UNIX Basics

http://www.cgi101.com/help/unixhelp.html
http://www.isu.edu/departments/comcom/unix/workshop/unixindex.html

Not familiar with UNIX? Never fear; here's a handy guide to all you need to know to get around in the UNIX shell.

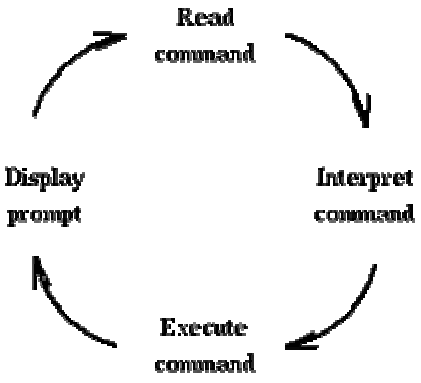
The UNIX Environment

To ensure security and organization on a system with many users, UNIX machines employ a system of user accounts. The user accounting features of UNIX provide a basis for analysis and control of system resources, preventing any user from taking up more than his or her share, and preventing unauthorized people from accessing the system. Every user of a UNIX system must get permission by some access control mechanism. (You will receive an account to access our UNIX network at the workshop.)

Once you've logged into the UNIX host, you'll be in the shell. What you first see on your screen may look something like this:

```
Last login: Thu May 5 18:07:13 EDT 2005 on /dev/pts/0 from
malouf224.malouf.wcslc.net
educ001@control(educ001)%
```

In this example, the % is called the "prompt". When you type, your typing will appear to the right of the prompt, and when you hit return, the shell will attempt to run the command you typed, and then display another prompt. The shell is perhaps the most important program on the UNIX system, from the end-user's standpoint. The shell is your interface with the UNIX system, the middleman between you and the kernel. The shell is a type of program called an *interpreter*. An interpreter operates in a simple loop: It accepts a command, interprets the command, executes the command, and then waits for another command. The shell displays a "prompt," to notify you that it is ready to accept your command.



Interpreter loop

Command Notation

The shell recognizes a limited set of commands, and you must give commands to the shell in a way that it understands: Each shell command consists of a command name, followed by command options (if any are desired) and command arguments (if any are desired). The command name, options, and arguments, are separated by blank space. Unlike DOS, the UNIX shell is case-sensitive, meaning that an uppercase letter is not equivalent to the same lower case letter (i.e., "A" is not equal to "a"). Most all UNIX commands are lower case.

The basic form of a UNIX command is: `commandname [-options] [arguments]`

The command name is the name of the program you want the shell to execute. The command options, usually indicated by a dash, allow you to alter the behavior of the command. The arguments are the names of files, directories, or programs that the command needs to access. The square brackets signify optional parts of the command that may be omitted.

Note: All of the commands shown below are the actual commands you can type at the UNIX prompt.

For example, the following command identifies the users currently logged in.

```
% who
```

When the following command flag option is added:

```
% who -q
```

a quick listing of users and the number of users on the local system is printed. All commands have online documentation, called man (manual) pages. Anytime you would like to find out more information about a command simply type: `man [commandname]`

```
% man who
```

Pressing the <enter> key will continue to the next line and the space bar will continue to the next page. If you want to exit the man page and return to the UNIX command prompt, simply type ‘q’.

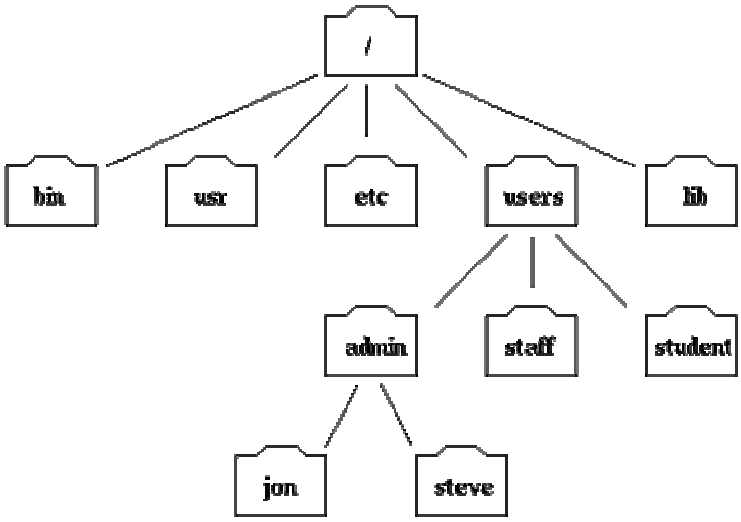
Aborting a Shell Command

Most UNIX systems will allow you to abort the current command by typing Control-C. To issue a Control-C abort, hold the control key down, and press the "c" key.

