An Introduction to Linux

CSE/IT 107L

NMT Department of Computer Science and Engineering

"...the Linux philosophy is 'laugh in the face of danger'. Oops. Wrong one. 'Do it yourself'. That's it."

— Linus Torvalds (inventor of Linux)

"Avoid the Gates of Hell. Use Linux."

— Unknown

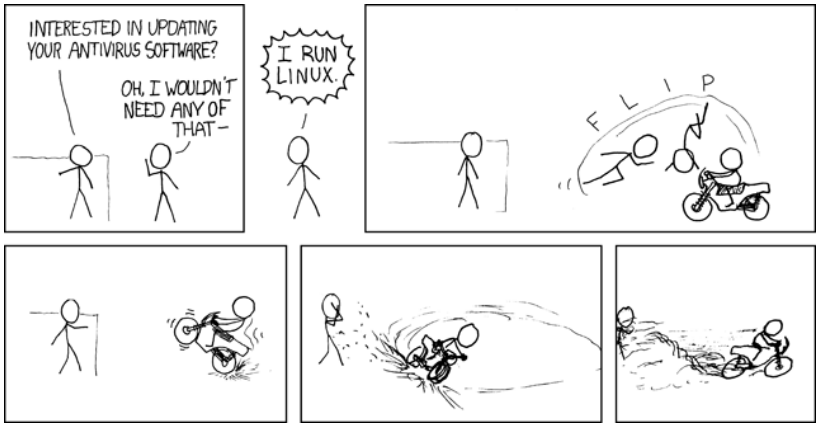


Figure 1: `http://xkcd.com/272`

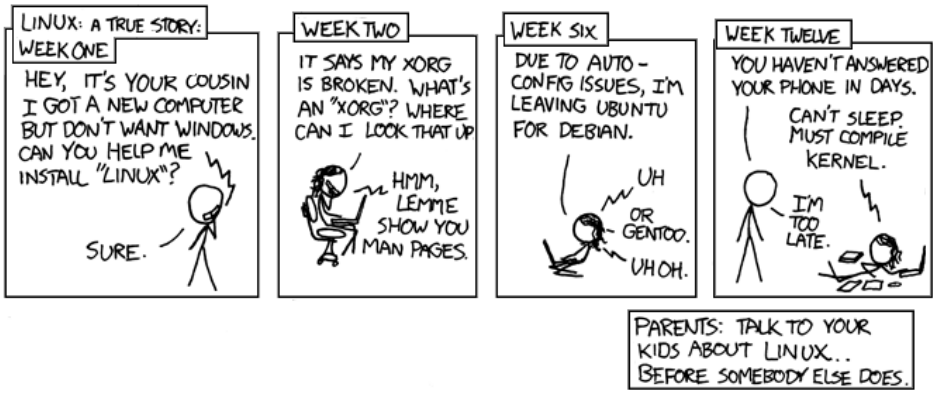


Figure 2: `http://xkcd.com/456`

Introduction

This lab will introduce you to the Linux operating system and its command line interface. You will learn how to log into, use, and navigate the operating system Linux, and how to create a compressed file called a tarball and submit it to Canvas. Many of the instructions are very detailed and specific. However, it is important that you understand the concepts behind the actions you take — you will be repeating them throughout the semester. If you have any questions, ask the instructor, teaching assistant, or lab assistant.

Contents

1 Log In

1.1 ITC Computer

If your lab computer is displaying a Windows login screen or you have logged into Windows, restart it and select Debian (a type of Linux) as your operating system. Please note that you are required to use `Linux` tools for all labs in this class. To log into an ITC machine, you must use your ITC username and password. If you do not have an ITC account, you must visit the ITC office (Gold building) to have one activated.

1.2 PuTTY

You may also use the Windows program PuTTY to login to your Linux system. Enter "rainbow.nmt.edu" as the hostname and click "Open." This should open a Linux terminal on your Windows desktop. If you are logged in to Windows on an ITC machine you may access your files by clicking on "Computer" in the Start menu and clicking on the `U:` drive.

Please be aware that we will use the Python modules Turtle and matplotlib in later labs, these require graphics which are not available through PuTTY.

2 Terminal

In Linux, you will perform most actions from the command line. In order to do this, you must first open a terminal. The following is just one of many methods for doing this.

1. Click on the `Menu` button at the bottom left of the screen.
2. In the text box that pops up as part of the menu, type in `Terminal` and click the application named "Terminal."

