

Basic Unix Skills on MEnet Systems

Version 1.22

R W Kaszeta

June 18, 1999

Contents

1	The MEnet Machines	1
2	The Basics	1
3	A Unix Overview	2
4	Finding Information	3
5	Basic Unix Commands	4
6	Modules	6
7	Editors	6
8	Email	6
9	Internet Applications	7
10	The X Window System	8
11	Printing	8
12	Programming and Compiling	9
13	Graphics	9
14	Scripting	10
15	Process Control	10
16	Archiving and Compressing	11
17	Other Useful Programs	12
A	Basic Unix Commands (alphabetically listed)	13
B	What are all these dotfiles, anyways?	14
C	Secure Passwords	15

Introduction

This course is meant to provide a fairly basic introduction to the MEnet computers for users that are new or unfamiliar with the Unix operating system. Our goal is not really to teach you the entirety of Unix, but to teach you enough to use the MEnet machines comfortably and learn how to find out the answers to most of your Unix questions yourself.

We encourage users to give us feedback and recommendations for improving this document. Please mail suggestions to me (kaszeta@me.umn.edu) or MEnet (menet@me.umn.edu).

1 The MEnet Machines

MEnet provides public Unix workstations in two public labs:

ME 10 is the largest MEnet lab. The left side of ME10, ME10a, contains 9 SGIs, 5 Suns, and 4 Linux machines. The right side of ME10, ME10b, contains 6 Sun workstations.

ME 472 contains 4 SGIs and a Linux machine.

In addition, MEnet maintains a number of servers, administration machines, experimental lab machines, and office machines. A complete list of machines administered by MEnet is available on the MEnet services page, <http://www.menet.umn.edu/services>.

As mentioned above, the MEnet workstations are of three types, Sun Microsystems running Solaris 2.5, Silicon Graphics running Irix 5.x/6.x, or Intel machines running Debian Linux 1.3/2.0. Generally, each of these machines is configured and used in a similar manner, but small differences (and what software is available) depend on the particular platform you are running on.

2 The Basics

2.1 Logging In

Logging into the Unix machines is simple, whether you are logging in via the network or logging into the machine itself (hereafter called a “console login”). You simply type in the username and password (that you selected when you opened your account) when prompted. Note that both your username and your password are *case sensitive* (in other words, upper and lower case are treated as different characters).

2.2 Logging Out

How you log out when you are done depends on whether you are accessing the machine via the network or from the console. If you are logged in via the network, simply exiting your shell by typing `exit` or `logout` will log you out.

If you are logged in from the console, you simple select “Log Out” or “Exit FVWM” from the *Root Menu* (you access the

Root Menu by clicking the mouse buttons on the screen background).

2.3 Changing Your Password

To change your password, simply log in and type `yppasswd`. It will prompt you for your old password and then ask for your new password.

It is important to choose a secure password to maintain system security. From time to time MEnet staff members run a program called “Crack” which checks for easily-guessed passwords. If “Crack” guesses your password you will have to change it immediately. Guidelines for choosing secure passwords are listed in appendix C.

Note that your password is important; it stops other users from gaining access to your account. Never give your password to *anyone* for *any reason*. Sharing of your password is grounds for immediate account closure.

Also, it is important to maintain the secrecy of your password. Thus, you should not write it down and *never* email it or store it in a file on your account.

2.4 Forget Your Password?

The way Unix stores passwords, it is not possible for the systems staff to find out your password. However, the systems staff *can* change your password to something you know. If you have forgotten your password, come see the systems staff in ME155 in person, or call us at 626-9800. For security reasons, we will not respond to password change requests via email.

3 A Unix Overview

3.1 What is Unix?

Unix is a computer operating system, originally written in 1969 by Ken Thompson and Dennis Ritchie. Unlike most operating systems, Unix is written as a “portable” operating system, and can be run on literally hundreds of different types of computers (for example, MEnet runs various versions of Unix on three different computer platforms: Intel x86, Sun Sparc, and SGI MIPS). This portability allows users to use a consistent interface on many different types of hardware without much additional learning required.

Unix has a lot of powerful features, many of which are not present or not as powerful in other operating systems. These features include

Multitasking: Unix is a *multitasking* operating system—it is capable of running multiple programs (for multiple users even) simultaneously.

In addition to multitasking, Unix also has *memory protection*, which keeps individual processes separate in memory from each other. The primary advantage is that this generally keeps a faulty program from crashing another program, or even worse, the operating system.

Multiuser: Unix is also *multiuser*. Multiple people can log into a workstation (even at the same time) to do their work. The operating system keeps users’ files and program separate from each other, so that only you can access your files, and nobody else’s programs will interfere with programs you may be running.

3.2 Networking

Unix includes a large collection of networking programs (in fact, most of the Internet “backbone” is run using Unix servers of one sort or another), and is configured to allow users to communicate effectively and easily with other machines on the network.

One example of this is the user’s ability to *remotely login* to a computer and use it, even if the computer is in another location¹. Thus, at any given time multiple users may be using the same machine as you to get their work done.

