIT:

```
A:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT         <-- get the VAX ready to send
Kermit-32> SET FILE TYPE BINARY
Kermit-32> SEND SUDS.DAT
<ctrl>-] C       <-- tell the PC to receive it
Kermit-MS> RECEIVE B:\
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q               <-- exit VMS Kermit
```

USING SERVER MODE TO MOVE FILES:

GET FILES FROM THE VAX TO THE PC:

```
A:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT             <-- Start Kermit on the VAX
Kermit-32> SERVER        <-- Go to SERVER mode
<ctrl>-] C             <-- escape to PC Kermit
Kermit-MS> GET              <-- GET a VAX file
Remote Source File:  XYZ.FOR
Local Destination File:  C:\XYZ.FOR
Kermit-MS> <more server commands if desired>
Kermit-MS> FINISH      <-- to quit SERVER mode
Kermit-MS> C         <-- Reconnect to the VAX
Kermit-32> Q               <-- Quit VAX Kermit
```

SEND FILES FROM THE PC TO THE VAX:

```
A:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT             <-- Start Kermit on the VAX
Kermit-32> SERVER        <-- Go to SERVER mode
<ctrl>-] C             <-- escape to PC Kermit
Kermit-MS> SEND             <-- SEND a PC file
Local Source File:  C:\ABC.DAT
Remote Destination File:  ABC.DAT
Kermit-MS> <more server commands if desired>
Kermit-MS> FINISH      <-- to quit SERVER mode
Kermit-MS> C         <-- Reconnect to the VAX
Kermit-32> Q               <-- Quit VAX Kermit
```

OTHER SERVER COMMANDS:

```
Kermit-MS> REMOTE DIR    <-- see your VAX DIR
Kermit-MS> REMOTE CD [.DEMO]   <-- change dir
Kermit-MS> REMOTE TYPE PD.TXT <-- type a file
Kermit-MS> REMOTE SPACE   <-- check diskquota
Kermit-MS> REMOTE WHO     <-- see SH PROC/ALL
```

NOTE: Don't use BYE to exit SERVER mode unless you want to be logged all the way off the VAX!

DOING NON-PROTOCOL FILE TRANSFERS:

While file-at-a-time and server mode transfers are by far the most common way of moving files to and from the VAX, there may be times when you simply want to move plain text files **without** any kind of error-checking protocol.

Be aware that non-protocol transfers may include undetected transmission errors – "real" text may be dropped (or "garbage" added to the text) without any warning whatsoever.

UPLOADING A TEXT FILE:

```
A:> KERMIT         <-- startup Kermit on the PC
<...logon as you normally would here...>
$ SET TERM/HOST    <-- allow VAX to valve flow
$ ED STUFF.DAT         <-- make a new VAX file
*I                        <-- goto insert mode
<ctrl>-] C             <-- escape to PC Kermit
Kermit-MS> TRANSMIT C:\STUFF.DAT   <-- send it
Kermit-MS> C              <-- reconnect to VAX
<ctrl>-Z             <-- exit from insert mode
*EX                          <-- exit from EDT
```

DOWNLOADING A TEXT FILE:

```
A:> KERMIT         <-- startup Kermit on the PC
<...logon as you normally would here...>
$ TYPE WAZOO.DAT     <-- DON'T hit return yet!
<ctrl>-] C             <-- escape to PC Kermit
Kermit-MS> LOG SES C:\WAZOO.DAT  <-- open file
Kermit-MS> C              <-- reconnect to VAX
<RETURN>              <-- execute TYPE command
<file displays>
<ctrl>-] C             <-- escape to PC Kermit
Kermit-MS> CLOSE SES    <-- close session file
Kermit-MS> C              <-- reconnect to VAX
```

NOTE: You CANNOT transfer binary (.EXE) files using non-protocol commands!

Note, too, that some special characters (including the control zee at the end of MS-DOS files) can cause problems when transferred using the non-protocol approach.

WHY NOT PROCOMM?

Users occaisionally ask why we support Kermit (rather than PROCOMM) for the PC:

There are a number of reasons:

(1) **COST:** We can give you a copy of Kermit for free, and you can use it for free forever. PROCOMM, on the other hand, is "user supported software:" you are obligated to pay the registration fee established by PROCOMM's authors if you elect to continue to use that package beyond a brief trial period.

(2) **KEYPAD SUPPORT:** Another important difference is that Kermit allows the PC keyboard to be completely remapped to closely emulate a VT100 terminal's keypad layout, while the shareware version of PROCOMM does not. Because EDT and EVE (as well as many other VAX applications) rely on user access to a VT100-like keypad, the lack of a viable VT100 keypad in the shareware version of Kermit is a serious deficiency for most VAX users.

(3) **GRAPHIC TERMINAL EMULATION:** Because many of our users generate mainframe graphics using SAS/Graph or DISSPLA, there is substantial interest in being able to preview TEK4010 graphics output on screen. Kermit can easily display TEK4010 graphics while the shareware version of PROCOMM cannot.

(4) **INSTALLED BASE:** We have a large historical installed base of Kermit users we are committed to supporting. Because we have limited PC communication support resources, attempting to support another communication package would severely impact our current communication support efforts.

X/Y/Z MODEM TRANSFERS?

MS-Kermit does not support up/downloads using the XMODEM, YMODEM or ZMODEM file transfer protocols, nor does the Computing Center currently provide VAX routines for those protocols. You are encouraged to use the Kermit file transfer protocol, instead.

KERMIT TRANSFER FAILURES AT 2400 BAUD+

If you encounter Kermit protocol file transfer failures (particularly at the higher baud rates), one thing that may help you successfully complete your transfer is to increase the number of retries Kermit will make before aborting a transfer. By default, Kermit will try four times to retransmit a garbled block; to increase the number of times it will retry sending a bad block up to the maximum of 63, try:

```
Kermit-MS> set retry 63  <-- 63 is max allowed
```

CANCELLING CALL WAITING:

The phone company's optional call waiting service [intended to alert you to a second caller while you're already on the phone] may cause your modem to hang up in the middle of a session with the VAX.

To prevent this, prefix your call to the VAX with the magic "cancel call waiting for this call" string, which is *70 <pause for two seconds> on Touch-Tone lines, or 1170 <pause for two second> on rotary lines. Normal call waiting service is restored automatically when you hang up.

```
For Kermit try: ATDT *70,,346-3565
```

NCSA TELNET:

WHEN SHOULD I USE PC NCSA TELNET?

Use NCSA TELNET for the PC only when your PC is directly connected to the campus ethernet via a 3Com ethernet card. If you are connecting to the VAX any other way (such as via a modem, through a hardwired Gandalf port, or through a UONet terminal server), you'll want to use Kermit instead.

LOGGING ON USING PC NCSA TELNET:

```
C:> TELNET OREGON
<logon as you normally would; you're a VT100!>
```

TRANSFERRING FILES USING PC NCSA TELNET:

```
C:> FTP OREGON
Login: JDOE                        <-- VAX Username
Password: BLACKY                   <-- VAX Password
*PUT C:\SAMPLE.DAT                <-- PUT: PC-->VAX
*GET [JDOE]EXAMPLE.FOR            <-- GET: VAX-->PC
*QUIT
C:>
```

See the FTP section for more information.

PC NCSA TELNET VAX KEYPAD MAPPING:

VT100 KEY	PC NCSA TELNET KEY
DELETE	BACKARROW
BACKSPACE	<ctrl> BACKARROW
CURSOR KEYS	KEYPAD ARROWS
KEYPAD 0-9	<alt> TOP ROW NUMBERS 0-9
PF1 - PF4	F1 - F4
KEYPAD DASH KEY	F5
KEYPAD COMMA KEY	F6
KEYPAD ENTER KEY	F10
KEYPAD DOT KEY	KEYPAD DEL KEY

PC NCSA AND EDT:

NOTE: NCSA maps the PC keyboard ONLY for EDT. Comparable mappings are NOT currently available for EVE or other editors.

FUNCTION	PC NCSA KEY
SAVE+EXIT	F9
SELECT	F7
CUT	F8
PASTE	F1 then F8
SCROLL UP	PG UP
SCROLL DOWN	PG DN
DEL CHAR	KP .
DEL WORD	F5

MANAGING YOUR PC NCSA SESSION:

COMMAND:	DOES:
ALT-H	Shows these commands
ALT-M	See the Console Message Screen
ALT-A	Adds a new telnet session
ALT-X	Exit current telnet session
ALT-N	Next telnet session
ALT-T	Start FTP File Transfer
ALT-E	Shell to DOS (exit returns to NCSA) [N.B.: DO *NOT* use SIDEKICK, FTP, FORMAT, or network prgms in shell!)
ALT-C	Capture toggle (use PRN for print)
HOME	Return to VT102 mode from TEK4010
ALT-P	Change NCSA Parameters
ALT-G	Graphics Menu
ALT-R	Reset VT102
ALT-S	Shoot ahead to end of lots of output
ALT-F3	Abort NCSA (try NOT to use this!)

WHEN SHOULD I USE MAC NCSA TELNET?

Use NCSA TELNET for the Mac only when your Mac is directly connected via Appletalk or Ethertalk. If you are connecting to the VAX any other way (such as via a modem, through a hardwired Gandalf port, or through a UONet terminal server), you'll want to use version 10.3 of Red Ryder or Mac Kermit instead.

USING NCSA TELNET FOR THE MACINTOSH:

(1) Launch NCSA Telnet by double clicking on the NCSA Telnet icon.

(2) Click and hold on FILE; come down to OPEN CONNECTION and release. Enter the host (to be connected to) and hit return.

(3) Proceed to logon.

(4) Click and hold on NETWORK; come down to SEND FTP COMMAND and release.

(5) Click and hold on FILE; come down to SET TRANSFER DIRECTORY. Choose the correct MAC origin/destination for files to be transferred.

(6) You are now ready to transfer files. If you want to go VAX–>MAC, use the PUT command. On the other hand, if you want to go MAC–>VAX, use the GET command.

(7) When you're done moving files, issue the CLOSE command to end FTP, then logoff the VAX. That's all there's to it!

PROBLEMS? Have you checked "FTP Enable" in the Files menu? Do you have an FTP password file specified in the config.tel file? If so, have you set a password with the telpass program?

MISCELLANEOUS MICROCOMPUTER TOPICS:

UPLOADING A 1-2-3 WORKSHEET:

(1) Start 1-2-3 and retrieve your worksheet with /FR worksheetname.

(2) Use /WCS to crunch the size of each column as much as you can without getting ***'s. (If you can't scrunch the total width of the worksheet to 240 characters or less, you'll have to move the worksheet in vertical "strips" and then "reassemble" the strips once the data is on the VAX. Also, note that editing files wider than 80 characters can be awkward in EDT on the VAX – strive for a file no wider than 80 characters if you can possibly do it!)

