Introduction to Unix Tutorial

ECS30 University of California, Davis

Table of Contents

This document is based on Sean Davis' Unix Tutorial. The following conventions will be used:

- Instructions will be in: this normal typeface and *this italics typeface*
- Things which you will type will be in: **`this bold typeface`**
- Keys which you will press will be in: **`<this bold typeface>`**
- Through this document, I've created links to other websites. You do not need to go through the sites that are linked. You should note that they may contain useful information should you find it necessary (or you are curious).

CSIF Computer Network

Students taking computer science classes work on the machines run by CSIF (Computer Science Instructional Facility), located in the basement labs of Kemper. If you are enrolled in a class this quarter, you will automatically have an account on this system (see below, "Local Logging in"). Currently the bulk of the system consists of about eighty PC (Intel) workstations, named pc01 to pc60 (yes, you need the 0 in "pc01", "pc02", etc.). The CSIF website has much more information than you may need for this course in their FAQ (frequently asked questions). It does not matter *which* computer you log in to in order to do your work. Your work will sync to your login

Local Computer Access

If you are trying to work from home, please skip to section "**Remote Computer Access: ssh**". If you are on campus, you can use the CSIF machines located in the basement of Kemper. If you are taking a computer science class, you will automatically get an account on the CSIF machines. To access an account, you will need two things: 1) your Username and 2) your Password. Your username is your UCD Kerberos username, and your password is your Kerberos password.

Type your login name, then your password. Do this now. If all goes well, the monitor will display the UNIX graphic user interface (GUI) of the computer for you.

The PCs do not start with a terminal window. To create a terminal window on a PC, click on the *f* icon at the left end of the toolbar, and then click on the Terminal item in the menu of System Tools. Open a terminal window now. For the rest of this tutorial will only be typing in one of these terminal windows.

Once you have created a terminal window, you will see a “command line prompt” indicating that Unix is ready to accept your instructions. This prompt will usually have your username, computer name, and end with a "$", but may be terminated by a "%". The prompt is generated by a program known as a *shell*. This is simply a program that accepts your commands and gets UNIX to execute them.

Remote Computer Access: ssh

If you are in the CSIF, then please skip this section. If you are at home, then we need you to connect to the CSIF before you continue. The first step is determining the name of CSIF computer with which to connect. The CSIF computers are named pc01 to pc60, with about five missing. You may simply choose any number in that range, say 17. Then the computer name would be pc17.cs.ucdavis.edu.

You can remotely log in to a CSIF machine from some other computer by using secure telnet software. The ssh software is already installed on a Mac, but must be installed for Windows. If you are on Windows machine please skip down to the topic “**ssh in Windows**”.

ssh in Mac OS

For Macs, there is a built-in UNIX terminal like on the CSIF computers.

The syntax for secure shell is: “ssh *computer_name”.* So in our example you would type: “ssh pc17.cs.ucdavis.edu*”* If you have never connected to the computer before, ssh will ask if you are sure of the remote computer's authenticity and you will see a response similar to this:

The authenticity of host 'pc17 (169.237.5.14)' can't be established.

RSA key fingerprint is f0:ad:72:d2:2e:84:ab:53:2e:06:60:7d:3e:38:17:3c.
Are you sure you want to continue connecting (yes/no)?

In such a case, just type yes to continue. Once the authenticity of the remote computer is established, ssh will prompt you for your username and password on the remote computer.

Now you are ready to make a real ssh connection. Type:

```
ssh pcXX.cs.ucdavis.edu<enter>
```

where `XX` is a number between 01 and 60. Is the pc you picked up (working) or down (not working)? From here on out, when you need to type something in, we will omit writing `<enter>`.

Once you have entered your Kerberos name and password, you should now see a prompt that reflects your username and the remote computer, e.g. [rsgysel@pc17 ~].

ssh in Windows

Since Windows does not come with ssh installed, you will need to download an ssh program. PuTTY is a free ssh program.

Once you have downloaded PuTTY, you will need to choose a CSIF with which to connect. You must randomly choose a computer name between pc01 and pc68. The following example assumes you chose pc17.

Start Putty now. You will see a dialog window with the cursor flashing in the “Host Name” box. Now enter the name of computer name you chose, including its domain. In our example, you would type: pc17.cs.ucdavis.edu. Once you have entered the computer name, you make sure that the “Connection Type” is “ssh”, and the “Port” is “22.” If you wish, you can save this connection information for later use my entering a descriptive name, e.g. “CSIF” or “pc17”, in the “Saved Sessions” box, and then clicking “Save.” Now click “Open”. Since you will have never connected to this CSIF computer before, PuTTY will produce the “PuTTY Security Alert” dialog box that asks if you trust this host. Click “Yes.”