Another example of Unix’s networking ability is its ability to share files across the network. On any Unix machine you log into, most of the files you are working with, such as your home directory, are not located on the local machine but are fetched from one of the servers in ME 155 over the network. Thus, on any MEnet machine you will see not only a similar work environment, but have access to the exact same set of files.

3.3 Filenames and Directories, and “Dot” Files

Unix is case sensitive when it comes to commands and filenames. Thus the file `file` would be different from the file `File`.

Additionally, Unix has no restrictions on the length of filenames, so filenames can be arbitrarily long. And unlike DOS, Unix treats “.” just as another file character, not as a divider between a filename and an extension. Thus you can have filenames such as “file.tar.gz”.

An important exception, however, is filenames starting with “.”, such as `.cshrc`. These files are treated just like other files except they do not normally show up in a directory listing (much like “Hidden Files” on other OSes).

Like any other operating system you may have used, Unix uses a hierarchical method of store files, with directories, subdirectories, etc. However, Unix uses a forward slash, `/`, as a separator between directories (unlike DOS which uses a backslash, `\`). Thus, the name `source/file.c` refers to the file `file.c` in the subdirectory `source`.

3.4 The Root Directory

All of the files on a Unix machine are stored in a single hierarchy, the “file system”. The very top of this file system, the “root directory”, is known as `/` (much akin to `C:\` on DOS machines)

¹In fact, in section 10, we’ll even show you how to run graphical programs on a remote machine and have the graphical output shown on your screen.

3.5 The Home Directory

When you first log into a Unix machine, it places you in your home directory (also known as `~`). Similarly, another user's home directory is known as `~username`). This directory is used only by you and programs you run, and is where most personal configuration files are stored. All of the work you do will be done in your home directory or subdirectories of it.

Note that a `cd` with no arguments will return you to your home directory.

3.6 Quotas

Your unix directory has a *quota*, a limit on the amount of storage space you are allowed. After this space is filled, no more files can be written in your directory. To find out both your current usage (and how much space you are allowed), type `quota` or `quota -v`.

If you are running out of room and trying to find out what is taking up so much space, use the `du` command to get a summary of how much is stored in each directory. You can also look at archiving and compressing your data, as described in section 16.

3.7 The Shell

Whenever you login to a Unix system you are placed in a program called the shell. It is called the "shell" because it sits between you and the actual operating system (usually called the "kernel"). Like most command-line driven computers, you can see its prompt at the bottom left of your screen. To get your work done, you enter commands at this prompt. The shell acts as a command interpreter; it takes each command and passes it to the operating system kernel to be acted upon. It then displays the results of this operation on your screen. For those users familiar with DOS, the shell is like `command.com`. Similar enough, in fact, that for most DOS commands there is a corresponding Unix command of nearly the same name:

DOS Command	Unix Command
<code>dir</code>	<code>ls</code>
<code>cls</code>	<code>clear</code>
<code>del</code>	<code>rm</code>
<code>copy</code>	<code>cp</code>
<code>move</code>	<code>mv</code>
<code>rename</code>	<code>mv</code>
<code>type</code>	<code>cat</code>
<code>cd</code>	<code>cd</code>
<code>md</code>	<code>mkdir</code>
<code>rd</code>	<code>rmdir</code>

On Unix machines, a number of different programs can be used as a "shell". By default, users are set up using a shell cause the "TC-Shell", or `tcsh` for short. Other available shells include the "C-Shell" (`cs` for short) and the "Bourne Again Shell" (`bash` for short).

3.8 Undelete

Due to the large numbers of files stored on the server and the multitasking nature of Unix, there is no "undelete" command.

So the first rule is to do your best not to delete anything you want to keep. If you do accidentally need something back, and it has been there more than a few days, then there is a chance we have a copy of it on our tape backups. Mail menet@me.umn.edu with your username and the name of the file you want restored, and we will attempt to restore it from tape. No guarantees, though.

4 Finding Information

4.1 Man Pages

Usually the first place to look for information is the "manual page". To look at the online manual page, type `man command` where "command" is the command you are trying to find information on.

4.2 The MEnet Web Page

Another useful place to look for information is the MEnet Web Page, <http://www.menet.umn.edu>. From this page you can find a number of useful references, such as the MEnet documentation page, the MEnet policies page, and other useful information.

4.3 FAQ

The MEnet home page includes the MEnet Frequently Asked Questions List (FAQ). Consult it before asking us any questions.

4.4 *Unix in a Nutshell*

A useful but thorough book for Unix beginners is *Unix in a Nutshell*, by Daniel Gilly, published by O'Reilly and Associates, ISBN 1-56592-001-5. Available in the Minnesota Book Center for \$20, it provides detailed explanations of unix commands, editors, and shell scripting. A similar guide to linux, *Linux in a Nutshell*, is also available. Two copies of each of these books are located in ME 10a.

4.5 Online Guides

The WWW has a number of good Unix guides. Two sites I recommend are: <http://www.geek-girl.com/Unixhelp/> and <http://albrecht.ecn.purdue.edu/~taylor/4ltrwr/html/unixman.html>

4.6 Other Users

One of the reasons we like having large workstation labs is that it allows users to learn from each other. Ask the people around you how to solve problems. You may learn something the systems staff doesn't know.