The terminal provides access to a command interpreter, also known as a shell. It is where you type text-based commands that are used to control the computer and run programs. On other operating systems you would use your mouse and the graphical user interface (GUI) to use the same features that these commands provide.

When you open the terminal prompt, you will probably see something like this:

```
[username@machinename ~]$ youcantypehere
```

This is where you type commands. We will be shortening this prompt to just a single $ to denote a terminal prompt for the rest of this semester's labs.

2.1 Example Terminal Sessions in the Labs

Throughout this semester's labs, we will be using specially formatted text to aid in your understanding. For example,

```
$ text in a console font within a gray box
```

is text that either appears in the terminal or that you should type into the terminal (almost) verbatim. Even the case is important, so be sure to keep it the same! However, a $ is used that to indicate that the code is from a terminal prompt. You do not type $, only everything after it.

Text in a `simple console font (like this)`, without a frame or background, indicates the name of a file, a function, or some action you need to perform.

3 File Hierarchy

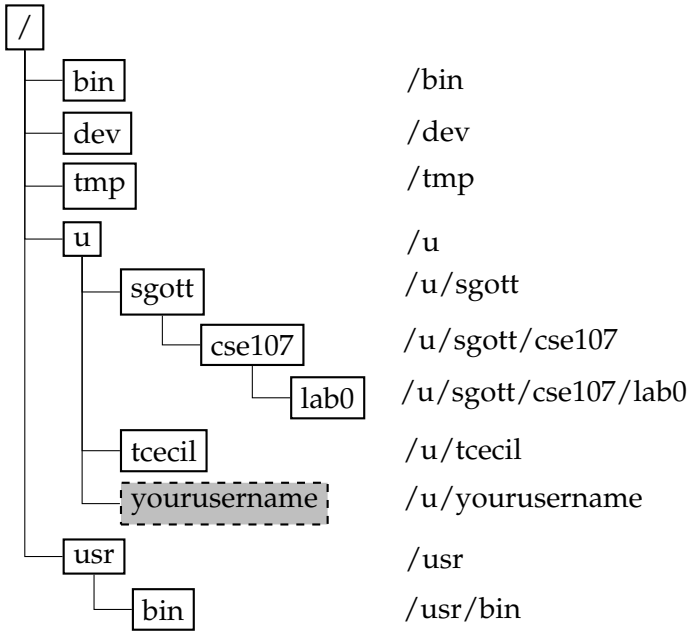
Most of the information stored on your computer appears in files and directories (directories are called folders on Windows.) In Linux, they are all stored somewhere within the root directory `/`. The files and directories on Linux machines are known as the file system tree.

A path consists of the names of directories and is a way to navigate from the root file system to a specific directory or file. If you include the root directory in the path this is called an absolute path. Another way to navigate the tree is to use relative paths, which navigate the tree based on the current directory.

An absolute pathindexpath, absolute starts at the root directory / and gives an absolute description of which directory or file you are refering to. For example, `/u/sgott/cse107/lab0/` refers to the directory as seen in the file tree below. A relative path always references your current directory. When you are in a shell, you always have a *current working directory* from which the relative path will go.

A relative path leaves off the first / and uses `..` to refer to the parent of the current working directory and `.` to refer to the current working directory. For example, if my current directory is `/u/sgott/cse107/`, then `lab0` refers to the absolute path `/u/sgott/cse107/lab0/`, while `../` refers to the absolute path `/u/sgott/`, and `../../tcecil` refers to the absolute path `/u/tcecil/`.

For example, this is a file system tree that resembles what you might find on the school's ITC computers. Absolute file paths are shown on the right, and a tree representation of the entire file system is shown on the left.



4 Commands

All commands that you type in a shell are structured in the following way:

```
$ commandname -options arguments
```

The command name comes first, followed by options that modify its behavior, and the arguments appear last. We will explore commands that take files (including Python programs) and directories as arguments. Remember that the $ just denotes a terminal prompt.

There can be any number of options and arguments, separated by spaces. For example, if the command is `python3` and you want it to run the Python program `hello.py` and you want to enter interactive mode after the file is run, you would type

```
$ python3 -i hello.py
```

in your shell. Options and arguments vary by the specific command. (Do not worry, we will explain what this specific command does later.)

