Introduction to the Raspberry Pi 3 Model B and Program Development in the Linux Lab

Objective

This guide is to learn how to setup, connect to, and develop programs for the Raspberry Pi 3 Model B board.

Part 1: Board setup

1. To begin, log on to the lab computer using your pawprint and password.

2. Insert your SD card into the Raspberry Pi at your workstation. Once this is completed, plug in the power cord (or press the inline switch on the power cord, if there is one). This will begin the booting process, which should take less than a minute to complete.

3. The computers in the lab run Ubuntu, and they will be used to develop the lab assignments. The development can be done with Eclipse and is described in Part-2 below. The lab computers will be used to login to the Raspberry Pis via ssh. To do this, open a terminal shell on your local machine and type in the command:

   ssh –Y pi@RASPBERRYPI_IP or ssh –Y root@RASPBERRYPI_IP

   In place of 'RASPBERRYPI_IP' use 128.206.19.[Last 2 digits located on the monitor label of your workstation]. Example: Monitor label is **ENGR-C1251-127**, then you would use **128.206.19.27**. Notice that **Y** is capitalized and the –Y option enables the graphical interface, without it you won't be able to use Eclipse properly. After running the ssh command you will be asked to enter a password. Use the password: roomc1251. This is the password for all the Raspberry Pi boards in the lab for both the pi and root accounts.

4. You should now be logged into the Raspberry Pi at your workstation.

5. When you finish using a Raspberry Pi, be sure to shut it down; otherwise you might corrupt your ARM/Linux file system and lose your work. To shut down type the following in the terminal:

   sudo shutdown –h now (or just shutdown –h now if logged in as root)

   After a few seconds, you will lose the connection to the Raspberry Pi from the terminal. Close your terminal and then wait 1 minute or so to allow the Pi to shut down. Now turn off your board by unplugging the power cord (or pressing the inline switch on the power cord, if there is one). You can now remove your SD card from the Raspberry Pi. Please be careful doing so, you don't want to damage the board or your SD card.

Part 2: Program Development with Eclipse on Raspberry Pi 3 Model B

1. To develop programs for the Raspberry Pi, the Eclipse IDE will be used. You must open Eclipse while being logged in to the Raspberry Pi through your local machine. To open Eclipse, type:

   eclipse &

2. After Eclipse opens it will prompt for a workspace path, enter: */home/pi/workspace*, which should be the default path.

3. Next, we need to create a new project, follow the instructions and screen shots shown below. They show how to create a "Hello World" C project, but you can take similar steps to create a C++ project. It is convenient to create a non-empty project, since this will create some directories for your source files and binaries, and a simple source file with common header files included and a main program that you can edit.

   (a) Click **File→New→Project** or click the **New Project** shortcut button underneath **File** on the top left of the IDE. Create a new C project by expanding the C/C++ folder and selecting **C Project**. Now click **Next**. You should now see the image below. Make sure to select '**Hello World ANSI C Project**' for project type and '**LinuxGCC**' for toolchains. Give a name to your project ("testProject" in this example). Click **Next**. (See figure-1)

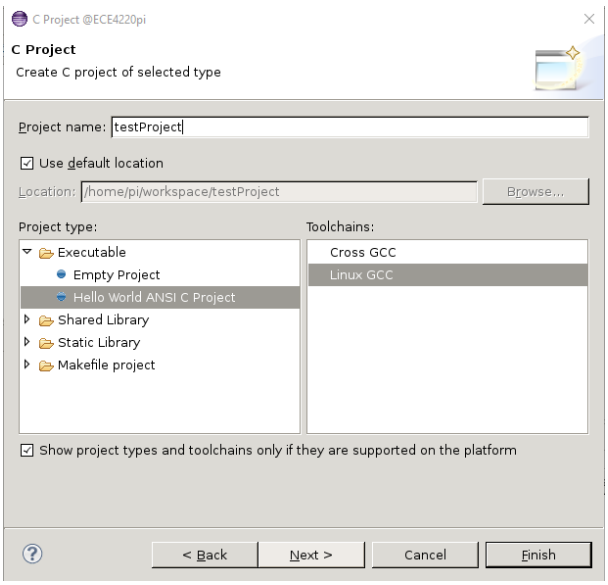

Figure 1: New project name with compiler selection

   (b) You should now see the image 2, which includes basic settings. You don't need to change anything here, but you can add your name as Author if you like. Click **Next**.