4.7 The MEnet Office

If all else fails, talk to the systems staff in ME 155 via email, menet@me.umn.edu, phone (626-9800), or in person.

5 Basic Unix Commands

5.1 Working with files

5.1.1 Copying Files

To copy files, use the `cp` command:

```
cp source destination
```

To copy multiple files to a common destination, simply list the multiple files:

```
cp source1 source2 destination
```

For more files, this is a good place to use wildcards. See section 5.5.

5.1.2 Deleting Files

To remove a file use the command:

```
rm filename(s)
```

A better idea is to have `rm` ask you about each file, this is done with

```
rm -i filename(s)
```

To remove a directory use the command:

```
rmdir directory_name
```

or

```
rm -r directory_name
```

You cannot retrieve a file or directory that has been removed; it has been removed permanently. If you *really* need it back, see section 3.8.

5.1.3 Moving and Renaming a file

To move a file to another location, use `mv` which is like `cp` except it removes the original copy;

```
mv source destination
```

To rename a file, you simply `mv` it from its original name to the new name.

5.1.4 Displaying a File

To type out a file to the screen, simply `cat` it:

```
cat filename
```

`cat` also works with multiple files (the name “cat” comes from “concatenate”)

```
cat filename1 filename2
```

If instead you’d like to page through a file (see it a screen at a time), `more` it:

```
more filename
```

Additionally, you could use `less`, a version of the `more` command with additional functionality. Also note the `head` and `tail` commands, which show the first few and last few lines of a file, respectively.

5.1.5 Determining what type a file is

The `file` command examines the content of a file and reports what type of file it is. To use the command enter:

```
file filename
```

Use this command to check the identity of a file, or to find out if executable files contain shell scripts, or are binaries. Shell scripts are text files and can be displayed and edited.

5.2 Working with and navigating through directories

5.2.1 Listing Files in a directory

You can use the `ls` command to list the files in a directory:

```
ls directory
```

`ls` assumes the current directory (also known as “.”) by default. To see the file sizes and other information, use the `-l` option

```
ls -l directory
```

and if you want to see all the files, include the “dot” files, use the `-a` option:

```
ls -a directory
```

5.2.2 Changing Directories

To change your current working directory use the command:

```
cd pathname
```

where `pathname` specifies the directory that you want to move to. Note that as mentioned above, typing `cd` by itself will take you back to your home directory.

5.3 File Permissions

Every file and directory in your account can be protected from or made accessible to other users by changing its access permissions.

You can only change the permissions for files and directories that you own.

To display the access permissions of a file or directory use the the command:

```
ls -l filename (directory)
```

This displays a one line summary for each file or directory. For example:

```
-rwxr-xr-x 1 kaszeta...
```

This first item, `-rwxr-xr-x`, represents the access permissions on this file. The following items represent the number of links to it; the username of the person owning it; the name of the group which owns it; its size; the time and date it was last changed, and finally, its name.

For each file, there are three different types of permissions:

r read the file or directory

w write to the file or directory

x execute the file or search the directory

Each of these permissions can be set for any one of four types of user:

u the user who owns the file (usually you)

g members of the group to which the owner belongs

o all other users

a all of these

The access permissions for all three types of user can be given as a string of nine characters:

```
user  group  others
r w x  r w x  r w x
```

5.3.1 Changing Access Permissions

To change the access permissions for a file or directory use the command

```
chmod mode filename
chmod mode directory_name
```

The "mode" consists of three parts: who the permissions apply to, how the permissions are set and which permissions to set. For example, to give yourself permission to execute a file that you own:

```
chmod u+x file1
```

This gives you execute permission for the file `file1`. As another example, to give everyone read access to `file2`,

```
chmod a+r file2
```

As a final example, to take away permissions you use a `-` instead of a `+`, so to take away read access to all users for `file3`, you'd type

```
chmod a-r file3
```

You can also set the permissions using *numerical mode*, where the permissions for a file as represented as a three-digit number. Read permission is given the value 4, write permission the value 2 and execute permission 1.

```
 r  w  x
 4  2  1
```

These values are added together for any one user category:

```
1 = execute only
2 = write only
3 = write and execute (1+2)
4 = read only
5 = read and execute (4+1)
6 = read and write (4+2)
7 = read and write and execute (4+2+1)
```

So access permissions can be expressed as three digits. For example:

```
chmod 640 file1  user  group  others
chmod 754 file1  user  group  others
chmod 664 file1  user  group  others
```

5.4 Redirection

Unix considers any device attached to the system to be a file. And that includes your terminal!

By default, a command treats your terminal as the standard input file from which to read in information. Your terminal is also treated as the standard output file to which information is sent from the command.

This action can be changed by redirecting standard input and standard output from and to any other file.

To redirect the standard input for a command use the `<` (less than) character followed by the name of the input file. For example:

```
mail menet < memo
```

This redirects the standard input to the mail command so that it comes from the file memo. The effect of this is to mail the contents of this file to the user menet.