4.1 Print the Working Directory with `pwd`

The directory you are currently in is called your "working" directory. The command called `pwd` **p**rints the **w**orking **d**irectory. Here is an example of running this command:

```
$ pwd
/u/yourusername
```

Note, first we type the command `pwd` in the terminal then press "enter" or "return." This causes the command to run and it prints its output (or result), which is the file path `/u/yourusername/`.

4.2 Change the Working Directory with `cd`

`cd` allows you to change your working directory to another location.

This is how it is used:

```
$ cd <path_name>
```

In the following example, we will change the working directory to the directory `/usr/bin`. Again, we type the command and the path we would like to go to and then press "enter." We also use the previously introduce command to make sure we have changed directories.

```
$ cd /usr/bin
$ pwd
/usr/bin
```

Using a special argument `cd -` returns you to your previous working directory.

The `-` is called a command line argument or option.

To move up one directory (one node) level use dot-dot:

```
$ cd ..
```

Where are you?

To move up two levels use dot-dot slash dot-dot:

```
$ cd ../..
```

To go to your home directory you can type

```
$ cd /u/yourusername
```

You can check that you are in your home directory by using the `pwd` command.

```
$ pwd
/u/yourusername
```

Alternatively, use a tilde (`cd ~`)

```
$ cd ~
```

Even easier than all these options is to just type `cd` without any options. This always takes you to your home directory.

```
$ cd
```

Argument to `cd`	Meaning
path	change directory to the supplied path
..	change directory to parent directory
~	change directory to home directory
-	change directory to previous working directory
no argument	change directory to home directory

4.3 Read the Manual with `man`

To view the manual for any command, use `man`

```
$ man commandname
```

For example,

```
$ man ls
NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort is  speci
       fied.
       ...
```

This shows the documentation for the `ls` command. A man page will usually include a *synopsis* that tells you all the possible options and usages of the command followed by a detailed *description*,

which is followed by a section that explains all the *options*. There can be many other sections in a man page, but most man pages will have at least these three.

Please try to make use of man pages before you ask for help — our first suggestion regarding anything that involves Linux commands will typically be for you to use the man page.

4.4 Listing the Contents of a Directory with `ls`

The command `ls` allows you to list the contents of a directory, that is, the files and directories that are located within.

```
$ cd
$ pwd
/u/yourusername
$ ls
<a list of file names and directories contained
with your working directory, which is currently
your home directory>
```

Adding some command-line options lets you see other file information:

```
$ ls -aF
.emacs.d/ Downloads/ hello.py
```

In Linux, any file whose name begins with a dot is a "hidden" file. In the example, the directory `.emacs.d` is a hidden directory. Most commands that list files in some way will hide files that begin with a dot.

The `a` option shows hidden files (a.k.a. dot-files), directories and the `F` option adds a `*` to the end of a file if is an executable file and a `/` if it is a directory. format.

4.4.1 Finding More Information

How can you see more command line options for `ls`? What is the option to sort contents of a directory by file size?

4.5 Make Directories with `mkdir`

Directories are very important for organizing your work, and you should get in the habit of creating a new directory for every lab this semester.

To create a directory, use the `mkdir` command as such:

```
$ mkdir newdirectorypath
```

For example, if you are in your home directory `/u/yourusername`, you can create the directory `cse107` in two ways:

Absolute path

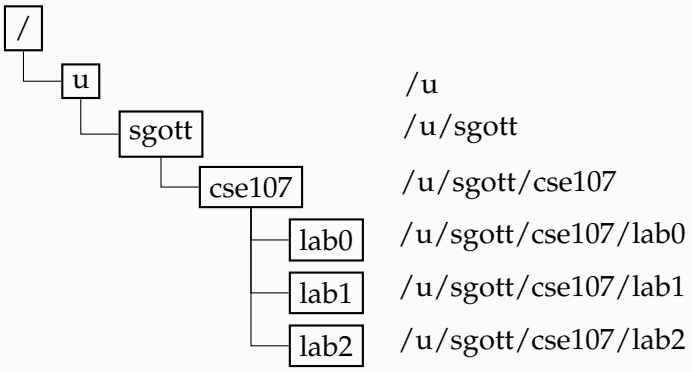
```
$ mkdir /u/username/cse107
```

Relative path

```
$ mkdir cse107
```

Habits

It's a good idea to have a directory for each lab inside a directory for the class. For example:



4.6 Read *The Linux Command Line*

Chapters 1 through 4 of *The Linux Command Line* by William Shotts cover the above material in much more depth. The book is available online as a PDF. Please read chapters 1 through 4 before lab 1.

