Operating Systems

Practical 5

Installing Software from Source-Code:

Example Gnuplot

Dr. Kevin Farrell

March 2025

Table of Contents

Legal... 4

1. Introduction and Objectives...5

2. Preliminaries: Installing Dependency Packages: the X11 Development Libraries...........6

3. Obtaining Gnuplot.. 8

4. Uncompressing and Extracting the Source-Code Tarball..10

5. Configuring the Source-Code/Package...11

6. Compiling the Source-Code..12

7. Installing the Resulting Application..13

8. Testing the Resulting Application...14

List of Figures

Figure 2.1: Select the "All" package types on the top left, and then type "libx11-devel" in the "Find" box. Tick the box beside version 1.7.0 Release 1.mga8 of"libx11-devel, and click apply.....7

Figure 2.2: Answer OK to install the additional "Dependencies" packages.....8

Figure 8.1: Ref.: www.gocomics.com/foxtrot.....16

Legal

[1] http://creativecommons.org/

[2] http://creativecommons.org/licenses/by-nc-nd/3.0/

1. Introduction and Objectives

One of the most common tasks that System Administrators and Software Developers do is the installation of software from source-code – either code they have written themselves, or source-code of packages that they have downloaded from the Internet.

In this practical, you are going to learn how to install a package from source-code onto your Linux system. The package in question is Gnuplot, a handy plotting tool. However, the methodology you will use, applies generally across the range of source-code packages available on the Internet.

Once, you have dealt with some preliminaries, there are a number of steps involved in this procedure, summarised as follows (the detailed steps are given later):

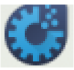
1. Obtain the source-code package, usually in compressed “tarball” format, and save it to a location in your home directory
2. Uncompress and extract the tarball
3. Configure the source-code/package (do as an ordinary user)
4. Compile the package (do as an ordinary user)
5. Install the resulting executable programs and associated files (libraries, manual pages, etc.) onto your system, for use by others (do as root)
6. Test that the procedure was successful (as an ordinary user)

2. Preliminaries: Installing Dependency Packages: the X11 Development Libraries

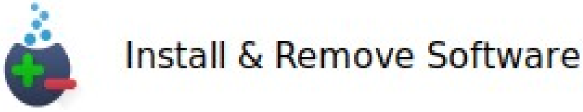
2.1. Boot your Mageia Linux guest Virtual Machine, and check that you have Internet access from it also. If you do not, fix this **before** you proceed. Ask for assistance if necessary!

2.2. In this section, we will show you how to install the X11 development libraries, which are needed by gnuplot so that it can display plot windows. These development libraries are part of the **X Window System**[3], which is the software responsible for the GUI on your Mageia Linux operating system. As gnuplot doesn't come with functions to display windows, the development libraries of the X Window System (which contain these functions) need to be **linked** with the gnuplot **object-code** during the linking phase of the compilation/linking that you are about to undertake in the next section. This is a common practice in software development, and such libraries are commonly referred to as “dependencies”. This will mean that the resulting gnuplot executable program will have the capability of displaying plot windows on your system.

