



LABORATORY SESSION 1

- **Content of lab session:** Quick introduction to Linux. Using the file editor *gedit* or *emacs*. Compiling a C program via the compiler *gcc*. Reading and understanding simple C programs, with different variable types and memory allocation of arrays. Letting the students write simple C programs. Using *gnuplot* for plotting curves and writing figures in a file.



L.1 Quick introduction to Linux

Fedora Linux Live ([version 25](#)) has been installed on a 32G USB flash drive. The first step is to reboot your machine from the flash drive.

- Power off your machine.
- Insert the flash drive.
- Power on your desktop machine and press F12.
- Select the USB device in the boot menu.
- Fedora Linux should start booting. On boot, you need to select again [Start Fedora-Workstation-Live 25](#) by moving the highlighted sentence up via the arrow keys. Fedora Linux operating system will start booting with many OK messages. Type Ctrl-C if it gets stuck on one of the booting jobs.
- A login prompt will show up for the account *liveuser* (password is required).
- Click on the *Activities* button located on the top left corner. Then at the favorites menu on the left, click on the second icon to open a Terminal. A right-click on the Terminal icon lets you open another new terminal window if needed.
- Notice, inside the terminal, the prompt is *liveuser@localhost-live*. In general, Linux or any Unix-based operating system will display a prompt indicating the user name and the host name (machine) in the format *username@hostname*.
- Inside the terminal, type the following command “*pwd*” then Enter. The command *pwd* stands for “*print working directory*”. By default */home/liveuser* is the home directory for your Fedora Linux user account.
- Type “*ls*” inside the terminal. This command will list all files and directories in your current working directory. Type “*ls*  *l*” to get a long listing (detailed listing). The symbol  represents an empty space. A directory is also known as a “*folder*”.

Linux is an operating system capable of running on all types of machines. Interaction between a user and the operating system is made via a graphical user interface, also called a desktop environment. Many desktop environments are available under Linux. Our current Fedora 25 Linux installation (2016-2017) is using GNOME 3 as the default graphical user interface. Another alternative is KDE.

Now, let us create a new file and do some simple manipulations using commands inside the Linux terminal.

- Type “*cat*  *example.txt*” then Enter. Continue typing, fill-in multiple lines (any text can be included). To end the file, type *Control-D*. Check that file *example.txt* has been well created by typing “*ls*”. What is the size (in bytes) of this file? Find its size with “*ls*  *l*”.

- In order to see (display on screen inside the terminal) the content of file *example.txt*, just type “*cat* *example.txt*” without the “>” sign.
- *cat* stands for concatenate. You can concatenate two files *file1* and *file2* and put the result into *file3* as follows “*cat* *file1* *file2* > *file3*”.
- Unix manual pages include information about commands, mathematical functions, and system functions. Type “*man* *cat*” to get information about *cat*. Type “*man* *ls*” to get information on *ls*. Type “*man* *cos*” or “*man* *tanh*” to see the manual page of the cosine function or the hyperbolic tangent function respectively. The **space** key allows you to scroll inside a long manual page. You can also use the keyboard buttons **b**, **f**, and **q**, **b** for backward, **f** for forward, and **q** for quit.
- Finally, you can delete (remove) the file *example.txt* by typing “*rm* *example.txt*”.

Notice that files and directories (or folders) can be **manually** handled via commands inside a terminal or **mouse-handled** via a file manager such as Nautilus. The file manager Files is shown as drawers in the favorites menu (under Activities), click on the Files icon to check the content of your Home folder.

Launch the Mozilla Firefox web browser from the Activities menu. Connect to the Internet via the TAMUQ authentication address bluesocket-1.qatar.tamu.edu, then go to <http://www.josephboutros.org/ecen210/lab1> and get the file **infinite_ops**. This file is a binary executable file. It runs $y=\cos(x)$ forever.

- Create a new folder by typing “*mkdir* *lab1*”. The full path of this folder should be */home/liveuser/lab1*. You can also use Nautilus to create the folder.
- Download *infinite_ops* and put it in *lab1* folder. Inside a terminal, you can type “*cd* *lab1*” to change directory and go into *lab1*.
- Make *infinite_ops* executable by typing “*chmod* *u+x* *infinite_ops*”. The letter “u” stands for user and “x” stands for executable.
- Run the program by typing “*./infinite_ops*” followed by Enter.
- You can stop the program execution by pushing *Control-C* (type it inside the same terminal where the program is running).
- Let us run the program in a background mode, type “*./infinite_ops* &” followed by Enter and again Enter. In the same terminal, now you can type another command, but the program is still running. For example, type “*xeyes* &”.
- How can we find the Unix process that is executing *infinite_ops*? The answer is *ps*, i.e. process status. Just type “*ps*” followed by Enter. You will see the command name (CMD) and the process id (PID). Stop the process by typing in the terminal “*kill* *PID*” where you should replace PID by the correct number representing the process id. The command *kill* sends a signal to a process. Linux is currently running a high number of processes in parallel. It can also run multiple users connected to the same machine in parallel.
- Run again the program by typing “*./infinite_ops* &”. In the same terminal, or in a second terminal (you can open as many terminals as you want), type “*top*”. *top* is an alternative to *ps*. *top* will display a list of processes. The first one should be *infinite_ops*, it should be taking 99% of CPU usage. Type **q** to quit *top*.