`http://linuxcommand.org/tlcl.php`

4.7 Other Commands

These commands are described in many Linux resources.

1. `cp` — Copy a file.
2. `mv` — Move a file.
3. `rm` — Remove a file.
4. `cat` — Print the contents of a file.
5. `less` — Print the contents of a file. Useful for larger files.
6. `touch` — Create an empty file.

4.7.1 Resources

- *The Linux Command Line*, William Shotts, `http://linuxcommand.org`. (Chapters 1 through 4.)
- *Summary of common Unix commands*, John Shipman, `http://infohost.nmt.edu/tcc/help/pubs/unixcrib/`.
- Manual pages using `man`.

5 Editing Files

An easy to use graphical file editor is *gedit*. To open it, click on "Menu" in the lower-left hand corner of you screen and type "gedit". Press enter to run the program.

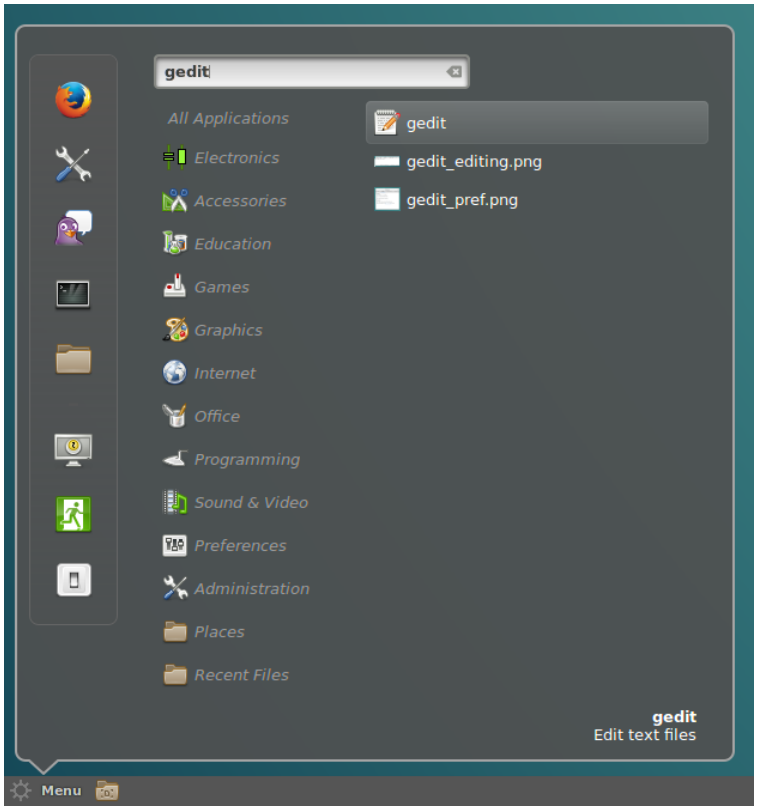


Figure 3: Opening `gedit`

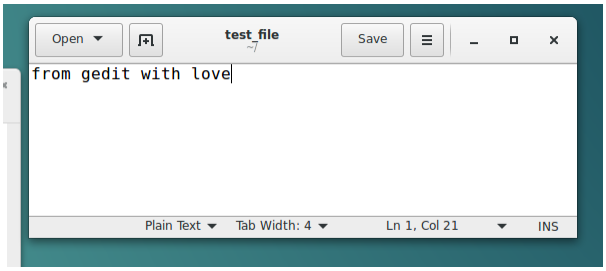


Figure 4: An example editing session.

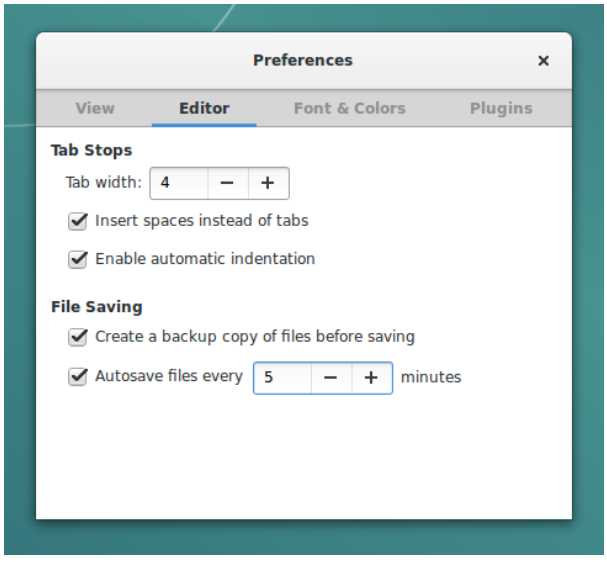


Figure 5: Useful settings for editing Python scripts using `gedit`.

