

**CS-202**

Lab Section 1

# Linux-GCC Primer, Console Input/Output, Sorting Basics

**Xinying Wang, Bashira Akter Anima, Yuchuan Liu**

University of Nevada, Reno



# Your Teaching Assistants

## Yuchuan Liu :

- [ycliu@nevada.unr.edu](mailto:ycliu@nevada.unr.edu)
- Office Hrs: Fri 4 pm-6 pm @ ECC

## Xinying Wang :

- [xinyingw@nevada.unr.edu](mailto:xinyingw@nevada.unr.edu)
- Office Hrs: Th 8 am - 10 am @ ECC

## Bashira Akter Anima:

- [banima@nevada.unr.edu](mailto:banima@nevada.unr.edu)
- Office Hrs: Mo 11 am - 1 pm @ ECC

Lab Section 2 –	10:00-10:50 am	@ SEM 231B
Lab Section 3 –	11:00-11:50 pm	@ SEM 231B

Lab Section 1 –	9:00-9:50 pm	@ SEM 231B
Lab Section 4 –	12:00-12:50 pm	@ SEM 231B

Lab Section 4 –	6:00-6:50 pm	@ SEM 231B
Lab Section 6 –	7:00-7:50 pm	@ SEM 231B

# Your Teaching Assistants

## Contacting your TAs :

- WebCampus “Discussion” board :  
This is checked and answered regularly.
- During your scheduled Lab times :  
Strictly for subjects pertinent to the Lab material / Quizzes!
- During the announced Office hours:  
For any other inquiry regarding the Course material / Projects.
- Email :  
Better used to schedule appointments if necessary.

# Lab Objectives

- Learn and Practice Programming under the Linux OS & environment.
- Learn certain C++ features not covered in class.
- Go through Debugging practices.
- Support and facilitate your overall success in this course.

# Grading and Requirements

## Lab Quizzes

- At the end of each Lab Section.
- Short (due in class) on-sheet coding assignments.
- Answers delivered on sheet at the end of the Section.
- You *should* use your ECC computer to put together & verify your code.
- You *can* use any other resource (open-book).
- Due to limited time, best consider inquiring the assistance of your TA.
- *Note:*  
Failure to attend labs and complete Quizzes will be reflected on your grade.

# Coding Style Tips

- Comments should be used where appropriate :  
Make them explanatory with as few words as possible !  
Not on every line of code !
- Code should be clearly formatted and indented.
- Variable names should help with code interpretation.
- Functions should be used where appropriate.  
Anything that expresses a distinct functionality (e.g. copying the content of a Cstring into another) should be made into a function.

# Coding Style Tips

```
int sumOfIntAbsolutes (int numA, int numB)
{
    → if (numA < 0)
    {
        → numA = -numA; //changes value of numA to absolute
    }

    if (numB < 0)
    {
        → numB = -numB; //changes value of numB to absolute
    }

    return (numA + numB);
}
```



# Coding Style Tips

```
int sumOfIntAbsolutes (int numA, int numB)
{
    numA = intAbsolute(numA); //changes value of numA to absolute
    numB = intAbsolute(numB); //changes value of numB to absolute

    return (numA + numB);
}
```

```
int intAbsolute (int num)
{
    if (num < 0)
    {
        return -num; //flip sign and return
    }
    else
    {
        return num;
    }
}
```



# Project Submission

## Basic Guidelines:

Include:	Don't Include:
<b>Source Code</b> – ALL source and header files (always tested to <b>Compile &amp; Run</b> )	<b>Executables</b> (we will Compile & Run your Source Code)
<b>Documentation</b>	<b>Extra Files</b>
<b>Makefiles</b> (when applicable)	<b>Empty Files</b>

*Note:* More specifics will be given when the first project is assigned.

# Linux Terminal Basics

## Syntax

`command` `[options]` `[arguments]`

Example:

`ls -a directory_name`

`ls` : command

`a` : option (signified with ‘ - ’)

`directory_name` : argument

# Linux Terminal Basics

## Linux Basic Commands

**pwd**

: print working directory

**ls** [**options**] [**directory**]

: list files and directories within the  
the given directory

(**-a** option shows hidden files too)

**cd** [**directory**]

: change directory

**mkdir** [**options**] [**directory**]

: make directory

**rmdir** [**options**] [**directory**]

: remove directory

# Linux Terminal Basics

## Linux Basic Commands

**rm** **[options]** **[file/directory]** : remove file or directory

**-r** option recursively deletes the directory

**-i** option prompts prior to deletion

**cp** **[options]** **[file 1]** **[file 2]** : copy file 1 to file 2

Will overwrite file 2 if it exists

**mv** **[options]** **[source]** **[destination]** : moves file or directory from source location to destination location

# Compilation / Execution

## Compilation Basic Command:

**g++** [options] [file 1] [file 2] ... [file N]

- Compiles, Assembles, and Links
- Creates executable file ( ' **a.out** ' by default)
  - o** option allows you to specify output executable file name

Example:

```
g++ -o executable_filename source_code_filename.cpp
```

# Compilation / Execution

## Running a Binary Executable File

After SUCCESSFUL compilation

(*Note*: always check the **g++** output to see if compilation succeeded, otherwise you might be running the result of a previous successful compilation !)

