

**UNLV**  
**School of Computer Science**  
**CS 135 Lab Manual**

**prepared by**

**Lee Misch**

**revised January 2009**

# CS 135 Lab Manual

Content	Page
Introduction.....	3
CS Computer Accounts.....	3
TBE B361 Computer Basics.....	4
Choosing an Operating System.....	4
Logging In to Window or Linux.....	4
Logging Out of Windows or Linux .....	4
Opening a Terminal Window in Linux .....	4
Changing CS Passwords.....	5
Changing EGR Passwords from TBE B361.....	5
Linux Primer.....	6
Conventions.....	6
Basic File Management Commands.....	6
Using Directories.....	10
Controlling Processes.....	12
Other Useful Information.....	15
Using the Emacs text editor.....	17
Compiling and Executing a C++ Program.....	19
Using make to Compile a C++ Program .....	19
Using SSH to Remotely Access School of Computer Science Servers.....	20
Using Alpine (Pine) to send/receive email.....	21
EGR email addresses and access.....	25
Separate Compilation of C++ Programs with Makefiles.....	25

## Quick Linux Command Lookup

Command	Page
cat – display file	8
cd – change directory	12
cp - copy	8
lpr - print	9
ls – display directory contents	7
man – online manual	6
mkdir – create a directory	12
more – display file	8
mv – rename file	9
passwd – change password	15
pwd – working path directory	11
rm – delete file	9
rmdir – delete directory	12

Please send any corrections and/or suggested additional topics to [lee@cs.unlv.edu](mailto:lee@cs.unlv.edu).

## Introduction

The following manual is an updated adaptation of several documents prepared by University and Community College System of Nevada System Computing Services, including "Managing Files Under Unix", "Using Directories in Unix", and "Running Processes Under Unix". Additional information has been adapted from the CS Lab website (<http://tux.cs.unlv.edu>) and handouts generated for CS 135/202 classes. Users of this manual should regularly check the CS lab website for updates.

## CS Computer Accounts

You may apply for a CS computer account in a variety of ways.

1. If you **do not** have a College of Engineering account and **do not** have a CS account, go to <http://accounts.egr.unlv.edu/> and complete the Online Account Application.

**Use only letters and digits in your desired login names.** Choose a name that you can live with; it will not be changed at a later date.

Make sure that you choose **CS-Computer Science** as your Department Affiliation/Major. Make sure you choose the appropriate Classification, do not leave this part of the form blank.

Documents with your login and 2 passwords (egr and cs) will be generated and can be picked up in TBE A-311 approximately 24 hours after your request is submitted. During fall and spring semesters, TBE A-311 is open every day from 8:00am to midnight. Bring a picture ID.

2. If you already have a College of Engineering account, but require a CS account, go to <http://tux.cs.unlv.edu/AccountApplication/>, provide your egr login name, and submit the form.

Read the information provided on the screen. If your request was denied follow the provided instructions. If your request was accepted, you can pick up your account information (login name and password) from TBE A-311. Bring a picture ID.

3. If you already have an Engineering and/or CS account, but cannot remember your login and/or password, or have any other problems creating a computer account, contact the School of Computer Science Systems Administrator, John Kowalski, in TBE B378E. Bring a picture ID.

## TBE B361 Computer Basics

The TBE B361 computer lab was created and is maintained for the use of undergraduate students taking computer science courses. The computers in the lab are *dual boot* machines, meaning they can run either Windows XP or Linux.

### Choosing an Operating System

If it is not already on, turn on the computer by pressing the power button.

1. A menu will be displayed on the screen offering the option of booting **Windows** or **Linux**.
2. Use the Up and Down arrow keys to select the operating system you want. Press Enter.

### Logging In

After selecting the operating system:

To Windows:

1. Press ***Ctrl-Alt-Del***
2. Enter your CS login (as User name) and CS password in the Log On to Windows screen
3. Click ***OK***

To Linux:

1. Enter your CS login in the Username space, press Enter
2. Enter your CS password, press Enter

### Logging Out

From Windows:

Select ***Log Off*** or ***Shutdown*** from the Windows Start menu.

From Linux:

Select ***Log Out*** or ***Shutdown*** from the fedora start menu (**f** icon in lower lefthand corner).

### Opening a Terminal Window in Linux

1. Bring up the fedora start menu (**f** icon in lower lefthand corner).
2. Move to System Tools and select Terminal from the menu.