The UNIX Filesystem Structure

All the stored information on a UNIX computer is kept in a *filesystem*. Any time you interact with the UNIX shell, the shell considers you to be located somewhere within a filesystem. Although it may seem strange to be "located" somewhere in a computer's filesystem, the concept is not so different from real life. After all, you can't just *be*, you have to be *somewhere*. The place in the filesystem tree where you are located is called the *current working directory*.

The UNIX filesystem is heirarchical (resembling a tree structure). The tree is anchored at a place called the root, designated by a slash "/". Every item in the UNIX filesystem tree is either a file, or a directory. A directory is like a file folder. A directory can contain files, and other directories. A directory contained within another is called the *child* of the other. A directory in the filesystem tree may have many children, but it can only have one parent. A file can hold information, but cannot contain other files, or directories.



Part of the filesystem tree

To describe a specific location in the filesystem heirarchy, you must specify a "path." The path to a location can be defined as an *absolute path* from the root anchor point, or as a *relative path*, starting from the current location. When specifying a path, you simply trace a route through the filesystem tree, listing the sequence of directories you pass through as you go from one point to another. Each directory listed in the sequence is separated by a slash. Note: UNIX provides the shorthand notation of ".." to refer to the parent directory. For example:

- The absolute path to the directory "jon" is: `/users/admin/jon`
- The relative path from the directory "users" to the directory "jon" is: `admin/jon`
- The relative path from the directory "student" to the directory "jon" is: `../admin/jon`

Directory Commands

In UNIX, your location in the filesystem heirarchy is known as your "current working directory." When you log in, you are automatically placed in your "home directory." To see where you are, type the command: `pwd` (which stands for print working directory)

```
% pwd
```

To list the contents of the directory: `ls`

```
% ls
```

The previous command simply gives the file or directory names. To find out more information about each file or directory, use the –l option. This option displays file permissions, ownership, size, and modification date.

```
% ls –l
```

To make a directory: `mkdir [directoryname]`

```
% mkdir subdir1
```

If a full path is not given, the directory is created as a subdirectory of your current working directory. You must have write permissions on the current directory to create a new directory. If you list (`% ls`) the contents of your directory, you will now see this subdirectory listed.

To change your directory: `cd [directoryname]`. Remember, you must specify either the absolute path or relative path for the directory.

To change to a subdirectory in your current directory, you can simply type the relative pathname of that subdirectory.

```
% cd subdir1
```

(You can type `pwd` at the prompt to confirm that you're actually there.) To move to the parent directory of your current directory

```
% cd ..
```

Otherwise, you can always specify the absolute path name of the directory

```
 % cd /cgate/common/education/bin
```

At any time you can get to your default home directory by simply typing `cd`. Note: you can also refer to your home directory as `~yourusername`. The following three commands are equivalent:

```
% cd
% cd ~yourusername
% cd /cgate/common/education/yourusername
```

To remove a directory, use `rmdir [directoryname]`

```
% rmdir subdir1
```

File Commands

UNIX filenames can't have spaces, slashes, or weird characters in them. Also, file names are case sensitive. File names are specified similar to directory references. You can refer to a file in your current directory simply by typing the file name. Otherwise files are fully specified when you list each directory branch needed to get to them.

To copy a file to a different directory: `cp [filename] [directoryname]`

```
% cp /cgate/common/education/bin/hostname.cmd ~yourusername
```

Note: `hostname.cmd` is a file located in the directory `/cgate/common/education/bin` and you made a copy of this file in your home directory (which is designated by the shorthand notation ~yourusername rather than the absolute path). If you list the contents of your directory, you will now see this file listed. In the above example, the absolute path name of the file is used. The following command is equivalent to the above command using the relative path name of the file.

```
% cp ../bin/hostname.cmd ~yourusername
```

If you were actually in the subdirectory, `/cgate/common/education/bin,` which contained the file you desired to copy, all you would have to type is `cp hostname.cmd ~yourusername`

You can also copy a file to a different file name: `cp [filename] [newfilename]`

```
% cp hostname.cmd copyhost.cmd
```

You now have two copies of the same file with two different names in the same directory. To simply rename a file without keeping the original file: `mv [filename] [newfilename]`

```
% mv copyhost.cmd newhost.cmd
```

To remove a file: `rm [filename]`

```
% rm newhost.cmd
```

Unlike windows, you can NOT back out of a remove. Be certain that you want to remove a file.