Now enter your Kerebos username, and then your Kerebos password. You should now see a prompt in the PuTTY window that reflects your username and the remote computer, e.g. [rsgysel@pc17 ~].

Navigating the UNIX file system: pwd, cd, ls

Each directory in the file system has a *path name* that uniquely specifies the location of that directory within the computer's file structure. As you move within the file structure, you can use the command pwd to display the path name of current location. pwd stands for "print working directory". Type:

```
pwd
```

The shell should display */home/* followed by your username, e.g. */home/ssdavis*. This is your *home directory*. This is the place where you "live" in the Unix file system. Each user has their own *home directory* that is completely under their control.

The cd change directory) command allows you to move to any other public directory in the file system. The syntax for cd is: "cd *path_name"*. Now change to the */bin* directory by typing

```
cd /bin
```

After executing the directory change, use

```
pwd
```

to see that you really are in the */bin* directory.
The ls (list) command prints the contents of a directory. The syntax is: "ls [-al] [*filename*]", where the "[ ]" indicate optional values. Type

```
ls
```

You will see a list of the names of the files in the */bin* directory. The */bin* directory holds many of the basic UNIX utility programs. Note that "ls" is listed, but not "cd" or "pwd". This is because "ls" is an independent program, but "cd" and "pwd" are commands built into the shell program.

Before you look into the options for ls, I want you back in your home directory. You could type "*cd /home/"* followed by your username, e.g. "*cd /home/rsgysel"*, but there is a short cut. Simply type

```
cd
```

You should now be in your home directory. To confirm this, type

```
pwd
```

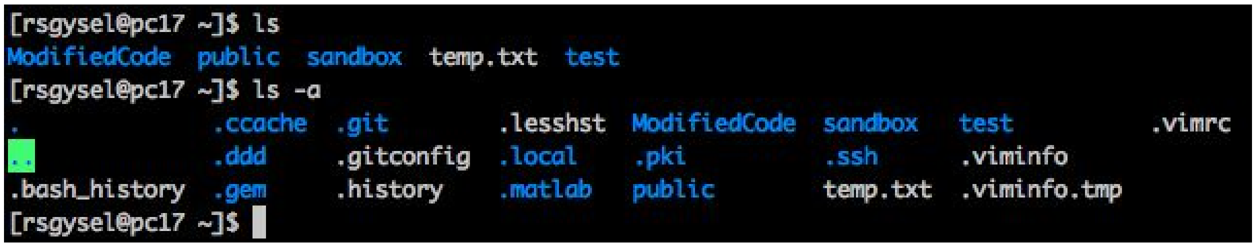
Now type

```
ls
```

If you have not created any files or directories in your account, then nothing will be displayed. Many shell commands have *options* that may follow the command after a space and a dash. The options change the behavior of the command. For ls, the options change what you see -- giving you specialized types of information which you wouldn't see using the basic command. Now type:

```
ls –a
```

The –a option stands for “all”. Now you will see *hidden* files that are used by the operating system or other programs. On my account I see the following, which will differ from your account:



```
[rsgysel@pc17 ~]$ ls
ModifiedCode  public  sandbox  temp.txt  test
[rsgysel@pc17 ~]$ ls -a
.              .ccache  .git        .lesshst  ModifiedCode  sandbox   test          .vimrc
..             .ddd     .gitconfig  .local    .pki          .ssh      .viminfo
.bash_history  .gem     .history    .matlab   public        temp.txt  .viminfo.tmp
[rsgysel@pc17 ~]$ 
```

Where did all these new files come from after using “ls -a”? Note that all of them start with a period. UNIX hides system files that are normally not needed by the user by starting them with a period. There are two unusual files listed, “.”, and “..”. These have special meanings in path names. “.” stands for the current directory.

Now type

```
cd .
```

(with just one period after cd). Now type

```
pwd
```

Note that you haven’t moved within the directory structure! “..” stands for the parent directory of the current directory. Now type

```
cd ..
```

(with two periods after cd). Now type

`pwd`

You should see "*/home*".

