Introduction

This document helps new or novice users of the UNIX or Linux operating systems do some common tasks and be introduced to some basic concepts about UNIX. For the most part, what is applicable to UNIX is applicable to Linux.

Some UNIX History

UNIX is an operating system that was first designed at AT&T's Bell Laboratories in 1969. While that makes UNIX somewhat ancient by common computer practices, its anomalous ability to maintain a presence through the years is partially attributed to its power, flexibility, and portability.

UNIX will run on platforms ranging from PCs and Macs, to mainframes and large supercomputers. To date, there have been many versions and releases of the UNIX operating system. Some of the most influential offshoots of the original UNIX release were the commercial versions developed by AT&T as releases of System V (pronounced "system five") and the licensed version to the University of California at Berkeley, called BSD (for Berkeley Software Distribution). Descendants of System V include Hewlett Packard's HP-UX, IBM's AIX, and Silicon Graphic's (SGI) Irix. Not falling into these categories is Linux, an open source operating system which, along with the many GNU open source tools and compilers, runs on a wide array of hardware architectures.

How Things Work

The operating system relies on the kernel which controls access to the computer, its files, allocating resources to the various activities taking place within the machine, maintains the file systems, and basically manages all things related to the operating system.

Users rarely interact directly with the kernel itself; instead the interface used is the shell. The shell interprets the user's commands and programs for the kernel, providing a manageable interface. There are several shells available, each one slightly different, but all sharing similar characteristics. Therefore, when "the shell" is mentioned in this document without further qualification, it applies to most, if not all, of the commonly used shells.

The most commonly used shells include the members of the Bourne and C shell families. Often, more than one shell is available to users on UNIX systems. These shells, while having individual names, are often called by their abbreviations (which also happen to be how they are invoked). The original shell, the Bourne shell (sh), is found on every system. It is not often used as a user's default shell these days. In the early 1970′s the C shell (csh) was developed providing more functionality than the Bourne shell and offering a command language

much like the C programming language. Other shells came later and many of these are in command usage today. Examples include the Korn shell (ksh) , a Bourne descendant, that brought together sh and csh functionality, and continues to be developed today; the Bourne Again shell (bash), the free Bourne descendant that brings together csh and ksh functionality; TC csh (tcsh), an improvement on csh; the Z shell (zsh), which resembles ksh in many respects but has extra functionality, and many others. These are the most commonly used ones though.

Logging In

To get to an interactive command interpreter, that is, a shell, you need to log into a UNIX system somewhere. This can be done either from a hard-wired keyboard/monitor or remotely, using a protocol such as Secure Shell (ssh). You can connect from one machine to the other in this way as well. Once you are connected, you will be prompted for a unique login and a unique password (which is not printed on the screen). Once logged in, you should see some messages and then a prompt. Your prompt can be a number of things, but it is often just a single character that designates that the shell is ready to accept a command. Common prompt characters are “$” (often in the Bourne based shells), “%” (in the C shell family), “>” (csh), and “#” (sh). The prompt is customizable and sometimes will contain other information such as the shell name or the present working directory.

Interacting with the Shell

Common to all shells is the way one gives a command. UNIX commands are generally one-word and lower case. Unlike Windows, UNIX is case sensitive. Optional flags are switches that modify how the command works and are often set off with a dash “-”, are case sensitive, and can be combined. A (sometimes optional) argument usually specifies the file or directory that is the target of the command’s action.

On a UNIX system a command will follow the format:

```
command  -flags  arguments
```

These commands allow the user to interact with the filesystem. The filesystem consists of a set of files which fall into three categories:

- ordinary files containing data in some form or another
- special files which include, but are not limited to, executable programs
- directories which contain information about a set of files and are used to locate a file by name.

Ordinary files are common all over the filesystem. The special files tend to be executables and include many of the programs you run. These are typically in certain directories which your environment is set up to see and

use: they include the commands used to manipulate files, access other systems, interact with the operating system, and many other things as well.

UNIX organizes the file system as a hierarchy of directories (often called a tree), although almost always pictured as an upside-down one. When you log in, you have accessed a certain directory, your home directory, in which there are files and other directories (or subdirectories).

Managing Files and Directories (Common UNIX Commands)

File and directory names can be up to 256 characters long. They may contain upper and lower case characters (including letters, numbers, punctuation, and other symbols).

> Try to avoid using characters like “/”, “>”, “!”, “?” or blank spaces in filenames, as they are sometimes hard to manipulate because of their special designations within the shell.

In UNIX, these are case sensitive. For instance, the name “Myxztplk.txt” refers to a file different file from “myxztplk.txt.” The file hierarchy starts from the root (or “/”) and makes its way down to the particular file name on the path. The full name for any file is the complete path to it, starting at the root, which means all the directories to it. The path name for the file “myxztplk.txt” in the directory “villains” in the directory “superman” which is under the root directory would be: /superman/villains/myxztplk.txt

Use the pwd command to list the name of your **p**rint **w**orking **d**irectory. You may refer to any files in your present directory without using their full path names. Often, you will not have a UNIX account that begins at the root directory. Most UNIX platforms have many users, and each user has a small slice of the available hard drive space. Thus, instead of starting out at the root of the drive, most user accounts will be somewhere down in the directory tree.

