Framebuffer Coding

ECE598: Advanced Operating Systems – Homework 7

Spring 2015

Due: Friday, 17 April 2015, 5pm

This homework involves getting the operating system producing framebuffer output.

1. **Download the homework code template**
   - Download the code from:
     `http://web.eece.maine.edu/~vweaver/classes/ece598_2015s/ece598_hw7_code.tar.gz`
   - Uncompress the code. On Linux or Mac you can just
     `tar -xzvf ece598_hw7_code.tar.gz`
2. For this homework you will need an HDMI connector and monitor so you can see the graphics display. If you do not have these, you can come and test your code in the TI Lab in the afternoon 2-5pm during ECE214 Lab.
3. Take a look at the mailbox code in `mailbox.c` and make sure you understand what it is doing.
4. Also look at the `framebuffer_init()` code in `framebuffer.c`
5. **Add code that initializes the framebuffer using `framebuffer_init()` (1pt)**
   - Init an 800x600x24bpp screen
   - You probably want to put this in `kernel_main.c` near the other init code.
6. **Add a command called "gradient" then when you type it at the prompt draws a vertical gradient in a color of your choice across the screen. (4pt)**
   - You will probably want to implement a function called `framebuffer_vline()` in `framebuffer.h` that is similar to the existing `framebuffer_hline()` function.
   - Draw one vertical line for each column on the screen. It's probably easiest to do this in a loop, incrementing the R, G, or B component of your choice each time. (You can also choose fancier colors but I'll leave that to you to figure out how you do that).
7. **Add a framebuffer console that prints text output to the screen as well as to the serial port. (4pt)**
   - Add a call to `framebuffer_console_init()` in `kernel_main.c`
   - Edit the `io.c` file so that the `write()` function calls `framebuffer_write(buf, count);` in addition to `uart_write(buf, count);`.
   - Implement the `framebuffer_putchar()` function in `framebuffer.c`. It is slightly more complex than the example given in lecture.
     (a) Use the `tb1_font` array to get the lines for the text. It is a double array, you can index it like `tb1_font[ch][y]` where ch is the ASCII char and y is the line of the font.

(b) You can use the `framebuffer_putpixel(color,x,y);` routine for drawing the font. The color argument is a 32-but value where the top 8bits are zero, the next 8bits are red, the next are green, and the bottom 8bits are blue.

(c) There are two cases to handle, foreground and background color. If the bit in the font is on, draw the foreground color. If the bit is off, draw the background color. If the background color is zero, draw nothing at all.

- Once you have this all working, when you boot your device the text should appear on the screen.

8. **Something Cool (1pt)**

   Do something cool with your homework. Below are just some suggestions for things you can do.

   - Use a different font for your console. The fonts are called "VGA Fonts" and you can often find them online. They tend to be 8x16 in size so you'll have to modify the font routines. You can use the `convert_font.c` utility in the tools directory to convert it to replace the `tbfont.h`.
   - Add a command that draws a horizontal gradient.
   - Add a command that draws a gradient and then updates it to give a shimmer or pulsing effect.
   - Add code that moves a shape around the screen that you can steer with the IJKL keys.

9. **Submit your work**

   - Run `make submit` in your code directory and it should make a file called `hw7_submit.tar.gz`. E-mail that file to me.