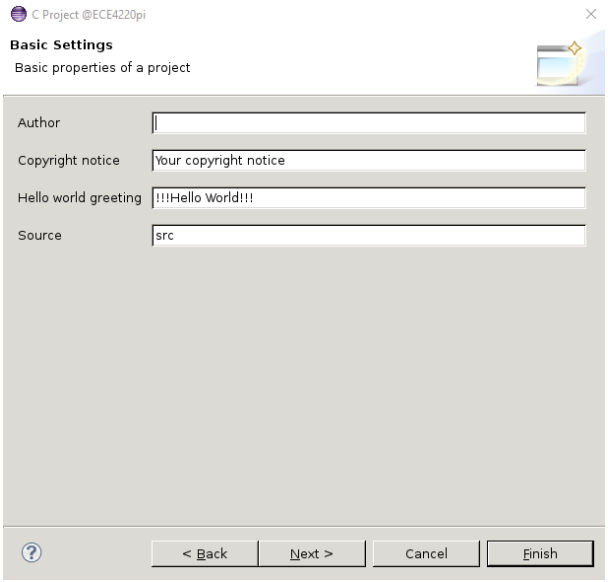

Figure 2: Project basic settings

(c) You should now see this image-3 which shows the selection configurations. Unclick **Debug** and then click **Finish** to create your first project. You may leave the Debug option if you would like to use the debugger for your project.

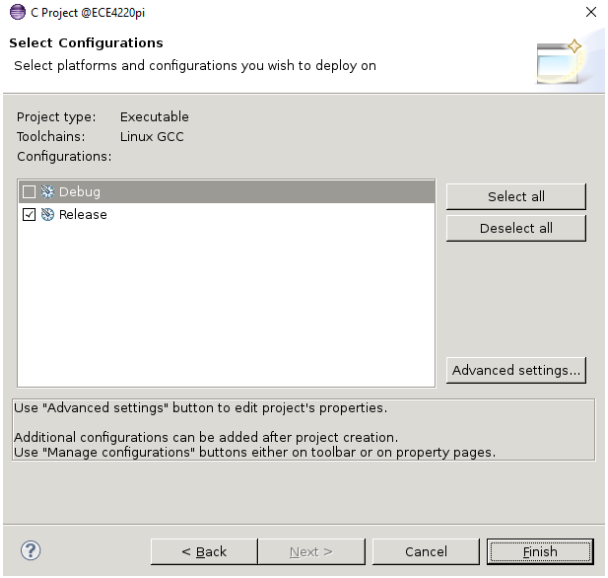

Figure 3: Build Options

(d) You may see the pop up shown if figure-4. If you do, click **Remember my decision** and click **Yes**.

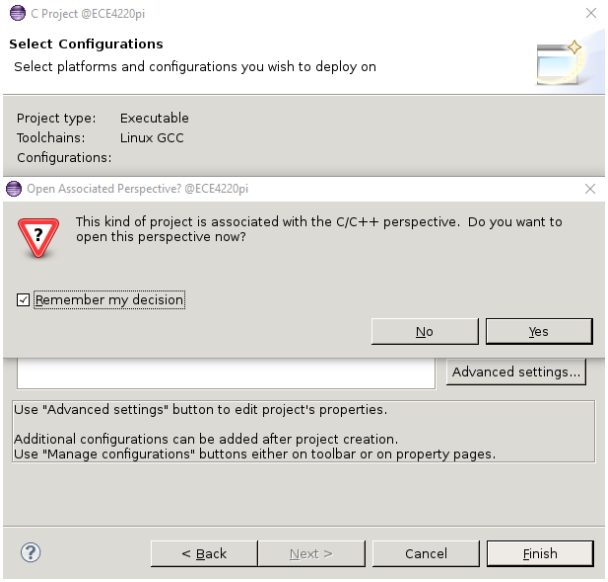

Figure 4: Select Configuration

(e) You have now created a project in Eclipse.