Similarly, To redirect the standard output from a command use the > (greater than) symbol followed by the name of the output file. If the file that you redirect standard output to does not already exist it will be created. For example:

```
a.out > file
```

This redirects the standard output from the a.out command so that it goes to the file file. A similar effect can be had with the >> redirector—it appends the standard output to the named file instead of overwriting the file.

5.5 Wildcards

Wildcard characters can be used to represent many other characters. Use them whenever you need to define a string of characters, such as a filename, for use with a command.

Useful wildcards are *, which matches any characters, and ?, which matches any single character. Note that unlike DOS, * even matches the . character.

For example, if you wanted to copy all the .jpg files to the image directory, simply

```
cp *.jpg image
```

6 Modules

The Modules package provides a single command line interface for manipulating your UNIX environment. The Modules interface enables users to easily add, change, and remove application specific paths, aliases, and environmental variables from your environment.

Instead of asking where something is located, you just inform Modules which package you want, using the module load command and it adjusts your environment ‘on the fly’ so that you are able to use the package or application.

For more information on the Modules package, consult <http://www.menet.umn.edu/docs/modules.html>.

7 Editors

To successfully use MEnet computers, it is necessary for users to become familiar with one of the standard MEnet text editors.

7.1 Nedit

nedit is probably the simplest and easiest to use editor available on the MEnet systems. It’s primary advantage is it’s simplicity, providing a fairly intuitive menu- and mouse-driven graphical interface for editing files. To edit a file using nedit, simply type

```
nedit filename
```

The disadvantage is that unlike the rest of the editors I discuss here, nedit requires a graphics terminal to use.

7.2 Pico

pico is a simple text-mode editor similar in features and usage to Nedit. It has a fairly intuitive interface with common commands shown right on the screen. To edit a file using pico, simply type

```
pico filename
```

7.3 Emacs

Emacs is a very powerful (but to some people, somewhat cryptic) Unix text editor, providing a detailed macro language, powerful commands, extensive configurability, and the ability to easily do content-based editing (i.e. load special formatting and editing rules for C code, FORTRAN, L^AT_EX code, etc.), and it runs in both text mode and graphical mode. For more information, you can load Emacs by typing emacs and starting the tutorial with ctrl-h t.

7.4 VI

VI is another very powerful (but again, some people find it cryptic) text-mode editor, available on just about every Unix machine ever made. Additionally, it is very small and very fast. The best place to get information on vi is to look at the “VI Lovers Home Page”, <http://www.cs.vu.nl/~tmgil/vi.html>, or the “VI Editor FAQ”, <http://www.menet.umn.edu/docs/vi/>.

8 Email

One of the most important reasons people want access to MEnet machines is for email. A number of different email programs are available to MEnet users, the popular ones are listed below:

8.1 Elm

Elm is a simple, easy-to-use text-mode mailer for Unix. To use, simply type elm on the command line. For more help with Elm, consult the online Elm help, or look at the Elm manual on the MEnet documentation page, <http://www.menet.umn.edu/docs/elm/elm.html>.

8.2 Pine

Pine is a mailer similar in capability to Elm, but with a different interface. Simply run pine to load Pine. A good place to find more information on Pine is the Pine FAQ, <http://www.ii.com/internet/messaging/newsgroups/launchers/comp.mail.pine/>

8.3 Mailx

Mailx, also known as `Mail` or `mail` on most MEnet machines, is very simple command-line email program. Most people will not want to use Mailx for reading mail, but it is a very convenient program for sending short email messages. For example, to send a quick email to user “fred@abc.com”, you’d enter:

```
mail fred@abc.com
```

and Mailx would ask you for a subject. Then you simply enter the text of your message, ending with a `ctrl-d` or a single “.” on a line.

Similarly, this works with the Unix shell redirection operators, so to send the file `process.c` to “fred@abc.com”, you’d enter:

```
mail fred@abc.com < process.c
```

8.4 Other Programs

MEnet also has a number of additional email programs available. For `emacs` users, `emacs` includes *two* mail readers, `VM` and `RMAIL`. To use `VM`, you simply enter `M-x vm` inside of `emacs`, and similarly you use `M-x rmail` for `RMAIL`. For complete information on these programs consult the online `emacs` documentation.

8.5 Forwarding Email

Some users may wish mail addressed to their MEnet address be delivered to another account (like their gold/maroon account). To do this, all the user has to do is create a file called `.forward` in their home directory, and in this file place the email address where you wish your mail to be forwarded.

Note that you should take care that your forwarding address is correct—it’s a good idea to send a test message. Also make sure you don’t set up a *mail loop* (where you forward email to an address which forwards it back to you).

9 Internet Applications

9.1 Web Browsing

MEnet supports two web browsers for surfing the World Wide Web.

9.1.1 Netscape

Netscape Navigator versions 4.07 and 3.01 are installed on all MEnet machines. You can access it simply by typing `netscape` or `netscape3`, respectively.

Note that while Netscape Navigator provides a mail reader, we do not recommend its use, since it interacts poorly with the other unix mail readers.

