CIT 371 lab 16: Software installation continued and software maintenance

Parts of this lab must be done with the Web Console. See the Student VM Access document for information on accessing your VMs.

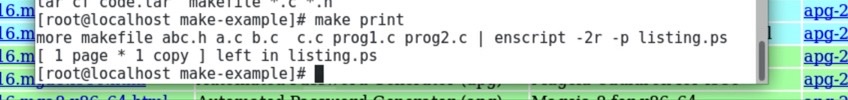
Log into Coivcenter, start your VM (VM2), and open a terminal window. Read about configure, make, make install in chapter 13 before doing this lab.

1. Installing Open Source software using configure, make, make install
2. *What does the* **configure** *instruction do? What does the* **make** *instruction do? What does the* **make install** *instruction do?* We explore this with an example. cd to ~, type **wget sappho.nku.edu/~huy1/CIT371/make-example.tar.gz**. We uncompress and untar this file using **tar xvfz make-example.tar.gz**. Read the tar man page. *What do the x, v, f and z all stand for?*
   1. **The configure instruction does is used to detect certain machine settings and set up needed files for make to do its job. This is meant to get ready and build the software on your system. The make instruction does GNU make utility to maintain groups of programs. This is used to build the software. The make install instruction will copy the built program, and its libraries and documentation, to the correct locations. The x stands for extract files from an archive, the v stands for verbose which will verbosely list files processed, the f stands for file=archive which means use archive file, and lastly the z stands for gzip and this allows for a filter the archive through gzip.**
3. Type **ls**, tar unpackaged files into its own directory, make-example. cd into make-example. Type **ls** to see the files here. There is a makefile file and C files (.c and .h). Examine the contents of makefile. *What do CC, LIBS, OBJS and SRCS represent (not what they store, but what type of information do they store)?* You might need to infer this after seeing what they store. These are typical variables used to define options for the gcc command. *What is gcc?*

**So, CC represents a variable to hold the gcc. LIBS represents a special library where we can hold special requests in this library. OBJS represents object files where they can store files objects. The SRCS represents source files that we can store the source of the files for this make-example. Gcc is GNU Compiler Collection.**

1. makefile has several different “areas”, denoted as **name: instructions**. In this example, these areas are “all”, “prog1”, “prog2”, “.c.o”, “install, “depend”, “clean”, “tar”, “print”. You can call an individual area using **make *area*** as in **make install** or **make tar**. This makefile is simple and only needs a make command. We will look at a larger makefile later. *What would you use* ***make tar*** *to do? Note: Please open the makefile and observe the content to answer this question. What is the output from* ***make print****?*

**We would use the make tar to do used to rip a collection of files and directories into highly compressed archive files. The output from the make print is below .**

****

1. Type **make all**. *What was the output? What new files have appeared?* To run one of these programs, type **./*name***. Run both new programs. *What was the output of each?*

**Text

Description automatically generated**

**This was the output. There was a lot of new files added such as prog1, and prog2. The prog1 output was function a, function b, function c. For prog2 the output was function b, function a, function c.**

1. Run **make install**. *What happens?* *Why?* su to root and repeat the **make install** instruction. *What is the output now?* *What does make install do (in this case)?*

**We are unable to install it due to not being the root user. We had to move the prog1 and prog2 files to /usr/local/bin in order to install the files. They also had to remove the .o , .c , .h , files so it can download. Make install allows us to install the prog1 and prog2 files in this case.**

1. Type **ls**. *What are the contents of the directory now? Why?* Type **which prog1**. *Where is it located?* Run prog2 by typing **prog2** (without the ./). *Why did this work whereas in step 1d, you had to type ./prog2?* Look again at makefile. *Why did you have to run make (or make all) before make install? Summarize the difference between make and make install.* Look again at makefile. *What would you use make clean for?*

**Code.tar, listening.ps, and makefile are the only thing left. This is because the rest of the files have been moved due to the make install, we just did. Prog1 is located in the /usr/loca/bin/prog1 . This worked due to the moving of the file, we can now run the file without the ./ . When we moved the file to this directory we are able to now run the file where need be. You have to run make or make all before make install because make is used to build the software. The make install instruction will copy the built program, and its libraries and documentation, to the correct locations. The difference between these two is that one will allow you to build the software while one will make you copy and move the software and install. We would run make clean if we wanted to package the source code.**