(f) Now you can build your project by clicking the hammer on the top toolbar of the IDE. For simple programs (such as a "Hello World" one), you can run your project from the Eclipse in-built terminal. However, for most programs it is better to run them from a Linux terminal. To get to your project you can use the **cd** command in the terminal and your executable will be in **/home-/pi/workspace/testProject/Release/** and the executable name will be the same as the project name (by default; that can be changed in the settings).

Part 3: Program Development with Eclipse on Local Machine

1. To open Eclipse on local machines, either type: eclipse & in the terminal shell or search for "eclipse" in the **Ubuntu-dashboard**.

2. To create projects, follow the same steps shown in the previous section. The paths may be different, as you would be running Eclipse on the local machine, not on the Raspberry Pi.

You can use Eclipse or any other editor on the local machine to edit your source files. If you are creating programs to be run on the Raspberry Pi, you can copy/sync the files (and even Eclipse projects) from the local machine to the Pi. Then, you would compile and run on the Pi. To do this, you need to open up a terminal that is looking at your local machine. You then can use the rsync command below to sync over the files to your Raspberry Pi.

rsync -av SOURCE DESTINATION

where,

SOURCE: The location of files to sync on local machine (say: /home/students/workspace/testProject)

DESTINATION: The location you are sending the files to on the Raspberry Pi (say: /home/pi/workspace)

Example: rsync -av /home/students/workspace/testProject pi@RASPBERRYPI_IP:/home/pi/workspace

Then, on a terminal where you are logged in to the Pi, you can search for the files/directories that you copied. You can compile your c/c++ files using gcc/g++ directly on the terminal.

Helpful Materials

1. You can find documentation on the Raspberry Pi here: https://www.raspberrypi.org/documentation/

2. On Canvas you will find manuals and other documentation on the Pi. These manuals contain information about the location of ports and interrupt registers. Go to **Modules→Other Material**

3. There are extra files and binaries that will be referenced in the Labs. These will be posted on Canvas, as well. You will need to download them and then copy them to your SD card.

Some additional considerations

Throughout the semester, you will need to create Kernel modules, include additional libraries, etc. A few things to keep in mind:

1. When creating a kernel module, you CANNOT include any libraries. Modules CANNOT use *printf*, or *fopen*, or most of the otherwise usual functions in C/C++ because most of these functions are NOT available to the kernel itself (of which the modules are part), but only to the high-level user applications. All function calls available to modules are functions of the kernel itself and those references are resolved during the operation of *insmod* (install module) – not during linking (i.e. gcc).

2. When you are using *pthreads*, you need to add the parameter: **–lpthread** to your linker options when you build your project. To do this in Eclipse, go to the **Project-Properties**, and under where it says **Linux GCC Linker** click on **Libraries**. On the right this will bring up places to add libraries to link to and their paths. Simply click on the add button under the libraries (denoted ***–l***) and type in *pthread*. The pthread library is in a known location so we do not need to add a path of where to find the library. Also make sure in your program you include the header: *pthread.h*.

3. A library is referred to by an option in gcc called "*-l*". That is, when you say "*gcc ... -lpthread ...*" you are basically telling gcc to include a library called *libpthread.so*. If that library cannot be found in the usual system directories, you may add another option: "*-L*", with which you provide a path (directory) where that same library can be found (e.g. "*... -L/home/mylib ...*").