COMP 412, Fall 2021

Lab 2

A Brief Note About Tar Archive Files 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.

   For some languages (e.g., python or java), the first-level directory will contain an appropriately named script (e.g., 412alloc for lab 2). For other languages (e.g., C, C++), the first level directory will contain a Makefile.

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, it will not have access to your home filesystem on CLEAR, so a reference to your filesystem (e.g., ~keith/comp412/lab2/src) will fail.

3. Once you have created the directory, you can create a proper tar archive by executing the simple command:

   tar cvf ../netid.tar .

   in that first level directory, 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 "./412alloc" or "./Makefile".

In previous years, a significant number of students have had trouble creating a tar archive that works with the autograder. To help you, we have added a tarfile verifier to the lab 2 code base. It can be found, on CLEAR, in **~comp412/students/lab2/verify.tar**. To run the verifier, create a clean directory in your own filesystem. Copy the tar file into the directory and unpack it with the command:

tar xvf ./verifier.tar

Copy your submission (the tar archive you have created) into the subdirectory "FilesToTest" and execute the verifier with the command:

./Verify

The verifier will create a new file named "REPORT" that highlights any problems that the verifier found. For a well-formed tar archive, the verify should report:

No error messages from unpacking the tar file
README specifies the correct name and netid
412alloc found with correct permissions

If there are problems in any of these three steps, the verifier tries to provide more information.

The verifier **does not** execute your code. It will not detect errors that occur when ./412alloc is invoked from the command line. Among the common errors it **does not** detect are:

A syntax error in a shell script
An absolute path name in a shell script (e.g., /storage-home/k/keith/412/lab2/bogus.c)
A relative path name to a file which **comp412** cannot access (e.g., ~keith/412alloc)

You need to ensure that your code works. We recommend that you unpack the distribution into a clean directory on CLEAR, far away from your source directory. Lock down your source directory with **chmod** (see "man chmod"). Build and execute your submission.

Finally, your submission should contain what it needs to build and execute your code. It should not contain all of 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 (e.g., a repository for a project named "lab2").

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 September 2021