(3) Now print the worksheet to a file:

```
/PF                                  <-- Print File
 TEMP                   <-- print file is TEMP.PRN
 R                                        <-- Range
 A1..S100              <-- select A1..S100, say --
                      (note -- this is where you'd
                    split the files into strips if
                   it is over 240 characters WIDE)
 OML0                     <-- Option Margin Left 0
  MR240                       <-- Margin Right 240
  MT0                             <-- Margin Top 0
  MB0                          <-- Margin Bottom 0
  P100                          <-- Pagelength 100
  OU                          <-- Other Unformated
  Q                  <-- Quit (print options menu)
  G                                          <-- Go
  Q                          <-- Quit (print menu)
```

(4) Proceed to upload TEMP.PRN:

```
A:> KERMIT         <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT      <-- get the VAX ready to receive
Kermit-32> RECEIVE
<ctrl>-] C        <-- have the PC send the file
Kermit-MS> SEND B:TEMP.PRN
Kermit-MS> C             <-- reconnect to the VAX
Kermit-32> Q                  <-- exit VMS Kermit
```

NOTE: You may need to remove a <CTRL>-Z from the very end of the file

UPLOADING A PC EXCEL WORKSHEET:

(1) Start Excel and retrieve your worksheet [use File Open, giving your worksheet name]. Make sure any missing data has been given a code (such as -9) if the data is to be read into SPSSx, or a single period (.) if the data is to be read into SAS.

(2) Click on File, and come down to Save As.

(3) Click on Options. Now click on CSV (comma seperated variables).

(4) Now click on the end of the filename, and enter C:\TEMP.CSV or whatever you'd like to call the flat file you're creating.

(5) Click on OK to execute the SAVE AS.

(6) Quit Excel by clicking on File Exit.

(7) Proceed to upload C:\TEMP.CSV

```
C:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT      <-- get the VAX ready to receive
Kermit-32> RECEIVE
<ctrl>-] C       <-- have the PC send the file
Kermit-MS> SEND C:\TEMP.CSV
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q               <-- exit VMS Kermit
```

NOTE: You may need to remove a <CTRL>-Z from the very end of the file.

(8) On the VAX, read the CSV files in free format. [See pps. 63-65 of the 3rd Edition SPSSx Users Manual, or SAS Basic Manual Version 5th edition pps. 134-135].

NOTE: Before you can read a CSV file in SAS, you must first edit it a little since SAS gags on commas:

```
$ ED TEMP.CSV     <-- Edit the CSV format file
*S/,/  /W/NOT <-- Replace commas with 2 spaces
*EX             <-- Resave the CSV format file
```

UPLOADING PC MS-WORD DOCUMENTS:

METHOD 1: FILES THAT HAVE HARD RETURNS

(1) Start Word and load your document with <ESC> TL documentname.

(2) Confirm carriage returns are present at the end of each line using <ESC> O V Partial

(3) Save your document (unformatted) using:

```
<ESC> TS TEMP.DOC <TAB> <SPACE>
```

(4) Upload TEMP.DOC like any other file:

```
A:> KERMIT        <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT      <-- get the VAX ready to receive
Kermit-32> RECEIVE
<ctrl>-] C       <-- have the PC send the file
Kermit-MS> SEND B:TEMP.DOC
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q               <-- exit VMS Kermit
```

NOTE: You may need to remove a <CTRL>-Z from the very end of the file.

METHOD 2: TEXT WITHOUT HARD CARRIAGE RETURNS

(1) Set your margins with

```
<ESC> FDM
```

so that top=0" bottom=0" right=0" left=0".

(2) Set your printer type to plain printer (TTY):

```
<ESC> PO Printer: TTY
```

(3) Print the document to a file:

```
<ESC> PF TEMP.PRN
```

(4) Upload TEMP.PRN as show in part (4) above.

NOTE: You may need to remove a <CTRL>-Z from the very end of the file.

UPLOADING PC WORD PERFECT DOCUMENTS:

Start Word Perfect, naming the document you want to upload:

C:> WP XYZ.DAT

Hit <CTRL-F5> (Text In/Out)

Hit a 1 (Dos Text)

Hit a 1 (Save)

Enter a NEW filename: C:\TEMP.DAT <RETURN>

Hit F7 to exit Word Perfect; enter Y to confirm exit. [NOTE: If Word Perfect asks you if you'd like to save your document, DO NOT re-use the same name you used above!]

Upload TEMP.DAT like any other file:

```
C:> KERMIT          <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT        <-- get the VAX ready to receive
Kermit-32> RECEIVE
<ctrl>-] C        <-- have the PC send the file
Kermit-MS> SEND C:\TEMP.DAT
Kermit-MS> C            <-- reconnect to the VAX
Kermit-32> Q                 <-- exit VMS Kermit
```

NOTE: You may need to remove a <CTRL>-Z from the very end of the file.

UPLOADING DBASE .DBF FILES:

METHOD 1: FIXED FORMAT

```
C:\>DBASE

.USE C:\TRIAL.DBF
.COPY TO C:\TEMP.DAT SDF
.QUIT

C:\> KERMIT       <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT      <-- get the VAX ready to receive
Kermit-32> RECEIVE
<ctrl>-] C       <-- have the PC send the file
Kermit-MS> SEND C:\TEMP.DAT
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q               <-- exit VMS Kermit
```

NOTE: You may need to remove a <CTRL>-Z from the very end of the file.

METHOD 2: FREE FORMAT

```
C:\>DBASE

.USE C:\TRIAL.DBF
.COPY TO C:\TEMP.DAT DELIMITED WITH BLANK
.QUIT

C:\> KERMIT       <-- startup Kermit on the PC
<...logon as you normally would here...>
$ KERMIT      <-- get the VAX ready to receive
Kermit-32> RECEIVE
<ctrl>-] C       <-- have the PC send the file
Kermit-MS> SEND C:\TEMP.DAT
Kermit-MS> C          <-- reconnect to the VAX
Kermit-32> Q                <--exit VMS Kermit
```

NOTE: You may need to remove a <CTRL>-Z from the very end of the file.

UPLOADING PC SAS DATASETS:

NOTE: Confirm that you have the right baud rate and com port set in C:\SAS\RLINK\VMS.SCR Also make sure AUTOEXEC.SAS includes the line:

```
FILENAME RLINK 'C:\SAS\RLINK\VMS.SCR';
```

(1) Logon with Kermit as you normally would. At the "$" quit Kermit but DON'T LOG OFF:

```
$                         <-- normal VAX prompt
<CTRL>-] C                <-- exit to MS-Kermit
MS-Kermit> QUIT                 <-- quit Kermit
C:>                        <-- normal PC prompt
```

(2) Now start SAS/PC:

```
C:> SAS
```

(3) When the editor screen comes up, enter:

```
00001 OPTIONS REMOTE=ASYNC1;
00002 LIBNAME ONPC 'C:\';  RUN;
```

SUBMIT these commands by hitting <F10>.

(4) Now SIGNON remote SAS:

```
=====> SIGNON
```

(5) Declare your VMS libname; do the UPLOAD:

```
00001 LIBNAME ONVMS '[]';
00002 PROC UPLOAD DATA=ONPC.FILE1
00003   OUT=ONVMS.FILE1; RUN;

======> RSUBMIT
```

(6) When done SIGNOFF and exit SAS:

```
=====> SIGNOFF
=====> ENDSAS
```

(7) Now re-start Kermit and logoff the VAX:

```
C:> KERMIT
$ LOGOFF
```

MACINTOSH MISSING VALUE PROBLEMS:

Users uploading Macintosh data files from StatView 512+ (and possibly other applications as well) may be surprised to find that they have missing values that look like <Y-> when viewed on the VAX in EDT. Users will often be even more surprised to find out just how arcane a a global search and replace on that special character can be (and the <Y-> IS just one character, an ASCII 165 decimal).

The simplest approach to fixing this problem is to do it in two steps:

First, use the MACMISS command to replace the <Y->'s with the much more tractable character "#". Then, use your favorite text editor to do a global search and replace on the "#" characters, changing them to an acceptable missing value (such as -99 for SPSSx).

For example:

```
$ SET COMMAND DISK$USER1:[JOE.MACMISS]MACMISS
$ MACMISS FOO.RAW FOO.DAT
$ ED FOO.DAT
*S/#/-99/W
*EXIT
```

NOTE: The MACMISS command can handle variable length records up to 2048 characters in length

MISCELLANEOUS TOPICS:

THE COMPUTING CENTER VAXES:

The following summary includes information on both OREGON and DONALD because some resources (such as printers and tape drives) are shared by all machines comprising the VAX cluster.

SYSTEM NAME:	OREGON	DONALD
MAIN USE:	ACADEMIC	ADMINISTRATIVE
SYSTEM:	VAX 8800	VAX 6430
VAX "MIPS:"	2 x 6.1 MIPS	3 x 7.0 MIPS
MAIN MEMORY:	112 MB	128 MB
DISK:	18.2 Gigabytes	
TAPE DRIVES:	3 125 IPS 1600/6250 BPI drives	
PRINTERS:	1 1200 LPM Band, 1 2000 LPM Band 1 NON-PostScript Laser Printer	
PLOTTER:	1 Zeta 8	
DIALINS:	16 300/1200 baud, 8 2400 baud	

COMPUTING RESOURCES ON CAMPUS:

A common question is "Where can I go if I just want to use a Mac or a PC for a couple of hours to write a term paper?"

The Computing Center does not provide any public access general purpose microcomputers labs. However, general purpose hourly microcomputer **are** available at the following locations:

The EMU Computer Lounge (Fee charged; Macs, Kaypro AT clones, Apple IIs, dot matrix and laser printers, plotter)

The Chiles Computing Center (Fee charged; Macs, HP Vectra AT clones, printers)

Social Science Instructional Lab (Basement PLC) (Fee charged; created Fall 1989)

The Knight (Main) Library (Free; 2 Macs, Imagewriter)

The Math Library (Free; 3 Apple IIs)

SUMMARY OF VAX 8800 CHARGES:

CPU Time (per CPU minute M-F 8AM-5PM)	$1.80
CPU Time (M-F 5-11PM; Sun 4-11PM)	$0.90
CPU Time (all other periods)	$0.45
CPU Time (LOW$BATCH batch jobs)	$0.09
Connect Time (per hour M-F 8AM-5PM)	$0.96
Connect Time (M-F 5-11PM; Sun 4-11PM)	$0.48
Connect Time (all other periods)	$0.24
Disk Space (per 1000 blocks per month)	$3.00
Printout (per 100 pages--stock forms)	$6.30
Tape Rental (per tape per month)	$1.80
2400' Tape Purchase (per tape)	$15.00
Tape Storage (per tape per month)	$1.00

SOME COMPUTING CENTER PHONE NUMBERS:

Dial-in modems (300/1200 baud)	346-4391
Dial-in modems (2400 baud only)	6-3565
Computer Operator (CC Machine Room)	6-4382
Consulting Receptionist (207 CC)	6-4412
Downstairs Receptionist	6-4403
Upstairs Receptionist	6-4394
Accounting Clerk/Connie French (107 CC)	6-4404
Documents Room/Betsy Shaw (205 CC)	6-4406
E-Mail Postmaster/Rich Haller (237 CC)	6-4394
Microcomputer Sales Program (202 CC)	6-4402
Stat Consulting/Joe St Sauver (235 CC)	6-4394

THE COMPUTING CENTER NEWSLETTER:

You can get added to the mailing list for the Computing Center's monthly newsletter, **Computing News,** by sending a request to the newsletter's editor, Joyce Winslow, at VAX username JWINS

Subscriptions are free for University of Oregon users, and for interested users from other academic institutions. Invididuals not affiliated with a university or college can subscribe for $5/year to cover print, postage and handling costs.