Also note that Netscape typically keeps a large cache of recently loaded pages on disk, usually around 5 megabytes in size. If you are running out of space on your account it is probably a good idea to reduce the size of this cache through the “user preferences” option in the menus.

9.1.2 Lynx

Lynx is a text-mode browser which does not require any graphics to work. Its most useful application is users that are trying to access the WWW via a modem dialin line and don’t have any graphics access. Simply type `lynx` to use.

9.2 FTP

One of the most common methods used to move files between different machines and accounts is the “File Transfer Protocol”, also known simply as “FTP”.

The easiest version of FTP to use is the program `NCFTP`, accessed as `ncftp`. To transfer files, you have to first connect to a machine, and then either send files to that machine, or retrieve files from it.

9.2.1 Opening a connection

Once `NCFTP` is running, you open a connection to a machine by typing

```
open -u machinename
```

It will then ask you for an account name and password, and connect you to that account on the remote machine.

9.3 Navigating in FTP

When you are in FTP, you can navigate through directories using the same commands as you would at the shell prompt, `ls` and `cd`. Similarly, you can change the local working directory (where you are on the local machine) with the `lcd` command.

9.3.1 Downloading Files

Once connected, you can retrieve files from the remote machine using the `get` command:

```
get filename
```

which will retrieve the remote file and place it on your local machine in the current directory.

With `NCFTP` you can retrieve multiple files at a time, either by listing the multiple files:

```
get file1 file2
```

or by using the Unix wildcard syntax:

```
get file*.c
```

Type `help get` at the `ncftp` prompt for help.

9.3.2 Uploading Files

To send files to the remote machine, you use the `put` command, which otherwise works like the `get` command. Type `help put` for more information.

9.3.3 Anonymous FTP

The above examples assume you have an account on the remote machine. Often this is not the case. Many machines are set up with *anonymous FTP*, which allows you to access files without an account. To log into a machine using anonymous FTP, you either login in as user `ftp` or `anonymous`, or simply `open` the ftp session without the `-u` option given above.

10 The X Window System

The graphical user interface on the Unix Workstations is known as “The X Window System”, or more simply just “X”.

10.1 Running Remote X11 Applications

One of the most powerful capabilities of X is the ability to run *graphics* programs on a remote machine and have them display on your current machine. This is done by setting an *environment variable* called `DISPLAY` telling the program where to send its graphical output. The steps for doing this are:

1. Allow the remote machine (“zorro” in this example) to display graphics on your own machine:

```
xhost +zorro
```

2. Rlogin to the remote machine:

```
rlogin zorro
```

3. Set the `DISPLAY` variable to the name of the machine you are on (“lobo” in this example), followed by a “:0”:

```
setenv DISPLAY lobo:0
```

4. Run your program:

```
xv &
```

11 Printing

11.1 What Format?

The MEnet printers are *Postscript* printers, meaning that any graphics you print to them have to be Postscript (`.ps` files). Additionally, most of the MEnet printers are also capable of printing plain text, so you can print source code and other text files. Typically, most MEnet applications that can print do so by generating a postscript file.

11.2 Submitting a Print Job

To submit a print job on an MEnet printer, you use the `lpr` command:

```
lpr -P printername file.ps
```

where `printername` is the name of the printer you wish to print to. The public lab printers are:

me10a is the HP 4000N printer on the left side of ME 10.

me10a is the HP 4000N printer on the left side of ME 10, duplex printing.

me10a is the HP 4000N printer on the left side of ME 10, short-side duplex printing.

me10b is the HP 4MV printer on the right side of ME 10.

color155 is the Color HP 5MV printer in ME155.

me472 is the HP 4MV in ME 472.

These printers are for work-related printing *only*. For personal printing, the same printers are available by appending a ‘p’ to the queue name, i.e. `me10p`, `me472p`, ...

For information on printing in general, check out the web page <http://www.menet.umn.edu/docs/printing>, for information on duplex printing see <http://www.menet.umn.edu/docs/printing/duplex.html>, and for information on color printing see <http://www.menet.umn.edu/docs/printing/color.html>.

11.3 Checking on and Canceling Print Jobs

The print system works as a *queue*, where the first jobs submitted are the first jobs printed. To see the current status of the queue for a particular printer, do

```
lpq -P printername
```

which lists the current jobs waiting to be printed.

Occasionally you may wish to cancel a print job (it is printing incorrectly or you sent it to the wrong printer or something). To do this you use the `lprm` command:

```
lprm -P printername jobid
```

where `jobid` is the job number listed with the `lpq` command.

Sometimes `lprm` doesn’t work... in that case contact the MEnet staff to clear the print job.

11.4 How Much Have I Printed?

MEnet has implemented a print accounting system. To see how many total pages you have printed on MEnet printers, consult the MEnet print accounting page, <http://www.menet.umn.edu/stats/printer/>.

11.5 Ghostview

Due to the cost of printing materials, it is recommended that you preview your files before printing them. To preview your files, use Ghostview. On the Sun and SGI machines,

```
ghostview file.ps
```

and on the Linux machines:

```
gv file.ps
```

to get a graphical view of what your file will look like when printed.

