

HOW TO Use Linux, Text-Edit, Compile, and More

Using the Linux/UNIX operating system (to work in your eng101 directory):

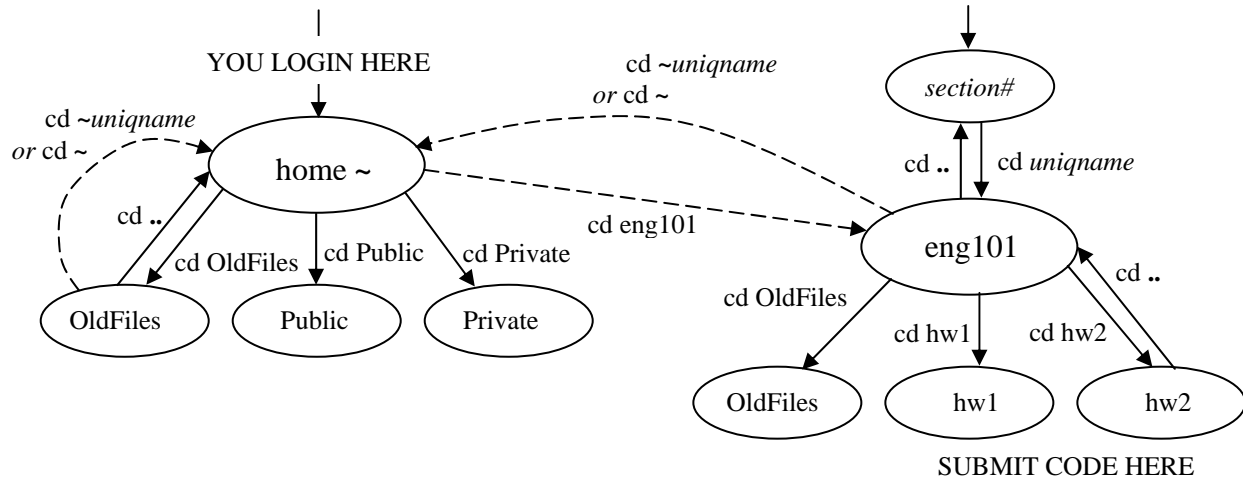


Figure 1. Diagram of the Linux/UNIX directory structure. The ovals represent directories and the arrows show where the cd commands take you. The dashed arrows represent links. Ovals underneath other ovals represent subdirectories whereas the ovals overhead represent parent directories.

1. Start a terminal window by right-clicking on the desktop then select **New Terminal**. You will see a prompt, which is the name of your computer, like this:
ruby%
2. To display the path of your working directory (or where you are), at the prompt type:
ruby% pwd
Your home directory path should be:
/afs/engin.umich.edu/u/firstletter/secondletter/username
This is where you keep your personal (non-Eng101) files.
3. To change from your home directory to your Eng101 directory, type:
ruby% cd eng101
4. Again, type:
ruby% pwd
Your Eng101 directory path should be:
/afs/engin.umich.edu/class/f03/eng101/section#/username
This is where you will be writing, saving, compiling, and running your Eng101 programs.
5. To make your homework submission subdirectories, type:
ruby% mkdir hw1
Repeat for homeworks 2 through 9. Note that Linux/UNIX is case-sensitive.
6. To list the contents of your working directory, Eng101, type:
ruby% ls
You should see your hw1 subdirectory now created.
7. To change to your hw1 subdirectory, type:
ruby% cd hw1
8. To change back to your Eng101 parent directory, type:
ruby% cd ..

Some useful Linux/UNIX commands:

pwd	prints path of <u>w</u> orking <u>d</u> irectory	cp <i>from to</i>	<u>c</u> opies a file
ls	<u>l</u> ists directory contents	mv <i>from to</i>	<u>m</u> oves a file
ls -l	<u>l</u> ists <u>l</u> ong detailed directory contents	rm <i>name</i>	<u>r</u> emoves a file
cd <i>name</i>	<u>c</u> hanges <u>d</u> irectory (into subdirectory)	mkdir <i>name</i>	<u>m</u> akes a <u>d</u> irectory
cd ..	<u>c</u> hanges to parent <u>d</u> irectory	rmdir <i>name</i>	<u>r</u> emoves a <u>d</u> irectory
cd ~	<u>c</u> hanges to your home <u>d</u> irectory	<i>command --help</i>	shows options <u>h</u> elp

Using the Emacs text-based text editor (to write text files and C++ source code):

1. In a terminal window, type:
ruby% emacs filename &
The **&** (ampersand) allows Emacs to run in the background, detached from the terminal.
2. Press **<ctrl>+x** then **<ctrl>+s** to save the text to a file.
3. Press **<ctrl>+x** then **<ctrl>+c** to exit Emacs.
 - Press **<ctrl>+x** then **<ctrl>+f**, then type a filename to open or create a file.
 - Press **<ctrl>+g** to cancel an Emacs command at any time.
 - Press **<ctrl>+h** then **t** to start the emacs built-in tutorial.
 - Other text-based text editors you may try are **vim**, and **pico**.