## Changing CS Passwords

1. Log into **Windows XP**
2. Press **Ctrl-Alt-Del** to bring up the menu with the Change Password option
3. Select a password that conforms to the following rules
  - minimum length of 8 characters
  - must be a **MIX** of upper and lower case letters
  - at least one special character must be included (#s are considered special)
  - **DO NOT** use blanks, your login name, or dictionary words
4. You will be required to enter your new password twice before the change is complete

## Changing EGR Passwords from TBE B361

A College of Engineering computer account is not necessary to access the machines in TBE B361. However, if you want to access your College of Engineering email from the lab, you will need to know your egr account password. **It is recommended that you change your egr password to match your cs password.**

1. Log into your CS account (Windows or Linux)
2. Use ssh to connect to **student.egr.unlv.edu**  
**Linux**
  - open a new **Terminal** window (see page 4 for instructions)
  - at the prompt type: **ssh student.egr.unlv.edu** (press Enter)
  - if the system asks if you want to continue, type: **yes**
  - when prompted for your password, type in your **current egr** password

### Windows

- double click the **SSH Secure Shell Client** icon
  - click the **Quick Connection** button
  - type: **student.egr.unlv.edu** as Host Name
  - type: your login names as the User Name and click Connect
  - type: your **current egr** password
3. At the prompt, type in the command: **passwd**  
This is the Linux command to change a password. Follow the directions provided. Your old and new passwords will **NOT** be displayed as you enter them, so type carefully.
  4. To log out of student, type: **logout** or press **Ctrl-d**

# Linux Primer

## Conventions

A **system prompt** is a sequence of symbol(s) that are displayed by the operating system indicating that the system is ready to accept input.

A **command** is a directive to the computer to perform some task.

An **argument** is any string of characters added to a command to modify its results. A file name is one kind of argument. Individual letters preceded by a hyphen are another type.

Example:

```
[lee@bobby ~]$ cat -n testdata
```

In the given example, `[lee@bobby ~]$` is the system prompt, `cat` is the command, `testdata` and `-n` are arguments.

**Dummy words** are words/numbers that should be replaced by the user's own information. They are often used when presenting a general example of a command. Dummy words will be presented in *italic type* in examples.

Example:

```
[lee@bobby ~]$ more filename
```

In all examples, text entered by the user will be displayed in **boldface type**.

When entering a command, the user must always press the Enter key to complete the entry.

## Basic File Management Commands

A useful Linux command reference guide can be found at [http://tux.cs.unlv.edu/refs/linux\\_commands.html](http://tux.cs.unlv.edu/refs/linux_commands.html).

### **man command – on-line manual pages**

The **man** command provides the user with access to an on-line manual for Linux commands.

Example:

```
[lee@bobby ~]$ man command
```

## ls command – list

The **ls** command displays an alphabetical list of all the files in your current directory.

Example:

```
[lee@bobby ~]$ ls
```

Optionally, the user may specify a directory name:

```
[lee@bobby ~]$ ls directoryname
```

This command will display an alphabetical list of all the files in the specified directory.

Sample command and output:

```
[lee@bobby ~]$ ls cs135sum06  
a.out  data07  hw06.cpp  hw07.cpp  testdatadir
```

-l (long argument)

The **-l** argument of the **ls** command lists the files in the specified directory in long format. The long format displays the types of access, number of links, owner, size in bytes, and time of last modification for each file.

Sample command and output:

```
[lee@bobby ~]$ ls -l cs135sum06  
total 36  
-rwx----- 1 lee csfac 19447 Jun  1 12:14 a.out  
-rw----- 1 lee csfac   14 May 31 12:58 data07  
-rw----- 1 lee csfac  1648 May 30 13:28 hw06.cpp  
-rw----- 1 lee csfac  2576 Jun  1 12:14 hw07.cpp  
drwx----- 2 lee csfac  4096 Jun  2 12:46 testdatadir
```

The first column (the one containing 10 dashes and/or letters) indicates the type of access, or what you and other users can do to the file or directory.

The first character indicates whether the object is a file or a directory. A 'd' means directory; a hyphen (-) means file.

The next 3 characters refer to the owner's (user's) permissions to read (r), write (w), and execute (x) the file/directory. The r means that you can look at the file/directory. w means that you can write to or save in the file/directory. If the x is present it means that the file is executable or the directory can be searched. Generally, all the files/directories in your account will have rw permissions. Only directories and executable files will have the x permission set.

The next 6 characters refer to permissions given to the 2 remaining levels of file/directory ownership (group and other). Generally, these permissions should not be allowed (should show a hyphen).

The second column in the display shows the number of links to a file or directory.

The third column shows the owner of the file/directory. If you are listing the files in your home directory, your login should appear in this column. The fourth column shows your group.

The next 3 columns indicate the size of the file/directory in bytes, the date and time (using a 24-hour clock) when the file or directory was last modified, and the name of the file/directory, respectively.

### **cat command - concatenate**

The **cat** (concatenate) command allows the user to display the entire contents of a text file on the screen.

Example:

```
[lee@bobby ~]$ cat filename
```

### **more command**

The **more** command allows the user to display the contents of a text file one screen (or page) at a time.

Example:

```
[lee@bobby ~]$ more filename
```

The first page of the file will appear on the screen. To see one more line of the file, press the Enter key. To see the next page of the file, press the space bar. If you do not want to see the remainder of the file, quit the more command by entering the letter **q**.

