CSE 391, Autumn 2019
Homework 8: Program Management and Build Tools (make)
Due Tuesday, November 26, 2019, 1:00 PM

This assignment focuses on using automated build tools such as `make`. **Electronically turn in TWO files: `homework8.txt` and `Makefile`.**

Task 1 of 2: Build and Run a Program

> *Self-Discovery:* SourceForge is a useful web site that allows developers to host their open source projects for free. SourceForge hosts each project's code and makes it available for any to download free of charge.

For this task you will download, compile, and run a piece of open source software from the web using `make`. The program is called **`jp2a`** (JPEG to ASCII converter). Here are the steps to follow:

1. Find the `jp2a` home page using your favorite search engine, and **download** the program's **source code** as a `.tar.gz` file **from SourceForge**. (The download page will have other files available for download, such as a Windows *binary* version. You **don't** want those files. Click "Browse all files" and look for the `.tar.gz` file, something like `jp2a-1.0.6.tar.gz`, probably in the 1.0.6 folder inside the `jp2a` folder)
2. Once you've downloaded the file, **decompress** it into the current directory.

   The `tar` command compresses/decompresses files in the Unix TAR ("tape archive") file format. TAR merges many files into a single archive; however, unlike ZIP, TAR does not compress the contents. Therefore most `.tar` files are then subsequently also compressed with a separate compression algorithm called GNU ZIP ("gzip"), which yields a `.tar.gz` file. (This format was used over ZIP because of patent issues.)

   To extract and decompress the `jp2a-1.0.6.tar.gz` file you will need to figure out the appropriate options to the `tar` command, which can be found in the `man` pages or the lecture slides.
3. After decompressing the archive, change into the directory just created and **run the `configure` program (see note below)** to set up the build process for your particular computer type/architecture. The `configure` program analyzes your system and produces a `Makefile` that will work for your computer to build and install the program.

   (*Note:* The `jp2a` compilation process requires that your computer have the `gcc` and `make` programs and an installed library called `libjpeg`. The CSE shared `attu` server and the CSE VM already have these, but your own Linux box might not.)

Normally `configure` sets up the `Makefile` to install the app to the directory `/usr/local/bin`, but since you may not have root access on the system you're using, you may not be able to install it there. We suggest you tell `configure` to install the app to your current directory, the directory where you decompressed the `jp2a` source files. To do so, run the command as follows:

```
./configure --prefix=$(pwd)
```

If it worked properly, you'll see several lines of output such as:

```
checking for a BSD-compatible install... /usr/bin/install -c
config.status: executing depfiles commands
```

4. Assuming that the `configure` program completes successfully, you are ready to **compile and install** the program using `make`. Run: `make`, and once it completes, run: `make install` . If each command works, it will output several mostly incomprehensible messages such as:

```
gcc  -g -O2 -o jp2a  html.o term.o curl.o jp2a.o image.o  -ljpeg -lcurl -lncurses
make[2]: Leaving directory `/homes/iws/zorahf/391/hw8/jp2a-1.0.6/src'
```

5. If `make install` worked properly, you should now have a `bin/` subdirectory within your `jp2a` source folder. In this folder should be a newly built executable called `jp2a`. You can **run the program** while in that directory by typing:

   `./jp2a` *`[options]`* `filename.jpg`

6. In order to make the `jp2a` command available to you regardless of your current directory you will need to add the directory that contains the `jp2a` executable to your `PATH`. This can be done by adding a line to your `bash_profile` that modifies your `PATH` environment variable. See the lecture slides for how to do this.

   If you have done this correctly, the command `which jp2a` should print the location of the `jp2a` executable on your machine. You can also now run the `jp2a` program from any location on your machine using the following: (note the `./` prefix is no longer required)

   `jp2a` *`[options]`* `filename.jpg`

7. To complete this task you should download a `.jpg` file of your choice from the web and convert it to ASCII using `jp2a`. Use Google Image Search to find an image. You can get the file onto `attu` by using `wget` *`URL`* if needed.

The `jp2a` program should output an ASCII version of the image. Redirect the `jp2a` ASCII output to **capture the output in a file** named `homework8.txt` and submit this file as part of your assignment turnin.

(*Side note*: `jp2a` has several **options** that you can learn by typing `jp2a --help`. There isn't a `jp2a man` page because we haven't fully installed the app or its `man` files into your system. A particularly useful option is `--background=light`, which causes white/light backgrounds to be drawn in a lighter color. We found that this option made the ASCII output of some of our test images, such as a Homer Simpson drawing, look much better.)

Task 2 of 2: Write a Makefile

For this task you will **write a `Makefile`** for a small set of C program files provided by the instructor. Download the resource file `hw8.tar.gz` from the course web site and decompress it to your homework directory. These files represent a linked list library stored in `linkedlist.c` and `linkedlist.h` along with some client program C files that use this library to perform simple tasks.

Your `Makefile` should have the following **six properties**:

- A target that **builds an object file named `linkedlist.o`** from the source code found in `linkedlist.c`. If `linkedlist.c` or `linkedlist.h` is modified, the `linkedlist.o` file should be rebuilt. In other words, it depends on both of those files. (You can test this by `touch`ing the `.c` or `.h` file and then re-running `make`.)
- A target that **builds an executable file named `list_check`** from the source file `use_ll_2.c` and the compiled object file `linkedlist.o`. If `linkedlist.o` or any of its dependencies are modified, **`list_check`** should be rebuilt.
- A target that **builds an executable file named `list_run`** from the source file `use_linkedlist.c` and the compiled object file `linkedlist.o`. If `linkedlist.o` or any of its dependencies are modified, **`list_ run`** should be rebuilt. (You can test the **`list_check`** and **`list_run`** programs by running them once they have been compiled.)
- A **target named `clean`** that removes the **`list_check`** and **`list_ run`** executables along with any `.o` files from the directory.
- The `Makefile`'s **default target** (the one that runs if **`make`** is not given any parameters) should build both the **`list_check`** and **`list_run`** executables.
- Use at least **one of `make`'s advanced features**. For example, declare at least one variable and use it in your rules, and/or try to use some of the special variables such as `$^` or `$<` .

For reference, our `Makefile` is 16 lines long (11 non-blank, non-comment "substantive" lines).