11.6 mpage and nenscript

Additionally, there are two useful programs that can preformat your document.

mpage reformats your document so that it prints 2-up or 4-up (multiple pages printed on a single page by shrinking them down). For example, to print 2 pages per page,

```
mpage -2 -P printername file.ps
```

Do a `man mpage` for more information.

nenscript does similar preformatting of text file, and places a header on the page. For example, to print your text file two-up with headers, do

```
nenscript -2rG -P printername file.txt
```

Do a `man nenscript` for more details.

12 Programming and Compiling

12.1 C

On the Suns and SGIs, there are two different C compilers available, `cc`, the vendor-supplied C compiler, and `gcc`, the GNU C Compiler from the Free Software Foundation. The Linux machines only have `gcc`. The syntax for these compilers is nearly identical. To compile a C program whose source code is in the file `program.c`, the syntax is

```
gcc program.c
```

(or `cc`, if that's what you are using). If the compilation is successful (no errors), it creates the executable file `a.out`.

You can specify a different name for the output executable with the `-o` option:

```
gcc -o program program.c
```

does the same compilation as above, but places the resulting executable in `program`.

This is just the briefest overview of the C compiler. For more information, look at the man pages for `cc` and `gcc`, or consult a C programming handbook.

12.2 FORTRAN

Compiling FORTRAN code is similar to compiling C code except that the compiler is `f77` on the Suns and SGIs, and `g77` on the Linux machines. Otherwise, the usage is nearly identical.

13 Graphics

A number of programs are available on MEnet machines for viewing and editing graphics files

13.1 XV

XV is a rather powerful program for viewing (and printing) graphics. It can handle most of the common formats (gif, tiff, pcx, jpeg, etc.), and can do some simple image manipulation (color adjustment, rescaling, etc.). To run XV on a file, you simply call

```
xv imagefile
```

13.2 ImageMagick

ImageMagick is a suite of graphics manipulation programs. Their greatest feature is the ability to deal with almost any graphics format you can encounter. The most useful programs in this suite are `display`, which displays graphics files, `convert` which easily converts one format to another, and `identify`, which tells you what format and size a graphics file is.

13.3 Xanim

Xanim is also a graphics viewer, but for animated graphics instead of images. Like XV, Xanim supports a wide number of animation formats, including MPEG, Quicktime, and Indeo. To view an animation with Xanim, simply run

```
xanim animationfile
```

13.4 Movieplayer

The SGIs also have an animation player called Movieplayer, which although supporting fewer animation types than Xanim tends to run a bit faster. To run Movieplayer, simply type `movieplayer`.

13.5 XFig

XFig is a simple drawing program for producing vector (i.e. rescalable) graphic images. It is capable of exporting its image in a large number of graphics formats, including PostScript and JPEG. To run XFig, simply type `xfig`.

13.6 Gscan

There is a scanner available for use in ME 10. To use it, simply load your image onto the scanner and run `gscan` on any of the SGI's. Gscan is capable of exporting images in either SGI RGB or TIFF format (which can be converted to more convenient formats using XV).

13.7 Other Programs

Other graphics commands may be available, consult the MEnet Applications Page for more information.

14 Scripting

One very powerful feature of the Unix shell environment is the ability to generate “shell scripts”, which are a collection of commands that can be run together in a sequence (much like the “batch files” that users can write in DOS).

The first line of script always looks like the following:

```
#!/bin/csh
```

The `#!` characters tell the operating system to treat the program as script instead of a simple text file, and the characters following the `#!` tell the operating system what interpreter is supposed to process the commands in the file (in this case, `/bin/csh`).

14.1 C-shell Scripts

The simplest scripts to write are C-shell scripts, since they use the exact same syntax as the default user command line. So you can create simple C-shell scripts just by starting a file with `#!/bin/csh` and then listing out the commands you would type if doing each step by hand. For example, if you want to run your program (called `a.out`), print the output to the printer, and then clean up and delete the output file, the script would look like:

```
#!/bin/csh
a.out > file.out
lpr -P me155 file.out
rm file.out
```

Note that for the operating system to know that the file is a script, you must also make sure it is marked as executable using `chmod +x file`. Many of the control structures available in C, such as “for” loops and “if” statements, have similar structures in `csh`. `man csh` for more information, or consult the “Advanced Unix Skills on MEnet Systems” course handout, available from the MEnet documentation page.

14.2 bash scripts

For more complicated scripts, users may find the syntax of another shell, GNU Bash, more convenient. `man bash` for more info, or consult the “Advanced Unix Skills on MEnet Systems” course handout, available from the MEnet documentation page.

14.3 Other Scripting Languages

Additionally, a number of additional unix scripting language tools are available to users. One of the most highly recommended tools is `perl`, a language designed for conveniently manipulating text strings, numbers, and files. `man perl` for more information, or better yet, read either *Learning Perl* by Randal Schwartz, or *Programming Perl* by Larry Wall, Tom Christianson, and Randal Schwartz. Both are available at the [Minnesota Book Center](#).