### **cp command - copy**

The **cp** (copy) command is used to copy the contents of one file to another file.

Example:

```
[lee@bobby ~]$ cp sourcefile destinationfile
```

This command will copy the contents of *sourcefile* into *destinationfile*. Both files will exist in the current directory when the command is completed. **NOTE:** If a file with the name *destinationfile* exists BEFORE the **cp** command is executed, the old *destinationfile* will be replaced by the contents of *sourcefile*.

## **mv command - move**

The **mv** (move) command is used to move the contents of one file to a new file (rename a file). This command can also be used to change the name of a directory.

Example:

```
[lee@bobby ~]$ mv oldfile newfile
```

The name of *oldfile* is changed to *newfile*.

## **rm command - remove**

The **rm** (remove) command is used to permanently delete a file from your account.

Example:

```
[lee@bobby ~]$ rm filename
```

The file called *filename* will be deleted from the current directory.

## **lpr command – line print**

The **lpr** command is used to send a file to a printer.

Example:

```
[lee@bobby ~]$ lpr filename
```

The file called *filename* will be sent to the default printer for your computer.

**-P** (printer argument)

The **-P** argument of the **lpr** command allows you to specify the name of the printer to which the file should be sent.

Sample command:

```
[lee@bobby ~]$ lpr -Pponderosa mydata
```

This command will result in the file called *mydata* being sent to a printer called *ponderosa*.

## pquota – check print quota

The **pquota** command allows a user to check how many pages have been printed from his account. You may be required to enter your egr password to access this information. If your print job does not come through the printer, use this command to determine if you have exceeded your limit.

Example:

```
[lee@bobby ~]$ pquota
```

At the start of each semester, each CS student account is given a 50 page quota on ponderosa (the TBE B361 student printer). If you have exceeded your print quota for ponderosa, you may purchase additional pages (\$0.02 per page) from the CS systems administrator, John Kowalski (TBE B378E).

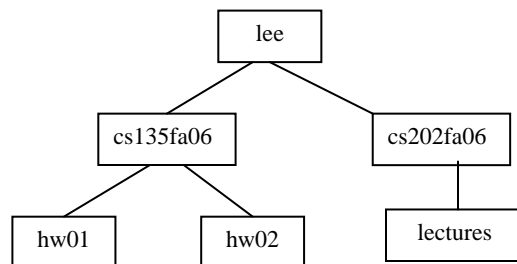
## Using Directories

### What is a directory?

A **directory** is a place to store files and other directories (like a Windows folder). Directories are used to organize the content of your account.

A directory created inside another directory is called a **subdirectory**. Forward slashes are used to separate subdirectory names (`/cs135sum07/hw01/`). A directory that contains a subdirectory is called a **parent directory**. Subdirectories may, in turn, be parents to other directories. A parent directory is symbolized by two periods (`..`).

Example of a directory structure:



In the example above, *lee* is the parent directory of *cs135fa06* and *cs202fa06*. *hw01* and *hw02* are subdirectories of *cs135fa06*. *cs202fa06* is the parent of *lectures*.

## Path Names

The root directory, designated by a forward slash (/), is the highest level in the system. Every directory starts from the root. The full name of a file beginning with a forward slash is called the **absolute path name** of the file. It specifies the location of the file starting with the root directory. Each successive subdirectory in the path must be preceded by a slash.

You may create and access files in your home directory. Your **home directory** is the directory created for your account. It is where you are placed when you log in to the computer. ~/ (tilde, slash) can be used as a shortcut that stands for your home directory.

A **relative path name** does not begin with a slash. It tells the computer to look for the specified file or directory relative to your current position in the directory structure.

Assume you are working in the *lee* directory in the sample directory structure shown above. The following command uses a relative path to display the content of the *hw02* directory.

```
[lee@bobby ~]$ ls cs135fa06/hw02
```

If you were working in the *cs135fa06* directory and wanted to display the contents of the *cs202fa06* directory (without changing directories), the command would be:

```
[lee@bobby ~]$ ls ../cs202fa06
```

The `..` refers to the parent directory of *cs135fa06*.

### **pwd command – print name of current/working directory**

The **pwd** command displays the absolute path to your current directory.

For instance, if you were working in the *hw01* directory in the sample directory structure and invoked the `pwd` command, the response will be,

```
[lee@bobby ~]$ pwd
/lee/cs135fa06/hw01
```

## **cd command – change directory**

The **cd** command is used to change from one directory to another.

Example:

```
[lee@bobby ~]$ cd directorypath
```

This command changes the current working directory to the specified directory.

To return to your home directory, type:     **cd**   or   **cd ~**

To move one directory up (to parent directory), type:     **cd ..**

## **mkdir command – make a directory**

The **mkdir** command is used to create a new directory. The specified directory will be created as a subdirectory of your current working directory.