notepad++ is an editor similar to *gedit* is available exclusively on Windows. You may also type the commands `vim` or `emacs` to edit files from the terminal. There are many other learning resources for both Vim and Emacs. Ask a TA, grader or tutor if you want to learn more about these editors.

Text editor	Download	Learning Resource
gedit		
Vim	`http://www.vim.org`	`https://www.openvim.com`
Emacs	`https://www.gnu.org/software/emacs`	`https://emacs.sexy`
notepad++	`https://notepad-plus-plus.org`	

6 Recording Your Sessions with `script`

The `script` command creates a file and writes the commands you type and their output. This is useful for keeping track of which commands you've been using and their output. For example,

```
$ ls
other files
$ script test_output
Script started, file is test_output

$ pwd
/u/yourusername
$ cd /
$ ls
bin cdrom dev etc lib root sbin sys usr var
$ exit
Script done, file is test_output
$ ls
other files command_line_output
```

Note `exit` is used to end the recording session and save the script file. Only use this command if you have started recording using `script`.

7 Exercises

See Section 4.6 for a reading assignment and Section 4.7 for a list of commands you should learn.

Exercise 7.1 (command_line_output).
This exercise is easier to perform if you use the `script` command for recording your commands.

1. Start recording your commands and all their output to a file named `command_line_output`.
2. Print your current working directory.
3. Create a file called `first.py`.
4. Within this directory, create a new directory called lab0.
5. Create empty files called `test1` and `test2` within the directory `lab0`.
6. Create a hidden and empty file called `hideme`.
7. List the files in the directory, inclding hidden files.
8. Remove the file `test1`.
9. Create a new directory called `testfiles` within `lab0`.
10. Move the file `test2` to this new directory.
11. List the files in the directory `lab0`, including hidden files.
12. Finish recording your commands.
13. Submit this script file and the file `first.py`. See the next section to learn how to compress your files and submit them.

8 Compress Your Exercise Files with `tar`

You need to submit your work as a tarball file with extension `.tar.gz`, which is a compressed file (like a `.zip` file). It should contain all files used in the exercises for this lab. All required files and the filename of the tarball are listed at the end of the lab.

To create a tarball file, we will use the Tar (`tar`) program. This command has a similar function to zipping utilities on other operating systems: the program combines many files and/or directories into a single file. Gzip is used in Linux to compress a single file, so the combination of Tar and Gzip do what Zip does. However, Tar deals with Gzip for you, so you will only need to learn and understand one command for zipping and extracting.

Ensure you are in your `lab0` directory, then type the following command, replacing `firstname` and `lastname` with your first and last names:

```
$ tar czvf cse107_firstname_lastname_lab0.tar.gz command_line_output first.py
```

This creates the file `cse107_firstname_lastname_lab0.tar.gz` in the directory. The resulting archive includes the files named `command_line_output` and `first.py`. The archive file is called a

tarball.

To check the contents of your tarball, run the following command:

```
$ tar tvf cse107_firstname_lastname_lab0.tar.gz
```

You should see a list of the output files that you compressed.

8.1 Command Line Options

Option	Purpose
`-c`	create an archive
`-x`	extract an archive
`-t`	list files in an archive
`-z`	compress using gzip
`-f FILENAME`	specify the archive name
`-v`	list which files are processed

8.2 Uploading to Canvas

Now that you have your tarball ready, you may submit it to Canvas. Navigate to `https://nmt.instructure.com`, login, and select the CSE 107L class. You should see "CSE/IT-107L" in the upper-left hand corner of the webpage. Click on "Assignments", find the correct assignment and upload your tarball. Please note that graders will post the rubric on your assignment, so check the assignment once you have received a grade to read the rubric.

Index of New Terminology, Programs, and Linux Commands

Submitting

You should submit your code as a tarball that contains all the exercise files for this lab. The submitted file should be named

`cse107_firstname_lastname_lab0.tar.gz`

Upload your tarball to Canvas.

List of Files to Submit

Exercises start on page 9.