COMPUTING CENTER HOURS:

NOTE: The Computing Center reserves the right to adjust these hours as needed for hardware or software maintenance purposes. (One days notice will usually be given).

BUILDING HOURS

```
Monday-Friday   7:30 AM - 11:45 PM
Saturday        9:00 AM -  4:45 PM
Sunday          4:00 PM - 11:45 PM
```

Beginning Sunday of week five during each term except summer, and ending the Sunday before finals, the building opens at noon on Sundays.

OPERATOR COVERAGE

```
Monday-Thursday 8:00 AM - 11:30 PM
Friday          8:00 AM - 10:00 PM
Saturday        no coverage
Sunday          4:00 PM - 11:30 PM
```

During hours without operator coverage, SYS$PRINT line printer output, SYS$PLOT plotter output and tape mounts are unavailable.

DOCUMENTS ROOM:

```
Monday-Friday  9:30 AM - 5:00 PM (term time)
Monday-Friday  1:00 PM - 4:00 PM (vacations)
```

ASCII CHARACTER TABLE:

DEC	HX	CHR	DEC	HX	CHR	DEC	HX	CHR
0	00	^@ NUL	44	2C	,	88	58	X
1	01	^A SOH	45	2D	-	89	59	Y
2	02	^B STX	46	2E	.	90	5A	Z
3	03	^C ETX	47	2F	/	91	5B	[
4	04	^D EOT	48	30	0	92	5C	\
5	05	^E ENQ	49	31	1	93	5D	]
6	06	^F ACK	50	32	2	94	5E	^
7	07	^G BEL	51	33	3	95	5F	_
8	08	^H BS	52	34	4	96	60	’
9	09	^I TAB	53	35	5	97	61	a
10	0A	^J LF	54	36	6	98	62	b
11	0B	^K VT	55	37	7	99	63	c
12	0C	^L FF	56	38	8	100	64	d
13	0D	^M CR	57	39	9	101	65	e
14	0E	^N SO	58	3A	:	102	66	f
15	0F	^O SI	59	3B	;	103	67	g
16	10	^P DLE	60	3C	<	104	68	h
17	11	^Q DC1	61	3D	=	105	69	i
18	12	^R DC2	62	3E	>	106	6A	j
19	13	^S DC3	63	3F	?	107	6B	k
20	14	^T DC4	64	40	@	108	6C	l
21	15	^U NAK	65	41	A	109	6D	m
22	16	^V SYN	66	42	B	110	6E	n
23	17	^W ETB	67	43	C	111	6F	o
24	18	^X CAN	68	44	D	112	70	p
25	19	^Y EM	69	45	E	113	71	q
26	1A	^Z SUB	70	46	F	114	72	r
27	1B	ESC	71	47	G	115	73	s
28	1C	^\ FS	72	48	H	116	74	t
29	1D	^‘ GS	73	49	I	117	75	u
30	1E	^= RS	74	4A	J	118	76	v
31	1F	^- US	75	4B	K	119	77	w
32	20	SPC	76	4C	L	120	78	x
33	21	!	77	4D	M	121	79	y
34	22	"	78	4E	N	122	7A	z
35	23	#	79	4F	O	123	7B	{
36	24	$	80	50	P	124	7C	|
37	25	%	81	51	Q	125	7D	}
38	26	&	82	52	R	126	7E	~
39	27	’	83	53	S	127	7F	DEL
40	28	(	84	54	T			
41	29	)	85	55	U			
42	2A	*	86	56	V			
43	2B	+	87	57	W			

APPENDIX 1: EDT

SETTING UP YOUR TERMINAL TO USE EDT:

```
$ SET TERM/VT100 <RETURN>    <-- identify your
                            terminal as a VT100
```

GETTING INTO FULL SCREEN MODE EDT:

```
$ ED MYPROGRAM.FOR <RETURN>
*C <RETURN>
```

PUTTING TEXT INTO A FILE:

Simply begin typing in full screen mode!

CORRECTING MISTAKES:

```
<DELETE>                  <-- delete a character
<CTRL-X>               <-- delete an entire line
```

MOVING AROUND: THE CURSOR KEYS

Use the four arrow (cursor) keys to scoot!

SAVING A FILE AND EXITING FROM EDT:

```
<CTRL-Z> EX <RETURN>
```

SAVING A FILE (AND CONTINUING TO EDIT):

```
<CTRL-Z> WR MYPROGRAM.FOR <RETURN> C <RETURN>
```

ABANDONING A BUTCHERED FILE:

```
<CTRL-Z> QUIT <RETURN>
```

MOVING AROUND: USING THE KEYPAD

<PF 1> <KP 5> <-- top of file
<PF 1> <KP 4> <-- bottom of file

<KP 5> <KP 8> <-- up 16 lines
<KP 4> <KP 8> <-- down 16 lines

<KP 5> <KP 0> <-- back to beg. of line
<KP 4> <KP 0> <-- for. to beg. of line

<KP 5> <KP 2> <-- back to end of line
<KP 4> <KP 2> <-- for. to end of line

<KP 5> <KP 1> <-- backward one word
<KP 4> <KP 1> <-- forward one word

<KP 5> <KP 3> <-- backward one character
<KP 4> <KP 3> <-- forward one character

NOTE: Each of the preceding movement commands explicitly contains a <KP 4> or a <KP 5> "direction-setting" key. You actually need to hit a <KP 4> or <KP 5> ONLY when you are CHANGING direction. Thus, if you are doing multiple movement commands in the same direction you do NOT have to hit the direction-setting key before each command.

USING THE KEYPAD TO DELETE TEXT:

```
<KP PERIOD>     <-- mark start of text to del.
<cursor keys>  <-- move to end of text to del.
<KP 6>                    <-- delete the text
<PF1 <KP6>          <-- undelete deleted text

<PF 4>                      <-- delete a line
<PF 1> <PF 4>       <-- undelete deleted line

<KP MINUS>                <-- delete one word
<PF 1> <KP MINUS>   <-- undelete deleted word

<KP COMMA>               <-- delete character
<PF 1> <KP COMMA>   <-- undelete deleted char
```

FINDING TEXT:

```
<PF 1> <PF 3> TRIPE <KP 5>          <-- search
                           upwards for "TRIPE"

<PF 1> <PF 3> TONGUE <KP 4>         <-- search
                         downward for "TONGUE"
```

TO FIND THE SAME STRING AGAIN:

```
<PF 3>
```

SUBSTITUTING TEXT:

```
<KP PERIOD>         <-- mark start of new text
CHEVROLET               <-- enter the new text
<KP 6>                    <-- cut the new text
<PF 1> <PF 3>        <-- look for the old text
BUICK
<PF 1> <KP ENTER>     <-- sub new text for old
```

TO REPEAT THE SUBSTITUTION:

```
<PF 1> <KP ENTER>
```

SUBSTITUTING IN LINE MODE:

```
<CTRL-Z> S/HAM/TURKEY/W <RETURN> C         <--
                 substitute "TURKEY" for "HAM"
                     throughout the whole file
```

MOVING TEXT:

(1) Put the cursor at the beginning of the text to be moved.

(2) Mark the text by hitting a <KP PERIOD>

(3) Put the cursor at the end of the text to be moved. Text to be moved will highlight.

(4) Cut the text by hitting a <KP 6>. The marked text will disappear.

(5) Move the cursor to the location where the cut text belongs; hit <PF 1> <KP 6> to paste the cut text.

Repeat step 5 to paste the text at additional locations, as needed.

USING MULTIPLE BUFFERS IN EDT:

Normally, when you're using EDT you're working in the MAIN buffer (maybe also cutting and pasting text to the PASTE buffer).

However, you can also work in additional buffers if you need to edit more than one file at one.

```
[WE ASSUME YOU'RE IN EDT AT THE * ALREADY]

*=BRAZIL      <-- goto a buffer called BRAZIL,
               creating it if it doesn't exist

*SH BUFF       <-- see what buffers exist; the
          current buffer is marked with an "="

*CLEAR ARGENTINA      <-- zap buffer ARGENTINA

*INCLUDE CHILE.DAT =CHILE         <-- put file
         CHILE.DAT into buffer CHILE (creating
             buffer CHILE if it doesn't exist)

*MOVE 15:20 TO =PERU 128  <-- move lines 15-20
        from the current buffer to immediately
             below line 128 in the PERU buffer
         (creating the PERU buffer if need be)

*COPY ALL TO =ORIGINAL      <-- make a copy of
              the current buffer in the buffer
              called ORIGINAL (creating buffer
         ORIGINAL if it doesn't already exist)

*WRITE OIL.DAT =VENEZUELA      <-- write out a
           copy of buffer VENEZUELA to OIL.DAT
```

CAUTION: Text left in a buffer unsaved will be discarded when you exit or quit EDT! Don't forget to save your work!

INSERTING SPECIAL CHARACTERS:

```
<PF 1> 012 <PF 1> <PF 3>     <-- Insert a <FF>
```

TAKING CARE OF IRRITATIONS:

o **SILENCING THE BELL:** Many people like to turn the error warning bell off while using EDT, relying solely on the error messages which will flash at the bottom of the screen:

```
<CTRL-Z> SET Q <RETURN> C <RETURN>
```

o **ENABLING WORD WRAP:** It can be irritating to have to hit the return key at the end of each line. If you enable word wrap, EDT will automatically move to the next line as each line of text fills up, pulling any uncompleted word from the full line to the new line for you to complete. To enable word wrap at column 60 type:

```
<CTRL-Z> SET W 60 <RETURN> C <RETURN>
```

o **SWITCHING TO WIDE SCREEN:** To put your terminal into compressed-print mode so that you can see a full 132-column-wide line on your CRT screen, enter:

```
<CTRL-Z> SET SC 132 <RETURN> C <RETURN>
```

Note that most VT100-emulating PC's cannot emulate 132-column compressed-print mode.

o **SAVING YOUR SETTINGS:** You can save your normal EDT settings by creating a text file called EDTINI.EDT with the appropriate commands in it. For example, if you usually do word processing, you may want to save the settings:

```
SET NOTRUNCATE
SET WRAP 60
SET QUIET
SET MODE CHANGE
```

USING THE KEYPAD TO ENTER NUMBERS:

To get honest-to-god numbers rather than special EDT commands from your VT100 keypad, do the following:

```
$ ED EDTINI.EDT
*C
DEFINE KEY 7 AS "I7^Z."   <-- Note: really
DEFINE KEY 8 AS "I8^Z."       type in the
DEFINE KEY 9 AS "I9^Z."       caret (^) as
DEFINE KEY 4 AS "I4^Z."       shown!
DEFINE KEY 5 AS "I5^Z."
DEFINE KEY 6 AS "I6^Z."
DEFINE KEY 1 AS "I1^Z."
DEFINE KEY 2 AS "I2^Z."
DEFINE KEY 3 AS "I3^Z."
DEFINE KEY 0 AS "I0^Z."
DEFINE KEY 16 AS "I.^Z."
<CTRL-Z>
*EX
```

From then on, whenever you start EDT (from the same directory that contains the special EDTINI.EDT file), your numeric keypad keys will really generate numbers!

