CS 35L Spring 19 Section 4/5/8 Notes Week 7 Thursday

Zhaowei Tan

Make Basics:

1. Basic format:
   [target]: [prerequisites]
     [recipe]
2. To execute a specific target, run
   make [target]
   If you simply type make, the first target in Makefile will be executed
3. The target is by default considered as a file. Namely the recipe is used to generate the target file. The prerequisites are used to check if the target has to be compiled or not. The prerequisites can be another target or files. Make will check the prerequisites (by timestamp) to determine whether the target needs a recompilation.
4. Imagine you have a target named clean, which is a convention to clean up all the compiled files. By default, clean is regarded as a file. Therefore, make checks the file clean to see if it exists, and then "compiles" it (executes the corresponding recipe).
   However, if you happen to have a file named "clean" in your directory, make clean will never work because "clean" exists and it does not have any dependency. To fix this, you could add a target called .PHONY and declare clean as prerequisite of .PHONY will fix the issue.
5. You can assign a variable using NAME = VAR. To use it later, use $(NAME). It's conventional to have variables like CC, CFLAGS, etc.
6. Some automatic variables can be used to specify target, prerequisites, etc. Check https://www.gnu.org/software/make/manual/html_node/Automatic-Variables.html

Steps to compile statically and dynamically

1. Compile the files with -c (means without linking) flag. If you want to create dynamic library, add -fPIC flag.
   $ gcc (-fPIC) -c [file.c] -o [file.o]
2. To create a static library, use
   $ ar rcs [libxxx.a] file.o
   To create a shared library, use
   $ gcc -shared [file.o] [libxxx.so]
3. To link the library statically/dynamically
   $ gcc -L[path] -l[xxx] [your_main.o] -o [your_main]
   where [your_main.o] is the object file without linking that depends on your library, [path] is the location of the library, and [xxx] is the name of the library without prefix "lib"
4. By default, your libraries are in /usr/lib(64)
   To make sure the linker can find your library, either add your directory path to $LD_LIBRARY_PATH, or use -ldl -Wl,-rpath=[path] when compiling.