HOMEWORK 6

Userspace Console (screen) Library

Issued	4/15/14
Due Date	4/24/14 **before class** (even if there is a strike)

NOTE: As a reminder, homework is considered handed in when it is committed to your personal subversion repository before the deadline. Changes made after the deadline will be ignored.

Goal

For this homework, we will be designing a helper library which makes accessing the screen in our OS more user friendly. This is a pure userspace library.

Evaluation

The homework will be evaluated using the following factors:

- Code Quality: Your code should compile without warnings or errors
- Documentation of the code -- The code is clearly documented, explaining what the intent is and how it is accomplished.
- Correctness of execution -- The code performs the functions requested.
- Creativity -- Your solution is different. Make sure to **document** your code or we might miss your special trick!
- Optional tasks -- You implemented some or all of the optional requests, or went beyond what was requested in some other way.

What to hand in

As for all the other homework assignments, you will need to commit your homework to your assigned subversion repository, under the correct directory (`homework/homework6`).

Use EXACTLY the requested file and directory names, including case!

Please commit the following files:

- All the `.c` and `.h` files required to build your library and demo program(s).
- The `Makefile` (which should build the above programs)
- A text file `DESCRIPTION` containing a description of what your code does, how you achieved it, clearly listing which tasks you completed.
- The test program demonstrating your library.

Is is OK if you commit more files than requested, but make sure the files requested above are present and exactly named as specified. If you add extra C files (header, code) make sure to update your Makefile.

Details

Environment

Your code needs to compile under your virtual machine (ubuntu). While a part of your code will eventually be used in our own kernel, for the purpose of this homework, all development happens under linux.

A simple `Makefile` is provided. Feel free to enhance this makefile.

Kernel Screen Interface

This homework implements some of the interfaces proposed in the group project. The `os_link` files represent the interface to our kernel. This file provides some functions to close a file descriptor, open a file descriptor and read/write using a file descriptor.

We use the file descriptor abstraction to communicate with the screen driver in the kernel. (Note that in this case, `os_link` **emulates** talking to a screen driver by using the `curses` library, a POSIX library for manipulating a text console).

The file descriptor associated with the screen can be obtained by calling `getscreenfd()`. Make sure to close this file descriptor before the program ends (a good place for this is in your `screen_done()` function).

The screen driver supports reading `sizeof(ScreenInfo)` bytes at a time only. Trying to read more or less bytes will result in an error. When reading `sizeof(ScreenInfo)` bytes, the data returned will be a `ScreenInfo` structure (defined in `os_link.h`), containing the horizontal and vertical size of the screen in characters. (To keep things simple, we pretend the monitor and screen driver only support monochrome characters).

The screen is updated by writing exactly one screen worth of characters to the screen file descriptor. Writing more or less will result in an error. In other words, it is not possible to update only a portion of the screen.

This is where your library comes in. Your library simplifies screen access by implementing the functions described in `lumOS_screen.h`. It does so by keeping a full screen worth of characters in memory (in an array for example), making it possible to update a subset. Once the update is made, it can then write the full array to the screen driver, which will update the screen.

An example of this principle is shown in `lumOS_oslink_example`.

In addition to enabling partial updates, `lumOS_screen.h` also provides functionality to scroll the screen, implements the concept of current position (cursor) and wraps text if needed.

Task 1: Library

Implement each of the functions described in `lumOS_screen.h` in the `lumOS_screen.c` source file. There should be *no need to modify `lumOS_screen.h` *. In fact, if you do need to modify `lumOS_screen.h`, please explain on Piazza why `lumOS_screen.h` needs to be modified.

Task 2: Demonstration program

Complete `lumOS_screen_demo.c`. Each function described in `lumOS_screen.h` should be utilized in this demo program. Make sure to demonstrate scrolling of the screen and text wrapping. Your demo program should be able to work with any screen size equal to or larger than 40 columns and 25 lines.

Help and hints

- The `valgrind` tool can help you find bugs in your program. http://valgrind.org/ It can be installed using `apt-get` under ubuntu.
- ~~There should be no need to use `malloc` in `lumOS_screen.c`.~~
- `lumOS_screen.c` should use the functions from `os_link.h` to update the screen. A small example program, which directly uses the `os_link.h` functions (`lumOS_oslink_example.c`) shows how to interact with the screen using `os_link.h` functions.
- Don't duplicate code. Many `lumOS_screen.h` functions can be implemented in terms of other `lumOS_screen.h` functions.

Copyright 2016 The Board of Trustees
of the University of Illinois.webmaster@cs.uic.edu

WISEST
Helping Women Faculty Advance
Funded by NSF

MAKE A GIFT TO THE DEPARTMENT OF COMPUTER SCIENCE

Open House Information