USING EDT TO FORCE TEXT INTO UPPERCASE:

Assume you have a file called TOFU.DAT that has a mixture of upper and lowercase text. You want to force all the text to be uppercase only. To use EDT to "automatically" do this, use:

```
$ COPY  DISK$USER1:[JOE.EDT]UC.EDT  TOFU.JOU

$ ED/RECOVER TOFU.DAT <-- Then wait a minute!
*EXIT
```

RECOVERING FROM A CRASH:

From time to time, an EDT editing session may end abnormally. Most EDT sessions which end abnormally do so because the user has hit a <CRTL>-Y or because the user has let his terminal time out. When this happens, EDT leaves a journal file in your directory which can be used to recreate your editing session, up to the point where it abnormally terminated. To recover from an abnormal exit of this kind by applying the journal file to the file you had been working on, type (at the dollar sign):

```
$ ED/RECOVER PROGRAM1.PAS
```

where PROGRAM1.PAS is the name of whatever file you were editing when EDT aborted.

NEED MORE HELP?

If you are at the dollar sign, type:

```
$ HELP EDIT/EDT
```

When you are in full-screen EDT you can get help by hitting <PF 2>.

If you want to order an EDT manual, you may do so by calling Digital Equipment Corporation at (603) 884-6660, or by writing to DEC at P.O. Box CS2008, Nashua, NH 03061. Request the VAX EDT Reference Manual (Order Number AA-Z300A-TE).

```
-----------------------------------------
|PF1      |PF2      |PF3      |PF4      |
|         |         | FINDNXT |  DEL L  |
|   GOLD  |  HELP   |         |         |
|         |         |  FIND   |  UND L  |
|         |         |         |         |
-----------------------------------------
|7        |8        |9        |-        |
|   PAGE  |   SECT  | APPEND  |  DEL W  |
|         |         |         |         |
| COMMAND |  FILL   | REPLACE |  UND W  |
|         |         |         |         |
-----------------------------------------
|4        |5        |6        |,        |
| ADVANCE | BACKUP  |   CUT   |  DEL C  |
|         |         |         |         |
| BOTTOM  |   TOP   |  PASTE  |  UND C  |
|         |         |         |         |
-----------------------------------------
|1        |2        |3        |ENTER    |
|  WORD   |   EOL   |   CHAR  |         |
|         |         |         |         |
|CHNGCASE | DEL EOL | SPECINS |  ENTER  |
|         |         |         |         |
-----------------------------|         |
|0                  |.        |         |
|       LINE        | SELECT  |  SUBS   |
|                   |         |         |
|    OPEN LINE      |  RESET  |         |
|                   |         |         |
-----------------------------------------

-----------------------------------------
|         |         |         |         |
|    ^    |    |    |         |         |
|    |    |    |    | <------ | ------> |
|    |    |    V    |         |         |
|         |         |         |         |
-----------------------------------------
```

```
Delete      Delete characters
Linefeed    Delete to beginning of word
Backspace   Backup to beginning of line
<CTRL-A>    Compute tab level
<CTRL-D>    Decrease tab level
<CTRL-E>    Increase tab level
<CTRL-K>    Define key
<CTRL-R>    Refresh screen
<CTRL-T>    Adjust tabs
<CTRL-U>    Delete to beginning of line
<CTRL-W>    Refresh screen
<CTRL-Z>    Exit to line mode
```

APPENDIX 2: MICRO EMACS

SETTING UP TO USE MICRO EMACS:

```
$ ED LOGIN.COM
*I
     $ @SYS$SYSTEM:UNSUPPORTED_SETUP
     $ DEF/JOB TERM "vt100"  (or "tvi912")
     <CTRL-Z>
*EX                     ***Note: the vt100 or
$ @LOGIN                tvi912 MUST be lowercase
```

NOTE: Some EMACS commands include special control sequences such as <CTRL>-S which may interfere with normal communication protocols that also use those control sequences.

In order to make EMACS work properly, you will probably need to turn off flow control. This means, if you're connecting via UONet, connect to the VAX as usual, then issue the command:

```
UONet> do noxon
```

If you're also using Kermit, you may also need to use the Kermit command:

```
<CTRL>-] C
Kermit-MS> set flow=none
Kermit-MS> C
```

GETTING INTO MICRO EMACS:

```
$ EMACS MYPROGRAM.FOR <RETURN>
```

GOING FROM INSERT TO OVERSTRIKE:

```
<CTRL-X> M OVER <RETURN>        <-- overstrike

<CTRL-X> <CTRL-M> OVER <RETURN>     <-- insert
```

CORRECTING MISTAKES:

```
<DELETE>                <-- delete a character
```

SAVING A FILE AND EXITING FROM EMACS:

```
<ESC> Z
```

MOVING AROUND:

```
<CTRL-N>                  <-- go down one line
<CTRL-P>                    <-- go up one line
<CTRL-B>                  <-- go back one char
<CTRL-F>                 <-- go ahead one char

<ESC>  <                       <-- top of file
<ESC>  >                    <-- bottom of file

<CTRL-V>                       <-- scroll down
<CTRL-Z>                         <-- scroll up

<CTRL-A>                 <-- beginning of line
<CTRL-E>                       <-- end of line

<ESC> B                     <-- back one word
<ESC> F                    <-- ahead one word
```

DELETING TEXT:

```
<ESC> <space>   <-- mark start of text to del.
(...move to end of text to delete...)
<CTRL-W>                   <-- delete the text

<CTRL-K>             <-- delete to end of line

<ESC> <CTRL-H>      <-- delete previous word
<ESC> D                 <-- delete next word

<CTRL-H>            <-- delete previous char.
<CTRL-D>            <-- delete next character

<CTRL-Y>              <-- undo the last delete
```

SAVING A FILE (AND CONTINUING TO EDIT):

```
<CTRL-X> <CTRL-S>
```

ABANDONING A BUTCHERED FILE:

```
<CTRL-X> <CTRL-C> Y
```

FINDING TEXT:

```
<CTRL-R> TRIPE <ESC>                          <-- search
                                    upwards for "TRIPE"

<CTRL-S> TONGUE <ESC>                         <-- search
                                 downwards for "TOUNGE"
```

TO FIND THE SAME STRING AGAIN:

```
<CTRL-R> <ESC>                    <-- search upwards

<CTRL-S> <ESC>                  <-- search downwards
```

SUBSTITUTING TEXT CONDITIONALLY:

```
<ESC> <CTRL-R>
BUICK
<ESC>
CHEVROLET
<ESC>
Y (Yes) | N (No) | ! (All!) | . (Enough.)
```

SUBSTITUTING TEXT GLOBALLY:

```
<ESC> R
MISISIPI
<ESC>
MISSISSIPPI
<ESC>
```

MOVING TEXT:

```
(go to start of text to be moved)

<ESC> <space>   <-- mark start of text to move

(go to the end of the text to be moved)

<CTRL-W>                     <-- delete the text

(go to where the text is to now to be put)

<CTRL-Y>               <-- undelete deleted text

(repeat <CTRL-Y> to paste text again)
```

"WORD PROCESSING" IN EMACS:

ENABLING WORD WRAP: It can be irritating to have to hit the return key at the end of each line. If you enable word wrap, EMACS will automatically move to the next line as each line of text fills up, pulling any uncompleted word from the full line to the new line for you to complete. To enable word wrap at column 60, enter:

```
<CTRL-X> A SET $FILLCOL 60 <RETURN>
<CTRL-X> <ESC> WRAP <RETURN>
```

COUNTING THE WORDS IN A DOCUMENT: Frequently, one is aiming at producing a document of a particular length, say 5000 words. EMACS can count the number of words in a document for you with the command:

```
<ESC> <CTRL-C>
```

SAVING YOUR SETTINGS:

You can save your normal EMACS settings by creating a text file called EMACS.RC in your default directory with the appropriate EMACS commands in it.

ENCRYPTING/DECRYPTING FILES:

ENCRYPT (WITH PASSWORD MONTREAL):

```
$ EMACS RECIPE.DAT
<create text to be encrypted>
<ESC> E <RETURN> MONTREAL <RETURN>
<CTRL-X> M CRYPT <RETURN>
<ESC> Z
```

DECRYPT (WITH PASSWORD MONTREAL):

```
$ EMACS -K RECIPE.DAT
MONTREAL <RETURN>
<CTRL-X> <CTRL-M> CRYPT <RETURN>
<proceed to work with plain text>
<ESC> Z
```

NOTE: Upper and lowercase letters are different when used as encryption keys

You must make at least some change, however minor, to the file you're working on before attempting to save it with <ESC> Z since EMACS doesn't bother writing out a file unless it has been changed (encryption is not considered to "change" a file!).

Remember to delete any unencrypted copies of your file after encrypting!

WARNING: Be quite CAREFUL with your encryption key – if you mis-type it when initially encrypting a file, or if you forget it, there is NO WAY to recover your file. In fact, ask yourself right now, "Do I REALLY need to encrypt this file in the first place?"

NOTE: You may also want to consider using the DES encryption routines described elsewhere in this book.

PROTECTING YOURSELF FROM "ACCIDENTS":

From time to time, something bad may happen to cause your EMACS editing session to terminate abnormally. For example, the system may unexpectedly crash due to a hardware problem. In such a case, you stand to lose all the work you've done since the last time you saved your file.

For your own protection, you may want to set the EMACS autosave facility to automatically save your work every so-many characters, as you work. To autosave ever 100 characters, for example, use the command:

```
<CTRL-X> A SET $ASAVE 100 <RETURN>
<CTRL-X> M ASAVE <RETURN>
```

With autosave, you may still lose work, but at least you can effectively limit your exposure.

NEED MORE HELP?