Example:

```
[lee@bobby: ~]$ mkdir dirname
```

## **rmdir command – remove a directory**

The **rmdir** command is used to remove a directory. Before a directory can be deleted, it must be empty (no files or subdirectories can be in the directory).

Example:

```
[lee@bobby: ~]$ rmdir dirname
```

## **Controlling Processes**

The following information on canceling processes can also be found online at:  
<http://tux.cs.unlv.edu/processControl.html>.

### **Canceling a Process**

If a program is caught in an infinite loop or must be terminated early for some reason, *Ctrl-C* can be used to terminate the process.

Sometimes a process may "run away" (not terminate properly). The following command checks the processes that are currently running for a user with the login name janedoe.

Sample command and output:

```
[lee@bobby ~]$ ps -ef | grep janedoe
UID          PID  PPID  C  STIME TTY          TIME CMD
janedoe     2233   2231  0  16:57 ?           00:00:00 /usr/sbin/sshd
janedoe     2234   2233  0  16:57 pts/0       00:00:00 -sh
janedoe     2254   2234  0  16:57 pts/0       00:00:00 ps -ef
janedoe     2255   2234  0  16:57 pts/0       00:00:00 grep janedo
```

The numbers in the PID column are the process identification numbers for the jobs. If you want to terminate one of these jobs, use the kill command.

Example:

```
[lee@bobby ~]$ kill -9 PID
```

Replace PID with the actual process ID number for the job to be terminated.

Example:

```
[lee@bobby ~]$ kill -9 2233
```

This command will terminate the command "/user/sbin/sshd"

## Redirecting Input and Output

By default, the **standard input** device is the keyboard and the **standard output** device is the screen (monitor). Linux allows the user to change the standard input and output destinations through a process called **redirection**.

### Output Redirection (>)

The greater than sign (>) is used to change the destination of standard output.

Example:

```
[lee@bobby ~]$ ls > dirlist
```

This command will place the output from the ls command into a file called "dirlist" rather than to the screen. If the file "dirlist" existed prior to the command, its old content would have been destroyed.

Example:

```
[lee@bobby ~]$ ./a.out > output
```

This command will send the results of executing the object file "a.out" to the file called "output".

## Append Output Redirection (>>)

Two greater than signs (>>) allows you to add the output of a program or command to the end of an existing file rather than destroying its contents.

Example:

```
[lee@bobby ~]$ cat feline >> canine
```

This command will add the contents of "feline" to the end of "canine".

## Output Error Redirection (>&)

If you want to include any error messages that might appear along with the results of your process, add an ampersand (&) after the greater than sign.

For example, to save the error messages from a g++ compile to a file called "errorlist" the command would be

```
[lee@bobby ~]$ g++ prog.cpp >& errorlist
```

## Input Redirection (<)

The less than sign (<) is used to change the standard input device to a specified file.

Example:

```
[lee@bobby ~]$ ./a.out < inputfile
```

The command in the example runs the executable file "a.out" which will try to get its input from the file "inputfile".

Input and output redirection can be used in the same command line.

Example:

```
[lee@bobby ~]$ ./a.out < testdata > myresults
```

Here the program stored in "a.out" is run, getting its input from "testdata" and writing its output to "myresults".

## Pipe Redirection (|)

The vertical line (|) is used to send, or pipe, the output from one command or program as input to another command or program.

For example, the command

```
[lee@bobby ~]$ ls | more
```

will cause the computer to list the files and directories in your current directory, one screen (page) at a time.

## Other Useful Information

### **passwd command – change password**

The **passwd** command is used to change a user's authentication tokens. This command can be used to change College of Engineering computer account passwords.

**NOTE: THIS COMMAND WILL NOT CHANGE YOUR CS ACCOUNT PASSWORD. TO CHANGE THAT PASSWORD YOU MUST LOG INTO WINDOWS.** (See the Change CS Password section of this manual.)

The command will prompt the user for his/her current password, followed by prompts for a new password and re-entry of the new password. **Note: passwords will not be displayed on the screen for security reasons. Type carefully.** The “unknown password response” message will be displayed after a pause for the system to reset your password. It is not an error message. Any other message displayed by the system indicates a problem in changing your password and you should try the **passwd** command again.

Sample session:

```
[lee@faculty:]% passwd  
Changing password for lee.  
Enter login(LDAP) password:  
New password:  
Re-enter new password:  
Unknown password response (26469).
```

## Wildcard Characters

Wildcard characters can be used to get a quick list of files and directories with related spellings. The two characters that act as abbreviations (wildcards) in names are the question mark (?) and the asterisk (\*).

A question mark in a name matches any single character. For example, if the command

```
[lee@bobby: ~]$ ls testdata??
```

is typed, the computer will search for all names that begin with "testdata" and end with exactly 2 characters. So, objects with the names testdata10, testdataxy, and testdata.1 would be displayed, but testdata1 and testdata004 would not.

