Linux Seminar

Department of Computer Science

A Brief History

In the late 1960s an operating system began development by Ken Thompson and Dennis Ritchie in Bell Laboratories at AT&T. Originally written in B and then changed to C for its portability, the release date of the first Unix operating system was November 3, 1971. In 1977 Berkley University decided to fork the Unix operating system and create BSD. The BSD project will eventually be forked by Steve Jobs into NeXT which is the core of OSX. Other Unix based systems come and go (System V and Solaris for example, as well as Minix for academia) but in 1991 Finnish graduate student Linus Torvalds started a project which later became the Linux kernel. What made this revolutionary was that this was the first free operating system. Below is the announcement Linus made on August 25, 1991 (age 21) in a Usenet posting to the newsgroup" comp.os.minix".

> Hello everybody out there using minix -
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since April and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the filesystem (due to practical reasons) among other things).
>
> I've currently ported bash (1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. Yes. it is free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 tasks switching etc.), and it probably never will support anything other than AT-hard disks, as that's all I have :-(.
>
> -Linus Torvalds

Eventually this operating system was named Linux and is widely adopted and deployed on millions of devices. Linux is one of the driving factors behind the open-source movement, seeing as how all of the code for the entire operating system is publicly available and anyone can join in and start writing code to help further the operating system and its cause.

Why Linux?

There are many reasons to delve into the Linux world. Aside from being a requirement for your class, knowledge in Linux and Unix based operating systems will diversify you in the job market and give you more options for programming solutions. Below are just a few of the many reasons to become well versed in Linux.

- Most of the time it's completely FREE
- Native support for many programming languages
- Complete customization freedom

- Easier to develop code
- Very secure
- Everything can be done through the (C)ommand (L)ine (I)nterface

Student Linux Account

Each student in the department is issued a Linux account if they have been enrolled in at least one Computer Science course since attending the university. This account may be used for school and personal projects responsibly. Server administrators reserve the right to disable any account for blatant misuse. This account uses the same username and password as your Texas State University netID. In addition, this account is given only 1GB of storage. If you require more storage feel free to email infra@cs.txstate.edu to request more.

Linux in our Department

Our department offers many Linux resources for students to use on and off campus.

We have two Linux servers that all students who have been enrolled in a Computer Science course at some point in their collegiate career will be able to log into. Students have access to the following servers. Your Linux directory will be the same on each server, no matter which one you log on to.

- zeus.cs.txstate.edu
- eros.cs.txstate.edu

You may also boot into an actual Linux Desktop environment on any computer in the department. To do this simply reboot the machine. You will be presented with a menu with multiple options on boot. There are currently two existing NetBoot options for Linux in the department, Ubuntu 12.04 and Ubuntu 14.01, that are available for use. These servers will mount your Linux directory, meaning you will have access to your files in the graphical environment.

Caution

When using your Linux account in the Linux Desktop Environment/NetBoot it is easy to fill up your quota with web browser caching and application data. If you fill up this quota you will be unable to log on to NetBoot images anymore. To fix this you need to either delete files from the Windows GUI through the mounted drive or log onto the CLI and delete files from there.

Connecting from Windows

In order to connect to Linux from Windows you must use certain programs that emulate a terminal interface (connecting this way is known as connecting via SSH). Two programs are installed on the Windows images for you to use (Putty and Secure Shell Client). Simply open these programs and supply the following information.

Secure Shell Client

- Click Quick Connect Button
- Hostname: one of the Linux servers
- Username: Linux account username
- Port: 22
- Hit ENTER
- Password: Password for Linux account (once prompted)

Putty

- Hostname: one of the Linux servers
- Hit ENTER
- Username: Linux account username
- Password: Password for Linux account (once prompted)

Transferring Files from Windows

Within the CS Department

All departmental Windows clients will automatically map your Linux account as a network drive. To access this drive simply click Start–Computer–Linux Folder

To transfer files from your personal machine you will need to get a Secure File Transfer Protocol Client (SFTP). If you installed the package for the Secure Shell Client, then a similar application was installed named *SSH Secure File Transfer Client*. The logon procedure for this is identical to the *Secure Shell Client*. Once logged on you will see an application with a split window. The left side of the application represents files on the local machine, while the right side of the application represents files on the remote server. To transfer files simply drag and drop the appropriate file to the desired location.

Directory Layout

Just as in Windows or OSX, the directory structure of Linux file systems is like a tree. You have a directory with files and possibly other directories within it. The default location when you log in is known as your home directory. All your files and directories will be in this directory, with nesting down deeper as you get more directories and files.

