

CSE 459.22 – Programming in C++

Lab 1

Assigned: 30th Mar, 2006

Due: 12rd Apr, 2006 11:59 PM

Late Penalty: 20 points per day

Goal

- To practice editing and compiling a C/C++ program in the UNIX environment
- To refresh your mind about C programming

Points

This lab is worth 100 points

Instructions

1. INTRODUCTION AND SETUP

- **Login**
 - Press any key on the keyboard and wait for the login prompt.
 - When you see the "login:" prompt, type in your username (usually your last name) and hit return.
 - At the "password:" prompt, type in the last four digits of your social security number, followed by your first and last initial in lower case. For example, "5670bd". This is your default password.

NOTE If you have any problems with login, please contact the help desk at 2-6542 or DL 894 or help@cse.ohio-state.edu

- **Enter user name and password.**
 - On your first login, the system will prompt you to change your password.
 - In future, you can use the *passwd* command to change your password. Make sure that you remember your new password.
- **Open an xterm Window in the UNIX environment, and you will see the following prompt** `/~~~~~/username/ >`
Now, you are ready to play with the Unix system!

2. REFRESHER

- Remember to click on the title bar to activate the window you want to work in.
- Remember, this system is case-sensitive.
- Recommend you to create a new directory with name “cse459”, which can help to organize all your files for this class in one place.

mkdir cse459 <return>

cd cse459 <return>

- The UNIX command *man* will tell you about a specific command. For example, *man mkdir* will give the description of unix command *mkdir*.
- Some important UNIX commands:
 - *ls -l* list files
 - *cp* copy files
 - *more* display file contents on screen
 - *mv* move/rename files
 - *rm* remove files
 - *mkdir* make directory
 - *cd* change directory
 - *rmdir* remove directory
 - *lp* print command
 - *emacs* invokes the Emacs editor
- At the prompt, the up arrow key brings up previously typed commands in reverse order; the down arrow key brings up previously typed commands in forward order.

3. DESCRIPTION OF THE ASSIGNMENT

Your *main()* program should give the user a menu to choose from, as shown below.

I. Enter a date

Q. Quit

Your program should proceed based on the user’s choice.

a) If the user chooses I (Input)

read a date from the keyboard as mm/dd/yy;

convert it to the output format if input is right, otherwise, give error message;

return to the main menu and wait for user to choose the next action;

b) If the user chooses Q(quit)

exit from the program;

Specification of Input and Output :

- The correct input format is: mm/dd/yy, for example: 01/01/01.
- The correct output format is: mm/dd/yyyy, where month and day are not zero filled. An example could be 1/1/2001.

- Assume we are interested only in years between 1950 and 2049, a Y2K correction is necessary for a 2-digit year, i.e. if (year < 50) year += 2000; else if (year < 100) year += 1900.
- If the input is not in correct format, your program needs to display an error message. For example, if 1/1/01 has been entered as a date by the user, error message can be "Error: Invalid format". If input is 13/01/06, then error message can be "Error: Invalid Month".
- **Hint:** you can use functions *strtok* and *atoi* from the C library to parse the input and extract month, day and year.

The following labs will base on the date class in lab1. Try to get this lab working !

Compilation

Save your program as `lab1.cpp` and compile it with `g++` by entering the following command in the xterm window:

```
g++ -o lab1 lab1.cpp <return>
```

Remove all compilation errors and warning messages before running the program. Recompile each time you make a change to the program. By finishing, your program should compile and return to the prompt without displaying any error messages.

Running

Run your program by entering the following command in the xterm window:

```
lab1 <return>
```

Test your program with several cases you may think of, such as "1/1/01", "01012001", "01/01/50", "12/12/01" (These are just examples). It should comply with the specifications given above.

Submission

When you finish with the lab, you need to turn it in for grading. The *submit* command submits your lab electronically. You **MUST** use the *submit* command to turn in your labs. The format of *submit* command is as follows:

```
submit classname labname files-to-submit
```

where,

classname is the name of the CIS 459.22 section that you are enrolled in. Your classname is `c459.22ab` .

labname is the lab you are working on (`lab1`, `lab2`, etc.). For this lab, labname is `lab1` .

files-to-submit is a list of the files that make up the lab. For now, it only contains `lab1.cpp` .

```
Ex: submit c459.22ab lab1 lab1.cpp
```

NOTE:

- All of the files in a lab **MUST** be submitted using one command. If you use two *submit* commands, the second one **erases** the files from the first submission.
- Your programs **MUST** be submitted in source code form. Make sure that you submit the "lab1.cpp" file only. Do not submit the object file.
- Each *submit* command **MUST** be entered on one line without pressing Enter. If the line you are entering is too long, it wraps onto the next line.
- It is **YOUR** responsibility to make sure your code can compile and run on CSE server stdsun.cse.ohio-state.edu, if you develop the code using other systems. Points will be deducted if your code does not compile on stdsun servers.