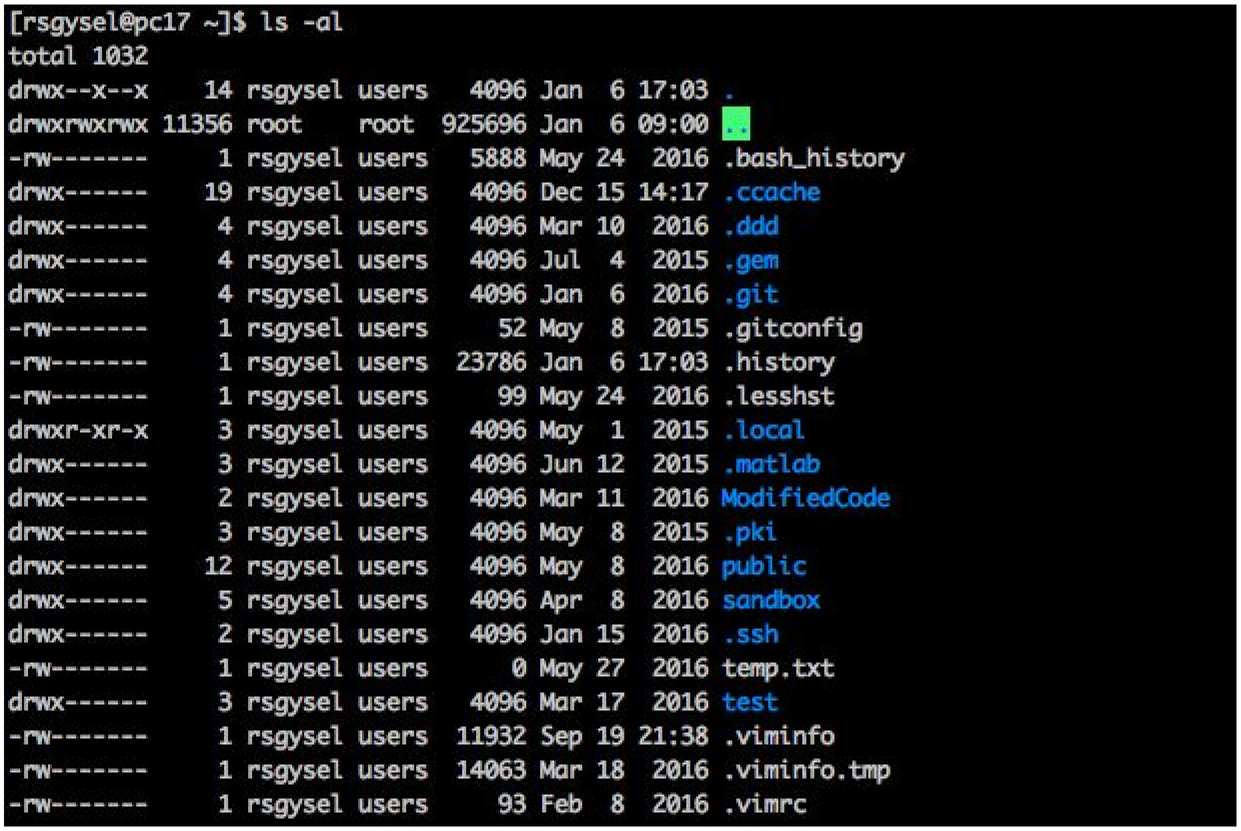
Now return to your home directory by simply typing

`cd`

Now type

`ls -al`

The –l option stands for long. The shell will display a line of information for each file. I get:



```
[rsgysel@pc17 ~]$ ls -al
total 1032
drwx--x--x    14 rsgysel users   4096 Jan  6 17:03 .
drwxrwxrwx 11356 root    root  925696 Jan  6 09:00 ..
-rw-------     1 rsgysel users   5888 May 24  2016 .bash_history
drwx------    19 rsgysel users   4096 Dec 15 14:17 .ccache
drwx------     4 rsgysel users   4096 Mar 10  2016 .ddd
drwx------     4 rsgysel users   4096 Jul  4  2015 .gem
drwx------     4 rsgysel users   4096 Jan  6  2016 .git
-rw-------     1 rsgysel users     52 May  8  2015 .gitconfig
-rw-------     1 rsgysel users  23786 Jan  6 17:03 .history
-rw-------     1 rsgysel users     99 May 24  2016 .lesshst
drwxr-xr-x     3 rsgysel users   4096 May  1  2015 .local
drwx------     3 rsgysel users   4096 Jun 12  2015 .matlab
drwx------     2 rsgysel users   4096 Mar 11  2016 ModifiedCode
drwx------     3 rsgysel users   4096 May  8  2015 .pki
drwx------    12 rsgysel users   4096 May  8  2016 public
drwx------     5 rsgysel users   4096 Apr  8  2016 sandbox
drwx------     2 rsgysel users   4096 Jan 15  2016 .ssh
-rw-------     1 rsgysel users      0 May 27  2016 temp.txt
drwx------     3 rsgysel users   4096 Mar 17  2016 test
-rw-------     1 rsgysel users  11932 Sep 19 21:38 .viminfo
-rw-------     1 rsgysel users  14063 Mar 18  2016 .viminfo.tmp
-rw-------     1 rsgysel users     93 Feb  8  2016 .vimrc
```

