**09 – Building a FOSS Project with**

**the GNU autotools**

**Activities**

COMP190 – Tools and Techniques for Software Development

Dickinson College

Fall 2020

Prof. Grant Braught

In this activity you will put together a lot of what you’ve done thus far in the course to work with a real open source project – Freeciv – “a Free and Open Source empire-building strategy game inspired by the history of human civilization.” If you are familiar with the game Civilizatoin, Freeciv is an independently created FOSS version of that game, much in the same way that LibreOffice is an independently created FOSS version of MS Office.

You’ll fork and clone a real open source project (well since this is a course, you’ll fork and clone my fork of it). You’ll build the project… well eventually you will… there will be a few challenges along the way. The challenges you’ll encounter are very common in trying to get setup to make contributions to an open source project, so having some experience with them will be helpful. In the process you’ll learn about the traditional Unix build tools (autogen, configure, make), and some techniques for resolving missing dependencies. Finally, you’ll add a new feature to the program and make a pull request (to my fork) for it.

**Freeciv:**

Freeciv is FOSS and thus is developed by a community of volunteers. The Freeciv web page is a good place to begin exploring this project.

1. Visit <http://www.freeciv.org/> and explore its contents to answer the following questions:

a. When was the latest news item posted? What version of Freeciv is it about?

b. How many different languages has Freeciv been translated into?

c. Who are the current project administrators?

d. What is the # and title of the most recent bug (not feature) that has been reported on the issue tracker. Hint: Be sure you are looking at the “Issues” and not the “Overview.”

e. Under what license is Freeciv released? Hint: Navigate to the GitHub site and look for the COPYING file.

f. What is the URL of the Forum where Freeciv community members can ask questions of each other?

g. What is the main heading on the page that is intended for new developers that might be interested in contributing to the project?

**Starting Clean:**

When you have decided to start working with a new project it can be beneficial to start with a clean install of your OS in Virtual Box. In Activity 05 question #1, before learning about system administration and installing software you took a snapshot of the system. This was so that if things went wrong you could revert to that clean state. We will make use of that snapshot now so that we have a clean state to start from in working with FreeCiv.

2. You can revert to snapshots that have been taken using the VirtualBox Manager.

Graphical user interface, text, application

Description automatically generated

a. Ensure that you do not have any virtual machines running.

b. Click on your VM in the left pane.

c. Locate the snapshot you created in A05 #1 in the right pane of the manager. It will likely be at the top of the list.

d. Click on that snapshot.

e. Click on the “Restore” button at the top.

i. There will be a check box to take a snapshot of the current state before reverting. I recommend checking this box so that you can go back to the state of the machine as it is now if you would like.

f. When you have restored your clean snapshot from A05, click on Current State and start the virtual machine.

g. There is no response required for this question.

3. Confirm that you are in a clean install. None of the code for the Calculator should be present and the git command should not work. What output do you see when you run git?

**Update and Upgrade:**

It is also a good idea to update and upgrade all of the software on the VM before going into a new project. This can reduce some of the complications due to differences in OS versions.

4. First you’ll want to ensure that all of the packages installed with the apt (or Synaptic) package manager are also up to date. Recall from A05 that the following two commands do this:

sudo apt-get update

sudo apt-get upgrade

Run these commands and wait for them to complete (it could take a few minutes!).

5. Use the software updater (orange swirly thingy in the bottom right):



a. Choose “Install updates.” Installing the updates may take a while (10 minutes) depending upon the speed of your internet connection. So you might want to shift to doing some other work while that happens rather than just sitting and waiting.

b. It is not necessary to view the logs or to do anything with the window that opens with the CD.

c. Reboot the system when you are asked.

6. Doing the updates sometimes means that some packages will no longer be needed and can be removed. Run the apt commands from #4 again to see which ones, if any can be removed.

a. List any packages that are no longer required.

b. Remove those packages. What command did you use? Hint: Read the output from the upgrade command.

c. Now when you run the upgrade command you should see:

﻿0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

This tells you that your machine is fully up to date with the latest software.