²On occasion this does not work, so you can try either the more forceful `ctrl-\` or `kill` the process from another terminal session

15 Process Control

When you enter a command it invokes a program. While this program is running it is called a *process*. At any time, a user can have any number of processes running. There are several commands which allow you to manage the processes that belong to you. You cannot do anything with processes belonging to other users.

15.1 Foreground Processes

Generally, when you run a program, the parent process (your shell) waits for the command to complete and for its child process to die. Until then you are unable to enter another command. Commands entered in this way are known as *foreground processes*.

To cancel the process that is currently running the user simply types `ctrl-c`².

Similarly, a foreground job can be *suspended* (which is like killing but allows the job to be returned to the foreground) by entering `ctrl-z`. You can return to the suspended job by typing `fg`.

15.2 Background Processes

The opposite of a foreground job is a background job, which continues running while returning the shell prompt to you. To start a job in the background, you simply end the command with `&`. For example, to start your computational program `cruncher` in the background, you’d type:

```
cruncher &
```

If this program produces output, it is generally a good idea to redirect this output to a file using the methods described in section 5.4.

At any time, you can use the `jobs` command to show you all the jobs currently run by the shell.

15.2.1 Placing a Foreground Job Into the Background

You can place a foreground job in the background by:

1. Suspending it with `ctrl-z`.
2. When you suspend it, note the *job id* (the number in square brackets).
3. Background it by typing `bg %jobid`.

15.3 nohuping a job

Generally, when you exit a shell (or logout), all the background processes in that shell exit. You can cause the job to continue running with the `nohup` command:

```
nohup a.out &
```

which causes `a.out` to continue running even after you log out. This is known as a “background job”, and allows you to run long computations without tying up the computer console.

Note that MEnet policy requires all such processes to be “niced” (by at least the default value of 4), which makes them yield the CPU to other processes on the machine. To nice a job, you simply start it with `nice`:

```
nice +4 nohup a.out &
```

Higher values of 4 (up to 20) can be used to make the job “nicer” to other jobs. To change the nice value of an already running process, you use

```
renice +n PID
```

where PID is the process ID of the process (see below). Note that you can only make jobs nicer, you can’t “un-nice” them.

15.4 Checking on running jobs

To monitor the state of your processes use the `ps` command³. Typically, `ps` produces a list of all the processes owned by you and associated with your terminal. To get a list of all processes on the machine, use `ps -ef` on the Suns and SGIs, and `ps -aux` on the Linux machines. To get a list of your own processes, type `ps -u yourusername` on the Suns and SGIs, and `ps -x` on the Linux machines.

The output of this command typically shows three useful pieces of information, the *Process ID* or “PID”, which is a number unique to that process, the “CPU” column, which shows how many CPU hours the code has used, and the name of the command you are running.

The `top` command is similar to `ps`, but shows only the processes using the most CPU on the machine.

15.5 Killing Jobs

Each process you start is usually completed within a few seconds. Sometimes a background process or a process without a controlling terminal hangs up and you will need to destroy this process by killing it.

To kill a process use the `kill` command.

```
kill PID
```

where PID is found using the `ps` command as detailed above.

Sometimes the process is “hung” and not responding to a `kill` command. In these cases you can try a `kill -1` or a `kill -9` which are more forceful ways of killing a process. In general, you should try a plain `kill` first, as it generally gives the application a chance to exit gracefully (i.e. clean up temp files, lock files, and such).

16 Archiving and Compressing

16.1 gzip

The most useful tool for storing data is `gzip`, which compresses files so that they take less space on the disk.

To compress a file, simply type

```
gzip file
```

After the `gzip` runs, you’ll notice that the file `file` has been replaced with the compressed version, `file.gz`, which is usually significantly smaller than the original.

To uncompress a file, simply run `gunzip` on it:

```
gunzip file.gz
```

which will replace the `.gz` file with the original.

16.2 tar

Another useful archival program is `tar`, which combines a number of files together into a single “archive file”.

The syntax for this command is

```
tar key name ...
```

where `key` is specified by a plethora of options (see abridged list below and unabridged list in the man pages) and `name` is the archive file name.

Here are some of the more commonly used keys:

c Creates a new archive.

f Used for taring to a file.

t Lists the contents of a tar file.

v Turns verbose on.

x Extracts selected files. If no file argument is given, the entire contents of the tar file is extracted.

Here is typical syntax for creating and extracting tar files:

```
tar cvf file.tar directoryname
```

creates `file.tar` containing all of the files in `directory`, while

```
tar xvf file.tar
```

extracts the contents of `file.tar`.

³Note that the `ps` command behaves differently on different systems, so `man ps` for usage info

16.3 Zip Drives

The Iomega Zip drives (100 MB storage drives similar to floppies) are accessible on either of the public Linux machines, `chipolte` in room 10 and `jalapeno` in room 472.

To use the zip drive, insert a DOS format disk in the drive and run the `jazip` program. Your disk will appear as the directory `/dosz`. Simply copy/move your file to `/dosz` to place it on the zip disk.

When you are done, simply run `jazip` again to unmount the disk (you cannot eject the disk until it is unmounted).

Blank DOS Format Zip disks are available in the bookstore.

