COMP 412, Fall 2022

A Brief Note About Tar Archives and the Autograder

In all three programming exercises in COMP 412, you will submit your code in the form of a tar archive file. In effect, you are preparing a distribution of your software. That distribution will be read and processed by an autograder — a fairly straightforward python program.

Many of you have not used **tar** before COMP 412. Many of you have not handed your software off to a blind evaluation system. This note is intended to help you ensure that your tar archive correctly encapsulates your project and to maximize the likelihood that the autograder finds, unpacks, and executes your code correctly. Points are deducted for submissions that require manual intervention.

Your submission should be a self-contained distribution of your code—that is, a file system subtree that has everything needed to build and execute the code, along with a README file that tells both a human and the autograder about the code. To create that distribution, you should:

1. Pick a directory to be the first-level directory in the distribution. It should contain a **README** file that specifies your name and your netid (see lab handout). It should contain everything required to build (if necessary) and to execute your submission.

   If your submission includes a **Makefile**, the **Makefile** should be in the first-level directory. The **Makefile** should implement the targets clean and build.

   > **make clean** should prepare the directory for the build target. Typical actions include removing the results of any prior build, such as **jar** files, **.o** files, and the executable itself.
   >
   > **make build** should build the actual executable. It should perform any compilation steps, link steps, and archive or library construction that the submission needs.

   If your submission includes an executable script (often needed for java or python codes), the script should be in the first level directory. It should be named appropriately (e.g., 412fe for lab 1) and have its permissions set to include owner and group execute.

2. Your code may live in the first-level directory. You may place the code in a subtree of the first-level directory. We view that as a matter of taste, driven primarily by the number of files you create for the lab.

   Your script or Makefile must reference the code in the distribution's directories. The autograder will create a fresh directory, buried deep in some subtree of **~comp412** on CLEAR; it will build and execute your submission in that directory. Most importantly, the autograder will execute as **comp412**, so it will not have access to your home filesystem on CLEAR. Thus, a script that references files in your home directory will fail.

3. Once you have created the directory, you can create a proper tar archive making the distribution directory your current working directory and executing the command:

tar cvf ../netid.tar .

using your own **netid** in place of the string "**netid**". The "v" option on the tar command will have it list all of the files it includes in the archive, by their pathnames. Those pathnames should be relative pathnames—that is, they should begin with "./". For example, we would expect to find "./README" in the list, along with either "./412fe" or "./Makefile".

To assist you in understanding how to build a tar archive that works with the autograder, we will provide you with a stripped-down version of the autograder. You should run the stripped-down autograder on your submission. It will provide you with feedback. Either it will report a score, in the **results** file, or it will produce error messages in the log file.

Note that when **you** run the autograder, it has access to your file system. (After all, you own your file system!) When **we** run the autograder, it does not have access to your file system. (After all, we are not you!) Thus, the autograder will not discover some common errors. Among these are:

An absolute path name in a shell script (e.g., **/storage-home/k/keith/412/lab1/412fe** rather than **./412fe**)

A relative path name that is not accessible to **comp412** (e.g., **~keith/comp412**)

Finally, your submission should contain what it needs to build and execute your code. It should not contain all your test files, copies of the test blocks or report blocks (later labs), testing scripts, or a complete archive of the language's library structure. Similarly, your build process should be self-contained. Several students in previous years have used tools that left large hidden subtrees in **~comp412**. While that works in your file system, it may break in the testing environment. In particular, if multiple students import conflicting files into the same hidden file system.

If you have problems, please report them on Piazza. You can use either a public post or a private post. We will see it either way.

Updated February 2022