Using the GEdit graphical text editor (to write text files and C++ source codes):

1. In a terminal window, type:
ruby% gedit filename &
The **&** (ampersand) allows GEdit to run in the background, detached from the terminal.
2. Use the menu with the mouse to open, edit, and save your text.
 - Other graphical text editors you may try are **nedit**, and **kwrite**.

Using the g++ compiler (to make executables of your C++ program):

1. In a terminal window, change to the directory that contains your code, and type:
ruby% g++ filename.cpp -o executablename -s
where "**filename.cpp**" is the name of the C++ source code ready to be compiled, "**-o executablename**" denotes the outputted executable program name, and "**-s**" strips the executable thus making the executable smaller.
ex: **ruby% g++ myprog.cpp -o runmyprog -s** or
ex: **ruby% g++ myprog.cpp -o runmyprog** (makes a stand alone executable)
ex: **ruby% g++ myprog.cpp -s** (executable is named "**a.out**" by default)
2. To run the compiled C++ program, type the executable name:
ex: **ruby% runmyprog** or
ex: **ruby% ./runmyprog** (ensures program in working directory is run) or
ex: **ruby% a.out** (if no executable name was specified at compiling)

Using the **cp**, **mv**, and **rm** commands (to copy, move, and remove your files):

To copy a file from a source to a destination, type:

```
ruby% cp sourcepath/filetocopy destinationpath/newfilename  
ex: ruby% cp myprog.cpp backupdir (simplest usage) or  
ex: ruby% cp myprog.cpp ~/eng101/hw1 (with dest. path) or  
ex: ruby% cp myprog.cpp ~/eng101/hw1/equation.cpp (w/new name)  
ex: ruby% cp ~/myprog.cpp hw2 (with source path from home) or  
ex: ruby% cp ../sample.cpp hw2 (with source path from parent) or  
ex: ruby% cp ../sample.cpp ~/eng101/hw2/mysample.cpp (full usage)
```

To move a file from a source to a destination, replace **cp** with **mv** in above examples. Note that moving a file deletes the source file; so use only when necessary.

To remove or delete a file, type:

```
ruby% rm path/filetoremove  
ex: ruby% rm myprog.cpp (simplest usage) or  
ex: ruby% rm ~/eng101/myprog.cpp (with path)
```

Note that removing a file permanently deletes the file. There is no undelete; so be careful!

Using the **enscript**, **lpq**, and **lprm** print commands (to print your text files and C++ code):

To print a text file, typically change to the directory that your file is in, then type:

```
ruby% enscript options -Pprintername path/filename  
ex: ruby% enscript -Pb505pierpont myprog.cpp (simplest usage) or  
ex: ruby% enscript -2Gr -Pb505pierpont myprog.cpp  
(with 2-column, fancy header, landscape formatting)
```

- Printer names can be found on or near the physical printer in any CAEN lab like: **b505pierpont**, **b507pierpont**, and **b521pierpont**.
- You may also use the non-formatting standard **lpr** print command instead.
ex: **ruby% lpr -Pb505pierpont myprog.cpp**

To check the print queue, type:

```
ruby% lpq -Pprintername  
ex: ruby% lpq -Pb505pierpont
```

To remove your own print job from the print queue, type:

```
ruby% lprm -Pprintername jobnumber  
ex: ruby% lprm -Pb505pierpont 30
```

Using additional resources:

CAEN provides excellent technical notes on the following topics: *Email • AFS • Communications • Printing • Databases • Connecting to CAEN • UNIX • Terminal Servers • Web Resources • Programming • Applications • Mathematics • Text Editors • Graphics & CAD • and more.*

Technotes are found at: <http://www.engin.umich.edu/caen/technotes> .

Submission headings:

All source files should include a commented heading up top, using `//` :

- Your actual name
- Your username
- Eng101, Section #, Your GSI's name
- Homework #
- Date
- Filename of source code
- Description of program (i.e. title, what it does, inputs, outputs, etc.)

Indentation style:

Indent where appropriate to make your source code easier to read. You should indent with tabs or a consistent number of spaces inside of:

- Loop statements (while, for)
- Selection statements (if/else)
- Braces { }

Commenting:

Comment chunks of code with a short description for clarity and modularity. There is no need to comment every single statement. The following should be commented:

- Non-descriptive variable declarations
- Non-trivial assignment statements or calculations
- Loops (while, for) and selection statements (if/else)
- User-defined functions

Re-grades:

Homework, in-lab, and exam re-grade requests are made by submitting an explanation of your re-grade request written on a separate sheet of paper, stapled to your original unmarked homework/in-lab/exam paper. Write your name and section number on the request and submit it to your GSI as soon as possible. DO NOT write anything on the original graded homework, in-lab, or exam.