Since EMACS is in the Computing Center's "Unsupported Software" category, consulting and EMACS classes will generally not be available. However, an EMACS tutorial IS available. After setting up your account for EMACS use as described at the beginning of this appendix, type:

```
$ EMACS EMACS_LIBRARY:[ETC]EMACS.TUT
```

You can also check out a very well written Micro EMACS manual from the Computer Center Documents Room, if you prefer to learn your computer skills from more traditional printed materials.

APPENDIX 3: EVE

SETTING UP YOUR TERMINAL TO USE EVE:

```
$ SET TERM/VT100 <RETURN>    <-- identify your
                             terminal as a VT100
```

GETTING INTO EVE:

```
$ EVE MYPROGRAM.FOR <RETURN>
```

PUTTING TEXT INTO A FILE:

Simply begin typing!

TOGGLING BETWEEN INSERT AND OVERSTRIKE:

```
<KP ENTER>
```

CORRECTING MISTAKES:

```
<DELETE>                 <-- delete a character
<CTRL-U>               <-- delete an entire line
```

MOVING AROUND: THE CURSOR KEYS

Use the arrow keys to move around.

SAVING A FILE AND EXITING FROM EVE:

```
<CTRL-Z>
```

ABBREVIATING EVE COMMANDS:

You may abbreviate commands entered after hitting <PF 4> as along as any such abbreviation is unambiguous.

SAVING A FILE (AND CONTINUING TO EDIT):

```
<PF 4> WRITE FILE <RETURN>
```

ABANDONING A BUTCHERED FILE:

```
<PF 4> QUIT <RETURN>
YES <RETURN>
```

MOVING AROUND:

```
<KP 5>                                       <-- up arrow
<KP 2>                                     <-- down arrow
<KP 1>                                     <-- left arrow
<KP 3>                                    <-- right arrow

<CTRL-E>                          <-- move to end of line
<CTRL-H>                         <-- move to head of line

<KP ->                                <-- move by lines
<PF 3>                          <-- set move direction

<KP .>                                 <-- up a screen
<KP 0>                               <-- down a screen

<PF 4> BOT <RETURN>             <-- bottom of the file
<PF 4> TOP <RETURN>                <-- top of the file

<PF 4> WHAT LINE <RETURN>         <-- current line?
<PF 4> LINE 150 <RETURN>          <-- goto line 150
```

USING THE KEYPAD TO DELETE TEXT:

```
<KP 7>           <-- mark start of text to del.
<cursor keys>  <-- move to end of text to del.
<KP 8>                     <-- delete the text
<KP 9>               <-- undelete deleted text

<KP COMMA>                 <-- delete one word
```

FINDING TEXT:

```
<PF 1> TRIPE <RETURN>   <-- search for "TRIPE"
<PF 1> <PF 1>  <-- search for the string again
<PF 3>               <-- toggle search direction
```

NOTE: The FIND routine in EVE is queerly case sensitive.

If you search for a capitalized string or mixed-case string, EVE will generally find only EXACT matches for that string.

On the other hand, if you search for an all-lowercase search string, EVE will disregard case in finding matching text, matching against capitalized, lowercase or mixed strings with equal alacrity.

NOTE: You can tailor EVE's behavior to be more liberal in what it matches – for example, you can instruct EVE to find matching strings than span a line break (i.e., a string that is partially on one line, but terminated on the next line). You can also tell EVE to treat tabs and spaces interchangably in finding matches. See SET FIND NOWHITESPACE below.

Also note: if you need still more powerful character matching capabilities (including regular expressions), see the following panel on FINDING REGULAR EXPRESSIONS.

FINDING REGULAR EXPRESSIONS:

Regular expressions are a way of expressing potentially quite complicated pattern matching rules.

Two different ways of expressing regular expressions are offered in EVE: UNIX-like regular expression characters, and VMS-specific regular expressions characters.

You are generally encouraged to use the UNIX-like SET WILDCARD ULTRIX unless you need the peculiar pattern matching capabilities of the SET WILDCARD VMS command:

```
<PF 4> SET WILCARD ULTRIX
```

Once you've SET WILDCARD ULTRIX, you can then do things like find lines which BEGIN with the letters x, y, or z:

```
<PF 4> WILDCARD FIND
^[xyz]
```

ULTRIX AND VMS REGULAR EXPRESSION CHARACTERS:

ULTRIX	VMS	MATCHES ---------------
.	%	Exactly any one char
^	\<	Beginning of line
$	\>	End of line
[hwkx]	\[hwkx]	Any of h, w, k, x
[^iul]	\[~iul]	Any NOT of i, u, l
*	\.	Repeat 0 or more times
+	\:	Repeat 1 or more times
	\w	Tab, space, or linebrk
	\d	Any decimal digit
	\o	Any octal digit
	\x	Any hex digit
	\a	Any alphabetic char
	\n	Any alphanumeric chr
	\s	Alphanumeric, or + $ _
	\l	Any lowercase letter
	\u	Any uppercase letter
	\p	Any punctuation
	\f	BS TAB LF VT FF CR
	\^	Any control char
	\+	ASCII 128 thru 255
\	\	"Take literally" or "Don't make magic!"

SET FIND WHITESPACE:

If you enter the command:

```
<PF4> SET FIND WHITESPACE
```

EVE will then treat spaces and tabs identically when searching for a match. EVE will also find a search string that happens to be split from one line to another.

To restore EVE to its default mode of distinguishing spaces and tabs, and to restrict search string matches to those found on a single line, enter the command:

```
<PF 4> SET FIND NOWHITESPACE
```

REPLACING TEXT:

```
<PF 4> REPLACE
Old string: beans
New string: corn
Replace? Yes, no, all, last or quit: LAST
```

DOING MULTIPLE TEXT REPLACEMENTS:

```
GLOBAL REPLACEMENT:    instead of LAST use ALL
SEQUENTIAL REPLACEMENT:         use YES and NO
```

NOTE: REPLACE is queerly case sensitive.

If you give REPLACE a capitalized string or a mixed-case string to find, it will generally find only EXACT matches for that string. Moreover, REPLACE will then use EXACTLY the new string you specify, respecting capitalization as entered.

On the other hand, if you give REPLACE an all-lowercase search string, REPLACE will disregard case in finding matching text, matching against capitalized, lowercase or mixed strings with equal alacrity. REPLACE will endeavor to preserve the capitalization pattern found in the old string when substituting a version of the new value.

MOVING TEXT:

(1) Put the cursor at the beginning of the text to be moved.

(2) Mark the text by hitting a <KP 7>

(3) Put the cursor at the end of the text to be moved. Text to be moved will highlight.

(4) Cut the text by hitting a <KP 8> The marked text will disappear.

(5) Move the cursor to the location where the cut text belongs; hit <KP 9> to paste the cut text.

Repeat step 5 to paste the text at additional locations, as needed.

INVISIBLY MARKING TEXT:

(1) Go to location to be invisibly marked.

(2) Create a mark called FOO: <PF 4> MARK FOO

(3) Continue about your business, moving through the file as necessity dictates

(4) To return to FOO, use: <PF 4> GO TO FOO

NOTE: You can be in one buffer and GO TO a mark in another buffer!

To see all defined marks use: <PF 4> SHOW

MAKING THE CURSOR FOLLOW END-OF-LINE:

```
<PF 4> SET CURSOR BOUND
```

To let the EVE cursor go past end of lines:

```
<PF 4> SET CURSOR FREE
```

MULTIPLE BUFFERS:

```
<PF 4> BUFFER BRAZIL  <-- goto a buffer called
            BRAZIL, creating it if it doesn't
                                already exist

<PF 4> SHOW BUFFERS       <-- see what buffers
        exist; use <arrows> <return> to select

<PF 4> SHOW SYSTEM BUFFERS  <-- see the system
        buffers; generally don't delete these!

<PF 4> DELETE BUFFER ARGENTINA     <-- zap the
      ARGENTINA buffer. If prompted: D=Delete,
       W=Write it then delete, Q=Don't delete.
          Don't delete system created buffers!

<PF 4> GET FILE CHILE.DAT         <-- put file
     CHILE.DAT into buffer CHILE.DAT, creating
        the buffer if it doesn't already exist
```

[To move text from one buffer to another, use the procedure outlined for moving text, but move from the original buffer to the new buffer before pasting the text]

```
<PF 4> WRITE FILE OIL.DAT        <-- write the
          current buffer into the file OIL.DAT
```

USING MULTIPLE WINDOWS:

Unlike EDT, in addition to allowing you to work with multiple buffers, EVE will also allow you to create multiple independent on-screen windows.

```
<PF 4> SPLIT WINDOW     <-- Create two windows

<PF 4> SPLIT WINDOW 3   <-- Create three of em

<PF 4> NEXT WINDOW        <-- Goto next window

<PF 4> PREVIOUS WINDOW  <-- Go back one window

<PF 4> ONE WINDOW    <-- Make curr. window big

<PF 4> DELETE WINDOW   <-- Delete curr. window

<PF 4> ENLARGE WINDOW 3   <-- Make the current
                       window grow three lines

<PF 4> SHRINK WINDOW 3    <-- Make the current
                     window shrink three lines
```

RUNNING DECSPELL FROM EVE:

```
<PF 4> SPELL
```

TEMPORARILY EXITING TO THE "$":

```
<PF 4> SPAWN
```

NOTE: Use $ EOJ to return to EVE

ENHANCING EVE:

EVE was created with the expectation that it would be extended or enhanced. "EVE" is, after all, an acronym for **E**xtensible **V**AX **E**ditor!

o **DEFINING NEW EVE KEYS:** To define a key so that it automatically executes an EVE command, use:

```
<PF 4> DEFINE KEY
SHOW BUFFER     <-- Eve command to be assigned
<KP 6>  <-- Hit the key to get that assignment

<PF 4> SH KEY    <-- Scrutinize key definition
<KP 6>                    <-- ... of this key!

<PF 4> UNDEFINE   <-- Clear the key definition
<KP 6>                    <-- ... of this key!
```

o **LEARNING A SEQUENCE OF KEYSTROKES:** Sometimes you'll find yourself repeatedly entering the same sequences of five, ten, or even more keystrokes. If that's the case, you can make EVE "learn" those keys, and you can then "do them" by hitting a single key. For example, maybe you need to enter the word "foo" at the start of a number of lines:

```
<PF 4> LEARN
<CTRL H> foo <DOWN ARROW> <CTRL R>
```

Anytime you hit a <CTRL F> from now until the time you finishing this editing session, the word "foo" will be inserted at the beginning of the current line, and EVE will then move you down to the next line.

FURTHER EXTENSIONS: You can go far further in customizing EVE. For more information on this, read "Extending the EVE Editing Interface to the VAXTPU Text Processing Utility," section 3.11 of the Guide to Text Processing on VAX/VMS, one of the smaller orange VAX manuals.