2.3. Open the Mageia Control Centre by clicking on the button on the panel that looks like this:



2.4. In the main Mageia Control Center window click on the “Install and Remove Software” icon:

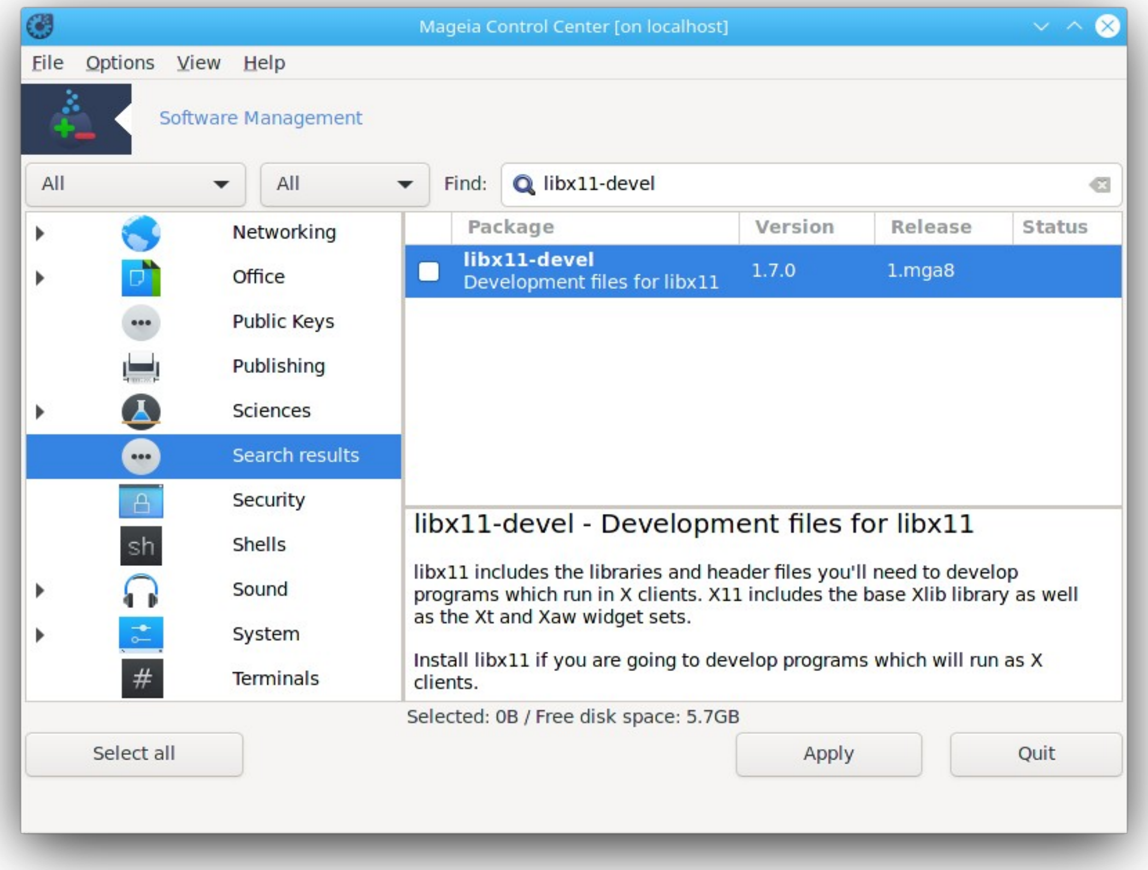


Install & Remove Software

[3] The “11” in X11 refers to the major version number of the X Windows System software.

2.5. At the top left, you should see a drop-down menu which says “**Packages with GUI**”. Click on this and select the word “**All**” in ***this*** drop-down menu (***not*** in the box to the right), so that all package types will be visible. You should then have two drop-down menus, both of which say **“All”**.

2.6. Then, in the “Find” box, type **libx11-devel** and hit the return key, which should show you this package - see Figure 2.1 below.



Find: libx11-devel

Package	Version	Release	Status
☐ **libx11-devel** Development files for libx11	1.7.0	1.mga8	

