Unix Primer - Basic Commands In the Unix Shell

If you have no experience with the Unix command shell, it will be best to work through this primer. The last section summarizes the basic file manipulation commands.

1. Unix Shell

The *shell* is a command programming language that provides an interface to the UNIX operating system. The remainder of this tutorial presents basic commands to use within the UNIX shell.

1. Directories

The shell should start you in your home directory. This is your individual space on the UNIX system for your files. You can find out the name of your current working directory by typing:

```
% pwd
/Users/username
```

(The '%' designates your command line prompt, and you should type the letters 'p', 'w', 'd', and then "enter" - always conclude each command by pressing the "enter" key. The response that follows on the next line will be the name of your home directory, where the name following the last slash should be your *username*.) The directory structure can be conceptualized as an inverted tree.

No matter where in the directory structure you are, you can always get back to your home directory by typing:

```
% cd
```

(without specifying a directory name).

From your home directory, create a new subdirectory named "primer" for working through this tutorial:

```
% mkdir primer
```

You can remove an empty subdirectory with the following command (but don't do this right now):

```
% rmdir primer
```

(Note: if you do remove "primer", please create it again.)

Now change to the "primer" subdirectory, making it your current working directory:

```
% cd primer
```

1. Files

Files live within directories. You can see a list of the files in your "primer" directory (which should be your current working directory) by typing:

```
% ls
```

Since you just created the directory, nothing will be listed because the directory is empty. Create your first file using the pico text editor:

```
% pico first
```

The pico editor fills the entire console window. You can type text and move the cursor around with the arrow keys; the bottom of the screen presents the commands available. Type the sentence: "My first file." Then press "^O" (hold the left "control" key while pressing 'O') and "enter" to save the file, then "^X" (hold the left "control" key while pressing 'X') to exit pico. Now when you list your files, you will see file "first" listed:

```
% ls
first
```

You can view a text file with the following command:

```
% cat first
My first file.
```

("cat" is short for concatenate - you can use this to display multiple files together on the screen.) If you have a file that is longer than your 24-line console window, use instead "more" to list one page at a time or "less" to scroll the file down and up with the arrow keys. Don't use these programs to try to display binary (non-text) files on your console - the attempt to print the non-printable control characters might alter your console settings and render the console unusable.

Copy file "first" using the following command:

```
% cp first 2nd
```

By doing this you have created a new file named "2nd" which is a duplicate of file "first". The file listing reveals:

```
% ls
2nd     first
```

Now rename the file "2nd" to "second":

```
% mv 2nd second
```

Listing the files still shows two files because you haven't created a new file, just changed an existing file's name:

```
% ls
first   second
```

If you "cat" the second file, you'll see the same sentence as in your first file:

```
% cat second
My first file.
```

"mv" will allow you to move files, not just rename them. Perform the following commands:

```
% mkdir sub
% mv second sub
% ls sub
```

```
second
% ls
first   sub
```

This creates a new subdirectory named "sub", moves "second" into "sub", then lists the contents of both directories. You can list even more information about files by using the "-l" option with "ls":

```
% ls -l
-rw-r--r--   1 username     group          15 May 22 16:26 first
drwxr-xr-x   2 username     group         512 May 22 17:11 sub
```

(where "*username*" will be your username and "group" will be your group name). Among other things, this lists the creation date and time, file access permissions, and file size in bytes. The letter 'd' (the first character on the line) indicates the directory names.

Next perform the following commands:

```
% cd sub
% pwd
/Users/username/primer/sub
% ls -l
-rw-r--r--   1 username     group          15 May 22 16:55 second
% cd ..
% pwd
/Users/username/primer
```

This changes your current working directory to the "sub" subdirectory under "primer", lists the files there, then changes you back up a level. The ".." always refers to the parent directory of the given subdirectory.

Finally, clean up the duplicate files by removing the "second" file and the "sub" subdirectory:

```
% rm sub/second
% rmdir sub
% ls -l
-rw-r--r--   1 username     group          15 May 22 16:26 first
```

This shows that you can refer to a file in a different directory using the relative path name to the file (you can also use the absolute path name to the file - something like "/Users/*username*/primer/sub/second", depending on your home directory). You can also include the ".." within the path name (for instance, you could have referred to the file as "../primer/sub/second").

1. Other Useful Commands

The current date and time are printed with the following command:

```
% date
Thu May 22 17:39:04 CDT 2003
```

Remote login to another machine can be accomplished using the "ssh" command:

```
% ssh -l myname host
```

where "myname" will be your *username* on the remote system (possibly identical to your username on this system) and "host" is the name (or IP address) of the machine you are logging into. Note that "ssh" is secure unlike older programs such as "telnet".

Transfer files between machines using "scp". For example, to copy file "myfile" from the remote machine named "host", the command would be:

```
% scp myname@host:myfile .
```

(The "." refers to your current working directory, meaning that the destination for "myfile" is your current directory.)

If you are running X11, you should be able to start some applications from the command line, for example:

```
% matlab &
```

(The '&' character tells the console to run Matlab in the background - this way you immediately get a new prompt without first having to quit Matlab.)

In OS X, you can open file using the command "open". This command will open the file with the application corresponding to the file type. For example

```
 % open README.pdf
```

will open the file README.pdf using Preview. You can also open applications using the -a flag

```
 % open -a Safari
```

or open a file using TextEdit with the -e flag

```
 % open -e mytext.txt
```

1. Online Help

You can get online help from the "man" pages ("man" is short for "manual"). The information is terse, but generally comprehensive, so it provides a nice reminder if you have forgotten the syntax of a particular command or need to know the full list of options.

Use the "-k" option to provide a list of commands that pertain to a particular topic. For instance, try:

```
% man -k "copy files"
```

(and make sure to include the quotation marks). One of the commands listed should be the "cp" file copy command. Display the man page for "cp" by typing:

```
% man cp
```

1. Logging Out

It is very important to log out of your account whenever you are done using it, especially if you are on a public machine.

To close a shell window in a graphical environment, you can type:

```
% exit
```

Logging out from a graphical environment will require clicking on the appropriate icon. Logging out of a remote session can be done by either using the "exit" command or by typing:

```
% logout
```

1. Summary Of Basic Shell Commands

`% pico myfile`	*text edit file "myfile"*
`% ls`	*list files in current directory*
`% ls -l`	*long format listing*
`% cat myfile`	*view contents of text file "myfile"*
`% more myfile`	*paged viewing of text file "myfile"*
`% less myfile`	*scroll through text file "myfile"*
`% cp srcfile destfile`	*copy file "srcfile" to new file "destfile"*
`% mv oldname newname`	*rename (or move) file "oldname" to "newname"*
`% rm myfile`	*remove file "myfile"*
`% mkdir subdir`	*make new directory "subdir"*
`% cd subdir`	*change current working directory to "subdir"*
`% rmdir subdir`	*remove (empty) directory "subdir"*
`% pwd`	*display current working directory*
`% date`	*display current date and time of day*
`% ssh -l myname host`	*remote shell login of username "myname" to "host"*
`% scp myname@host:myfile .`	*remote copy of file "myfile" to current directory*
`% netscape &`	*start Netscape web browser (in background)*
`% man -k "topic"`	*search manual pages for "topic"*
`% man command`	*display man page for "command"*
`% exit`	*exit a terminal window*
`% logout`	*logout of a console session*