
Achtung, Atención, Attenzione, Oppmerksomhet, Attention

You **must** have Cygwin or Linux installed on your laptop before attempting this lab.

1 Introduction

The operating system of a computer is the coordinator of all the computer's activities, including memory allocation, process switching, and file access. At Wake Forest University students use the Windows operating system on their laptops. In CSC 112 you will use two versions of Unix: Linux on **your** laptop and Solaris on the Sun workstations. For this reason, this lab will introduce you to these environments for C++ programming. Specifically this lab covers,

1. Login to Linux and change your password.
2. Introduce the Linux environment: file structure and commands.
3. Create the directory structure required for this course.
4. Use Unix commands to copy/compile/execute an existing C++ program. (graded)
5. Write/compile/execute a C++ program. (description given at the end of this assignment). (graded)

1.1 A Brief History of Unix

In 1969, Ken Thompson developed the Unix operating system at Bell Laboratories. Initially Unix was only a tool for AT&T programmers; however, today many different "flavors" of Unix exist. These variations range from Solaris (which is on our Sun workstations) to Linux (version for Intel based computers). The Unix operating system was designed as a multi-platform system, which can run on computers with different central processing units. It has evolved into one of the most powerful multitasking operating systems in existence, especially for heterogeneous, networked systems of computers. The Unix operating system allows multiple users to access and share time on a single CPU. More than one process can be executed concurrently and each process communicating with others when necessary. Furthermore, computers at different locations can communicate with each other, sharing information and application programs.

2 Start Cygwin

Assuming you have created your home directory (see section 2 of lab 0 for instructions) you will issue the following commands every time you start Cygwin for CSC 112.

1. Move your mouse to the XP task bar and select `Start` → `Programs` → `Cygwin` → `Cygwin Bash Shell` starts the Cygwin bash shell window.
2. Enter the command `startx &` in the bash shell window, starts an `xterm`.
3. Once an `xterm` appears, enter the command `cd /home/userName` at the prompt.

2.1 Unix Commands

The standard format for Unix commands is

commandName options path1 path2

Where *commandName* is the command, *path1* is the source, *path2* is the destination, and *options* are the command options. Furthermore, Unix commands (as well as file and directory names) are **case sensitive**; therefore, entering `PWD` at the prompt will not return the current directory. The following is a list of important Unix commands you must know.

Command	Description
<code>pwd</code>	Returns the path name from root to the current working directory. This is called the <i>full path name</i> .
<code>ls</code>	Lists the current directory. There is a variety of command options for <code>ls</code> . For example, <code>ls -al</code> provides detailed information about each file in the current directory.
<code>mkdir <i>directoryName</i></code>	Creates a directory called <i>directoryName</i> .
<code>rmdir <i>directoryName</i></code>	Removes the directory called <i>directoryName</i> . Note the directory must be empty.
<code>cd <i>pathName</i></code>	Change directory to <i>pathName</i> . Note that entering <code>cd</code> without a <i>pathName</i> will return you to your home directory. Furthermore, <code>cd ..</code> moves back one directory.
<code>cp <i>fileName1 fileName2</i></code>	Copy file <i>fileName1</i> to file <i>fileName2</i> .
<code>rm <i>fileName</i></code>	Erase file <i>fileName</i> . Once erased you can not get the file back.
<code>less <i>fileName</i></code>	View the contents of the file called <i>fileName</i> (if it is ASCII).
<code>man <i>commandName</i></code>	Provides a description of the Unix command <i>commandName</i> including syntax and options.

3 Creating Your CSC 112 Directory Structure and Compiling an Existing C++ Program (Graded)

For this portion of the lab you will create a directory, copy a C++ program into this directory, compile the program, then execute the program. For this class (CSC112), all programming assignments must reside in certain directories for grading. This portion of the assignment will cover how these directories are created using the commands in the previous lab.

First, make certain you are currently in your home directory. Issue a `pwd` command.

```

Terminal
userName@userName-2007 ~
$ pwd
/home/userName

```

Now create a new directory called `Lab1` off your home directory. Issue the following commands to accomplish this.

```

Terminal
userName@userName-2007 ~
$ mkdir Lab1

```

Now list your home directory. You should see the directory you just created (among other files in your home directory).

```

Terminal
userName@userName-2007 ~
$ ls
Lab1/

```

For this lab all the program files will be contained in the `Lab1` directory.

3.1 Copy and Compile

A C++ program (source) file called `hello.cpp` is located at course web page. To obtain the file, use firefox for a web browser (type `firefox` at the prompt to start). Copy this program to your `Lab1` directory. After you have copied the file, compile and execute the program.

3.1.1 Gnu Compiler

We will use the gnu C++ compiler for this course. The command to compile a program is

```
g++ -o executableName.exe sourceName
```

Where *sourceName* is the C++ program (source) file and *executableName.exe* is the resulting executable file. For the `hello.cpp` compile the program so the resulting executable is called `hello`. The g++ compiler has more options, which will be introduced as they are needed. To execute the program enter the executable name at the prompt. **Keep these two files (source and executable) in your Lab1 directory, their existence (in your Lab1 directory) will give you credit for this portion of the lab.**

The path or file name for any command will be relative to the current working directory. For that reason, it is not always necessary to supply the full path name. For example using figure ??, suppose you are in the directory `usr` and want to move to `include`. You could enter `cd include`, instead of supplying the full pathname `cd /usr/include`

4 Program Assignment (Graded)

In this portion of the assignment you will write your own C++ program. To accomplish this you will need to start a text editor. Our Unix environment has a few text editors to choose from.