libx11-devel - Development files for libx11

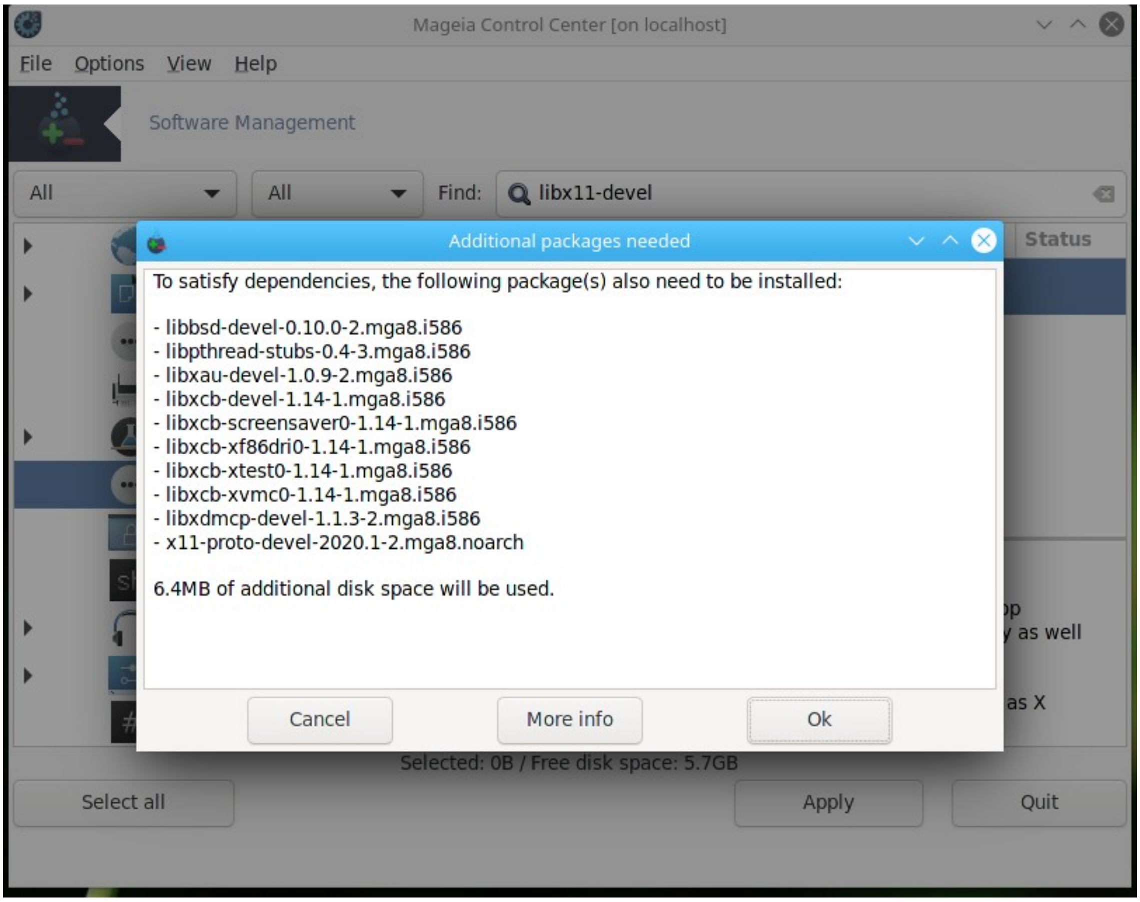
libx11 includes the libraries and header files you'll need to develop programs which run in X clients. X11 includes the base Xlib library as well as the Xt and Xaw widget sets.

Install libx11 if you are going to develop programs which will run as X clients.

Selected: 0B / Free disk space: 5.7GB

Figure 2.1: Select the "All" package types on the top left, and then type "libx11-devel" in the "Find" box. Tick the box beside version ***1.7.0*** *Release* ***1.mga8*** *of"libx11-devel, and click apply.*

2.7. Tick the box beside version **1.7.0** Release **1.mga8**, which should cause a new window to pop up asking you to confirm it is OK to install additional (dependencies) packages automatically selected – see below.



To satisfy dependencies, the following package(s) also need to be installed:

- libbsd-devel-0.10.0-2.mga8.i586
- libpthread-stubs-0.4-3.mga8.i586
- libxau-devel-1.0.9-2.mga8.i586
- libxcb-devel-1.14-1.mga8.i586
- libxcb-screensaver0-1.14-1.mga8.i586
- libxcb-xf86dri0-1.14-1.mga8.i586
- libxcb-xtest0-1.14-1.mga8.i586
- libxcb-xvmc0-1.14-1.mga8.i586
- libxdmcp-devel-1.1.3-2.mga8.i586
- x11-proto-devel-2020.1-2.mga8.noarch

6.4MB of additional disk space will be used.

Figure 2.2: Answer OK to install the additional "Dependencies" packages.

2.8. Answer “**OK**” to confirm that the dependencies will be installed. Then click “**Apply**”.

2.9. A summary box will appear asking is it okay to install the **libx11-devel** library along with several dependency packages. Simply click “**Yes**” to proceed.

3. Obtaining Gnuplot

3.1. Before you start this section, please download the “**OS Reflective Journal Template 2025**” file to your Linux system. Move it to your **Documents** directory, and rename it to:

```
Firstname_Surname_Practical5_ReflectiveJournal.odt
```

3.2. In the Exercises in the remainder of this Practical, please write your answers into your Reflective Journal on page 2 after the Table. When you have finished the Practical, please then answer the questions in the Table.