The asterisk matches zero or more characters. If the command

```
[lee@bobby: ~]$ ls testdata*
```

is typed, any name that begins with "testdata" will be displayed.

Wildcard characters can be used in any part of a name (beginning, middle, or end).

## Using Completion in the `bash` Shell

The `bash` shell provides users with the *command argument completion* feature. This feature allows a user to type in a partial command then press the **Tab** key to complete the command. Completion works best when there is exactly one possible match for the partial command that is typed in. If there are several possible matches, the command will be partially completed, allowing the user to type in the remainder of the command.

For example, if you have files called **assign1.cpp** and **hw1testdata** in your current directory and type in the following partial command and then press **Tab**:

```
[lee@bobby: ~]$ more as
```

the `bash` shell will automatically supply the remaining characters in the file name (**sign1.cpp**).

On the other hand, if you have files called **assign1.cpp** and **assign1testdata** in your current directory and type in the following partial command and then press **Tab**:

```
[lee@bobby: ~]$ more as
```

The bash shell will only partially complete the command as shown below. It is up to the user to supply more characters in the command, because there are 2 files that have the same set of starting characters.

```
[lee@bobby: ~]$ more assign1
```

If the user then added a ‘t’ to the command shown above and pressed **Tab**, the command would be completed as:

```
[lee@bobby: ~]$ more assign1testdata
```

### Accessing Prior Commands Using Up and Down Arrow Keys

The bash shell has a built-in command history that “remembers” prior commands issued. When editing, compiling, and running a program you will find yourself issuing the same commands repeatedly. To avoid the necessity of retyping commands each time, press the **Up Arrow** key to access a prior command. The **Up** and **Down Arrow** keys allow you to scroll through recently issued commands.

### Using the Emacs text editor

Emacs is a program designed to allow users to create, edit, and save text files. It can be used to create your program and test data files. It is recommended that you go to the SCS Computer Lab website ([tux.cs.unlv.edu](http://tux.cs.unlv.edu)) and print a copy of the GNU Emacs Reference Card.

If you are logged into Linux in the CS lab (TBE B361), you will be able to use the GUI provided for invoking commands. If you connect to a CS computer (bobby or cardiac) through SSH, you will have to enter commands via the keyboard (do NOT use the mouse).

Once you have logged into your CS account, to invoke emacs and begin editing a C++ program file, type the command:

```
[lee@bobby: ~]$ emacs aprog.cpp
```

If a file called **aprog.cpp** does not exist in your current directory, it will be created. If the file already exists, it will be opened for editing. NOTE: only include the .cpp extension for files that will contain C++ programs. Choose file names that are meaningful.

Once **aprog.cpp** is open for editing, you may move the cursor (with the mouse if in a graphical environment, with the cursor movement keys if using SSH) and begin typing. Refer to the Emacs Reference Card for specific commands.

**Emacs Commands:** On the Emacs Reference Card, commands are designated as C-letter and M-letter. The C stands for the Ctrl key. To enter a C-letter command, press the Ctrl key and the specified letter at the same time. The M stands for the Meta key. In Linux the Meta key is Esc. To enter an M-letter command, press the Esc key, release it, then press the specified letter.

For example:

**C-x C-w** (Ctrl-x Ctrl-w) will allow you to write to a specific file  
**M-d** (Esc d) will delete the word following the cursor

When you connect to the system using SSH, you may find that the Backspace key does not work as expected in Emacs (the Delete key sends a backspace). If so, you can change how the keyboard handles input (see the section called **Reconfiguring How the Terminal Handles Keyboard Input**).

**Error Recovery.** To abort a partially typed or executing command type: **C-g**.

**Saving a file.** To save your current file, the command is **C-x C-s**.

**Exiting Emacs.** To exit emacs, the command is **C-x C-c**.

NOTE: When you exit emacs, it will automatically save a copy of your old file (create a backup). The old file will be called **filename~**

For example:

```
[lee@bobby sampledir]$ ls
aprog.cpp  aprog.cpp~  testdata  testdata~
```

**aprog.cpp** and **testdata** are the most recently edited versions of a C++ program file and a data file, respectively. **aprog.cpp~** and **testdata~** are the files before the last edit/save were performed.

## Compiling and Executing a C++ Program

The C++ compiler that will be used when evaluating programming assignments in CS 135 is called **g++**. In order for a file to be recognized by the g++ compiler as a C++ program file, the name of the file must have an appropriate extension. The extension to be used for program file names in CS 135 is **.cpp**.

Command to invoke the g++ compiler:

```
[lee@bobby: ~] g++ progname.cpp
```

If the program file contains no syntax errors, an executable file with the name **a.out** will be created. This file should not be displayed as it is not in a human readable form. The program can now be executed with the following command.