SOME EVE "TRICKS":

o ENABLING WORD WRAP: It can be irritating to have to hit the return key at the end of each line. If you enable word wrap, EVE will automatically move to the next line as each line of text fills up, pulling any uncompleted word from the full line to the new line for you to complete. To enable word wrap at column 60 type:

```
<PF 4> SET RIGHT MARGIN 60
<PF 4> SET WRAP
```

o SWITCHING TO WIDE SCREEN: To put your terminal into compressed-print mode so that you can see a full 132-column-wide line on your CRT screen, enter:

```
<PF 4> SET WIDTH 132
```

Note that most VT100-emulating PC's cannot emulate 132-column compressed-print mode.

o SHIFTING RIGHT AND LEFT: Try:

```
<PF 4> SHIFT RIGHT 40
<PF 4> SHIFT LEFT 30
```

o MAKING TABS VISIBLE: Try:

```
<PF 4> SET TABS VISIBLE
```

o INCLUDING A FILE IN THE CURRENT BUFFER:

```
<PF 4> INCLUDE FILE THUD.DAT
```

o SAVING YOUR SETTINGS: You can save your normal EVE settings by creating a text file called EVE$INIT.EVE with the appropriate commands in it. For example, if you usually do word processing, you may want to save the settings:

```
SET WRAP
SET RIGHT MARGIN 60
```

RECOVERING FROM A CRASH:

From time to time, an EVE editing session may end abnormally. Most EVE sessions which end abnormally do so because the user has hit a <CRTL Y> or because the user has let his terminal time out. When this happens, EVE leaves a journal file in your directory which can be used to recreate your editing session, up to the point where it abnormally terminated. To recover from an abnormal exit of this kind by applying the journal file to the file you had been working on, type (at the dollar sign):

```
$ EVE/RECOVER PROGRAM1.PAS
```

where PROGRAM1.PAS is the name of whatever file you were editing when EVE aborted.

IMPORTANT NOTE: EVE journal files cannot be relied upon after you've hit a <CTRL-C> while editing, so if you do accidentally hit a <CTRL-C>, immediately save your file and start EVE anew so that you'll once again have a reliable journal file.

Also, if you've issued one or more WRITE FILE commands while editing, you'll need to make sure you explicitly specify the version number of the original file when recovering. NOTE: You may need to edit the .TJL file manually if the version you started from has been automatically purged. Ask a VAX consultant for help if need be!

NEED MORE EVE HELP?

If you are at the dollar sign, type:

```
HELP EDIT/TPU
```

When you are in EVE you can get help by hitting <PF 2>.

EVE manuals are also available for your perusal in the Documents Room.

```
-----------------------------------------
|PF1      |PF2      |PF3      |PF4      |
|         |         |         |         |
|   FIND  |  HELP   |  CHANGE |   DO    |
|         |         |   DIR   |         |
|         |         |         |         |
-----------------------------------------
|7        |8        |9        |-        |
|         |         |         |  MOVE   |
|  SELECT |  REMOVE | INSERT  |   BY    |
|         |         |  HERE   |  LINES  |
|         |         |         |         |
-----------------------------------------
|4        |5        |6        |,        |
|         |    ^    |         |         |
|         |    |    |         |  ERASE  |
|         |    |    |         |   WORD  |
|         |         |         |         |
-----------------------------------------
|1        |2        |3        |ENTER    |
|         |    |    |         |         |
|  <----  |    |    |  ---->  |         |
|         |    V    |         |         |
|         |         |         |  CHANGE |
-----------------------------|   MODE  |
|0                  |.        |         |
|                   |         |         |
|   NEXT SCREEN     |   PREV  |         |
|                   |  SCREEN |         |
|                   |         |         |
-----------------------------------------

-----------------------------------------
|         |         |         |         |
|    ^    |    |    |         |         |
|    |    |    |    | <------ | ------> |
|    |    |    V    |         |         |
|         |         |         |         |
-----------------------------------------
```

```
Delete      Delete characters
Linefeed    Delete to beginning of word
Backspace   Backup to beginning of line
<CTRL-A>    Compute tab level
<CTRL-D>    Decrease tab level
<CTRL-E>    Increase tab level
<CTRL-K>    Define key
<CTRL-R>    Refresh screen
<CTRL-T>    Adjust tabs
<CTRL-U>    Delete to beginning of line
<CTRL-W>    Refresh screen
<CTRL-Z>    Exit to line mode
```

APPENDIX 4: ERROR MESSAGES IN ENGLISH

ACP FILE EXTEND FAIL: This is the VAX's way of saying you're out of disk space.

Either delete some files you no longer need, have the Computing Center accounting clerk increase your permanent VAX disk quota, or use temporary SCRATCH space.

BACKWARDS QUESTION MARKS ON A VT320: Your VT320 terminal has an incorrect communications setting. This problem arises most frequently in public terminal areas frequented by EMACS users. To fix this problem:

```
o Hit SETUP at the top of your keyboard
o Use the arrow keys to move to COMM
o Hit ENTER key on the keypad
o Use the arrows to move to NO XOFF
o Hit ENTER key on the keypad
o (You should now see X-ON AT 64)
o Use the arrows to move to TO DIR
o Hit the ENTER key on the keypad
o Use the arrows to move to SAVE
o Hit the ENTER key on the keypad
o Hit SETUP at the top of your keyboard
o Hold down the control key and hit a W
  to redraw your screen properly!
```

BUFFER OVERFLOW (IN MAIL): You typed more than 255 characters without a carriage return. While it may appear that your lines are automatically "wrapping around", this doesn't actually take place.

You must remember to hit return as you approach the right margin of each line!

EDT WON'T GO INTO FULL SCREEN MODE: You see C* instead of the full screen display you expected to see because your forgot to declare your terminal to be a VT100.

Use EXIT to get out of EDT, then reset your terminal with:

```
$ SET TERM/VT100
```

"ENTER CLASS" PROMPT GARBLED: You are connecting to the VAX via the Gandalf using communication settings of 8 data bits, 1 stop bit and no parity. The "enter class" prompt appears garbled, but if you ignore the garbled prompt and enter OREGON (as you normally would), everything works fine from there on out.

This problem is due to the Gandalf switch sending seven bit characters rather than eight bit characters. If this problem really bothers you, change your communication program to use 7 data bits, 1 stop bit and SPACE parity.

However, in most cases, just leave your settings lone. Once you're past the "enter class" prompt everything will be fine. By not changing your terminal from the 8-N-1 setting, you'll be set correctly for most bulletin boards and other systems that use the 8-N-1 defacto standard.

FORTRAN "OUTPUT CONVERSION" ERROR: You're trying to write a number (or character string) too wide for its format specification. Expand the width of the field that shows asterisks in your output.

LINKER REPORTS AN ERROR ON EVERY LINE: Usually, you've mistakenly attempted to link a SOURCE file rather than an OBJECT file.

That is, you mistakenly said something like:

```
$ LINK MYJOB.FOR    <-- note the .FOR!!!
```

when what you **really** wanted to do was to link the .OBJ file (produced when you compiled your program with the $ FOR command) instead!

"PASSWORD IS EXPIRING" MESSAGE: It has been nearly six months since you last changed your account's password. Because your password is your primary protection against hackers and other people who'd like to misuse your account or damage your files, the system will force you to change your password every six months "for your own good."

To stop the password expiration messages, use:

```
$ SET PASS
```

You'll be prompted for your old password, a new password (DIFFERENT FROM THE OLD PASSWORD and AT LEAST SIX CHARACTERS LONG), and then you'll be asked to type the NEW password a second time (just to confirm you typed what you thought you typed!).

SEND/FILE CHOKES ON SOME ADDRESSES: The SEND/FILE command will only work for true BITNET sites! If you have an Internet style address (BLAH@FOOBAR.SPUTTER.EDU, say) instead, you can use mail to move text files:

```
$ MAIL
MAIL> S SUMMARY.DAT
To: IN%"BLAH@FOOBAR.SPUTTER.EDU"
```

SPSSX "MAILBOX" ERROR: Suddenly, for no discernable reason, SPSSx prints a horrible sounding message complaining about being unable to write to a "mailbox." Your program may or may not continue beyond this error message.

First of all, recognize that SPSSx isn't talking about your electronic communication (e-mail) mailbox; rather, SPSSx has its own internal "mailboxes" used in composing output from your SPSSx job. Occaisionally, SPSSx has trouble accomplishing that task, due entirely to its own internal shortcomings.

The word from the SPSSx support people in Chicago is that if your SPSSx job continues beyond the mailbox error, simply ignore it – it should have no substantive effect on any of your results.

If, on the other hand, your job dies after the message, the recommended approach is to try removing any "extraneous" material preceding the procedure that died, and try rerunning the job. If the job then doesn't croak, you should be all set.

If the job still aborts, come visit with me, Joe St Sauver, and we'll call SPSSx in Chicago.

TAPE DRIVE CAN'T BE "FOUND" THOUGH MOUNTED: Usually this means you omitted one of three critical spaces on the mount command. Note that there MUST be:

```
-- at least one space BEFORE the MU:
-- at least one space BETWEEN MU: and the
   tape reel number, and...
-- at least one space BETWEEN the tape
   reel number and MYTAPE: (or whatever
   you want to call the tape)
```

If you inadvertently miss one of the required spaces, dismount the tape and remount it using the MOUNT command with the required spaces. You'll need to refer to the actual drive name (i.e., MUA0: MUA1: or MUA2:) to do this; to figure out which is right for you, use the:

```
$ SH DEV/FULL MU:
```

command and look for your username.

We have also noticed some occaisions when BACKUP appears to "lose" logical names even when you're doing everything right; if this happens, the easiest thing to do is to refer to the tape by its physical device name (MUA0: or whatever drive you were assigned) rather than using the logical name until a fix is found for this problem.

TAPES COMMAND ENDS WITH A STACK DUMP (WHEN READING FILES FROM TAPES WITH MANY RECORDS): This is not necessarily an "error" per se. Apparently the author of the TAPES program didn't anticipate users reading more than 99,999 records with the program, and hence he allocated insufficient space for reporting the reading of 100,000 or more records. As nearly as we can tell, the resulting data file should nonetheless be just dandy, in spite of this "error."

TAPES COMMAND REPORTS A "PARITY ERROR:" One or more characters of data from your tape appear internally inconsistent.

Each character written to tape is written as a series of eight ones or zeroes plus a "parity bit". If the sum of the eight binary digits representing the character is even, the parity bit is set to one to make the sum of the eight digits (plus the parity bit) odd; if the sum of the eight binary digits is already odd, the parity bit is left zero.

When characters are read from the tape, the tape drive checks to make sure the sum of each character's eight bits (plus the parity bit) sums to an odd value. One or more of your characters failed this test.

Your best advice is to try re-reading the tape, perhaps on a different tape drive (you can specify a particular drive at MOUNT time by saying MUA0: or MUA1: or MUA2: instead of just MU:). If the parity error is consistently present, you may be out of luck.

TERMINAL GOES INTO "CHINESE" MODE: After working normally, your terminal suddenly ceases producing normal letters, numbers, and symbols and now produces strange symbols vaguely resembling bizarre oriental writing.