I.2 Utilize gedit/emacs for editing text files

We recommend the use of the two following text editors: *gedit* or *emacs*. Both text editors can be found in the menu under Activities. You can also open a terminal then type “*gedit* *filename*” or “*emacs* *filename*”, where *filename* is the name of the file you would like to open.

- Download *hello.c* from <http://www.josephboutros.org/ecen210/lab1> and place it in folder */home/liveuser/lab1*.
- Use *emacs* or *gedit* to open *hello.c*. You can do it through a terminal or through menus. Try both methods.
- The file *hello.c* is a simple C program. It just displays “Hello ECEN 210” on the screen. This simple message is displayed via line #11 inside *hello.c*. This line includes “`printf(“Hello ECEN 210! \n”);`”. Add a second line to display hello again, just add “`printf(“Hello again! \n”);`” inside file *hello.c* on line #12.
- Save the modification made above and quit the text editor.

gedit is easier than *emacs*. Both include coloring of C language source files. Copy&Paste inside *gedit* is done as usual via Control-C and Control-V. Copy&Paste inside *emacs* is done via the middle mouse button: Select a text using the mouse (this is copy!) and then paste by clicking on the middle button. The middle mouse button in Linux makes the copy&paste much faster than copy&paste in other operating systems such as Windows or OS X for Macs. Very recently, Apple adopted this feature in the Terminal application under Mac OS Sierra.

I.3 Compiling a C program with gcc

The text file *hello.c* is a simple C source file. It includes the main function to be executed. The main function starts with a curly bracket “{” and ends with a curly bracket “}”. Displaying a text on the screen is made via *printf*. The screen is known as *stdout* under Unix. The **Unix** operating system is the father of Linux. Unix was developed in the early 70s (50 years ago) for big mainframe machines. Nowadays, besides Linux, OS X for Macintosh is also Unix-based. Fast web servers, fast email servers, and fast computing machines at Google, Amazon, or Facebook are all running under Linux or a Unix-based operating system. The function *printf* sends the text to *stdout* (standard output). In order to use *printf*, the C file must include a header file that defines *printf*. The corresponding header file is *stdio.h* (standard input and output), it is included before the main function by writing “`#include <stdio.h>`”. You can read more about *printf* from the manual page “*man 3 printf*”.

The C file also includes comments. A comment starts with “*/**” and ends with “**/*”. It is also possible to add a full line for comments by starting the line with “*//*”.

The C source file can be converted into a binary file then into a binary executable file using a C compiler. We will use the standard GNU compiler named *gcc*. The *gcc* manual page includes a large amount of details on the use of *gcc*. You can also find web pages on the Internet giving tutorials on *gcc*.

Let us introduce gcc by compiling hello.c. The first step is to generate the binary file hello.o, then the executable file hello.

- Open a terminal, or use an already opened terminal.
- Go to folder lab1 where you can find hello.c.
- Type “gcc -c hello.c”. This will compile the C file and generate hello.o. Type “ls” or “ls -l” to check the binary file.
- Type “gcc -o hello hello.o” to create the executable file hello. Run the program inside the same terminal by “./hello” followed by Enter.

Let us study a more advanced C program.

- Download the file *infinite_ops.c* from lab1 folder on our course web page <http://www.josephboutros.org/ecen210/lab1>.
- Open *infinite_ops.c* using gedit or emacs. Read the C program. It includes 4 variables, an integer *iter* and 3 real numbers *x*, *y*, and *PI*. An integer is declared via “int”. A real number with double precision is declared via “double”. The program has an infinite *do-while* loop.
- Compile via “gcc -c infinite_ops.c”.
- Link via “gcc -o infinite_ops infinite_ops.o -lm”. This command will link the binary file to the standard mathematical C library that contains the cos() and atan() functions. The same command, after linking, will generate the executable file *infinite_ops*. Now, you can run it via “./infinite_pos”.

I.4 Writing a new C program to convert from Celsius to Fahrenheit

In this section, students are asked to write a C program that converts Celsius temperature into Fahrenheit temperature. The work is based on the original program celsius.c found on pages 24-25 in ECEN 210 textbook by Dr. K.N. King.

- Download the file *celsius.c* from lab1 folder on our course web page <http://www.josephboutros.org/ecen210/lab1>.
- Open *celsius.c* using gedit or emacs. Read this C program. You can also open the textbook on page 24, Chapter 2.
- Copy *celsius.c* into *fahrenheit.c*. Modify the C file *fahrenheit.c* and make it convert degrees from Celsius to Fahrenheit. Compile and run on different values.