Command to run your program:

```
[lee@bobby: ~] ./a.out
```

If the program contained syntax errors, a series of error messages will be displayed. You must take note of the lines at which the errors occurred and then go back to the original program file (the .cpp file) to locate and correct the mistakes. Save your updated file. Then, recompile.

## Using make to Compile a C++ Program

An alternative method for compiling a C++ program is to use the Linux make utility. Using make to compile a program stored in the file **program.cpp**, will automatically invoke the g++ compiler using the **-o** option and create an executable file called **program**.

The command to compile and create the executable is:

```
[lee@bobby: ~] make program
```

Note that the .cpp extension should not be included in this command even though the name of the program file is program.cpp.

In order to execute the program, type the command:

```
[lee@bobby: ~] ./program
```

## Using SSH to Remotely Access School of Computer Science Servers

**bobby.cs.unlv.edu** is a Linux general purpose login machine that is available to provide remote access to SCS computing resources for students currently enrolled in SCS courses. **cardiac.cs.unlv.edu** is also available for remote login. Although network and OS students **must** use **cardiac** for remote access, CS 135 students may also log into this machine.

In order to access **bobby** or **cardiac** from home you will need to download and install the **Secure Shell Client (SSH)** software.

1. Go to the SCS lab website (<http://tux.cs.unlv.edu>). Locate the User's Guide and then follow the Remote Access link.
2. Locate the link to the Secure Shell Client website and download the free, non-commercial version of the program.
3. Install it on your computer.
4. Follow the instructions provided for using the SSH client graphically.

### Creating a Profile

SSH allows users to create profiles so that different servers can be quickly accessed.

1. Locate the Secure Shell Client icon on your desktop and double click it.
2. Click on the Profiles button (located on the upper left side of the window).
3. Select the Add Profile option. Type in a name for the server to be accessed. For example, bobby.
4. Click the Profiles button again and select the Edit Profile option. Locate the server name just added from the list of profiles on the left. Type in the **complete server name** (bobby.cs.unlv.edu) as Host and your **CS login name** as User. Do not change any other information. Click OK

### Reconfiguring How the Terminal Handles Keyboard Input

By default, emacs uses the Delete key to send a backspace. To change this when using SSH:

1. Run the SSH client.
2. Click on the Profiles button and select Edit Profile.
3. Select the server to be reconfigured.
4. Select the Keyboard tab.
5. Click the Backspace sends Delete option. (The Delete sends Backspace option can also be checked.)
6. Click OK.

## Using Alpine (Pine) to Send/Receive Email

**Important note:** Pine is the previous name of the University of Washington email program. While the systems in the CS lab (TBE B361) are being updated some machines may still be running pine. The information provided in the following description applies to both Alpine and Pine. If using a computer that has not been updated, substitute the name “pine” for all references to “alpine”.

Alpine is a program designed to provide novice users access to sending, receiving, and organizing email.

### Starting alpine

To start alpine log into your account (Linux) and open a new terminal window (right click mouse and select New Terminal or from the start menu select System Tools and Terminal). Then, at the command prompt type: **alpine**

A display similar to the one shown below will be generated.

```
ALPINE 4.63      MAIN MENU                               Folder: (CLOSED)  No Msgs +
?      HELP                - Get help using Pine
C      COMPOSE MESSAGE     - Compose and send a message
I      MESSAGE INDEX       - View messages in current folder
L      FOLDER LIST         - Select a folder to view
A      ADDRESS BOOK        - Update address book
S      SETUP               - Configure Pine Options
Q      QUIT                - Leave the Pine program
```

```
Copyright 1989-2005. PINE is a trademark of the University of Washington.
HOST: postmaster-2.egr.unlv.edu  ENTER LOGIN NAME [lee] :
^G Help
^C Cancel      Ret Accept
```

### Logging into alpine

The cursor will be located at the bottom of the screen following the message

```
HOST: postmaster-2.egr.unlv.edu  ENTER LOGIN NAME [yourlogin] :
```

Press the enter key and the following message will be displayed

```
HOST: postmaster-2.egr.unlv.edu  USER: yourlogin  ENTER PASSWORD:
```

Type in your **engineering account password** and press the enter key. Your password will not be displayed to the screen for security purposes, so type carefully.

## Alpine main menu

Alpine does not have a graphical user interface. **DO NOT use the mouse** to select options. Use the cursor movement keys (the arrow keys) to move the selection bar or enter the specific character(s) to issue a command through the keyboard. You can always return to the main menu by typing the letter M (or m) unless you are composing a message. Commands in alpine are **not case sensitive**.

```
ALPINE 4.63      MAIN MENU                               Folder: INBOX  981 Messages  +
?  HELP          -  Get help using Pine
C  COMPOSE MESSAGE -  Compose and send a message
I  MESSAGE INDEX -  View messages in current folder
L  FOLDER LIST   -  Select a folder to view
A  ADDRESS BOOK  -  Update address book
S  SETUP         -  Configure Pine Options
Q  QUIT          -  Leave the Pine program

Copyright 1989-2005.  PINE is a trademark of the University of Washington.
                        [Folder "INBOX" opened with 981 messages]
? Help                P PrevCmd                R RelNotes
O OTHER CMDS > [ListFldrs] N NextCmd            K KBlock
```

## Alpine command menu

The alpine command menu is always displayed at the bottom of the screen. A brief description of each command is preceded by the character(s) needed to run the command. For example, in the main menu display shown above, if the character O is entered by the user, other available commands will be displayed.