**Getting (or git-ing ☺) Freeciv:**

7. You’ll be using git to get and manage the Freeciv code so it needs to be installed and configured.

a. Use apt-get to install git. What was the full command that you sued?

b. Configure git. Recall from Activity 07 that we used the following commands:

git config --global user.name <GitHubUserName>

git config --global user.email you@email.com

git config --global push.default simple

git config --global credential.helper cache

Note: If you use save instead of cache in the last command git will remember your password indefinitely so you will only have to enter it once. But be sure that you are the only one that will be using the machine if you opt for this.

8. Later in the activities you’ll be using the apt-file command. This command is not installed by default so we need to install it.

a. Type apt-file on the command line. What output is generated?

b. Install apt-file. What is the full command that you used?

c. At the end of the output from the install you will see a message about updating the cache. Update the cache. What is the full command that you used?

9. Go to the COMP190 Freeciv fork on GitHub:

* <https://github.com/dickinson-comp190/freeciv>

a. Create a fork of the COMP190 Freeciv repo into your own GitHub account. What is the URL of your fork?

b. Clone your origin Freeciv repo from your GitHub to your LinuxLite virtual machine. Freeciv is fairly large project so this may take a few minutes. What is the full command that you used?

c. Set the upstream repositories of your local repository to point to the COMP190 Freeciv repo. See A08 for the commands. Give the output a git remote -v command as your answer to this question.

10. Take a snapshot of your VM at this point. That way if things go really sideways during the build or install you can simply restore to here and you will not have to do the lengthy updates or reinstall git again. You should feel free to use snapshots as you move through this activity. But please note that each snapshot will likely consume multiple gigabytes of disk space.

a. What did you call your snapshot?

b. Paste a screenshot of the VirtualBox manager showing your snapshot here.

**Building Freeciv:**

As described in the video for today, many Unix projects (including FOSS) will use the standard automake tools to build the project’s executable programs. In this section you will walk through the process of building Freeciv from its source code using these tools. While in theory it is a simple straightforward process, in practice it rarely goes smoothly. Most commonly there are dependencies (other programs and software libraries) that the project uses that are missing and need to be installed. We’ll see plenty of those and learn some of the techniques for resolving them.

Recall from the video that the build process proceeds in three stages (autogen, configure, make).

*Autogen:*

11. The first step of the process is run the autogen.sh script.

a. Notice that autogen.sh ends with sh which was the extension we used for shell scripts. In fact, it is just a shell script. Use head to look at the top of the autogen.sh file.

i. What is the first line of autogen.sh?

ii. What interpreter does autogen.sh use?

b. Answer the following questions based on the video content for today:

i. What two files does autogen.sh read as input?

ii. Do these files exist in the freeciv directory?

iii. What two files will autogen.sh create as output?

iv. Do either of them currently exist in the freeciv directory?

c. Run the command: ﻿

./autogen.sh --no-configure-run

What is the final line of output from the autogen.sh script?

A few notes on autogen.sh:

* + Notice that a few of the lines of the output tell you that the script is running autoconf and automake.
  + The command line argument –no-configure-run is optional. Without it autogen.sh will continue on and run the configure script immediately after creating it. We separated it here to more clearly illustrate the process.

d. Do the files that autogen.sh was supposed to create exist in the freeciv directory now? If not revisit parts b and c of this question.

*Configure:*

12. Now that autogen.sh has created the configure script and the Makefile.in file we are ready to run the configure script. Just a little warning… this is not going to go smoothly… But we’ll try to control the frustration and we will get it working by the end.

a. The last line of the autogen.sh script said to “type configure”. What happens if you enter the command: configure? Let this be the first less in… you can’t take everything you read in the output of the automake tools literally…

b. Run the configure script using the command:

./configure

The script will run for a bit and then it will exit with an error. What error is reported?

c. What is the line of output just before the error message?

