Lab 04

First Semester 2014 Sharat

CS 251: Lab 04: [Code Warrior] Makefiles, compilation and linking

- Handed out: 15/08 Due: 18/08 Monday 11pm
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on the web page.

Overview

In this lab we will learn about "Makefiles", compiling and linking to libraries in Linux.

A large project has many components, possibly written by different group members, or even by people on the Internet. To produce the final executable, it is unnecessary to recompile code that has already been previously compiled into an intermediate stage. This was valuable when compilers were slow, but even today, it is very relevant (especially when we have optimizing compilers.) A file that tells the compiler what to compile when things change (i.e., when development is ongoing), and what not to, is called a Makefile.

The process of producing an executable involves intermediate files called "object files" and libraries. The final executable involves complicated shared libraries in today's run time environment.

Pre-tasks

1. All exercises in Lab 04 are to be done from the terminal. Before coming to the lab you should learn how to use the terminal and the command line interface (CLI) in Linux. Do not use the GUI for things like renaming, moving, copying, searching files. Read the first few chapters of Learing the Shell

   Now is the time to pick an editor that you are going to be comfortable with. (We heartily recommend the mother, father, and grand parents (so to speak) of all editors, `emacs` but we will be just fine with any other editor that you might want to use.)

2. Please also read the information in Tutorial 4 section of the web page. That's most of all you need to read for the lab, so don't scour the Internet and get confused!

3. Also to do before coming to lab is to place your submission for Lab 03 on the web. Link your Lab 02 HTML page to Lab 03 HTML page (any way you like). What will you put in the Lab 03 HTML page? Whatever you submitted to Moodle but in an unpacked form (i.e., non-zip, non tgz form).

   The purpose of this "linking" process is that when some one comes to the viva in the lab, the teaching team can get a complete summary of all your labs (so that we can ask questions).

4. (Important!) Also to do before coming to lab is a picture of an object that you are going to build in Box2D. This is going to be some sort of mechanical Rube Goldberg like machine simulation has to be on this page. Put a diagram to the actual machine internals that you want to design.

In order to do this, you will need to read about the machine that you want to design. Do not forget to put links to your sources.

Look at this example for inspiration. There are several others on the Internet. Box2D Mountain Bike You can discuss your proposed design with anyone you like, there are no restrictions. However, designs must be sufficiently different and once frozen, no discussions are permitted.

You must make the diagram in Inkscape and save it as a image file. Your design must contain at least 10 different moving elements. The base code has 7-8 elements. Note that your final design for the project can be different in details from this preliminary design. Again, do not forget to add a list of all the sources that you referred to for learning about the machine and Box2D

Create a page called `mydesign.html`. Link this file when you come for Lab 05 (i.e. for the next lab). However, show this file in a home area other than your web area to your TA when you meet him on Monday.

You should get the idea now. When you have come to Lab 04, you have already submitted Lab 03 to Moodle but you are linking up the page NOW. When you come to Lab 05, you would have already submitted to Moodle your Lab 04, but you will link it on that Monday when you have Lab 05.

Why are we doing this one lab apart? One, we want to ask viva questions. Two, we don't want your current Lab to be revealed to other groups. That is, placing on a public page comes only a few days after submission. Viva will cover all contents *except* the current lab (in this example, Lab 04).

The Tasks

1. Move the original `Makefile` present to `Makefile.orig`, and study it. Be prepared to ask intelligent questions in the lab.

2. You will now create your own makefile having some desirable property. Name your file as `makefile_groupXY`. For example, `makefile_group07` or `makefile_group13` (note the leading `0` in the first group).

   To get to the desirable property, perform the next few sections. (We will use upper case and lower case interchangeably for `makefiles` but you should use lower case).

3. Add a target so that it creates folders named `myobjs`, `mylibs` and `mybins` in the `cs251_base_code` (if there are no such directories present). Do not create the directories manually, and your Makefile will help create directories ONLY if they are not already present.