A command may be displayed as the caret (^) symbol followed by a letter as in the following example:

```
^C Cancel      Y [Yes]
               N No
```

The ^ represents the control (Ctrl) key. To enter the command, press the Ctrl key and the specified letter key at the same time.

## Composing a new email message

To start composing a message, either type the letter C or (if in the main menu) move the selection bar to the Compose Message option and press enter. If you have previously postponed messages, you may be asked if you want to continue that message. Answer no.

A new message template will be displayed.

```
ALPINE 4.63      COMPOSE MESSAGE                Folder: INBOX  981 Messages  +
To      :
Cc      :
Attchmnt:
Subject :
----- Message Text -----
```

--

```
^G Get Help   ^X Send       ^R Rich Hdr   ^Y PrvPg/Top ^K Cut Line   ^O Postpone
^C Cancel     ^D Del Char   ^J Attach     ^V NxtPg/End ^U UnDel Line ^T To AddrBk
```

Enter the email address(es) of the people you are sending a message to on the To: line. Separate multiple addresses with commas. If the recipient has an account on the engineering college network, you need only enter his/her login name ( the domain name, @cs.unlv.edu, is not necessary).

Alpine automatically will send a copy of all messages you send to a folder called **sent-mail**, so you do not have to put your address on the Cc: line to send yourself a copy.

**Do NOT send your CS 135 program files as attachments.**

The Subject: line should be used to provide a brief description of your message. **For CS 135 assignments, always put your name, section #, and assignment # on the subject line of your message.**

Once the message heading is complete, move the cursor down to the Message Text area of the screen and enter your message. If you are submitting a CS 135 programming assignment, use the **^R Read File** command. You will see the following prompt at the bottom of the screen.

```
File to insert from home directory:
```

Type in the name of your program file and press the enter key. The content of your file should appear in the message area. **Do NOT use copy-and-paste to insert your program.**

To send your message, use the **^x** command. When asked if you want to send the message, enter Y.

## Reading email

To read your email, go to the Folder List option from the main menu or type in the letter L. This command will take you to your Collection List (as shown below).

```
ALPINE 4.63    COLLECTION LIST                               Folder: INBOX  981 Messages  +
Maildirs
  Folders on mail.egr.unlv.edu/novalidate-cert in INBOX.
Mail on mail.egr.unlv.edu/novalidate-cert
  Folders on mail.egr.unlv.edu/novalidate-cert in INBOX.
Mail
  Local folders in mail/
```

The selection bar should be on Maildirs. Press the enter key. Your Folder List will be displayed (see sample below).

```
ALPINE 4.63    FOLDER LIST                               Folder: INBOX  981 Messages  +
-----
  Folders on mail.egr.unlv.edu/novalidate-cert in INBOX.
-----
INBOX          sent-mail[.]          saved-messages[.]
Drafts[.]      Junk[.]              Sent[.]
Trash

? Help        < ClctnList    P PrevFldr      - PrevPage A Add      R Rename
O OTHER CMDS > [View Fldr] N NextFldr    Spc NextPage D Delete W WhereIs
```

All new email will be located in INBOX. You may select a folder by using the cursor movement keys.

To look inside a folder, move the selection bar to the desired folder and press the enter key. You will see a message index (see below).

```
ALPINE 4.63    MESSAGE INDEX                               Folder: INBOX  Message 969 of 970  +
963 Aug  7 christine.wallace@ (1.9M) College calendar
964 Aug  8 Mario.Martin@unlv. (3091) CS 115 Instructor's reference Guide
965 Aug  8 UNLV_Today/UNLV@un (3695) News from UNLV Today
966 Aug  9 Fitch, Katie (13K) RE: CS 115 Instructor's reference Guid
967 Aug 12 william.culbreth@u (143K) Fw: Science and Engineering; Grants Re
968 Aug 12 Mario.Martin@unlv. (16K) Just a reminder
+ 969 Aug 13 swetha_r@egr.unlv. (1489) Fall 20008
N 970 Aug 13 christine.wallace@ (21K) Weekly Update and calendar for our "a

? Help        < FldrList    P PrevMsg      - PrevPage D Delete    R Reply
O OTHER CMDS > [ViewMsg] N NextMsg    Spc NextPage U Undelete F Forward
```

Move the selection bar to the message you wish to read and press enter. < or the letter I will return you to the index.