3.3. Using Firefox, download the **gnuplot-5.4.5.tgz** (Gnuplot Source Code Tarball) package from the **"Supplementary Resources"** sub-unit on the Operating Systems **Practicals** unit page on BrightSpace.

3.4. **Note: ALL of the remaining steps in this practical require you to use the command line (a "Konsole" terminal window), not the GUI.**

3.5. So, before you go any further, make sure you have a **Konsole** terminal window open.

3.6. By default, Firefox saves downloads to the **Downloads** directory. Using the Konsole, and from ***your*** **home** directory, move the file to your **programs** directory[4] by typing the command below. **Note**: if you haven't already created a **programs** directory in your **home** directory, please do so first.

```
$ mv Downloads/gnuplot-5.4.5.tgz programs
```

3.7. Incidentally, you can also obtain further information about gnuplot (including links to the latest and older versions) at the following link:

http://www.gnuplot.info

3.8. **Revision Exercises**: Write in your Reflective Journal, what the commands are for each task. If necessary, try looking up the answers on the Internet:

1. Command to create a directory
2. Command to move a file from one directory to another
3. Command to change directory

[4] Your home directory is /home/<username>. For e.g.: /home/kfarrell is the home directory of user kfarrell

4. Uncompressing and Extracting the Source-Code Tarball

4.1. Change into your **programs** directory, and list the contents to make sure you correctly moved the gnuplot tarball into it. Then, uncompress (gunzip!) and extract the tarball as follows:

```
    $ tar -zxvf gnuplot-5.4.5.tgz
```

4.2. **Exercise**: List the directory contents again. Write what you see into your Reflective Journal.

4.3. **Revision Exercise**: By reading the manual page for the **tar** command, write in our Reflective Journal, the explanation of each of the **four** switches: `-z, -x, -v, -f`, which you used with the **tar** command above. **Note**: In a manual page, you can use **/item** to search for **item**. Use the "n" key to search for the ***next*** instance in the manual page, and use the "N" key to search for the ***previous*** instance. When finished reading the manual page, hit "q" to quit.

4.4. If you have performed the tar command above correctly, you should have a new directory called **gnuplot-5.4.5**, which was created by the unzipping and then extraction of the tarball. The **gnuplot-5.4.5** directory is the main directory from which you will ***configure***, ***compile*** and run the ***installation*** program for Gnuplot.

4.5. Change to this new **gnuplot-5.4.5** directory, and list the contents. Let's point out a few important files in it:

- **README** = As the name suggests, you should read this file. It's normally the *first* file you would read, before you read anything else. As with all text files, you can use the **less** command[5] to read it. Use the **"q"** key to quit reading it with **less**, and return to the command prompt.
- **INSTALL** = You should (briefly) read this file next. It contains instructions on how to ***configure***, ***compile*** and ***install*** the program. You are not expected to understand all of this file – it is quite complicated for most second year OS students. So, the

[5] Use **"man less"** to read the manual page for the **less** command, and its usage.

aim here is just to get a small idea of what kind of information is provided to users.

- **configure** = The ***script*** for configuring the software. There is no need to read this file. It's also quite complicated, and the details of how it works are beyond the scope/level of the OS module.

5. Configuring the Source-Code/Package

5.1. You are now ready to configure the software. As stated above, the **configure** script is used to carry out this task. Essentially, what this script does is it checks to make sure your system has all of the required dependencies installed, and also determines where the application and associated files[6] will ultimately be installed; examples of these are, the GNU C compiler, various libraries, other programs that are required by Gnuplot. The default installation directory is **/usr/local**. However, this can be changed by the user by using an option with the **configure** command.

5.2. **Exercise**: What is the option to pass to the **configure** script to define an ***alternative*** installation directory, and how is it used? Hint: information on this is in INSTALL file you briefly looked at. **Note**: this is a challenging exercise! Write your answer in your Reflective Journal.

5.3. You will install in the default directory. To start the script, type the following:

```
$ ./configure
```

5.4. **Revision Exercise**: Why is the "**./**" necessary in the above command? Write your answer in your Reflective Journal.

5.5. After the script runs, you should be presented with a command prompt again. If there are any error messages, inform your Lecturer.

