Common Commands and Options

ls - list files in the current directory, or the files in another directory if a directory is given
common options
`-l` long or more complete file information `-a` list all files (even hidden)
example
```
ls -al /var/log/squid/access.log
```

cd - change the current working directory to the directory given
example
```
cd /etc/squid
```

cp - copy one or more files to another location
common options
`-a` maintain permissions and recursively descend into directories copying all files `-R` copy recursively including directories
examples
```
cp -R /home/joe /mnt/nfs/backup
```

mkdir - make a new directory
rmdir - remove an empty directory
common options
`-p` make or remove parent directories as needed, if possible
example
```
mkdir -p /home/cpxuser/src/squid-2.4-test
```

rm - remove one or more files
common options
`-R` recursively remove files from directories
example
```
rm -R /temp/work-dir/*
```

Network Configuration

Unix in general and Red Hat Linux in particular provide a number of simple tools for administering and maintaining the network functions in a server system.

If configuring network devices the important commands are **ifconfig**, **netconfig**, **route** and **ping**. To display the current network configuration, use **ifconfig**:

```
$ /sbin/ifconfig
```

To configure the Red Hat networking settings, use **netconfig** while logged in as root:

```
# /usr/sbin/netconfig
```

This will start the menu-driven interface for configuring the primary network interface on your system.

To test connectivity between hosts on a network, use **ping**:

```
$ ping 192.168.1.172
```

To temporarily alter a network interface address, you can also use **ifconfig**:

```
# /sbin/ifconfig eth0 192.168.1.175
```

Console Basics

Logging In
When you first sit down to a Linux machine, you'll see a login prompt, asking for your username first, and then your password. Note that Unix, unlike some other operating systems, is case sensitive in most regards including authentication. So enter your username and password exactly as the user was configured, including capital and lowercase letters.

Logging into a Linux machine can be done remotely, using either **ssh** or **telnet**. **ssh** is recommended, as it is a secure protocol. To log in via **ssh** simply type the name of your **ssh** client software, usually followed by the username you wish to login as and the remote address you'd like to log into. For example, using the *OpenSSH* **ssh** implementation, you would type the following command:

```
$ ssh -l root swell.mydomain.com
```

After entering your password, you will be authenticated as the root user and able to perform operations as though you were logged in on the console.

Privileged User or Superuser or root

Unix is a multi-user operating system, meaning that any number of people can have accounts on the system and any number of users can be logged in at the same time. To maintain sanity in an environment where many people share the same machine, or where the machine is exposed to the internet, the system needs some form of access control to keep users from interfering with other users or damaging the system. In Unix, this is generally referred to as *permissions*. When performing system administration tasks that alter the behavior of the system, you will usually need to be logged into the system as the **root** user.

When you are logged in as a normal, or unprivileged user, your shell command prompt will likely be a US dollar sign, or `$`. If logged in as the `root` user your prompt will be a hash mark, or `#`. Use the **su**

Getting More Information
Unix systems, like Linux, are usually self documenting if you know how to ask for help. The commands to know are man and info. The manpages and info documents while rarely easy to understand, can often provide just enough information to help you find deeper answers in a search of the internet or in a complete system reference.

To read a manpage or an info page on a command, simply type man or info followed by the command you'd like more information about. A good place to start is the documentation for each of the documentation readers.

```
$ man man
$ info info
```

Another nifty trick is the apropos option of either of these commands. Using the apropos option allows you to enter a word related to your problem, without having to know the actual command name you need to read about. The man apropos option is enabled with the -k option, while for info it is --apropos.

```
$ man -k network
$ info --apropos make
```

Note that while many manpages have been marked as deprecated in favor of their info counterparts, many people find info a difficult program to use and the manpages more comprehensible and easy to use. The manpages are also more complete, covering more software.

Directory Structure

For many new users, simply *finding* things is a difficult task. If you don't already know that configuration files usually live in /etc, it probably isn't the first place you'll look for that type of file. The Linux filesystem structure is consistent, and makes sense once you understand it, but it is not an intuitively obvious layout.