This is indicative of a terminal that has accidentally gotten stuck in the alternate "line drawing" character set. The easiest way to fix this problem is to get into something that intentionally uses the line drawing character set (such as the EDT full screen help display) which properly forces the terminal OUT of line drawing mode when done. That is, try:

```
$ SET TERM/VT100  <-- make sure you're a VT100
$ ED FOO.DAT  <-- get into EDT, using any file
*C              <-- change to full screen mode
<PF 2>         <-- ask to see the help display
<SPACE>         <-- hit a <SPACE> to exit help
<CTRL-Z>             <-- exit full screen mode
*QUIT                              <-- leave EDT
```

TERMINAL IS "FROZEN:" You probably hit the <CTRL-S> key. Try entering a <CTRL-Q> to undo the <CTRL-S>.

USER AUTHORIZATION FAILURE: You've mistyped either your username or your password (or both). Hit a return and try entering your username and password again.

If you still get the user authorization failure:

– Have you changed your password lately? Are you using the correct "new" value, or did you mistakenly enter your old one?

– If you have a student account, did you remember to enter the ST that precedes your five digit number?

– If you have a new account, did you give the accounting clerk 24 hours or so to actually create your account?

– Does someone else (like your secretary or graduate assistant) ever use your account? They may have changed your password and forgotten to tell you.

– Are you sure you're connecting to the right VAX? OREGON is the VAX 8800, a general purpose machine running VMS, but there are a number of other "VAXes" on campus. Is it possible your account is actually on another system run by your department, instead? (An account on one VAX generally won't automatically give you the ability to login to other machines on campus; virtually all of the systems are administered independently of one another.)

APPENDIX 5: DATA RESOURCES

CENSUS DATA:

Selected U.S. Census data is received on mag tape by the University's Bureau of Governmental Relations as well as by Professor Greg Hundley of the School of Management. While those parties are in no way responsible for helping you work with the Census tapes, you **may** be able to make private arrangements with them to obtain access to the census data you need for research work.

COMPUSTAT:

The Business School receives selected Compustat tapes containing firm financial statement information for the period 1968-1988. The tapes currently received include the Annual Industrial Tape, the Annual OTC Tape, the Annual Research Tape, and the OTC Research Tape. Contact Professor Ken Gaver of the School of Management for more information.

CRSP:

The Center for Research on Security Prices distributes a number of different compilations of security price data. The University of Oregon School of Business receives the NYSE and AMEX daily returns data series for 1962-date, as well as market index data series for those years.

Fortran and DCL routines to simplify extracting selected price data have been prepared by the School of Business. For more information on access to those routines and on using the CRSP data, contact Ken Shah at x3359.

ICPSR: INTER-UNIVERSITY CONSORTIUM FOR POLITICAL AND SOCIAL SCIENCE RESEARCH:

ICPSR makes available numerous high quality social science-oriented datasets. Datasets typically include both raw data, and an online (or pre-printed) codebook describing the data. Because of the variety of datasets available, datasets are typically orded on an as-required basis, although some of the more popular dataset may already be on hand. If a dataset you need has to be specially ordered, the cost involved is typically only the cost of a magnetic tape plus postage (i.e., $25-$30).

An excellent introduction to the ICPSR datasets is the ICPSR Guide to Resources and Services, updated every couple of years.

For more information, contact Professor Val Burris, Sociology Department, x5001.

STAT PACK DEMO DATASETS:

Several of the statistical packages provide demonstration datasets that may be of interest to instructors or other individuals.

SPSSx's Introductory Statistics Guide and Advanced Statistics Guide refer to a guitar dataset, a Western Electric dataset, etc. To see which SPSSX datasets are available, use:

```
$ DIR SPSSX$DISK:[SPSSX.SPSSX031.EXAMPLES]
```

A brief codebook for each of these datasets can be found in Appendix B of the SPSSx Introductory Statistics Guide.

SAS makes a set of sample programs and data available for each of its products – see:

```
$ DIR SAS$DISK:[SAS.SAS518.SAMPLES...]
```

BMDP shows a number of moderate-size interval-ratio level datasets in Appendix D to Volume 2 of the 1988 BMDP Statistical Software Manual. [If you want to use these data sets you'd need to enter these sets by hand at this time – sorry!].

MINITAB also makes available a number of demonstration datasets for your use. To see them, use:

```
$ DIR MINITAB$DISK:[MINITAB]*.DAT
```

Information on the datasets can be gleaned from the examples in the MINITAB manual using the data.

APPENDIX 6: SUPERCOMPUTING

GETTING A SUPERCOMPUTER ACCOUNT:

Applications suitable for a supercomputer are usually computationally intensive, involve scientific visualization and/or might profit from vectorization (i.e., make extensive use of certain types of do loops).

If believe you have an application appropriate for a supercomputer, please contact Rich Haller at 346-4394 for information on obtaining a supercomputer account.

USING THE CRAY X-MP AT NCSA:

The following panels describe use of the Cray X-MP running UNICOS (a flavor of UNIX) at the National Center for Supercomputer Applications at the University of Illinois. Depending upon your application requirements and the availability of resources, you may use that machine, or some other supercomputer.

CONNECTING TO THE X-MP MACHINE AT NCSA:

```
enter class  UONET  <-- needed only if logging
                        on via modem or Gandalf

UONet> C UX.NCSA.UIUC.EDU   <-- go to the Cray

login: u12345                <-- your CRAY login
Password: abc123456       <-- your CRAY password
Account: xyz               <-- your CRAY account

TERM = (vt100) vt100-nonl <-- declare the term
ux 1%                      <-- the system prompt
```

CHANGING YOUR PASSWORD:

```
% passwd              <-- change your password
Changing password for u12345
Old password: abc123456
New password: 888xyz888
Re-enter new password: 888xyz888
```

UNICOS FILES AT NCSA:

UNICOS file names can be up to 14 characters long, and can include letters, numbers, dots and underscores. Filenames are case sensitive.

UNICOS files can be stored three places:

(1) You have half a megabyte of UNICOS rotating disk storage for files that need to be kept in the root directory (such as .login and .profile) and for other frequently used files (such as command scripts). This storage is permanent. See the pathname of this home directory with:

```
% echo $HOME
```

If you exceed your disk allocation, NCSA will send you mail advising you to reduce your usage.

(2) You also have about 400 megabytes of temporary rotating disk storage for use as scratch space. Temporary files are automatically purged after 72 hours of non-access. See the pathname of this directory with:

```
% echo $SCR
```

(3) In addition to the permanent and temporary UNICOS disk storage resources just mentioned, you have virtually unlimited file storage via the Common File System (CFS) using a mix of rotating disk and magnetic tape storage. (Media choice is automatic and beyond your control, and is determined by file size and access frequency). The CFS has a per-file limit of 200 million (decimal) bytes, but larger files can be handled by breaking the files into 200 million byte (or smaller) chunks for storage. See the section on CFS commands for more information on using CFS.

BASIC UNICOS COMMANDS:

NOTE: UNICOS commands are CaSe SeNsItIvE!

LOOKING AT YOUR DIRECTORY:

```
% ls                       <-- see your file names
% ls -l       <-- see lots of info about files
% ls -l -a           <-- see even dotted files
```

TYPING AN INDIVIDUAL FILE:

```
% cat guava                      <-- display guava
% pg guava        <-- display a page at a time
```

DELETING A FILE:

```
% rm papaya                      <-- delete papaya
  papaya: ? y       <-- reply with a lowercase
                        y to confirm deletion
```

COPYING A FILE:

```
% cp mango breadfruit            <-- copy mango to
                                       breadfruit
```

RENAMING A FILE:

```
% mv mango pineapple           <-- rename mango to
                                        pineapple
```

SEARCHING FOR A STRING:

```
% grep -i 'roast seeds' *          <--
search all
                   files for an occurence of
                "roast seeds", ignoring case
```

HELP:

```
% man -k drivers             <-- look for a manual
                                 page on 'drivers'
```

FORWARDING MAIL BACK TO OREGON'S VAX:

In your permanent UNICOS root directory make a file called .forward that contains the address you want to forward your mail to. I.e.,

```
JSMITH@OREGON.UOREGON.EDU
```

WORKING WITH UNICOS DIRECTORIES:

SEE WHERE I CURRENTLY AM:

```
% pwd                <-- print working directory
```

MAKE A NEW DIRECTORY:

```
% mkdir vegetables         <-- make a vegetable
                      dir under the current dir
```

CHANGE DIRECTORIES:

```
% cd berries      <-- change to berries subdir
% cd ..                <-- come back one level
% cd $HOME     <-- come back to main directory
% cd ~         <-- come back to main directory
% cd ../nuts        <-- go back one level, then
                       down to the nuts subdir
```

MOVING FILES INTO A DIRECTORY:

```
% mv seeds ~/myarticle/.     <-- move the file
                 "seeds" to the myarticle dir

% cp xyz ~/mybook/abc  <-- copy the file "xyz"
                  to the mybook dir, renaming
                               it to be "abc"
```

USING THE TAR AND COMPRESS UTILITIES:

```
% tar -cfv my.tar fall.dat    <-- tar fall.dat
% compress my.tar          <-- compress my.tar

% uncompress my.tar    <-- decompress my.tar.Z
% tar -xfv my.tar             <-- detar my.tar
```

USING THE COMMON FILE SYSTEM:

```
% cfs              <-- start Common File System

? list lo=g  <-- list all info about CFS files
               in the current CFS subdirectory

? save apples        <-- save (new) file apples
? replace pears   <-- overwrite (existing) CFS
                    file pears with file pears
? store plums      <-- save/replace file plums
                  to CFS regardless of whether
                      it already exists or not
? save info="feb89 data" apricots     <-- save
                apricots with descriptive info
? save lclfile:cfsfile   <-- save file lclfile
                           using name cfsfile

? get grapes      <-- retrieve grapes from CFS
? get lclfile:cfsfile     <-- retrieve cfsfile
                          and call it lclfile

? delete raisins   <-- delete CFS file raisins

? modify aval=024681/r/-/s  /012345   <-- make
         u12345's directory readable by u24681

? add fruit     <-- creates a CFS fruit subdir
? dir=fruit     <-- change dir to fruit subdir
? dir=/012345    <-- change dir to user u12345
                  root CFS dir (note leading u
                      is replaced with a zero)

? remove fruit     <-- remove the fruit subdir

? ls               <-- do a UNICOS ls (you can
                 also do a cd or a pwd or a !)

? end
```

You can also enter CFS commands as:

```
% cfs command1, command2, command3, ...
```

Be aware that there are two CFS groups, "a" and "b". By default, you store CFS files on group a, but, if you have something that's truly irreplacable, you may also want to store an archival copy on grp=b, however note that physical CFS failures are extremely uncommon.