Command Line Interface

Finally, we get to the main draw of using a Linux system, the CLI or Command Line Interface. Below is a list of commonly used Linux commands, their description and examples of their usage. **All examples will be valid Linux commands and in italics.**

ls

This command is used to show the contents of your current directory.
example: *ls* or ls -a
Lists all the files in the current directory
example: *ls program**
Lists all the files whose name starts with program in the current directory
example: *ls *.cpp*
Lists all the files whose name ends with .cpp in the current directory
example: *ls -l*
Lists all the files with more information of each file such as size, date created, etc in the current directory

pwd

This command is used to list the whole path of your current directory.

mkdir

This command is used to create a new directory. example:
mkdir foo1
Creates a directory named foo1 in the current directory.

cd

This command is used to change directories to a new directory.

example: *cd myDir*
Changes to a directory called myDir. It **must be in your current directory for this to work!**
example: *cd myDir/foo*

Changes to a directory called foo that is inside the directory myDir.

example: *cd ..*

Changes to the previous directory.
example: *cd ~*
Changes to my home directory.

Note: Directory names are case sensitive.

For example, if you name a directory as LaB1, then your command should be cd LaB1 (You can type cd L and press tab).

If you name you directory as Lab Programs then, your command should be cd ‘Lab Programs’ (You must use quotes if you have spaces in your directory name).

mv

This command is used to move files and directories to a new location. example:

mv test.cpp ../newDir
Moves test.cpp to the directory named newDir. Recall that the two dots from above. We went up a level and then into the directory named newDir.

cp

This command is used to copy files and directories to a new location.

example: *cp test.cpp ..*
Moves test.cpp to the previous directory.

rm

This command deletes files.
example: *rm test.cpp*
Deleted test.cpp

wc

This command counts the number of words in a file.
example: *wc -w essay.txt*
Counts the number of words in the file essay.txt
example: *wc -l essay.txt*
Counts the number of lines in the file essay.txt

diff

This command compares the contents of two files and displays the line numbers where there are differences. Suppose you wrote a program (name it p1.cpp) and later made some changes to it (name it p2.cpp). To see the line numbers where you made changes, type

example: *diff p1.cpp p2.cpp*
It will show something like 1,2c1,2 which means 1,2 before the letter c are the line numbers of the file a.cpp and 1,2 after the letter c are the line numbers of the file b.cpp. So, changes are made in these lines.

chmod

This command is used to change the permissions of a file.

Symbol	Meaning
u	User
g	Group
o	Other
a	All
r	Read
w	write (and delete)
x	execute (and access directory)
+	add permission
-	take away permission

example: *chmod a+rw notes.doc*
Gives read and write permission for the file "notes.doc" to everyone
example: chmod go-wx notes.doc
Removes write and execution permissions for the group and others i.e., they can only read the file.

The command "ls -l" can be used to check the file permission: Enter the command *ls -l notes**
In the file information, you should see -rwxr--r-- after executing the above two commands.

file

This command classifies the named files according to the type of data they contain, for example ascii (text), pictures, compressed data, etc...

example: *file **

find

This command searches through the directories for files and directories with a given name, date, size, or any other attribute you care to specify. It is a simple command but with many options - you can read the manual by typing man find.

example: *find . -name "*.txt" -print*
Searches for all files with the extension .txt, starting at the current directory (.) and working through all sub-directories, then prints the name of the file to the screen.

example: *find . -size +1M -ls*
Finds files over 1Mb in size and displays the result as a long listing.

history

The shell keeps an ordered list of all the commands that you have entered. Each command is given a number according to the order it was entered.

example: *history*
Displays the list of all the commands in the history

example: *!10*
Will execute the 10th command in the history.

kill

The kill command allows you to terminate a process from the command line. You do this by providing the process ID (PID) of the process to kill. Do not kill processes willy-nilly. You need to have a good reason to do so. In this example, we will pretend the shutter program has locked up.

We can search for the shutter process and obtain its PID as follows:
ps -e | grep shutter

Once we have determined the PID—1692 in this case—we can kill it as follows:
kill 1692

free

The free command gives you a summary of the memory usage with your computer. It does this for both the main Random-Access Memory (RAM) and swap memory. The -h (human) option is used to provide human-friendly numbers and units. Without this option, the figures are presented in bytes.

free -h

less

The `less` command allows you to view files without opening an editor. It is faster to use, and there is no chance of you inadvertently modifying the file. With `less` you can scroll forward and backward through the file using the Up and Down Arrow keys, the PgUp and PgDn keys and the Home and End keys. Press the Q key to `quit` from `less`.

To view a file, provide its name to `less` as follows:

```
less lab1.cpp
```

sudo

```
sudo command allows a permitted system user to run a command as root or another user, as
defined by the security policy such as sudoers.

$ sudo apt update

Note: You cannot run this on the CS Dept server since, you do not have the admin rights.
You can only run it on your own laptop.
```

passwd

The `passwd` command lets you change the password for a user. Just type `passwd` to change your own password.

