

College of the Holy Cross
Introduction to GNU/Linux

Basic Concepts

Welcome (or welcome back) to the College of the Holy Cross! This document will introduce you to the Swords 219 workstations. It should be particularly helpful if you are new to GNU/Linux, but even if you have used *nix in the past you may learn something new.

The Math/CS server `radius` runs the operating system “GNU/Linux”. The term “Linux” refers only to the *kernel*, the single program that manages memory, disk storage, CPU time, and input/output. “GNU” refers both to the system software—including most of the programs you will run—and to the philosophy of Free (*libre*, not *gratis*) Software.

Unlike Microsoft Windows and Apple MacOS, GNU/Linux is largely command driven. There are graphical programs, and a desktop environment to rival the common PC operating systems, but many tasks are most easily accomplished at a command prompt.

Multi-User Systems

There are less obvious but more important differences between GNU/Linux and common PC operating systems than their user interface. First, GNU/Linux is a *multi-user* operating system. Policies are enforced by a system of “accounts” and “passwords”, “home directories”, and “permissions”.

User Accounts In order to use `radius`, you must have an *account* on the machine. Your account has a unique name and is allotted a “folder” called your *home directory*. Similarly to your high school locker, your home directory stores your course work, email messages, and other private data.

Passwords Access to your account, and therefore to your home directory, is controlled by your *password*. Your password is the combination to your locker; *it must not be shared with anyone*. There are excellent security reasons for demanding that only you know the password to your account. **Sharing your password is grounds for losing your computer privileges.**

A good password must be easy to remember (otherwise you’ll be locked out of the system) but difficult to guess (otherwise someone else could break in to your account). Choose a password consisting of at least 8 characters, including letters, digits, and punctuation. To make your password easier to remember, choose a mnemonic spelling of something you can remember, such as `Sm1;)l3y`.¹ The following are **poor** choices of password: Any literal word that can be found in a dictionary in any language, your name or any part of it, spelled forward or backward, or any string that is all letters, all digits, or all punctuation.

Directories As in Windows or MacOS, information on a GNU/Linux machine is stored in chunks called *files*. Files are, in turn, grouped in *directories* with names like

`/usr/bin` `/etc` `/usr/local/share` `/home/stu/luser`

¹Smiley, get it? Don’t choose this as your password; it’s easy to guess now!

(the last might be the home directory of Leon User). Directories are organized into a “tree”: Each directory (“node”) has a unique *parent directory*. The parent of Leon User’s home directory is `/home/stu`. The parent of `/home/stu` is `/home`. Above `/home` is the *root directory*, `/`. Every file on `radius` lies in some subdirectory of `/`, which is its own parent. As you become more experienced, you’ll learn where to look for programs and data files, but this knowledge is not necessary to use `radius`.

Ownership and Permissions Every file on `radius` has an *owner*. Files in your home directory are usually owned by you. Files in system directories like `/bin` are usually owned by the administrator account, `root`.

Every file potentially provides three kinds of access: read (`r`), write (`w`), and execute (`x`). Read permission allows you to view the file’s contents, write permission grants the right to modify (“edit”) or delete the file, execute permission allows you to run a file, if the file is a program.

You have write permission only in your home directory and in the system temporary directory, `/tmp`, and do not even have read permission for critical system files. On a multi-user system like `radius`, access restrictions are essential. You don’t want classmates copying or deleting your work (a hilarious prank, is it not?), and no one but the system administrator should be able to read the file containing users’ passwords or execute the `reboot` command. Ownership and permissions are built into the operating system at a fundamental level, unlike (say) Windows 9x which provides no real protection of individuals’ data.

Using GNU/Linux

This section gives a quick introduction and basic tips for system use. If you are new to GNU/Linux, log on to a Sun workstation and experiment with commands for practice.

Logging In When you sit down at a workstation, you are faced with a *login manager*. Type in your name and press `return`, then type in your password. (You will not see anything when you type your password. This prevents others from seeing how many characters your password contains.) If your account name and password are recognized, you will eventually see a `desktop`, which should look generally familiar. You can use the mouse to select and click on icons. In GNU/Linux, you single click rather than a double click. The mouse has three buttons. The left button generally selects an icon, while the right button may cause a menu to pop up. The middle button’s function varies from program to program.

Organizing Your Files Cultivate sensible work habits from the start. Maintain your files in an orderly fashion, by grouping related files in a single directory, and by giving files short, descriptive names. To create a directory, type `mkdir` followed by the name of the new directory. If you are taking CS 261, you might do `mkdir CS261`, then put all your files related to the course in that directory. Directory names should not contain spaces or other punctuation. (When you learn more about file names you can relax this restriction a bit.) As with all names in GNU/Linux, case matters; `CS261`, `Cs261`, and `cs261` are three different names.

Running Commands

The GNU shell `bash` is a command interpreter and scripting language. You type a command at a prompt, press `return`, and the system responds. To run `bash`, select a terminal icon from a drop-down menu.

When you first open a terminal window, your *working directory* is your home directory. Type `pwd` (print working directory) to see its name. To see what files are in a directory, type `ls` (list). Like most GNU/Linux commands, `ls` accepts *options* that modify the effect of the command. For example, `ls -l` (long listing) shows files' permission, owner and group, size, and modification time.

To change your working directory, type `cd` (change directory) followed by the name of the directory to move to. Typing `cd` by itself takes you back to your home directory, so you can't get very lost. Typing `cd ..` takes you to the parent of the working directory. Try doing this command a few times and see what happens. If you poke around in system directories, you're likely to get a "permission denied" message. This does no harm, but is merely GNU/Linux' gentle way of saying "keep out".