Here is a brief introduction to the information displayed in the long format.

- File permissions and file types (leftmost field)
    - *drwxr-xr-x*
        - The d at the very left says the file is a *directory*.
        - The next letters rwx say that the owner (myself) has *read*, *write* and *execute* permissions for the file. For a directory, execute permission

        means I can cd to that directory. Read permission means I can use the command ls to see the contents of the directory. Write permission means I can create additional files *in* that directory.
        - The next two groups of three show that other users (*group* and *world*) are only allowed read and execute permissions.
    - *-rwxr--r--*
        - The dash at the very left indicates an *ordinary file*, as opposed to a directory. Here the read, write and execute permissions mean the ability to look at the file (using commands like more, see below), to *edit* the file (using a text editor, see below), or *execute* the file (if it is an executable program -- otherwise this permission doesn't mean much).
    - *-rw-r--r--*
        - This set of permissions shows an ordinary file which can only be read and edited by the owner, and only read by everyone else.
- Links field (second from left): number of short cuts to this file from other directories. In linux, we call these *symbolic links (symlinks) or hard links*.
- Owner field (third from left)
- Group field (fourth from left)
- Size field (fifth): shows the file size in bytes
- Date field: when the file was last modified
- Name fields (last).

Notice there were no pathnames after any of the ls commands. That means that you saw the contents of the current working directory (e.g. “ls” is equivalent to “ls .”). If you do put a pathname afterwards, then you will see the contents of the directory identified by that pathname. Now type

```
ls /
```

In this example, the pathname you gave ls was the *root* directory (indicated by a sole backslash), which is the top directory in the Unix file system. All the files you see listed there are subdirectories of the root. Notice the subdirectory *home*. That is the subdirectory in which the home directories of everyone having an account are located.

Creating and Deleting Directories : mkdir, rmdir

To create a directory use the mkdir command. It’s syntax is mkdir {*directory_name}*+. The "{}+" indicate that there must be at least one directory name given. Now type the commands

```
mkdir first
ls -l
```

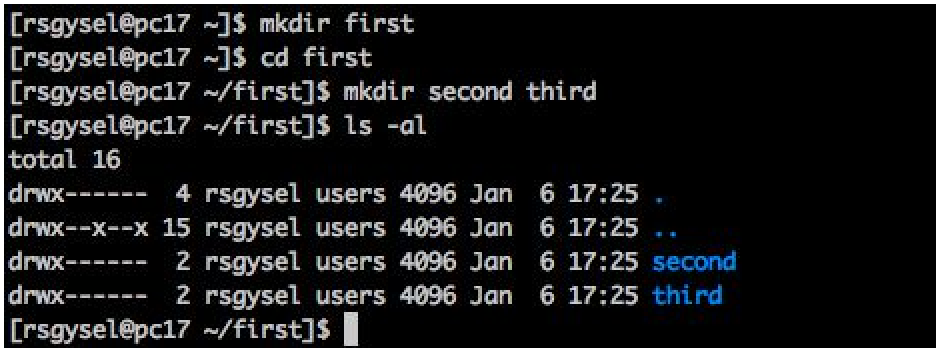
You should have a directory named *first* now in your file list. Now type

```
cd first
pwd
```

The “cd first” command *changes directory* to the directory *first*. Now type

```
mkdir second third
ls -a
```

You should see:

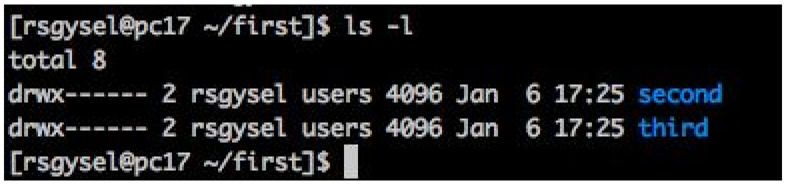


```
[rsgysel@pc17 ~]$ mkdir first
[rsgysel@pc17 ~]$ cd first
[rsgysel@pc17 ~/first]$ mkdir second third
[rsgysel@pc17 ~/first]$ ls -al
total 16
drwx------  4 rsgysel users 4096 Jan  6 17:25 .
drwx--x--x 15 rsgysel users 4096 Jan  6 17:25 ..
drwx------  2 rsgysel users 4096 Jan  6 17:25 second
drwx------  2 rsgysel users 4096 Jan  6 17:25 third
[rsgysel@pc17 ~/first]$ 
```

If you type

```
ls -l
```

You will see a more condensed directory listing, which is what you will usually be interested in:



```
[rsgysel@pc17 ~/first]$ ls -l
total 8
drwx------ 2 rsgysel users 4096 Jan  6 17:25 second
drwx------ 2 rsgysel users 4096 Jan  6 17:25 third
[rsgysel@pc17 ~/first]$ 
```

The syntax for the remove directory command is “rmdir {*directory_name*}+”. Now type

```
rmdir second
```

to remove the directory named *second*. Check that only *third* is listed when you use ls.

Copying, Moving, and Deleting Files : cp, mv, rm

Use pwd to make sure that you are still in */home/user_name/first.* There are two syntaxes for the copy file command:

1. cp *source_file_name destination_file_name*
2. cp {*source_file_name*}+ *destination_directory*

Now you will copy a file from user rsgysel's home directory into the current directory.

Type

```
cp ~rsgysel/CSatHome.txt myfile
```

Use ls to make sure that you were successful. You should see only *myfile*, and *third* listed. Now copy this file down into the *third* directory by typing

```
cp myfile third
```

You could check your work by cd'ng into the *third* directory and then using ls, but there is simpler way. Just type

```
ls third
```

to display the contents of the *third* directory. You should see only *myfile* listed.

The mv command has three syntaxes:

1. "mv *old_file_name new_file_name"* to change the name of a file
2. "mv {*filename}*+ *directory_name"* to move file(s) into a directory
3. "mv *old_directory_name new_directory_name"* to change the name of a directory.

You will try each one in order. Now type

```
mv myfile wombat
```

to rename *myfile* to *wombat*. Use ls to check your work. You should see only *third* and *wombat* listed. Now type

```
mv third/myfile .
```

(note the period indicates the current directory) to move *myfile* to the current directory. Again use ls to check your work. You should now have *myfile, third,* and *wombat* in the current directory. Now type

```
mv third fourth
```

to rename the *third* directory to *fourth*. Use ls to check your work. You should now see *fourth, myfile,* and *wombat* listed.

The syntax of the command to remove files is “rm {*file_name*}+”. Now type

```
rm myfile
```

Use ls check your work. Your current directory, *first*, should now contain only one directory, *fourth*, and one file, *wombat*.

Command line autocomplete

The command line has an autocomplete function that is similar to the way autocomplete works with search engines. Type

```
cp w<tab>
```

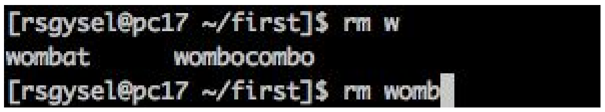
You will notice that the line will autocomplete to “cp wombat “. Type

```
wombocombo
```

so the command entered is “cp wombat wombocombo”. Use ls to notice that you have two files in this directory, *wombat* and *wombocombo*. Again, type (do not hit **`<enter>`** here)

```
cp w<tab>
```

and you should see the following:



```
[rsgysel@pc17 ~/first]$ rm w
wombat      wombocombo
[rsgysel@pc17 ~/first]$ rm womb
```

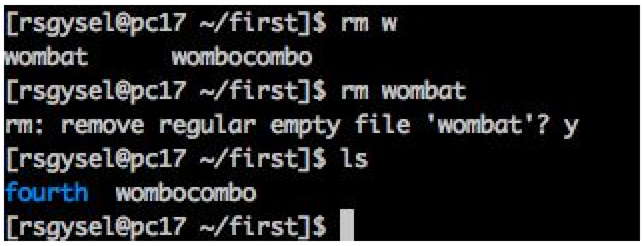
Notice the following:

1. Autocomplete doesn’t know if you want *wombat* or *wombocombo*
2. Autocomplete automatically extends your *w* to *womb,* since *womb* is a common prefix of both *wombat* and *wombocombo*

Typing

```
a<tab>
```

Then confirming the file removal and printing the directory contents results in:



```
[rsgysel@pc17 ~/first]$ rm w
wombat      wombocombo
[rsgysel@pc17 ~/first]$ rm wombat
rm: remove regular empty file 'wombat'? y
[rsgysel@pc17 ~/first]$ ls
fourth  wombocombo
[rsgysel@pc17 ~/first]$ 
```

Use autocomplete with **`<tab>`** *as much as possible*. If you are typing every character that appears on the command line, *you are probably doing it wrong.* Saving time in this way is an important skill for programmers to cultivate.

