CS-102 Design Principles: ncurses

A *library* is a file containing object code that a system (or project) supplies so that you can use their work in your own project without having to compile their code. (Libraries serve a secondary purpose: some companies don't want you to be able to look at their code!). Typically a library comes with header files, which provide the function and/or class declarations that you may use.

Downloading and building a library: ncurses

The ncurses library provides advanced functionality for the terminal, like color, alignment, windows, and menus. Here are the steps to download, install, and build code using the ncurses library.

Note: to install on decker, you would need to have root access (and only sysadmins have this), so we will be installing ncurses in your project directory.

1. Clone this repository into a folder on decker:

   https://github.com/HamiltonCollegeCS/ncurses.git

2. Go to the new folder and type the command:

   ```
   wget https://ftp.gnu.org/pub/gnu/ncurses/ncurses-6.2.tar.gz
   ```

   This grabs the file `ncurses-6.2.tar.gz` from the GNU project's ftp server and places it in the current directory. (Run `ls` to verify.)

3. This file is a tarred (files stuck together into one file) and compressed file, containing all of the code for the ncurses library. Untar and uncompress it with the command

   ```
   tar -zxf ncurses-6.2.tar.gz
   ```

   You should now have the directory `ncurses-6.2` in your current directory. Feel free to explore the files in this directory if you'd like.

4. We will now compile the code and install it in a directory. Run the command `cd ncurses-6.2` , followed by `mkdir install` to make a directory to install the library.

5. We now need to configure the project to install, with the command

   ```
   ./configure -prefix=`realpath ./install`
   ```

Where the quotes around the realpath, are back ticks (next to `1` on most keyboards). The configure command checks that your system meets the necessary requirements of the project. A failure here would mean that other libraries, header files, or compilers will be needed to compile and install the library.

6. We are now ready to build. To compile the library run

   ```
   make
   ```

   which compiles all of the code into a library.

7. To install the code into the directory we created earlier, run

   ```
   make install
   ```

Congratulations! You've successfully built your first library: inside the `install` directory, you'll find versions of the library in the `lib` directory. Library files end in `.a` on Linux systems. You'll also find header files in `include`.

Using ncurses

A `CMakeLists.txt` file has already been provided for you that shows how to specify a library and header files to use when building our code. If you choose to look at it, check out the lines that contain "ncurses", there is another library and program in here as well, which we will explore next. Running `cmake ./` and `make` will build the program `ncurses_test`, which you can then run with `./ncurses_test`. See what it does!

Manually compiling with ncurses

In order to manually compile a library in your code (for example, if you didn't use cmake), you would run the following command (on one line):

```
g++ -I./ncurses-6.2/install/include/ -L./ncurses-6.2/install/lib/
    ncurses_test.cpp -lncurses
```

Where the path to header files (also called an *include* directory is specified after `-I`, the path to libraries files is specified after `-L`, and the library name is specified after `-l` as the last argument[1].

[1] Note that the name of the library file is `libncurses.a` , but the convention is to supply the part after `lib` and before `.a` .

Practicing with ncurses

Take a moment to explore the code in the file `ncurses_test.cpp`. Using the following resources, adjust the code however you see fit, to get practice with this new library:

- ☐ http://www.cs.ukzn.ac.za/~hughm/os/notes/ncurses.html
- ☐ https://invisible-island.net/ncurses/ncurses-intro.html
- ☐ https://www.linuxjournal.com/content/getting-started-ncurses
- ☐ http://www.paulgriffiths.net/program/c/curses.php

Another library: OpenCV

OpenCV is a library aimed at real-time computer vision. We have installed OpenCV on gemini for you (it takes time to compile). When you ran `make` earlier to compile `ncurses_test`, you also built a a program called `opencv_test`, which uses the OpenCV library.

Run the command `./opencv_test` to run the program. It will only work if you are connected to gemini using the standard way: with `GEMINI.XLAUNCH` (on Windows) and `gemini.command` (on Mac), or by launching `XMING` from Citrix.

Explore the `CMakeLists.txt` file and see all of the library files are required for OpenCV. Also notice that the libraries end in `.so`, which means these are *shared object* libraries. With standard `.a` libraries, their object code is incorporated into your program (making it larger). With shared object libraries, the object code is loaded while the program is running. These are also called dynamically linked libraries.

Here are some resources for OpenCV, including the documentation for the library installed on gemini:

- ☐ https://www.tutorialspoint.com/opencv/index.htm
- ☐ https://docs.opencv.org/2.4.5/doc/tutorials/tutorials.html
- ☐ Documentation: https://docs.opencv.org/2.4.5/