4. Before we build our final target, we want the makefile to check if Box2D is installed in external/include and external/lib. If it is not installed, the makefile will help automatically untar the source `Box2D.tgz` in the `external/src` folder and install Box2D.

   We emphasize this. The makefile will make this happen. You will NOT `cd` to the directory and actually perform this action.

5. Declare a target and the recipes for compiling the source files present in `src` to object files and place them in the `myobjs` folder.

6. Declare a target "executable", and the recipes for generating the executable from the object files. This executable must be called `cs251_exe_groupnumber`. If your group number is 1, then the file is called, `cs251_exe_01`, if it is 23, then it is called `cs251_exe_23`.

   At this point, you want to test that everything is going as per plan. That is to say if you hand the distribution to your partner without any object modules or executable and without Box2D installed, the Makefile should enable the partner to run the program as in the previous lab but with a simpler `make` command.

   Now for something different.

7. Declare a variable `STATIC_LIB`. If this variable is set to `TRUE` then the makefile must help compile all the files except `main.o` into a static library called `libCS251.a` and place it in the `mylibs` folder. Define a target for this purpose.
8. If `STATIC_LIB` is not `TRUE` then the makefile must help compile all the files except `main.o` into a **shared** library called `libCS251.so` and place it in the `mylibs` folder. Define a target for this purpose.
9. In summary, depending on the value of `STATIC_LIB` the appropriate library target is called. Now define another target called `exelib` that uses the libs produced in the previous steps to generate the executable – you will have to link `main.o` with the libraries to get the executable. The library path must be passed to gcc using the `-L` linker flag (check this out in the original given Makefile) and the name of the library can be specified by using the `-l` flag like `-lCS251`. This executable must be called `cs251_exelib_groupnumber`. If your group number is 1, then the file is called, `cs251_exelib_01`, if it is 23, then it is called `cs251_exelib_23`.
10. Finally test your Makefile by typing the commands `make <target-name>` and ensuring that the two executables (defined above) and the appropriate library is generated as per the value of `STATIC_LIB`.
11. Add a `clean` target to your makefile that removes all the generated files and leaves the `cs251_base_code` directory in its pre-compilation state.
12. Add a `distclean` target to your makefile that does everything that the above clean target does, and removes the Box2D install and untarred source files, leaving only the `Box2D.tgz` intact.

Files To Submit

Submit a `tar.gz` of a folder containing ONLY the makefile and the usual readme.txt file and nothing else.

- Inside the folder the makefile should be named `makefile_groupXY` for example `makefile_group07`, `makefile_group13` etc.
- The README file should mention the name of the targets for each task. For example for task 3 the name of the target may be `setup`. Also mention the usual information in README file like group members, group number and name, honour code, citations etc.
- The folder and its compressed version should both be named `lab04_groupXY_final` for example folder should be named `lab04_group07_final` and the related `tar.gz` should be named `lab04_group07_final.tar.gz`

How We Will Grade You

Points will be assigned for all the tasks, which have been gathered into three buckets. However, the time you take for each bucket below might be different and we will take that into account. We are not putting the exact distribution of points within each bucket, because we believe that all of this is important. But to give partial marking, we will be assigning some marks for subtasks so that we don't have a zero or 100 situation. The last bucket carries more weightage than any of the other two buckets

- Setup and creation of executable : This includes task 1 to 6. We will run your makefile to check if the relevant targets are created and they perform the required functions correctly.

- Clean and distclean : This includes task 11 and 12. We will run `make clean` and `make distclean` to check if the relevant generated files and installation is removed correctly.
- Runtime and Libraries : This includes task 7 to 10. We will check if appropriate static/shared libraries are created based on the value of `STATIC_LIB` and whether they are linked to create the final executable.

Final note: To be a software developer, you need to know only the very basics about Makefiles (my opinion). To be a code warrior, however, Makefile is a necessary tool to understand, if not master. (Replace "software developer" by HSC and "code warrior" by JEE Advanced.)