You can also change the password of another user account, but you must use `sudo`. You will be asked to enter the new password twice.

```
sudo passwd username
```

alias

alias is a useful shell built-in command for creating aliases (shortcut) to a Linux command on a system. It is helpful for creating new/custom commands from existing Shell/Linux commands (including options):

```
$ alias cs1428='cd Fall2020/cs1428/labs'
```

The above command will create an alias called cs1428 for Fall2020/cs1428/labs directory, so whenever you type cs1428 in the terminal prompt, it will put you in the Fall2020/cs1428/labs directory.

at

at command is used to schedule tasks to run in a future time. It runs a task once at a given future time without editing any config files:

For example, to shut down the system at 23:55 today, run:

```
$ sudo echo "shutdown -h now" | at -m 23:55
```

Note: This command works only on your personal laptops.

clear

clear command lets you clear the terminal screen, simply type.

```
$ clear
```

WARNING! In CLI there is no recycle bin. Usually the system will ask you to make sure you want to delete this file (unless you have those turned off). Once you delete the file it is gone. Make sure you really want to delete that file before you use this command.

Text Editors

In CLI code is edited in text editors. This guide explains basic usage of two common text editors, Vim and Nano.

Nano

Nano is a very basic and simplistic text editor. While very easy to pick up and learn, it does not have some of the more advanced features of Vim that allow for quick development and editing. It is meant more for a very fast edit. Below is a table of basic nano commands.

nano test.cpp	opens a file named test.cpp. If the file does not exist it will open a blank file.
ctrl + o	saves the file
ctrl + x	exits the file
ctrl + v	jump down
ctrl + y	jump up

Vim

Vim is a very powerful and complex text editor. It has a bit of a learning curve but once learned can increase productivity immensely. It is the more modern version of the classic text editor *vi*, which on fresh Unix and some Linux based installs will most likely be the only text editor preinstalled.

Vim work in two modes, command mode and insert mode.
To enter insert mode from command mode, press the i key.
To enter the command mode from the insert mode, press the ESC key.

Insert Mode

When in insert mode you can use the arrow keys to navigate around and type freely like you would normally. On all of the Linux machines maintained by the department the settings for Vim are setup to be as natural to you as possible, however, on fresh installs you may have to set the configurations to allow you to use the arrow keys or backspace keys for instance. Do not worry about this if you are planning on just using department servers.

Command Mode

The magic of Vim comes from its command mode. Below is a table of basic Vim commands for beginners.

vim test.cpp	opens a file named test.cpp. If the file does not exist it will open a blank file.
:q	exits the program. Will not work if work is unsaved.
:w	saves the program.
:x	saves the program and exits.
/text	searches and highlights all instances of the word text in the file.
v and arrowkeys	select text
y	copy
yy	copy whole line
d	cut
dd	cut whole line
P	paste

C++ Compilation

To compile C++ programs, we use the *g++* compiler that is native to almost all Linux distributions. There are two stages to compilation, compiling and linking, but for the purpose of this tutorial we will force this to all happen with one command. The syntax to compile is as follows: *g++ -o test test.cpp*. This will compile the code and create an executable named test. If we do not use the -o option and just do *g++ test.cpp* the executable will be named a.out.

If there are any errors in the code, it will dump them to the console. If the code compiles correctly it will appear as if nothing happened.

To run the executable, simply type *./executableName (ex: ./a.out)* and it will run.

Exercise

Now, after all that information here are some exercises for you to practice your new Linux skills. If you are doing this in a seminar and have issues ask one of the instructors to help you, if you are in the lab you can always ask one of the Lab Staff for assistance as well.

- **First** Create two new directories named foo1 and foo2
- **Second** Change directory into the foo1 directory
- **Third** Write a basic Hello World program in C++ within this directory
- **Fourth** Compile this program, naming the executable to hi
- **Fifth** Run this program
- **Sixth** Copy this program to the foo2 directory
- **Seventh** Edit the Hello World program in foo2 to print something different than the original
- **Eighth** Compile the program without specifying an executable name
- **Ninth** Run this newly compiled program
- **Tenth** Delete the Hello World program in directory foo1

Final Thoughts

The *nix world is big and amazing. You can easily get lost for hours learning many things about the operating system and its commands. The best way to learn, not only Linux but Computer Science in general is by doing and using the Google. The likelihood of you being the first person on earth to ever have that question is astronomical, you just have to learn how to ask Google to give you what you want. If you do not have access to Google, you can always read the manual for each command. To use this, you simply type **man** *command* and it will display the documentation for the command (press q to quit).

Challenges

These questions require you to use Google or man pages to use the commands you already know in different ways.

1. Delete a directory
2. Delete an item without it asking you the safety question
3. Copy a directory into another directory
4. List the files in a directory according to date last modified.
5. Rename a program with the cp command.