An N to the left of the message means that it is new and has not been opened for reading.

## Managing your email

**Save** - To save an email message use the save (S) command. By default, alpine will save to the **saved-messages** folder and mark the message for deletion (a D will appear to the left of the message). If you want to organize your saved email, you can specify the name of a folder to save to. If the folder does not already exist, alpine will ask you if you want to create it.

**Delete** – To mark a message for deletion use the delete (D) command. alpine will not actually remove the message unless you expunge it upon quitting the program.

## Quitting alpine

To exit alpine, go to the main menu and select Quit or type the letter Q. You will be asked if you really want to quit alpine. if any messages were marked for deletion, you will be asked if you want to expunge the deleted messages. If you answer yes, the messages will be permanently removed. If you answer no, you will exit alpine, but the messages marked for deletion will remain in the INBOX folder.

## EGR email addresses and access

Creation of an EGR computer account provides you with an EGR email address. Your EGR address will be [yourlogin@egr.unlv.edu](mailto:yourlogin@egr.unlv.edu). Mail sent to [yourlogin@cs.unlv.edu](mailto:yourlogin@cs.unlv.edu) will automatically be sent to your EGR address.

To access your EGR email you **MUST** know your login name and your **EGR** password.

- to access your EGR email on the Web, go to <http://mail.egr.unlv.edu>
- to access your EGR email through the CS network (TBE B361, bobby or cardiac) use alpine

## Separate Compilation of C++ Programs with Makefiles\*\*

In C++ (and many other programming languages) a project may be composed of more than one source file. The Unix/Linux **make** utility provides an efficient method for specifying the dependency relationships between a set of files. By creating a text file called Makefile (or makefile), a programmer can list the commands required to form an executable file from the source files. The **make** program will automatically keep track of source files that have changed and recompile them if necessary.

The syntax for invoking **make** is

```
[lee@bobby: ~]$ make prog
```

where prog is the name of the executable file you want to create. **make** will perform the commands specified in Makefile (makefile).

A Makefile consists of a series of entries. Each entry consists of a line containing a colon (a dependency line) and one or more command lines that start with a tab. The dependency line begins with a target (usually a file to be created), followed by a colon, and then a list of the files that are required to generate the target. The command line **MUST** be tab-indented and shows how to build the target from the dependent files. A pound sign (#) is used to insert comments. All text following the # on a line will be ignored by **make**.

For example here is a Makefile for a complex number program. We will assume you have 3 files: testcomplex.cpp (a C++ client program designed to test a complex numbers package), complexImp.cpp (a file that contains the functions implementing the complex number data type), and complex.h (a header file that contains the type declarations and function prototypes for the complex number data type).

```
# testcomplex is dependent on testcomplex.o & complexImp.o
testcomplex : testcomplex.o complexImp.o
    g++ -o testcomplex testcomplex.o complexImp.o
testcomplex.o : testcomplex.cpp
    g++ -c testcomplex.cpp # -c - don't run the linker
complexImp.o : complexImp.cpp
    g++ -c complexImp.cpp
clean : # remove unnecessary object files
    rm *.o
```

To generate the executable file testcomplex, the command is:

```
[lee@bobby: ~]$ make testcomplex
```

**\*\* DO NOT use separate program files for programming assignments given in CS 135.**

### What Happens

When the command: make testcomplex is invoked the default descriptor file (Makefile or makefile) is used and the target "testcomplex" built. If the necessary object files do not already exist, **make** will perform the commands specified in the descriptor file to generate the target. If no target is specified in the call to **make**, the first target is made.

You may also select specific targets to be created. For example, if you wanted to create complexImp.o, **make** could be invoked with: make complexImp.o.

Not all targets need be files. In the example above, clean is a phony target. The command: make clean will cause all .o files in the current directory to be removed.

Additional information can be found in "An Introduction to the Unix Make Utility" (<http://www.mtsu.edu/~csdept/FacilitiesAndResources/make.htm>).

## **g++ Compiler Options Used** - from g++ man pages

- **-c**  
Compile or assemble the source files, but do not link. The linking stage simply is not done. The ultimate output is in the form of an object file for each source file.

By default, the object file name for a source file is made by replacing the suffix `.c`, `.i`, `.s`, etc., with `.o`.

Unrecognized input files, not requiring compilation or assembly, are ignored.

- **-o afile**  
Place output in the file called afile. This applies regardless of whatever sort of output is being produced, whether it be an executable file, an object file, an assembler file or preprocessed C code.

Since only one output file can be specified, it does not make sense to use `-o` when compiling more than one input file, unless you are producing an executable file as output.

If `-o` is not specified, the default is to put an executable file in `a.out`, the object file for `source.suffix` in `source.o`, its assembler file in `source.s`, a precompiled header file in `source.suffix.gch`, and all preprocessed C source on standard output.