File and directory permissions

UNIX supports access control. Every file and directory has associated with it ownership, and access permissions. Furthermore, one is able to specify those to whom the permissions apply. Permissions are defined as read, write, and execute. The read, write, and execute permissions are referred to as r, w, and x, respectively. Those to whom the permissions apply are the user who owns the file, those who are in the same group as the owner, and all others.

Read permission
: For a file, having read permission allows you to view the contents of the file. For a directory, having read permission allows you to list the directory's contents.

Write permission
: For a file, write permission allows you to modify the contents of the file. For a directory, write permission allows you to alter the contents of the directory, i.e., to add or delete files.

Execute permission
: For a file, execute permission allows you to run the file, if it is an executable program, or script. Note that file execute permission is irrelevant for nonexecutable files. For a directory, execute permission allows you to `cd` to the directory, and make it your current working directory.

To see the permissions on a file, use the `ls` command, with the `-l` option.

```
% ls -l /etc/passwd
```

The output should look similar to this:

```
% -rw-r--r-- 1 root sys 41002 Apr 17 12:05 /etc/passwd
```

The first 10 characters describe the access permissions. The first dash indicates the type of file (d for directory, s for special file, - for a regular file). The next three characters ("rw-") describe the permissions of the owner of the file: read and write, but no execute. The next three characters ("r--") describe the permissions for those in the same group as the owner: read, no write, no execute. The next three characters describe the permissions for all others: read, no write, no execute. The words after the permission characters designate the user/owner (root) and group (sys).

Viewing the contents of a file

Text files are intended for direct viewing, and other files are intended for computer interpretation.

To view a file (read-only): `more [filename]`

```
% more hostname.cmd
```

Pressing the <enter> key will continue to the next line and the space bar will continue to the next page. If you want to exit the file and return to the UNIX command prompt, simply type ‘q’.

To displays lines from the beginning of a file: `head [filename]`

```
% head /etc/rc
```

If no options are given, the default is 10 lines. An optional argument of —lines can be used to specify the number of lines to display. For example, to see the first fifteen lines of the /etc/rc file.

```
% head -15 /etc/rc
```

The tail command works like head, except that it shows the last lines of file: `tail [filename]`

```
% tail /etc/rc
```

Because we did not specify the number of lines as an option, the tail command defaulted to ten lines.

Logging out

When you're ready to quit, type the command

```
% exit
```

Basic Command Summary Sheet

Display Current Directory	`pwd`
List Contents of Directory	`ls`
Change Directories	`cd [directoryname]`
Make Directories	`mkdir [directoryname]`
Remove Directories	`rmdir [directoryname]`
Copy File to a Directory	`cp [filename][directoryname]`
Copy File to a New Name	`cp [filename][newfilename]`
Rename a file	`mv [filename][newfilename]`
Remove Files	`rm [filename]`
View File	`more [filename]`
View Beginning of File	`head [filename]`
View End of File	`tail [filename]`
Create/Edit File	`vi [filename]`
Close terminal session	`exit`

Inserting text:

`i`	insert text starting where the cursor is
`a`	append text to the right of the cursor
`o`	begin inserting text at the line below

Deleting text:

`x`	delete character
`dw`	delete word
`D`	delete to end of line
`dd`	delete line

Changing text:

`r`	replace single character under cursor
`R`	replace sections of text by overwriting
`~`	change the case of the letter under the cursor

Moving:

`l`	move forward one character
`h`	move back one character
`j`	move down one line
`k`	move up one line
`w`	move forward one word
`b`	move back one word
`$`	move to end of line
`0`	move to beginning of current line
`G`	move to last line in file

Saving files and quitting vi:

`ZZ`	write changes and quit vi
`:w`	write changes and continue editing
`:wq`	write changes and quit vi
`:q`	quit vi, there haven't been any changes
`:q!`	quit vi and disgard changes