- `./a.out`
- or
- `./<executable_filename>`

*Note*: **Ctrl-C** will terminate a program that is currently executing.

# Hands-on Code Submission

Install the **NoMachine™** Client:

<https://www.nomachine.com/>

Connect to the **CSE** Ubuntu Virtual Machine Environment:

<https://unr.canvaslms.com/files/3623763>

- 1) Right click on the desktop screen, choose “Create Document” → “Empty File”, name it **proj\_1.cpp**
- 2) Double-click to open it (with the default text editor – gedit ), write your code and save it.
- 3) Back on the Desktop screen, click the Blue sign on the top-left, and then click on “Terminal Emulator”
- 4) In the terminal give the command “**cd Desktop**” to go to the Desktop folder
- 5) In the same terminal give the command “**g++ -o proj\_1 proj\_1.cpp**” to compile your code
- 6) If it compiles correctly you will have a **proj\_1** file created which is your executable
- 7) You can run to test the executable by giving the command “**./proj\_1**” in the same terminal screen.  
If it compiles and runs, you can take the source code (**proj\_1.cpp**) and put it in an archive file (zip, tar.gz) together with the documentation file. Then upload this compressed archive file on WebCampus.
- 8) You will find the program that creates compressed files by clicking on the Blue sign on the top-left, and then go to “Accessories” → “Archive Manager”. You can add the files to compress via drag-and-drop.



# Console Input / Output *by-Example*

## Terminal/Console Input

```
#include <iostream>
using namespace std;
```

```
int main ( ) {
```

```
    int int_value;
    char cString_value[10];
```

```
    cin >> int_value;
```

- Waits for and captures console input.
- Attempts to interpret as int.
- Stores result in variable **int\_value**.

```
    cin >> cString_value;
```

```
    return 0;
```

```
}
```

# Console Input / Output *by-Example*

## Terminal/Console Input


```
#include <iostream>
using namespace std;

int main ( ) {
    int int_value;
    char cString_value[10];

    cin >> int_value;

    cin >> cString_value;

    return 0;
}
```

- 
- Waits for and captures console input.
  - Attempts to interpret as C-string (**char** array).
  - Stores result in variable **cString\_value**.

# Console Input / Output *by-Example*

## Terminal/Console Input

```
#include <iostream>
using namespace std;

int main ( ) {
    int int_value;
    char cString_value[10];

    cin >> int_value;

    cin >> cString_value;

    return 0;
}
```

*Note:* Make you type up to 9 characters (C-strings require +1 NULL-terminating character) otherwise you might get weird results...

# Console Input / Output *by-Example*

## Terminal/Console Output

```
#include <iostream>
using namespace std;

int main ( ) {
```

```
    int int_value;
    char cString_value[10];
```

```
    cout << "Give me an Integer" << endl ;
```

```
    cin >> int_value;
```

```
    cout << "Give me a String" << endl ;
```

```
    cin >> cString_value;
```

- Prints a literal string.
- Appends a new-line at the end.

- Prints another literal string.
- Appends a new-line at the end.

```
    cout << "Number: " << int_value << " and String: " << cString_value << endl;
```

```
    return 0;
```

```
}
```

# Console Input / Output *by-Example*

## Terminal/Console Output

```
#include <iostream>
using namespace std;

int main ( ) {

    int int_value;
    char cString_value[10];

    cout << "Give me an Integer" << endl ;
    cin >> int_value;
    cout << "Give me a String" << endl ;
    cin >> cString_value;
```

➤ Prints a literal string, an `int`, a C-string, a new-line, etc...

```
    cout << "Number: " << int_value << " and String: " << cString_value << endl;
```

```
    return 0;
```

```
}
```

# Sorting Basics

## Sorting

Order a set data from lowest to highest based on a given key

- List of test scores, an ID number, etc.
- Order by first or last name alphabetically.

**Bubble Sort** ( Simple and intuitive -but inefficient- way to sort data ) :

- [Step 1] Compare each pair of adjacent elements from the beginning of an array and, if they are in reverse order, swap them.
- [Step 2] If at least one swap has been done, repeat [Step 1].

# Sorting Basics

**Bubble Sort** ( Example ) :

5 1 7 -5 9 Unsorted

5	1	7	-5	9	$5 > 1$ , swap
1	5	7	-5	9	$5 < 7$ , ok
1	5	7	-5	9	$7 > -5$ , swap
1	5	-5	7	9	$7 < 9$ , ok

1	5	-5	7	9	$1 < 5$ , ok
1	5	-5	7	9	$5 > -5$ , swap
1	-5	5	7	9	$5 < 7$ , ok

1	-5	5	7	9	$1 > -5$ , swap
-5	1	5	7	9	$1 < 5$ , ok
-5	1	5	7	9	$-5 < 1$ , ok
-5	1	5	7	9	Sorted



**CS-202**

Time for Questions !