5.6. List the contents of the directory again.

5.7. You should now notice that there are some new files. In particular, notice that there is a new file called **"Makefile"**.

[6] Examples of associated files are manual pages, documentation, various libraries.

5.8. Makefiles are a sort of “recipe” file containing **three** essential elements:

- **Targets**: what you want to create
- **Dependencies**: what are needed before you can create the Target
- **Commands**: the commands that run to create the Target – once you have all of the Dependencies

5.9. The three elements together, **Targets**, **Dependencies** and **Commands** are referred to as a **Rule**.

5.10. A Makefile is ***interpreted*** by the **make** command. That is, the **make** command reads the Makefile and causes some action to happen.

5.11. In our case, some of the **macros** and/or **commands** in the Makefile, tell the compiler where to find the Gnuplot *C* code, and what specific flags/switches it must use with the compiler to compile that code; For e.g.: the “-O2” switch tells the compiler to optimise the code with “level 2” optimisations, which should make the resulting **gnuplot** program run faster (than if there were no optimisations).

6. Compiling the Source-Code

6.1. Now that you have configured your environment and the package, you need to compile the code. To do this, we will use the **make** command as described below.

6.2. Sometimes, one provides the name of the **target** after the **make** command in order to create that target. However, if you don’t provide a target-name, then the **make** command will look for the first “***target***” in the Makefile and try to create that. By convention, this target is usually called **all**. The convention is that users create a Makefile and have the first Rule (target, dependencies and commands) compile everything. Once this target is found, the “***commands***” associated with that target will run (assuming all of the dependencies exist!); i.e. all of the code will be compiled.

6.3. The compilation takes a while. To start it, simply, type:

```
$ make
```

6.4. After a few minutes, the compilation finishes, and you are presented with the command prompt again. If the compiler provides any errors, inform your Lecturer.

7. Installing the Resulting Application

7.1. With the compilation complete, you are now ready to install the application onto your system. This needs to be done so that *all* users will be able to run it and be able to access documentation and/or manual pages about it. However, you cannot perform the installation as an ordinary user, since ordinary users do not have the permissions to write to system directories, such as the default Gnuplot installation directory **/usr/local**. Only *root* (the System Administrator) can do that! Therefore, you need to log in as *root* from the command prompt, using the 'substitute user' command, su, as follows:

```
$ su -
```

7.2. Enter the root password when prompted: **qwerty** (on the Mageia 8 i586 32-bit system I provided) or a root password you set if you have your own Linux system. Once you have become *root*, notice how the command prompt changes from a **"$"** symbol to a **"#"** symbol. You can check to see what user you are logged in as by typing:

```
# whoami
```

7.3. Since you are now *root*, your current working directory is no longer the **gnuplot-5.4.5** directory. Instead, it is the home directory of the root user: `/root`

7.4. <u>**Revision Exercises**</u>: Write your answers in your Reflective Journal:

1. What is the command to print your ***current working directory*** on the console (screen)?
2. What ***is*** your current working directory?

7.5. Before, you can perform the installation, you need to navigate back to your own (ordinary **student** user's) **gnuplot-5.4.5** directory as follows:

```
# cd  ~student/programs/gnuplot-5.4.5
```

7.6. **Note**: there is NO SPACE between the “~” and your username[7].

7.7. Once you’ve done that, check that you are in the correct working directory. Then, install the application as follows:

```
# make install
```

7.8. Notice how you used the **make** command again. The new ***target*** this time is called `install`. It will read the Makefile, to find the target called `install`. If you see any errors, inform your Lecturer.

7.9. Once the installation is complete, logout as *root* by typing:

```
# exit
```

7.10. You should now be back to being an ordinary user again. You can check to see what user you are logged in as by typing:

```
$ whoami
```

8. Testing the Resulting Application

8.1. Congratulations! You have now configured, compiled and (hopefully) installed Gnuplot. However, as every good Scientist will tell you, unless you test it, your job is not done! The first part of that testing, is to see if the files exist in the required locations; i.e. in the installation directory. This task is encompassed in the next two exercises.

8.2. **Exercise**: Have a look in **/usr/local**. Write in your Reflective Journal, the names of the directories and files you see.