Displaying Files: cat, less, head

The syntax of the command to display a file is: "cat {*file_name}+"*. Now type

`cat wombocombo`

The contents of the file just flew by on the screen. For short files this might be fine, but usually you will wish to see the file a little at a time.

You can use "less {*file_name*}+" to scroll through a file. Try this with

`less wombocombo`

Use the up arrow and down arrow to scroll through the file.

Sometimes you will just wish to see the first few lines of a file. You can then use the head command. The syntax is "head [-n] {*file_namel}+"*, where "n" is the number of lines to be displayed, e.g. head -3 snake. If you leave out n, then head defaults to the first 10 lines. Try this with

`Head -10 wombocombo`

The Terminate Key : `<ctrl>-c`

<ctrl>-c usually causes instant termination of any running program. <ctrl>-c is also often written ^C in UNIX literature.

For example, the ping utility checks the quality of the network connection between your computer and a remote computer. It will continue to check the quality until you terminate it by typing ^C. The syntax of ping is "ping *computer_name".*

Now type

```
ping pc3
```

Your screen will start filling with lines indicating the quality of the connection to pc3.

Terminate ping by typing (<enter> is not necessary here)

```
<ctrl>c
```

ping should stop, and you should see the command prompt again. You can use ^C in most situations when a command or program is causing the computer to perform in an unsatisfactory manner. Often time students learning to code will accidentally write an *infinite loop*. When this happens, use <ctrl>-c to terminate your program.

Transferring Files: sftp

If you work on the CSIF, you will want to download your programs in order to submit it to Gradescope for grading. ***Do not copy and paste your program into a Word processor. Doing so will result in a file that can not be compiled.***

You can use the sftp (Secure File Transfer Protocol) command to retrieve or place files in remote locations, provided you have permission to access the location where the files are. WinSCP is a free Windows sftp program with a GUI. Mac OS

The syntax of sftp is: "sftp *computer_name*". As with ssh, you do not have to include the domain of the computer if you are ftping from a computer in the same domain. You are now going to go through the process of getting your wombat file from another computer.

Before continuing, move to your home directory, then login to a CSIF computer with

```
sftp computer_name
```

As in ssh, if you are trying to access a computer that is new to you, sftp will ask you to OK its authenticity; do so. You will be asked for your username. In this case you can simply hit enter, because the username it has detected is proper for the destination computer. If your account is under a different name, then you have to enter that name. Typically, many ftp sites let you access them under the user name anonymous, using your e-mail address as the password. In the CSIF you will still need to type in your CSIF password.

You can navigate in ftp using the cd, pwd, and ls. ftp also has a dir command which works like ls -l to show more information about files. Now type

```
pwd
```

and you will see that you are in your own home directory. Now type

```
cd first/fourth
```

to move the remote ftp session into the *fourth* directory on the CSIF computer. You can use lcd to change your local (on your computer) directory. Now type

```
lcd first
```

to move to your *first* directory on the computer in front of you. Now type

```
lpwd
```

to see where you are on the computer in front of you. Now type

```
ls
```

to see the files available to you on the remote computer. You should *wombocombo* listed. The "get" command retrieves files from the remote computer. Now type

```
get wombocombo
```

to retrieve *wombocombo* into your *first* directory.

If you had wanted to move multiple files you could use mget which has the syntax of “mget {*file_name*}+”. If you had wanted to transfer a file from the local computer to the remote computer you would use put, or mput. Now that you are done transferring files type bye to end the ftp session. To see if *wombocombo* is really in your *first* directory you can type ls first.

Text editors

Two popular command-line text editors available on the CSIF are *vim* and *emacs*. In another tutorial we will cover *vim*. You will be using a text editor to write your programs.

Logging Out: exit

You are done with the tutorial! Type

```
exit
```

to close the command line session. Then logout using the System drop down menu of the toolbar. Click the "Logout" or the "Continue logout" button to complete the logout process. Congratulations, you’re done!