UNX511 Lab 4: I/O Control

Due: Sunday, June 8, 2025 (11:59pm)

In this lab you will make use of I/O control (ioctl) to get screen information. You will use ioctl's to get the fixed screen information and the variable screen information, from which can be derived the visual type, the accelerator (if any), the screen capabilities, the x-resolution, the y-resolution, and bits per pixel. You will direct all error messages to a file.

- The header file for the screen hardware can be included as: **linux/fb.h**.
  Be sure to include this header file in your C/C++ code.
- The device file for the screen hardware can be found at: **/dev/fb0**.
- You have to open this file as **read only** and **non-blocking**. For a list of **open()** flags, see Chapter 4 of The Linux Programming Interface.
- You will ask the user to make a selection from one of three options:
  1. Fixed Screen Info
  2. Variable Screen Info
  0. Exit
- For fixed screen info, use the ioctl **FBIOGET_FSCREENINFO** along with the structure **struct fb_fix_screeninfo**. Report the following fixed screen information:
  screen visual (see FB_VISUAL_),
  screen accelerator (see FB_ACCEL_), and
  screen capabilities (see FB_CAP_).
  Report the numbers for these in comments in Lab4.cpp.
- For variable screen info, use the ioctl **FBIOGET_VSCREENINFO** along with the structure **struct fb_var_screeninfo**. Report the following variable screen information:
  screen xresolution,
  screen yresolution, and
  bits per pixel.
  Report the numbers for these in comments in Lab4.cpp.
- Be sure to run your program as super-user (if you are working on your local machine).
- Any errors in the code should be streamed to the standard error channel **cerr** which should be redirected to a log file called **Screen.log** using dup2(). You can test your error logging by trying to run your program as a regular user.
- Be sure to make use of Linux's global variable errno by including the header file **<errno.h>**. The integer variable **errno** is set by system calls and some library functions in the event of an error to indicate what went wrong. Use **errno** with **strerror()** to give a text description of the error, as follows: **strerror(errno)**. Be sure to include **<string.h>** for **strerror()**. For instance:

```
ret = ioctl(.., ..., ...);
if(ret<0) {
    cerr << strerror(errno) << endl;
}
```

Study materials which help you in this lab:

- For documentation on **struct fb_fix_screeninfo** and **struct fb_var_screeninfo**, see fb.h.
- For documentation on Frame Buffer (screen) ioctl's, see The Frame Buffer Device API.
- For a tutorial on I/O control and device drivers in Linux, see Controlling Hardware with ioctls.

Assignment Submission:

- Complete all steps, Add all output-screenshot and explanations (if required) to a MS-Word file.
- Add the following declaration at the top of MSWORD file

```
/*************************************************************************
***
* UNX511-Lab4
* I declare that this lab is my own work in accordance with Seneca Academic Policy.
* No part of this assignment has been copied manually or electronically from any other source
* (including web sites) or distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
*
*************************************************************************
**/
```

- Please submit the Source code (zip all .c, .h, and makeFiles)
- Please answer the following two declarations:
    - D1) On a scale from 1 to 5, **How much did you use generative AI to complete this assignment?**
        - where:
        - **1** means you did not use generative AI at all
        - **2** means you used it very minimally
        - **3** means you used it moderately
        - **4** means you used it significantly
        - **5** means you relied on it almost entirely
        - **Your answer :**
    - D2) On a scale from 1 to 5, **How confident are you in your understanding of the generative AI support you utilized in this assignment, and in your ability to explain it if questioned?**
        - where:
        - **1** means "Not confident at all – I do not understand the generative AI support I used and cannot explain it."
        - **2** means "Slightly confident – I understand a little, but I have many uncertainties."
        - **3** means "Moderately confident – I understand the majority of the support, though some parts are unclear."
        - **4** means "Very confident – I understand most of the AI support well and can explain it with minor gaps."
        - **5** means "Extremely confident – I fully understand the generative AI support I used and can clearly explain or justify it if asked."
        - **Your answer :**

Important Note:

- **LATE SUBMISSIONS for labs**. There is a deduction of 10% for Late assignment submissions, and after three days it will grade of zero (0).
- This labs should be submitted along with a video-recording which contains a detailed walkthrough of solution. Without recording, the assignment can get a maximum of 1/3 of the total.
    - Note: In case you are running out of time to record the video, you can submit the assignment (source code + screenshots) by the deadline and submit the video within 24 hours after the deadline.