Some common commands to help you interact with the filesystem and your files are:

ls (list): List directory contents

ls	lists the contents in a directory in columns
ls -l	gives a longer/fuller listing including file permissions, size, date created (the “l” is for long)
ls -la	similar to the above but includes “dot”/hidden files (the “a” is for all)

cd (change directory): Move to a different directory

cd	returns you to your home directory
cd ../	moves up one directory level
cd ../..	moves up two directory levels
cd mysubdirectory	moves to the subdirectory mysubdirectory

cp (copy): Copy files and directories

cp file1.txt file2.txt	copies file1.txt to file2.txt
cp file1.txt mydirectory/	copies file1.txt to named mydirectory
cp file1 file2.txt mydirectory/	copies file1.txt and file2.txt to named mydirectory

mv (move): Move or rename files or directories

mv file1.txt newname.txt	renames file1.txt to newname.txt
mv mydirectory newname	renames directory to newname
mv file1.txt directory/	moves file1.txt to named directory
mv file1.txt directory/newname.txt	moves file1.txt to named directory and renames it to newname.txt

rm (remove): Delete files or directories

rm file1.txt	removes (deletes) file1.txt
rm -i file1.txt	deletes named file1.txt after prompting to make sure you wish to remove it (the “i” is for interactive)

BE EXTREMELY CAREFUL with rm. There is no undelete command!

mkdir (make directory): Create directories

mkdir mysubdirectory	creates a directory within the current directory called mysubdirectory

rmdir (remove directory): Delete directories

rmdir mysubdirectory	removes (deletes) directory mysubdirectory within the current directory (if mysubdirectory is empty i.e. has no files in it)

chmod (change mode): Change file or directory access permissions

chmod u+w file1.txt	adds write permission for the user to file1.txt ("write" is the ability to edit or delete the file)
chmod g+x file1.txt	adds execute permission for the group to file1.txt
chmod o-r file1.txt	removes read permission for the world (other) from file1.txt

The chmod command assists in allowing you to share files with another user or group or to restrict access ("lock down") directories of files to keep things private. The document How to Use UNIX and Linux File Permissions explains this in detail.

> Since UNC users' file space is sometimes in AFS (your home directory or any directory that starts with "/afs/") , AFS ACLs (Access Control Lists) control directory access permissions as well as basic UNIX file permissions.

more: List/view files one screen at a time

more file1.txt	scrolls forward through file1.txt one screen at a time. Press spacebar to go forward one screen, to go back one screen, or to quit

pwd (print working directory): Display path name of current directory

pwd	displays your current directory

cat: Concatenate files and list them to the screen.

cat file1.txt	prints the contents of file1.txt to your screen
cat file1.txt file2.txt	prints the contents of file1.txt and file2.txt to your screen
cat file1.txt file2.txt > file3.txt	concatenates the contents of file1.txt and file2.txt into file3.txt

diff: Compare two files and shows where they differ

diff file1.txt file2.txt	lists the differences between the files file1.txt and file2.txt

wc: Count characters, words or lines

wc -c file1.txt	prints the number of bytes in file1.txt
wc -l file1.txt	prints how many lines are in file1.txt

who: List users logged in the machine

who	lists users who are logged in the machine
who -m	shows you what the ip address of the machine you logged in from

ps: Find out what jobs you are running and what shell you are in

ps	lists your current running jobs and your shell you are in
ps -ef	lists jobs of others as well as yours

As can be seen, all of these commands have optional flags and many take arguments, all of which can be found in the man pages.

Getting Help (The UNIX man pages)

The command for the manual pages is possibly one of the most important commands you can know. It is:

```
man  somecommand
```

All standard commands (that come with the Operating System) have a man(ual) page. On it there should be information on using the command, what options there are, and what the options do.

To search the man pages for a keyword use the -k option:

```
man  -k  somekeyword
```

Creating and Editing Files

To create a new file for, say, a web page or text for a document, you will need to invoke an editor. Some UNIX distributions only carry a certain editor. Depending on who set up the system, one may or may not be present. Common UNIX editors include pico, vi, and emacs. While there are others (ne, ed, etc.) these are common to many UNIX setups. To open or edit a file, you need to invoke the editor and give it a file name for an argument. If the file doesn't exist, it creates a new file. For instance,

```
pico  newfile.txt
```

will open the file " newfile.txt" for writing using the pico editor. Different editors have different commands and syntax to do essentially the same commands (write the file, open the file, write but don't close the file) and are too numerous to list in this document. Many references are available online and in print.

Logging out

Always log out of your account when you are finished. Do not leave your login session available for others to use on public workstations or lab machines.

Different UNIX machines use different words to log out of an account. Among the most popular are exit, logout, bye, or Control-d.