Breakout / Lab 06

The Curse of the Text User Interface

CSCI 1730 – Fall 2015

Problem / Exercise

This lab is designed to get you familiar with the following:

- Compiling and installing C/C++ software and libraries that others have written.
- Linking third party libraries with your own code.
- Creating simple Text User Interfaces (TUIs) in C/C++ using the `ncurses` library.
- Get you started with Low-Level File I/O.

NOTE: You will be required to do some self-learning for portions of this breakout lab. References will be provided to help aid you in this endeavor.

The `ncurses` Library

The `ncurses` (new `curses`) library is a free software emulation of `curses` in System V Release 4.0 (SVr4), and more. It is a programming library that provides an API which allows the programmer to write text-based user interfaces in a terminal-independent manner. It is a toolkit for developing "GUI-like" application software that runs under a terminal emulator. Such user interfaces are generally known as Text User Interfaces (TUIs). It uses `terminfo` format, supports pads and color and multiple highlights and forms characters and function-key mapping.

In mid-June 1995, the maintainer of 4.4BSD `curses` declared that he considered 4.4BSD `curses` obsolete, and encouraged the keepers of Unix releases such as BSD/OS, FreeBSD and NetBSD to switch over to `ncurses`. As of the writing of this document, the latest version of `ncurses` is 6.0, which was released on August 8th, 2015.

1 Group Brainstorm

You are NOT allowed to use the computers during this time.

Breakup into groups based on your seating and brainstorm about how to solve the problem or exercise. Make sure everyone understands the problem, and sketch out potential ways to move towards a solution. Perhaps something that was discussed during lecture might be useful?

NOTE: For this breakout lab, determine what each group member already knows about the topics enumerated at the beginning of the "Problem / Exercise" section. Also talk about what you expect to learn from this breakout lab.

2 Submit Individual Brainstorm

You may use a computer from this point forward.

Login to eLC and submit a version of your group's brainstorm, written in your own words. You may add additional information if you want. You need to write enough in order to convince the grader that you understand the problem or exercise and that you have a plan for moving forward towards a solution. Please include the last names of the other people in your group in your submission. The brainstorm submission should be available on eLC in your assignment dropbox. We prefer that you submit your individual brainstorms before the end of your breakout period, however, you generally have until 3PM on the day of your breakout (as indicated on eLC) to submit them.

NOTE: Submissions that do not include an individual brainstorm will not be graded.

NOTE: Once you have completed your individual brainstorm, please use the remainder of the lab period to actually begin work on the lab. Failure to do this may result in the taking of attendance.

3 References

You may find the following reference materials useful:

- `https://www.gnu.org/software/ncurses/`
- `http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/`

4 Compiling & Installing the `ncurses` Library

Follow the instructions presented below in order to compile and install the `ncurses` library to your home directory.

NOTE: I encourage you to take notes about what is happening. The following procedure is similar for most GNU libraries and software.

1. Login to your Nike account.
2. Go to `http://ftp.gnu.org/gnu/ncurses/` in your browser and copy the link for the latest version of `ncurses` (e.g., the link to `ncurses-6.0.tar.gz`).
3. Download `ncurses` using the `wget` command and the link you copied in the previous step. There should now be a `.tar.gz` file in the current directory (e.g., `ncurses-6.0.tar.gz`). **PROTIP:** Remember, you can lookup details about the `wget` command in the Linux Manual.
4. Extract the file you downloaded by using the `tar` command along with the `zxvf` command line options. **PROTIP:** You should lookup what each of those options to `tar` do by reading about them in the Linux Manual. For example, you might type the following:

   ```
   $ tar zxvf ncurses-6.0.tar.gz
   ```

   This will create a new sub-directory that contains the extracted files (e.g., the `ncurses-6.0` sub-directory).
5. Change into the sub-directory containing the extracted files.
6. Issue the following commands in order to configure the compilation:

   ```
   $ sed -i '/LIBTOOL_INSTALL/d' c++/Makefile.in
   $ ./configure --prefix=$HOME --with-shared --with-debug --enable-widec
   ```

   The first command is specific to `ncurses`, however the second command is something that you will likely issue before compiling most third-party software and libraries on Linux. I recommend issuing the following command in order to see what different options to the `./configure` script do:

   ```
   $ ./configure --help
   ```
7. If no errors occur, then issue the following command to compile the `ncurses` library:

   ```
   $ make
   ```
8. If no errors occur, then issue the following command to install the `ncurses` library:

   ```
   $ make install
   ```
9. Now change back into your home directory and delete both the `.tar.gz` file that you downloaded and the sub-directory containing the extracted files. You no longer need them.

5 Let the Linker know where to find the ncurses Library

Follow the instructions presented below in order to ensure that the `ncurses` library is added to your linker path.

1. Add the following line to your `.bash_login` file. If you already have `export` statements, you should place this near them.