17 Other Useful Programs

17.1 L^AT_EX

The Unix workstations don't have a word processor installed on them. However, they do include L^AT_EX, a document preparation system which produces very high quality documents (with well-typeset math expressions) using a compiler-like system. For more information on using L^AT_EX, consult...

17.2 Wincenter

The Unix workstations include a system called Wincenter, which allows you to access one of our Windows NT 3.51 workstations (and the associated Windows Applications such as Word, Excel, and EES) from the Unix console.

To run Wincenter on the SGIs and Suns, you first type

```
wincenter_on
```

followed by

```
wincenter
```

On the Linux machines, you type

```
wincenter servnt
```

17.3 Calendar Programs

MENet machines also include a number of calendar/scheduler applications. The three most common ones are Ical, Xcalendar, and Xcal, and have similar features, but different interfaces. To run these programs, run `ical`, `xcalendar`, or `xcal`, respectively. For more information on their usage, consult the man pages.

17.4 The MENet Application List

On the MENet services web page, <http://www.menet.umn.edu/services/>, we maintain a fairly up-to-date list of software available on our machines. Please consult this list if you are looking for software for a particular purpose.

A Basic Unix Commands (alphabetically listed)

Command	Description
cat	display a file
cd	change directory
chmod	show the permissions of a file
du	how much space is each file using?
elm	a mail reader
emacs	a text editor
exit	exit current shell (usually logs you out)
f77	a FORTRAN compiler
file	tell what type a file is
find	find files
finger	find out about another user
gfinger	Who is logged in to all MEnet machines
ftp (host)	transfer files to/from another machine
gcc	a C compiler
grep	look for a string in a file
gscan	use the scanner
gunzip	uncompress files
gzip	compress files
head	show the top lines of a file
jazip	use the ZIP Drive
kill	kill unwanted processes
less	page through a file (like more)
logout	log out
ls	show directory in alphabetical order
lynx	text-based web browser
mail	a simple mail program
mkdir	make a new directory
more	page through a file
nedit	a simple graphical text editor
netscape	graphical web browser
nice	make a process use less CPU
nn	another news reader
pine	another mail reader
ps	what processes are running on this machine
quota	how much disk space do I have?
rlogin (host)	connects to another machine
rmdir	remove a directory (must be empty)
tail	show the bottom lines of a file
tar	archive files
telnet (host)	connects to another machine
tin	news reader
top	what processes are running on this machine
vi	a common Unix text editor
w	who is logged in?
who	who is logged in?
yppasswd	change your password

B What are all these dotfiles, anyways?

Here are some of the common "dot files" for user configuration. Your account might not have all of these. For more information consult the MEnet documentation page, <http://www.me.umn.edu/docs>.

File	Description
<code>.Xdefaults</code>	
<code>.Xresources</code>	Where X11 Configuration is stored.
<code>.afpvols</code>	Where appletalk information is stored.
<code>.cshrc</code>	
<code>.login</code>	
<code>.logout</code>	Customization files for <code>tcsh</code>
<code>.forward</code>	Tells where you want email forwarded to
<code>.fvwmrc</code>	
<code>.fvwm2rc</code>	Configuration for the <code>fvwm</code> window manager
<code>.macbin</code>	Where appletalk files are stored
<code>.netscape</code>	Where Netscape keeps its cache and configuration
<code>.plan</code>	
<code>.project</code>	Information shown to other users when they <code>finger</code> you
<code>.xinitrc</code>	
<code>.xsession_linux</code>	
<code>.xsession_sun</code>	Startup files for X11

C Secure Passwords

C.1 Important Password Guidelines:

- Sharing accounts is explicitly forbidden. Never give your password to any of your co-workers. If they need access to your files or the MEnet systems, we can create a separate account for them.
- Never give your password to anyone. This includes System Staff. The system staff never needs to know your password to fix problems with your account. They have administration privileges already.
- Never write down your password.
- Remember to change your password periodically.
- Use different passwords for accounts on different systems.

C.2 Choosing a Good Password:

The only way to get a reasonable amount of variety in your passwords (I'm afraid) is to make them up. Work out some flexible method of your own which is NOT based upon:

- modifying any part of your name or name+initials
- modifying a dictionary word
- acronyms
- any systematic, well-adhered-to algorithm whatsoever

For instance, *never* use passwords like:

File	Description
alec7	it's based on the users name (and it's too short anyway)
tteffum	based on the users name again
gillian	girls name (in a dictionary)
naillig	ditto, backwards
PORSCHE911	it's in a dictionary
12345678	it's in a dictionary (and people can watch you type it easily)
qwertyui	ditto
abcxyz	ditto
0oooooooo	ditto
Computer	just because it's capitalised doesn't make it safe
wombat6	ditto for appending some random character
6wombat	ditto for prepending some random character
merde3	even for french words
mr.spock	it's in a sci-fi dictionary
zeolite	it's in a geological dictionary
ze0lite	corrupted version of a word in a geological dictionary
ze0l1te	ditto
Z30L1T3	ditto

I hope that these examples emphasise that *any* password derived from *any* dictionary word (or personal information), modified in *any* way, constitutes a potentially guessable password.