1. Another example. su to root for the remainder of this lab.
   1. Open Firefox, go to **https://github.com/Distrotech/apg**. Before continuing, type **which apg to** make sure this is no longer installed (if it is, type **yum remove apg**). From the web page, click “**Clone or download**”, from the pop-up window select **Download ZIP**, save the file. cd to ~/Download. A zip file has been saved there. This file is zipped, not tarred. Uncompress using **unzip apg-master.zip**, which creates the directory apg-master. cd into it, type **ls**. This installation contains more files. The make file is called Makefile but will still execute by the make command. Type **make**. Some warnings are issued by the gcc compiler, we can ignore them. A new program, apg, has appeared. *What color is the font and why?* Type **./apg**. *What is the output?* Type **make install**. Type **apg**.
      1. **The font that showed up is grey and has black, a pinkish/purplish color, and green.**
      2. *Text, letter

         Description automatically generated*
   2. *Why should we install software from its source code? Provide three reasons for doing this as opposed to installing software using yum or a GUI.*

**We should install software from its source code for multiple reasons such as it allows you to be specific and specify what you want to download, it allows you to modify the program as need be for your special occasion, or it can allow for a program to fix bugs and or add on features. We use source code over GUI because of the control we have over the download.**

1. We wrap up this lab by looking at software maintenance.
   1. Select **Applications** 🡪 **System Tools** 🡪 **Software Updates**. This program is similar to the Software program but in this case it lists only installed software titles that have updates available. *How many updates are available? What do updates do for you (read the sentence after the number of updates available)?* **Right click** on any entry and select **Unselect all**. Scroll down until you find The Linux kernel. **Select this** (click the checkbox). Select **Install Update**. This may take a minute or two, when done you will have 1 fewer updates and all packages will again be selected.
      1. **There are 836 updates. The updates will do package updates correct errors, eliminate security vulnerabilities, and provide new features.**
   2. Type **find / -name “\*.rpm”**. There are many rpm files under /var/cache/yum. These were automatically downloaded by yum. This allows us to use either yum or rpm to delete or update installed software. Read **rpm**’s **man page**. *What option do we use to delete a package? What option do we use to update a package? What does freshen do?*
      1. **We use -e erase option to delete a package. To update a package, we use -U. To freshen means to update a package but only to an earlier version that has been installed.**
   3. Using yum is far simpler as yum finds and downloads the rpm file for us and tracks down dependencies, installing needed libraries for us. *What yum options are available to update a package?*

We can run the command yum update package to update the package.

* 1. *Which approach should you use for updating installed software: rpm, yum, GUI, and why? Should you update software whenever updates are available? As a system administrator, how often should you check for updates? When might you not update software?*
     1. **You should use the yum approach when updating and installing software due to the simplicity and how you are able to manage what you can download with the rpm. You should update software sooner rather than later due to vulnerability attacks and security reasons, but if an update has something you do not approve of or have yet to look into, it is okay to understand more what you are about to update and download. As a system administrator I would be looking for updates weekly just to prevent the risk of attacks on our network or software. The only reason why I would say you wouldn’t want to update software is personal preference with how the software already is at but most of the time you want to be keeping an eye out on updates and staying updated.**

1. Occasionally you might want to delete software.
   1. *Under what circumstances might you want to delete any software?* In Linux, you do not want to just delete the executable file (probably stored in /usr/bin or /usr/sbin). *Why not?* 
      1. **You may want to delete software for test runs on other software, or a removal of a software for something new. You don’t want to just delete these files under /usr/bin or /usr/sbin because these types of files hold key executable files and are very valuable to your OS.**
   2. We should remove software using the GUI Software tool or yum. Select **Applications** 🡪 **System Tools** 🡪 **Software**, select **Applications** and then **GNOME Applications**. In your terminal window, type **which rhythmbox** (also known as “Music Management Application”). *Where is it located?* Notice in the Software GUI it is checked, meaning it is installed. We will delete it. **Select** this package in the Software GUI and click on **Remove**. *What icon appears next to it?* Click on the **Apply Changes** button. **Confirm** your choice and authenticate when asked. Once deleted, *how does this package appear differently from the others under GNOME Applications?* In your terminal window, repeat your **which** command. *What response do you get?*
      1. **It is located in /usr/bin/rhythmbox. The icon that appears next to it is the recycling bin. This package appears to be closed compared to open. When doing the which command again it comes back with the no rhythmbox in …. With all the possible options.**
   3. To delete using yum, the option is either erase or remove. Type **yum erase apg**. *Why can’t yum delete this package? Read yum’s man page. What does it say about using yum to remove yum?* We won’t actually delete anything using yum but its usually very easy.
      1. **Yum can’t delete this package due to the specifications on how we entered it and that when using yum it removes or deletes packages that are depended on the file. It talks about using remove yum and how much it does so we have to be specific and make sure we understand what we are deleting.**

Shut down your VM if desired, disconnect from the VPN if you are using it, and submit your lab report.