AE1PGA Lab 1

The first PGA lab is a straight-forward lab to check that you all know how to log on to the School of Computer Science's Linux server that we will be using in PGA, that you can enter a simple program from the textbook, and compile and run it successfully. This will make sure you are ready to try programming in your own time, and for next weeks lab exercises.

---

1 Logging On

You can graphically log on to our server by using the "X2Go" program. Depending on if you are using a fixed desktop computer on campus, your laptop on campus wifi (Eduroam), or your laptop off campus, there are different ways to connect.

1.1 Fixed desktop computer on campus

Go to the start menu → "_UNNC Software" → "X2Go Client". Do *not* just type "X2Go" into the start menu to search for the program as that may run older, incompatible versions. Then continue with the common instructions.

1.2 Your own laptop on campus

You can install the X2Go client on your own laptop. There are versions for Windows, OS X, and Linux). Once you have done that, you can configure it as described in "A - Session configuration" below and then follow the common instructions below. This will connect directly to our cslinux server from your laptop. This only works when your laptop is connected to the UNNC internal network (via Eduroam). It does not work off campus or in the dorms.

1.3 Your own laptop using the VDI

The university has a *virtual desktop infrastructure* that allows you to remotely connect to a University Windows desktop anywhere in the world (including from your dorms). From that, you can run X2Go normally as you would in 1.1 above. To use the VDI, follow the VDI Getting Started guide on the School of Computer Science Moodle page.

Common instructions

After starting X2Go, the X2Go control window will pop up. The configuration for a session to our linux server should already be set on the right hand side. Clicking on this will pop up a login box asking for your username and password. These are your standard UNNC username/password combination. Once they have been entered, X2Go will make a connection to the linux machine. This may take about 30-60 seconds, depending on how many students are doing it at the same time. X2Go has many settings - for now, don't change any of them. If it doesn't work, look at the possible problems below.

Once connected, you will be shown an entire new "virtual" desktop that behaves in a similar way to the Windows desktop. Everything within this window is running on the linux server. Note that the Windows taskbar is still at the very bottom of the screen. You can minimize the linux window and swap between that and other windows as normal. You can do similar if you are using other operating systems.

Possible problems

A - Session configuration

X2Go should be pre-setup with configuration for our Linux server but some machine seem to be missing it. If you try connecting and X2Go gives you an error saying "Hostname missing" or something similar, you will need to add it to the configuration. Click "Cancel" on the login box to make the session to appear on the right hand side column again. Then click on the "hamburger menu" on the bottom right of that session and select "Session preferences...". Make sure the "Host" setting is "cslinux" (all one word, lowercase). The "Session type" at the bottom should be "XFCE". Not other settings should be changed. Click ok and then try to log in again.

B - Missing Panels on desktop

If the application menu in the top left is missing, you might be missing the entire top panel. You can get the panel back by right-clicking on the desktop and selecting "Applications" → "Settings" → "Panel". You can then use the "+" button to add a new panel and drag it to the top of the screen. Change the "Length" to 100%. Then go to the "Items" tab and click the "+" button there. Select "Applications Menu" and click the "Add" button. You should then see the applications menu on the panel and can close each of the open windows.

2 Running a text editor

In the top-left corner, there is an applications menu. Go to the "Accessories" entry, then click on "gedit". This starts a text editor in a new window. We can use this to create and edit files. Gedit is a simple programmers editor that understands C source code, which means it can highlight the different parts of the source code to make them easy to read.

Type in the source code for the "Welcome" program on p72 of the textbook. Be careful to type it *exactly* as it appears but don't type the line numbers on the left, they are just to make it easier to refer to a part of the program in the book. (Actually, you don't really need to type in the lines that being with `//` either. These lines are *comments* which the compiler ignores, they are just there to help us humans make notes about the source code inside the source code file. Because the compiler ignores them, it won't make any difference if you copy these lines or not. I'd suggest copying everything for now, so you don't accidentally miss something important.)

Once you have typed it in, click the save button in the toolbar. Since this is a new file, it will open a file dialog for you to choose where to save the file to. Click on "Home" in the left hand column to select your *home directory* as the location to save the file (the Linux equivalent of your Z: drive). You then need to change the name of the file at the top of the dialog. Give it the name "welcome.c". Finally, click the save button and the file will be saved and the dialog will close. After it has saved you should see the source code in the window suddenly being highlighted. This is because we gave the file the ".c" extension, so gedit highlights it as C source code. If you open an already existing file with a ".c" extension, gedit will highlight it immediately.

3 Running a terminal

Go to the "System" program menu entry, then click on "Terminal". This should open a new window with a text prompt. This is where you enter text commands to be run. We will use it to run the compiler and to run our programs. This "command line interface" is the textual equivalent of the normal "graphical user interface" that you use with your mouse to run programs and manipulate your files. For our purposes, the textual interface gives us more power over exactly what we want the computer to do. There are many commands you can run from here. We will only use a few today. Others you will pick up in the labs throughout the course. If you are interested in knowing more, there are many books in the library about using the Linux command line.

The terminal has a *current working directory*. This is where all the commands that are run will look for files, if you haven't told them to look somewhere else. By default, the terminal starts in your *home directory*. In the terminal, this is represented by the ~ character in the prompt just before the cursor. In the previous step we saved out source code in our home directory so we should be able to find it easily.

4 Compiling the source code

Type the command `ls` and press return (the letters "el" and "s"). This will give you a list of all the files that are in the current directory (`ls` is short for "list"). You should be able to see your `welcome.c` file in the results. If you cannot, this means you either didn't do step 2 correctly or you didn't do this step correctly. Please check steps 2 and 3 and then ask for help if you still cannot see the file.

The compiler we will use is called `gcc`. A compiler is just a program like any other program, so we can run it on the terminal. If we just run `gcc` (by typing in "gcc" then pressing return) then we will get an error message, because `gcc` needs to be more given information when run to know what to do. When running a command in the terminal, we can give it extra *command line arguments* by typing them after the command name (we will see how we can use these in our own programs at a later date).

Run the command `gcc welcome.c -o welcome` ("-o" is dash then the letter "oh", not a zero). This is the command to compile the `welcome.c` file and turn it into a program called `welcome`. If it ran successfully then there will be no messages, you will just be returned to the prompt ready to type in your next command. If it was not successful then there will be an error message. For this task, this means you did not copy the program correctly. Go back to step 2 and edit the program by running gedit, opening the `welcome.c` file, changing it, saving it, and trying to compile again. As before, ask for help if you are having repeated problems. One problem to watch out for — if you accidentally type the name of your `welcome.c` file twice, the program generated by `gcc` will overwrite your C program file and you will need to type it all in again!

5 Running the program

The final step is to run your new program. Run the command `./welcome`. It should print the message "Welcome to C!" and then return to the command prompt. (The `./` bit at the beginning tells the terminal to run the program called `welcome` that is in the current directory, instead of looking for a system programming with that name. If you forget to type it, the terminal will probably give you an error message saying command not found.)

Once you have got this running, show one of the lab instructors so we can tick your name on the list.

6 Further work

Try entering the adding program from p76 of the book and get it running. This program is more complicated but you only need to type it in to get it to work. Make sure to save it with a different filename and alter the commands above to use that filename.

Like before, once you have got the second program running, show one of the lab instructors so we can tick your name on the list.

If you have been able to get both programs running then that is all you need to do in the lab this week. Read at least the descriptions of these two programs on pages 72-80, so you get an idea of how they work before we get to them in the lectures.

End