*Installing Missing Dependencies:*

Congratulations… you’ve just encountered your first missing dependency. We can use the apt package manager to install the missing dependencies. So that’s not the challenge. The challenge is to know which packages need to be installed. We could read the INSTALL document in the repository and try to get them all – which sometimes are helpful. Or we could run the configure script, as we did above, and it will tell us what dependencies we are missing. We’ll take the second approach as it is usually the most effective.

13. From the configure script output it should be clear that we need to install a version of the SQLite3 library that is at least 3.0.0. But we need to know what package that is in.

a. The apt-file command is useful for finding packages. Use the command:

apt-file search -l sqlite3.pc

(note: l is a lower case L, not a one, also don’t forget the .pc on the end.)

What packages are listed in the output?

b. Very often the things that a project will need to build are libraries. In Unix those libraries almost always begin with lib. In addition, the libraries that are required for development (i.e. building software) will end with the suffix -dev. Which package do we need to install the SQLite3 libraries for development?

c. Install the SQLite3 development libraries. What is the full command that you used?

14. After installing each missing dependency you will run the configure command again in hopes that all missing dependencies have been resolved – and eventually they will be…

a. Run the configure script again. What error messages are displayed this time?

b. Use apt-file to search for the package for the libcurl. What is the full command you used?

c. What packages are listed in the output?

d. Here there are 3 libraries and all of them are for development. Which to use? In this case it does not matter because these are just three different implementations of the same library. So install the one that uses Open SSL. What is the full command that you used?

15. You know the drill…

a. Run the configure script again. What error messages are displayed this time?

b. Find the package needed and install the dependency. What is the full command that you used?

16. You know the drill…

a. Run the configure script again. What error messages are displayed this time?

b. Did you receive the same error message as you did in question #15b?

c. If you answered no to part b great (but keep reading and complete this question anyway). If you answered yes, you have not installed the correct package to resolves the dependency. Even if this didn’t happen to you this time, it will happen sometimes. In these cases, you will just want to remove the incorrect package and try another. Use the man page for apt to figure out the command to would remove the package you installed in question 15b. Give that command here.

d. If your answer to part b was yes, use the command you determined in c to remove the incorrect package. Then return to question 15b. If your answer was no to part b, continue on to the next question without removing the package (or if you want to try your command from part c, run your command to remove the correct package and then go back to 15b and reinstall it).

17. You know the drill… no, no, no more please! Okay… I hear you… so confirm that the dependency above was resolved by running configure again. The last line of the output should now be:

﻿configure: error: can not build any of the supported clients

As long as that is the output, go on to the next question. If it is not, revisit questions 15b and 16 until it is.

18. Clearly resolving dependencies can get to be tedious, but when you have to get a project working there is really no choice but to persist. If you do you will get there. I did and I did. So, I’m going to save you some trouble. Below are the package names for the remaining dependencies that need to be installed:

libgtk-3-dev

libsdl2-dev

﻿libsdl2-mixer-dev

﻿libsdl2-image-dev

libsdl2-ttf-dev

libsdl2-gfx-dev

Install each of these dependencies. After installing all of these dependencies the configure script should complete without error. The output should end with a screen full of information that starts with:

﻿\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Configuration Summary \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If so, great you can move on to the next question. If not revisit the packages in this question to be sure each of them is installed correctly. Or if the configure script is indicating a different missing dependency use the apt-file command to find the appropriate package and install it. Once you see the Configuration Summary information, go on to #19.

*Make:*

19. The configure script has now completed.

a. What file is supposed to be produced by configure?

b. Does that file now exist in the freeciv directory? If so, go on to #20. If not, revisit question #18.

20. The next step is to use the make command to compile all of the code in the project into the executables that make up the game.

a. Run the command:

make

b. Go get a cup of tea or a soft drink… Freeciv is a big project so building it the first time will take a little while (10 minutes or so). Remember, back in the topic on compiled versus interpreted languages, when they said interpreted languages can be faster for development because they don’t have the extra compile step, well this is what they meant. It never seemed too bad to compile small Java programs in Eclipse, but it can take quite a while to build a larger project.

c. Make should terminate without any errors. If an error is reported run the commands:

make clean

make

d. If the error persists, return to question 18, run configure again and ensure that all of the dependencies have been satisfied.

e. If the error persists, get in touch so we can work through it.