I.5 Writing a C program to compute the dimensional weight with the ceil() function

In this section, students are asked to write a C program that calculates the dimensional weight using double precision real variables and a call to the mathematical C function *ceil(x)* which returns the smallest integer greater than or equal to *x*. Recall the following:

$$y - 1 < x \leq y, \quad \text{where } y = \text{ceil}(x), \quad x \in \mathbb{R}, y \in \mathbb{Z}.$$

The work is based on the original program dweight2.c found on page 23 in ECEN 210 textbook by Dr. K.N. King.

- Download the file *dweight2.c* from lab1 folder on our course web page <http://www.josephboutros.org/ecen210/lab1>.
- Open *dweight2.c* using gedit or emacs. Read this C program. You can also open the textbook on page 23, Chapter 2.

- Copy *dweight2.c* into *dweight3.c*. Modify the C file *dweight3.c*. Make use of double variables instead of int variables. Modify the calls to scanf and printf accordingly. Utilize *ceil()* instead of the fraction $(volume+165)/166$. Compile and run on different values.

I.6 Memory allocation of arrays (advanced section)

An array is a list of items. When items are numbers, the array is simply a one-dimensional table of numbers. For an array of size *n*, array elements are numbered from 0 to *n*-1. This is a well-known convention in C programming; we start counting from 0, not from 1.

Arrays may have a fixed or a variable size. For example, in an application where the user is always reading 10 integers, then the array named “table” can be declared and allocated by the following expression: *int table[10];*

If the number of array elements is unknown in advance, the user must read its size before allocating memory for the array. This is called dynamic memory allocation and can be performed in C language via two system functions *malloc()* and *calloc()*. For example, after reading the value of the size *n*, the user may allocate an integer table of size *n* as given in the following expression: *table=(int*) calloc(n, sizeof(int));*

- Download [fixed_size_array.c](#)
- Read and understand the content of *fixed_size_array.c*
- Compile (via gcc) and run the program *fixed_size_array*
- Download [variable_size_array.c](#)
- Read and understand the content of *variable_size_array.c*
- Compile (via gcc) and run the program *variable_size_array*

Matrices are bi-dimensional arrays. The students will encounter matrices in C language in future lab sessions. Briefly speaking, a matrix shall include an array of pointers where each pointer is pointing towards one matrix row. It is straightforward to create in C language multi-dimensional arrays such as 3D or 4D matrices.

I.7 Gnuplot for plotting curves

Gnuplot is a powerful tool for plotting 2D curves and 3D surfaces. Data can be read from one or many input files. Output can be displayed on screen (this is the default behavior) or it can be put in a file format such as *eps* (encapsulated postscript).

- Open a terminal, or use an already opened terminal.
- Type “gnuplot” followed by Enter. No need to use the ampersand “&”.
- Now, inside gnuplot, type “*plot sin(x)*” as a first example.
- You can get help by typing “*help*” inside gnuplot. For example “*help cos*”, “*help set grid*”, etc.

We would like to plot the result generated by a C program. Let us do it with *hyperbolic_tangent.c*. This program computes the hyperbolic tangent function defined as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Open a terminal and go to folder *lab1*.
- Download *hyperbolic_tangent.c* from ECEN 210 web page.
- Compile the C program as explained in previous sections.
- Run the C program *hyperbolic_tangent*.
- Check if the output has been well created. What is the name of the output file? What is the size of the output file (in bytes)?
- Launch *gnuplot* in the same directory */home/liveuser/lab1*
- Inside *gnuplot*, type the following
 - set grid
 - set style data lines
 - set xlabel "x"
 - set ylabel "Hyperbolic Tangent"
 - plot "tanh_function.dat" lw 3
 - set yrange [-1:1]
 - plot "tanh_function.dat" lw 3 t "tanh", x lw 2 t "y=x"
 - set term post land enh color
 - set output "image.eps"
 - replot
 - quit

The image file generated by *gnuplot* can be included in any document. You can display the image file on screen by typing "*evinceimage.eps*" or by clicking on its icon in the correct folder using the file manager Files (Nautilus). You can also use the old heavy-duty ghostview to open the eps file, "*gvimage.eps*".

- Draw a logarithmic spiral using *gnuplot*. Such a spiral is defined in polar coordinates by $r = ae^{b\theta}$, which is equivalent in Cartesian coordinates to $x = ae^{b\theta} \cos(\theta)$ and $y = ae^{b\theta} \sin(\theta)$. The angle θ will be replaced by the dummy parameter t inside *gnuplot*. Restart the application and type the following commands inside the *gnuplot* environment.
 - set parametric
 - set autoscale
 - set title ""
 - set key box
 - set key below
 - a=1.0
 - b=0.1
 - set trange [0:40]
 - set samples 500
 - plot a*exp(b*t)*cos(t), a*exp(b*t)*sin(t) lw 2 lc rgb 'red'

For more information, Please contact joseph.boutros@qatar.tamu.edu

Dr. Joseph Jean Boutros.