`bin` - System executable tools, including **cp**, **ls**, **mount**, **more**, and others. The bare minimum set of commands needed to bring up a functional (though not very featureful) system.

`boot` - The kernel boot images reside here. When a new kernel is installed it will be installed here.

`dev` - System devices, both physical and virtual. This is a pseudo-filesystem that provides a file interface to the components of the system. Necessary in a Unix system because in Unix, everything is viewed as a file.

`etc` - Configuration files reside here. Nearly every configurable service on your system has a configuration file or configuration directory located in this directory.

`home` - Home directory for users. Each user will have a home directory with the same name as their username located in this directory.

`lib` - System libraries live in this directory. A system library is an archive of program routines that provide user programs a standard method to access the various functions of the operating system.

`mnt` - Filesystems that are generally mounted temporarily usually go here. This would include floppy drives, CDROM drives, and other removable media. Some sites place NFS mounts in this directory.

`opt` - Locally installed components. This directory is currently rarely used on Linux systems.

`proc` - The Linux kernel creates this filesystem on boot and fills it with a wide range of files that provide access, either for configuration or for viewing the contents of, many of the system limits, settings, running processes, and a lot more.

`root` - The home directory for the `root` user.

`sbin` - System bin, or the location of most of the system configuration and administration binary executable files.

`tmp` - Temporary files, usually created by programs running on your system, reside here. These files are periodically cleaned from the system, and thus tmp should not be used for storage of anything you don't want to be deleted.

`usr` - Within this directory is a second tier of directories. Generally, user programs reside in one of the subdirectories of

Current Working Directory and Dots

The term current working directory refers to the directory you are in currently. You may find the working directory using the **pwd** command. Note the current directory is not in the path, thus simply typing the name of an executable in the working directory will not succeed in running it. To run a file in the working directory, you precede the filename with the path to the file. Unix provides a shortcut to represent the working directory: a dot. So to run install.sh:

```
$ ./install.sh
```

Two dots represents the directory above the working directory.

Globbing, Grepping and Finding Files

Linux provides some great tools for dealing with your files. The first is the built-in *globbing* that the shell provides, which allows you to enter just enough of a filename of directory to identify it, as well as locate all filenames that match some criteria, among other things. The **grep** command allows you to find some given piece of data in a number of files. Finally, the **find** command will help locate a lost file.

A glob in its most basic form is an ***** (often called "star" or "splat" when spoken) which matches any number of any character. For example, if I wanted to list all of the files in my directory with the letters "txt" in them, I could enter:

```
$ ls *txt*
```

Or to list the contents of every directory that contains a specific date in the name:

```
$ ls /home/joe/*07-31*/*
```

Other globs are possible too, the **?** matches one occurrence of any character.

Next, if we need to find an occurrence of some specific string of text in one of 5,000 files, we can use **grep**. The command is followed by the string to match against and a filename, a list of filenames, or glob of files to search for the text string. If it finds the string it will print a line identifying the file and display the line in which the string appears. To find the file containing "blue shoes":

```
$ grep 'blue shoes' /home/joe/*.txt
```

Finally, to find a file that we know the name of but can't locate amongst our thousands of files, we use **find**. It takes at least two arguments: the directory in which to begin the search, and the name of the file to find:

```
$ find . -name my-long-lost-file.txt
```

To look for the file in the current directory and each of its subdirectories.

Software Installation

Installation of software under Red Hat Linux is achieved using **rpm** or the Red Hat Package Manager. This package manager maintains a database of all installed packages, their versions, and their dependencies. If all software is installed via packages, your system will be easier to maintain and keep up to date. It must be run by the root user.

The following options are most commonly used:

`-i` install -U upgrade `-F` freshen `-v` verbose `-h` display a progress meter `-e` uninstall or erase

The `-U` option will install the package if a version is not already installed, while the `-F` option only performs upgrades of packages that are already present on the system.

For example, to upgrade Squid to the latest version:

```
# rpm -Uvh squid-2.4STABLE1-i386.rpm
```

To upgrade a number of packages from a directory, without installing any new packages:

```
# rpm -Fvh /home/joe/RPMS/updates/*.rpm
```