Simple commands such as `ls` or `cd` conform to the *nix philosophy: Perform one task well, and build up complex actions from simple pieces. The shell can "daisy chain" programs by sending the output of one to the input of the next.

Job Control Some commands run for a short time, then exit. Others start interactive programs, such as web browsers or text editors. When the shell spawns a process, it passes control to the "child", and cannot accept more commands until the child "returns". Because you usually want to run several interactive programs simultaneously without opening a terminal for each, the shell provides functionality called "job control". A process can be started in the "background" by typing a command followed by an ampersand (`&`). When a shell spawns a child in the background, control returns at once to the parent and you can type more commands.

Advanced Shell Features `bash` provides *tab completion* and *command line editing*. If you press the `TAB` key while typing a command, `bash` will match what you've typed so far against all possible completions, and will either complete the command as far as possible (if there is a unique choice) or (if you press `TAB` again) show you a list of possible completions. Tab completion can reduce the number of keystrokes by 60–80%, an obvious and substantial convenience. Second, `bash` permits "emacs-style" searching and editing of commands (including the arrow keys). With a couple of keystrokes, you can search through the previous 2500 commands you've run, and use the arrow and backspace keys to edit commands.

Every program has three "streams" by which it can exchange data with the system or other programs: *standard input* (`stdin`, usually a file or typed command line), *standard output* (`stdout`, usually a disk file or the screen), and *standard error* (`stderr`, usually the screen). Output can be written or appended to a disk file, input can be read from a disk file, and output of one command can be "piped" to the input of another. For example, the command

```
grep -v "the" names.txt | sort | uniq > tmp
```

finds all lines in the file `names.txt` that *do not* contain the string "the", sorts them alphabetically, removes duplicates, and writes the result to a file named `tmp`.

These features will become more useful as your use of GNU/Linux becomes more advanced. Some day you'll confidently type monster commands like:

```
ssh luser@radius.holtcross.edu cd CS261 && for FILE in *.cc; do
  mv $FILE $FILE.orig; sed 's/foo/bar/g' $FILE.orig > $FILE; \
done && tar -jcvf originals.tar.bz2 *.orig && rm -f *.orig
```

When the command² exits with an error because you've misspelled "holtcross", or you need to run the command again with a slight variation, you'll welcome command editing.

Software Overview

The Sun machines have a wide variety of software, including text editors, mathematical typesetting engines and document previewers, ray tracing and real-time mathematical visualization programs, web browsers, and compilers or interpreters for many common computer languages (C/C++, Fortran, Java, Scheme, and Perl, to name a few), as well as a full suite of shell utilities for manipulating text files in various ways.

Here is an incomplete list of programs you will probably learn to use at some point:

emacs: A Swiss Army knife that many mistake for just a powerful text editor.³ Invoke it by name with `emacs &`, then type `C-h t` (control-h t) for an interactive tutorial. Don't let the low-tech interface fool you.

gcc, g++, g77: The GNU compilers for C, C++, and Fortran. These are the programs you use to turn your source code into executable binary files.

tcsh and bash: Two shells that feature (among many other things) command line editing, command completion, and a history mechanism that allows you to re-run commands very easily. Each is also a full-featured programming language, with variables, decision statements, and loops.

L^AT_EX: The *de facto* standard for professional mathematical typesetting. L^AT_EX is a markup language and typesetting engine that allows camera-quality printed documents (especially those containing mathematics) to be written and stored in human-readable plain text form. L^AT_EX encourages good writing by forcing you to concentrate on the *structure* of your document rather than its *appearance*.

vi (vee eye): A small, fast text editor, designed to be run over slow data lines. Most commands are one or two characters, and they are *not* mnemonic. `vi` is a useful complement to `emacs`, though staunch advocates of each regularly get into flame wars on Usenet.

xdvi, gv, xpdf: Document previewers, the first for compiled L^AT_EX, the others for Postscript and Portable Document Format (PDF, often incorrectly called "Adobe Acrobat") files.

²This command, which can be run from anywhere in the world, securely logs on to `radius`, finds all C++ source files in the directory `CS261` and makes backup copies, substitutes every instance of "foo" with "bar" in these files, creates a compressed archive containing the originals, and finally removes the old files.

³See <http://vh213601.truman.edu/~jay/emacs.html>

pine: An easy-to-use text-based mail reader.

You'll probably want to learn these shell utilities. Type (e.g.) `ls --help` for built-in help, or `man ls` for the system manual page. In the man page reader, `SPACE` shows the next page and `q` quits. Typical usage is shown.

ls: List files in a directory. (`ls -lt`)

cp, **mv**, **rm**: Copy, move, or remove files. (`cp -p myfile.tex myfile-05.14.07.tex`)

ssh and **scp**: A secure remote login program that allows you to work on projects from anywhere, your dorm room or across the world. (`ssh luser@radius`)

less: A *file pager*, which shows you a text file one screen at a time. Hit `SPACE` for the next screen, or `q` to quit. (`less myfile.tex`)

lpr: The print command; usually requires options to be useful.

tar (tape archive): A utility for packing several files into a single file, conceptually analogous to ZIP or Stuffit. (`tar -jcvf cs261.tar.bz2 CS261`)

mkdir, **rmdir**: Create and remove directories.

grep (get regular expression): “The vice grips of Unix.” **grep** searches for patterns in text files, and behaves in any of several ways upon finding a match.

find: Locates files in a directory based on attributes such as name, permissions, modification time, or type of file.

sed (stream editor): Search and replace character strings in a file.

diff: Shows the differences between two files. Useful if you've been editing and want to see all the changes you've made.

chmod (change mode): Changes permissions on files.

Getting Assistance The easiest way to get help is to ask someone knowledgeable, either another student or a faculty member. Online information about most commands can be read by typing `man` (manual) followed by the name of a command; type `man man` for details!