   ```
   export LD_LIBRARY_PATH=$HOME/lib:$LD_LIBRARY_PATH
   ```

   PROTIP: You should try to figure out what this line is actually doing and why you needed to put it into your `.bash_login` file.

2. Logout of Nike, then log back in.

6 Test Code

Make sure that all of your files are in a directory called `LastName-FirstName-lab06`, where `LastName` and `FirstName` are replaced with your actual last and first names, respectively.

Follow the instructions presented below in order to test whether or not the `ncurses` library was properly installed.

1. Create a file balled `lab06test.cpp` and add the following to it:

   ```
   #include <cstdlib>
   #include <ncurses.h>

   int main() {
       initscr();                  // start curses mode
       printw("Hello World !!!"); // print Hello World
       refresh();                  // print it on to the real screen
       getch();                    // wait for user input
       endwin();                   // end curses mode
       return EXIT_SUCCESS;
   } // main
   ```

2. Compile the program:

   ```
   $ g++ -Wall -std=c++11 -c -lncurses -g -O0 -pedantic-errors lab06test.cpp
   ```

   PROTIP: Figure out why you needed to include the `-lncurses` option when compiling the code. What happens if you don't include that option?

3. Link the program into an executable:

   ```
   $ g++ -Wall -lncurses -g -o lab06test lab06test.o
   ```

   PROTIP: Figure out why you needed to include the `-lncurses` option when linking the object files into an executable. What happens if you don't include that option?

4. Run the program:

   ```
   $ ./lab06test
   ```

5. You will be able to tell if the program is working.

7 C++ Code

Make sure that all of your files are in a directory called `LastName-FirstName-lab06`, where `LastName` and `FirstName` are replaced with your actual last and first names, respectively. This portion of the lab should be written in a file called `lab06.cpp`.

Write a program with the following functional requirements:

- Your program should take operate in two ways:
  - If a single command line argument (`argc == 2`) is passed to the program, this will represent a text file that the user wishes to open.
  - If no command line arguments (`argc == 1`) are passed, then your program should display a window to the user using the `ncurses` library that prompts them for the name of the file they wish to open.
- Your program should display the contents of the text file using low-level file I/O and the `ncurses` library.
- If an error occurs with any of the system calls, create a windows that displays the error message to the user using the `ncurses` library.
- Your program should close the file when it's done working with it.

NOTE: While we have covered the low-level file I/O aspects of this breakout lab in lecture, you and your group members will need to consult the reference materials listed in the "References" section for information about the `ncurses` library.

Your program should adhere to the following non-functional requirements as well:

- Try to uses classes and standard object-oriented programming (OOP) techniques while writing the code for this breakout lab.
- Do NOT write everything in the `main` function. Breakup your code into multiple functions, as needed, in order to make your code more readable.
- Make sure that you comment all functions using Javadoc-style comments.

7.1 `Makefile` File

You need to include a `Makefile`. Your `Makefile` needs to compile and link separately. Make sure that your `.cpp` files compile to individual `.o` files. The resulting executables should be called `lab06test` (for the test program) and `lab06` (for the main program).

7.2 `README` File

Make sure to include a `README` file that includes the following information presented in a reasonably formatted way:

- Your Name and 810/811#
- Instructions on how to compile and run your program.
- A Reflection Section. In a paragraph or two, compare and contrast what you actually did to complete the problem or exercise versus what you wrote in your initial brainstorm. How will this experience impact future planning/brainstorms?

Here is a partially filled out, example `README` file: `https://gist.github.com/mepcotterell/3ce865e3a151a3b49ec3`.

NOTE: Try to make sure that each line in your `README` file does not exceed 80 characters. Do not assume line-wrapping. Please manually insert a line break if a line exceeds 80 characters.

7.3 Compiler Warnings

Since you should be compiling with both the `-Wall` and `pedantic-error` options, your code is expected to compile without `g++` issuing any warnings. For this lab, compiling without warnings will be one or more of the test cases.

7.4 Memory Leaks

Since this breakout lab makes use of dynamic memory allocation, you are expected to ensure that your `Matrix` implementation doesn't result in any memory leaks. We will test for memory leaks using the `valgrind` utility. For this lab, having no memory leaks will be one or more of the test cases.

8 Submission

Before the day of the next breakout session, you need to submit your code. You will still be submitting your project via nike. Make sure your work is on `nike.cs.uga.edu` in a directory called `LastName-FirstName-lab06`. From within the parent directory, execute the following command:

```
$ submit LastName-FirstName-lab06 cs1730a
```

It is also a good idea to email a copy to yourself. To do this, simply execute the following command, replacing the email address with your email address:

```
$ tar zcvf LastName-FirstName-lab06.tar.gz LastName-FirstName-lab06
$ mutt -s "lab06" -a LastName-FirstName-lab06.tar.gz -- your@email.com < /dev/null
```