**Installing & Running:**

When make has completed the executable files (binaries) will have been created. The final step will be to install the binaries into the system folders.

21. The Makefile also has the instructions for how to install the program. Running make with the argument install will install Freeciv into the system folders:

make install

22. The executable program ﻿freeciv-gtk3.22 has been installed into a directory that is on your PATH variable.

a. What directory has the program ﻿freeciv-gtk3.22 been installed into? (Hint: remember witch)

b. Run the freeciv program.

i. Click Start New Game

ii. Click Start

iii. Reduce the size of the window and then and paste a screenshot of the running program here.

Note: If you are used to playing Civilization with others you can do that with Freeciv too. You just need to run the server (freeciv-server) and connect to that rather than running a local game. All the details are in the docs on the Freeciv website.

c. In preparation for the next section let’s look at the sound options dialog (from the game menu choose options and local client. Then click the sound tab at the top). Notice that in this sound tab there is no volume control. Paste a screen shot of the sound options dialog here.

**Adding a Feature:**

23. Find issue #885685 in the Freeciv issue tracker. Hint: Use the search feature.

a. What is the feature that was added by this ticket?

b. Who added this feature?

c. What is the status of this ticket?

24. When the ticket is closed it means that the issue has been resolved. In fact, the code for the volume control has been merged into the real production version of the program.

a. Which 4 files were changed when this feature was added?

b. You can see the exact changes that were made by clicking on the link to the patch file (audiopatch.patch). A patch file is similar to a pull request and is just another way of submitting changes that some projects use. I find it a lot easier to see what was done when looking at the diff in side by side mode. The original file will be on the left and the new file will be on the right. The lines with filenames on them indicate that the following lines are in those files, and the line numbers on the left of each column are the line numbers in either the old or the new file.

i. What line number was changed in the audio\_sdl.c file?

ii. What line of code was added to line 163 of the options.h file?

25. Add the code for the volume control to the cod in your local Freeciv repository:

a. Create a new feature branch

b. Add all of the changes from the audiopatch.patch

NOTE: The line numbers in our files will be a little different than those in the audiopatch.patch file due to changes that were added after that patch was made. With a little effort you will be able to identify the correct locations to insert the changes.

c. Run make to build the project with these changes.

d. Running make will produce a new executable the client directory.

i. Change into the client directory

ii. Run the freeciv-gtk3.22 executable that is there using:

./freeciv-gtk3.22

d. Open the sound options dialog. Notice that it now has a volume control!

e. Paste a screen shot of the sound options dialog here:

26. Make a pull request for your changes to the COMP190 upstream repo.

a. commit the changes

b. push the branch to your origin repo on GitHub

c. Make a pull request for the changes

**Optional Extras:**

27. We use the autogen.sh, configure and make tools but we did not go into how to create the input files configure.ac and Makefile.in for use with the autotools. In practice, it is unlikely that you’ll have to make these for a new project because most new projects use more modern build tools. But, you never know, and if you are curious *The magic behind configure, make, make install* on thoughtbot by George Brocklehurst is a gentile introduction and a pretty short read:

* <https://thoughtbot.com/blog/the-magic-behind-configure-make-make-install>

29. The automake tools are just one, an admittedly old one, but not by any stretch an irrelevant one. But there are a ton of other build systems that can be used by projects and it might be worth it to at least know the names of a few of them:

* <https://en.wikipedia.org/wiki/List_of_build_automation_software>

30. I’ll call out the meson build system here specifically. Not because I think it’s the best, but because Freeciv seems to be migrating to it – though it does not work yet! So, I’m sure they would welcome contributions in that advance them in that direction.

* <https://mesonbuild.com/>
* <https://mesonbuild.com/Tutorial.html>