TRANSFERRING FILES WITH FTP:

You'll usually build files for the Cray on the Oregon VAX (or on a microcomputer) and then use standard FTP commands to transfer the files to Illinois.

For example, assume you have a number of files in the VAX subdirectory [.CRAFT] that you want to move to your scratch area on the Cray (and then eventually to the Cray's CFS). You'd do:

```
$ SET DEF [.CRAFT]
$ FTP UX.NCSA.UIUC.EDU
Name: u12345               <-- your Cray username
Password: abc123456        <-- your Cray password
Account: xyz                <-- your Cray account
*CD /scr1/u12345     <-- change to your scratch
                 area; note you can't use $SCR
*NOINTERACTIVE         <-- stop idiotic confirms
*MPUT *.*           <-- move all files in subdir
*QUIT
```

You will not be prompted for confirmation before existing files with the same filename are overwritten, so be careful you don't accidentally delete a file you want to keep.

USING NPRINT TO MOVE FILES:

To ship files from the Cray to the Oregon VAX, use:

```
% nprint -d jsmith -n oregon stats.f
```

MAKING CHANGES TO YOUR FILES WITH VI

If you need to make some changes to your files on the Cray, use the VI full screen editor.

VI on the Cray should generally work just like VI on the VAX (see the VI panel in the editor section for some basic hints on using VI), but be aware of a couple of special requirements:

(1) You **must** declare your terminal to be **vt100-nonl** rather than simply a vt100

(2) You **must** create a file in your root directory called .exrc containing the line:

set notimeout

You'll know you missed one or the other of these fixes if your cursor keys work badly (if at all).

EMACS?

EMACS is not currently available on the NCSA Crays.

COMPILING CRAY FORTRAN:

The following example assumes you want to run a little fortran program to calculate some univariate stats on data in file "stuff.dat"

```
% vi stuff.dat                  <-- make data file
i                             <-- goto insert mode
24.5
13.297
 ...
<esc> :wq                  <-- write the data file
% vi stats.f      <-- make stats.f source file
i
      PROGRAM STATS
      IMPLICIT NONE
      REAL X, SUMX, SUMXX, MINX, MAXX
      REAL XBAR, SSDEVX, STDDEVX
      INTEGER N
      DATA SUMX/0.0/, SUMXX/0.0/,
     1 MINX/99999.9/, MAXX/-99999.9/, N/0/
      OPEN (UNIT=10, FILE='stuff.dat',
     1 STATUS='OLD')
      OPEN (UNIT=20, FILE='summary.dat',
     1 STATUS='UNKNOWN')
100   READ (10, '(F10.5)', END=200) X
      SUMX = SUMX + X
      SUMXX = SUMXX + (X**2)
      N = N + 1
      MINX = MIN (MINX, X)
      MAXX = MAX (MAXX, X)
      GOTO 100
200   XBAR = SUMX / FLOAT (N)
      SSDEVX=SUMXX - (SUMX**2 / FLOAT(N))
      STDDEVX = SQRT (SSDEVX / FLOAT(N-1))
      WRITE (20, '(''  N  SIGMA X       '',
     1 ''MAX      MIN     MEAN    STDEV'')')
      WRITE (20, '(I3,5(2X,F7.2))')
     1 N,SUMX,MAXX,MINX,XBAR,STDDEVX
      CLOSE (10)
      CLOSE (20)
      END
<esc> :wq

% cft77 -e m stats.f   <-- compile stats.f and
                      generate stats.l listing
% segldr stats.o          <-- load object file
% a.out                     <-- run executable
% more summary.dat      <-- look at the output
```

IMSL:

To compile and link a Fortran program with the IMSL libraries, use:

```
% cft77 prog.f
% segldr -l imsl -D dupentry=ignore prog.o
```

NAG:

To compile and link a Fortran program with the NAG libraries, use:

```
% cft77 prog.f
% segldr -l nag -D dupentry=ignore prog.o
```

NCAR GRAPHICS:

```
% vi mygraf.f
i
      REAL X(20),Y(20)
      DATA X/1.3,2.5,3.1,4.2,6.5,
     1 8.7,9.3,10.2,11.3,13.754,
     1 14.1,14.2,15.44,16.49,17.22,
     1 18.88,18.99,20.00,21.2,24.0/
      DATA Y/10.2,8.8,6.5,5.6,4.4,
     1 3.4,2.2,1.1,0.,1.5,
     1 1.7,1.9,2.3,2.6,2.9,
     1 3.4,3.7,8.8,9.2,9.7/
      CALL OPNGKS
      CALL ANOTAT('X LAB$','Y LAB$',0,0,0,0)
      CALL EZXY(X,Y,20,'SAMPLE TITLE$')
      CALL CLSGKS
      END
<ESC> :wq

% cft77 -e m mygraf.f  <-- compile fortran src
% ncargldr mygraf.o    <-- link ncar libraries
% a.out                        <-- run program
% setenv GRAPHCAP t4010.g     <-- declare term
% setenv FONTCAP font0        <-- declare font
% cgmtrans GMETA            <-- generate graph
<return>                     <-- exit cgmtrans
```

DI-3000

DI-3000 is a Fortran-callable graphics library. (Note that some NCSA does **NOT** have all of the optional Precision Visual/DI-3000 add-on products).

```
% vi mygraf2.f
i
      REAL X(20),Y(20)
      DATA X/1.3,2.5,3.1,4.2,6.5,
     1 8.7,9.3,10.2,11.3,13.754,
     1 14.1,14.2,15.44,16.49,17.22,
     1 18.88,18.99,20.00,21.2,24.0/
      DATA Y/10.2,8.8,6.5,5.6,4.4,
     1 3.4,2.2,1.1,0.,1.5,
     1 1.7,1.9,2.3,2.6,2.9,
     1 3.4,3.7,8.8,9.2,9.7/
      CALL JEZINI(1)
      CALL JSZLIN(X,Y,20,'$SAMPLE TITLE$',
     1 '$X LABEL$','$Y LABEL$',
     1 0.0,24.0,0,12.0,1,0)
      CALL JEZSHW
      CALL JEZTRM
      END
<ESC> :wq

% setenv PVI_ROOT /usr/local/apps/graphics\
/pvi/release      <-- define PVI lib (2 line!)

% cft77 -e m mygraf2.f <-- compile fortran src
% di3load t14 mygraf2.o -GM  <-- link with PVI
% a.out  <-- run program (with Tek4010 output)
% mftran           <-- run metafile translator
```

GETTING STARTED WITH MOVIE.BYU:

MOVIE.BYU is one of several graphics packages available on the Cray X-MP. The following example will give you a brief introduction to MOVIE.BYU.

NOTE: You must be on a TEK 4010 compatible terminal or emulator (such as a PC with Kermit) to do the following example.

```
% utility    <-- lowercase; may not be abbrev.
> make volu elli
<SOLID ELEMENT DATA?> <return>
<COORDINATES ...> 0 0 0
<OUTER RADII ...> 30
<INNER RADII ...> 20
<STARTING AND ENDING LATITUDE ...> -45 45
<STARTING AND ENDING LONGITUDE ...> -45 180
<NUMBER OF ELEMENTS: ...> 6 6 6
MAKE-VOLU  >>>  <return>
MAKE  >>  <return>
> geom writ
<WRITE GEOMETRY FILE>  exp3.geo
<CHANGES?> no
GEOM  >>  <return>
> exit
<ABORT EXIT?>
no
```

NOTE: <CTRL>-C is TEK 4010 screen clear when you see a "More>" prompt.

```
% display    <-- lowercase; may not be abbrev.
:> 1    <-- that's the number one, not an "el"
geometry file:  exp3.geo
displacement file:  <return>
scalar function file:  <return>
:>  view.rotate
rotation coordinates:  30 -30
:>  image.line.hidden
:>  display
group to display:  <return>
<return>   <-- when finished looking at image
:>  exit
```

For more information on MOVIE.BYU see the MOVIE.BYU manual or HELP SOFT GRAPH MOVIE from UNICOS.

SOFTWARE AVAILABLE AT NCSA ON THE CRAYS:

```
Computational Fluid Dynamics:
  FIDAP, FLOW3D, KIVA, SALE3D

Chemistry:
  AMBER, AMPAC, CADPAC, CHARMm, DISCOVER,
  GAMESS, GAUSSIAN86, GAUSIAN88, GPRLSA,
  GROMOS, MACROMODEL, MM2/MMP2, MOPAC,
  RANFOLD, RNAFOLD

Electrical Engineering:
  HSPICE, MSC/MAGNUM, PISCES, SPICE, SUPREM4,
  UM-SPICE

Graphics:
  Blaze, MOVIE.BYU, NCAR(GKS), PVI Contouring,
  PVI DI-3000, PVI GK2000, PVI GRAFMAKER, PVI
  Metafile system, PVI TEXTPRO

Mathematics and Statistics:
  ACM Algorithms, BCSLIB, CALMATH, CAYLEY,
  CFTMATH, ECMFFT, EDA, EISPACK3, ELLPACK,

FFTLIB, IMSL (Versions 10.0 and 9.2), ITPACK,
  LAPACK, MINOS, MINPACK, NAGLIB, ODEPACK,
  PCGPACK, PORT, SCILIB, SLATEC, SMPAK, SPSSX,
  SSDLIN, TOEPLITZ, VECPACK, XMP

Solid Mechanics and General Engineering:
  ABAQUS, ANSYS, DADS, DYNA2D, DYNA3D, INGRID,
  MARC/MENTAT, MAZE, MSC/NASTRAN, NIKE2D,
  NIKE3D, ORION, PATRAN II, TAURUS, TOPAZ2D,
  TOPAZ3D

Systems and Utilities:

AS (assembler), BENCHLIB, CC, CDBX (debugger),
  CFT, CFT2, CFT77, NCC, OCC, PASCAL, SCC

For the latest info on Cray software, enter:

% help software
```

Note that software availability is currently quite fluid due to the recent conversion from COS to UNICOS – for example, the general purpose simulation language SLAM II is scheduled to be made available on the Crays shortly.

CONTACTS AT NCSA:

To be put on the NCSA mailing list for datalink (the NCSA technical newsletter) and access (the NCSA general newsletter), send a request to Tammie Hildreth, Documentation and Publications, NCSA, 152 Computing Applications Building, 605 E. Springfield Avenue, Champaign, IL 61820 or call 217-244-0072.

For a list of Cray technical documents, request a copy of the Technical Resources Catalog from NCSA Documentation and Publications by calling Kay Weidner at 217-244-8045.

To contact the consultants at NCSA, call 217-244-1144 or send mail to consult@ncsavms or consult@ncsaa.ncsa.uiuc.edu

NCSA also makes software for Macs, PCs, Suns and other systems available (at no charge) via anonymous FTP from ZAPHOD.NCSA.UIUC.EDU

INDEX:

"We hear only the questions to which we are capable of finding an answer." F. Nietzsche

"Ooh, are you ready now?" Kickstart My Heart, Motley Crue