8.3. **Exercise**:

[7] The combination of the tilde symbol “~” and username are equivalent to the user's home directory; for e.g.: ~kfarrell = /home/kfarrell

1. List the contents of **/usr/local/bin**. Write in your Reflective Journal, the names of the file(s) you see.
2. What type of file would be stored in a “bin” directory in Linux?
3. What are the letters “bin” an abbreviation of in Linux?

8.4. List the contents of your **gnuplot-5.4.5** directory again. You will notice a directory called **demo**.

8.5. Change into the **demo** directory, and list the contents. You will see that it contains lots of **“.dem”** or demonstration[8] files for Gnuplot. One particular file allows you to run **<u>all</u>** of the demonstrations. It's called **all.dem**. Run this, by typing:

```
$ gnuplot all.dem
```

8.6. Immediately, a window should appear showing a simple plot. If it does not, inform your Lecturer. The usual cause of this is that you have not completed Section 2. correctly.

8.7. Also, notice that your terminal window now contains a message “Hit Return to Continue”. Clicking on the terminal window, might cause it to be brought in front of the plot window, which can be annoying. If you wish, to keep the plot window visible at all times, you can do the following (although this is probably not necessary with a widescreen monitor!):

Right-click the plot window’s title bar → Advanced → Keep above others

8.8. That way, when you click on the terminal window to hit the return key for the next plot, the plot window will stay above the terminal window and therefore be visible all the time.

[8] The “.dem” extension is completely arbitrary. The authors of the demos could have used any extension; for example: “.mylovelygnuplotdemos”, but such an extension would be a bit inconvenient! Linux/UNIX doesn't need extensions, unlike that *other* OS – MS Windows!!! ;-)

8.9. With the three-dimensional plots, try using the left and middle mouse buttons whilst moving the mouse on the plots to change their orientation and shape.

8.10. Well done! You have successfully completed one of the most important tasks performed on a regular basis by Software Developers and System Administrators – the installation of software from ***source code.***

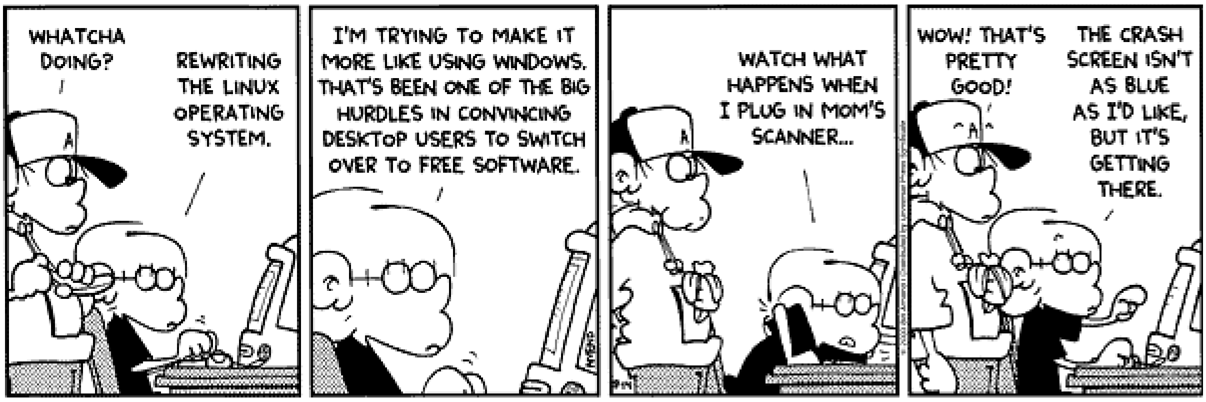

WHATCHA DOING?
REWRITING THE LINUX OPERATING SYSTEM.
I'M TRYING TO MAKE IT MORE LIKE USING WINDOWS. THAT'S BEEN ONE OF THE BIG HURDLES IN CONVINCING DESKTOP USERS TO SWITCH OVER TO FREE SOFTWARE.
WATCH WHAT HAPPENS WHEN I PLUG IN MOM'S SCANNER...
WOW! THAT'S PRETTY GOOD!
THE CRASH SCREEN ISN'T AS BLUE AS I'D LIKE, BUT IT'S GETTING THERE.

Figure 8.1: Ref.: www.gocomics.com/foxtrot