- `nedit` - Text editor with a GUI interface and several programming features.
- `xemacs` - Text editor with a GUI interface.
- `gvim` - King of all editors, but very difficult to learn. A right-of-passage for any CS major.

To start `nedit` enter `nedit &` at the Unix prompt. The `&` sign tells Unix to run the command in the background, freeing the terminal. If you have never heard of `gvim` then **don't** try it now; come by during office hours.

4.1 Program: Light Bulb Packaging

For this portion of the lab, create a new file called `lab1.cpp` in your `Lab1` subdirectory. Then write a program that solves the following problem. Be certain you follow the programming points at the end of this section.

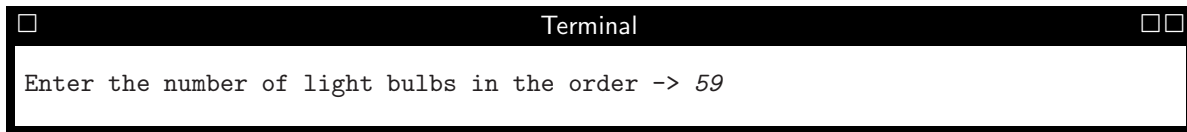
The Pluf Light Bulb Company is a world leader in manufacturing one watt light bulbs. These fragile light bulbs must be packaged in special containers, where each container must be filled to capacity (reduced breakage). The four different types of containers and their costs are listed in the following table.

Name	Capacity (bulbs)	Cost (\$)
huge	20	5.00
large	15	3.50
medium	7	2.00
small	1	0.75

The packaging manager needs a program that calculates the minimum number of containers required for any size order (you may assume integer input greater than zero) and total cost (bulbs and packaging). Assume bulbs are \$1.55 each.

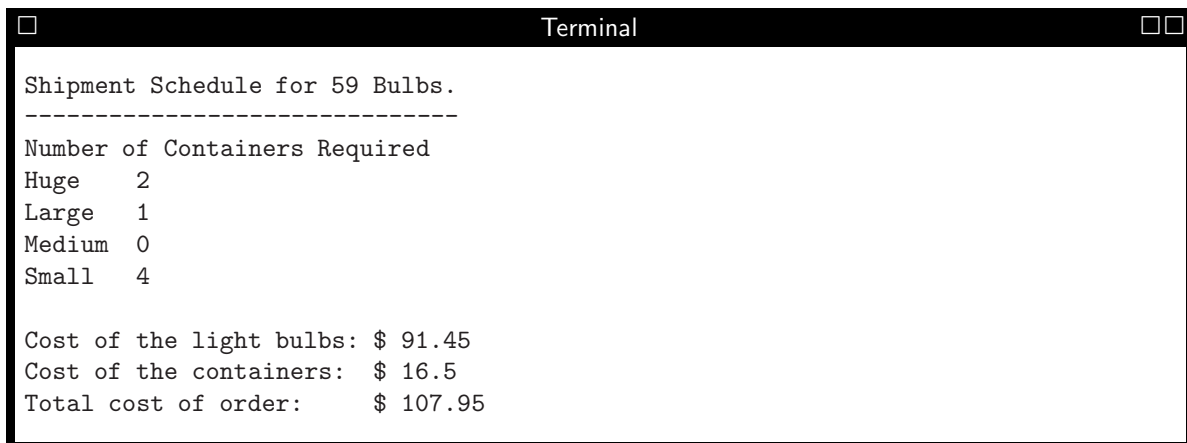
4.2 Program Description

First prompt the user for the number of light bulbs in the order. Below, the user input is 59.



```
Terminal
Enter the number of light bulbs in the order -> 59
```

After reading the number of bulbs in the order, calculate and display the minimum number of containers required, the cost of the bulbs, the cost of the packaging, and the total cost of the order. For 59 bulbs the output would be



```
Terminal
Shipment Schedule for 59 Bulbs.
-----
Number of Containers Required
Huge      2
Large     1
Medium    0
Small     4

Cost of the light bulbs: $ 91.45
Cost of the containers: $ 16.5
Total cost of order:      $ 107.95
```

4.3 Programming Points

You must turn-in printed and electronic copies of your programs. This must be done before the assignment due date. Printed copies must be given to Dr. Fulp while electronic copies will be submitted using the `blackboard.wfu.edu` web site. You **must** adhere to all of the following points to receive credit for this program.

1. Name your source (C++ program) file `lab1.cpp` This file must be located in your `Lab1` directory.
2. The program must have at least four functions:
 - `main`
 - `inputOrder` - Get the order information
 - `calcOrder` - Calculate order shipping schedule and cost.
 - `outputOrder` - Print shipment schedule and cost.
3. **Selection and repetition structures are not allowed.**
4. All prompts and output must appear as described in the previous section. Please note the example output carefully.
5. All code **must** follow the style guidelines for this course and be documented.

5 Electronically Submitting Your Program

You will use BlackBoard to electronically turn-in programs. Go to the `blackboard.wfu.edu` web-site and register for the CSC 112 Section A Fall 2007 course. Once registered, you can turn-in your programs electronically using this site. More information about BlackBoard is available at

<http://www.wfu.edu/Library/ITC/training/blackboard/>