CS 310 Homework 1: Serial Port Driver

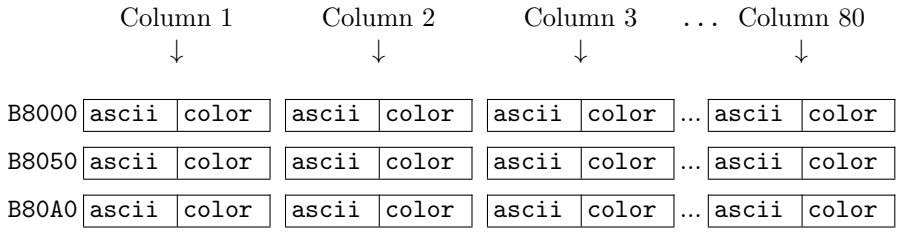
Spring 2021

September 1, 2022

1 Introduction

In this homework assignment, you're going to build a terminal driver for your kernel that allows it to print to the terminal. This homework will be done in your OS repo, not in Linux. The i386 PC has a special memory region reserved for terminal output. To write characters to the screen, we just need to write ASCII characters into video memory.

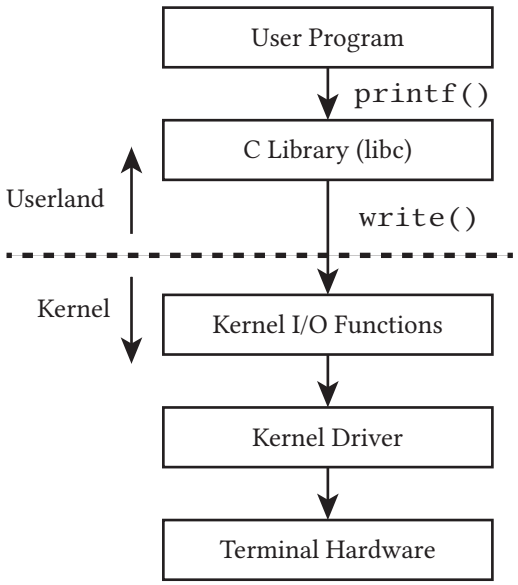
Video memory is basically just a large array that starts at address `0xB8000`. Each element of the array is two bytes long: one byte tells the computer what ASCII character to display, and the second byte tells the computer what color to use. The first element of the array controls the character that is displayed in the upper left corner of the screen—the first character on the top row. The second element of the array (at address `0xB8002`) controls the character in the second position on the first row, and so on. Each row is 80 characters wide. A diagram of video memory is shown below.



On the course website, I have linked to C file called `rprintf.c` that includes a function called `esp_printf`, an OS-free version of `printf`. You can use `esp_printf` mostly the same way as you would use regular textttprintf—it supports `%c`, `%d`, `%x`, etc. It doesn't support printing floating point, but we shouldn't need that.

There is one minor difference between `esp_printf` and regular `printf`—the interface to the terminal driver. The regular interface to the terminal driver is shown below. `printf` normally calls `write` to write a character string to the terminal, but this won't work for us because we haven't implemented `write` yet. Instead, `esp_printf` lets us pass a function as the first argument. The function that we pass will print a single character to the terminal. So a typical call to `esp_printf` looks like this:

```
esp_printf(putc, "Hello World!\r\n");
```



The deliverables of this homework assignement are:

1. Write a function called `putc` that writes a single character to the terminal:

   ```
   void putc(int data);
   ```

   Successive calls to `putc` should write characters to the terminal at sequential locations on the screen. It should not keep writing characters to the upper left corner of the screen. To make this work, you will need to keep track of the offset of the last character that you wrote to video memory and increment that offset each time you write a new character.

2. Use `esp_printf` to print out the current execution level to the terminal.

3. When you reach the bottom of the screen (past the 24th row of characters), your terminal driver should scroll the contents of the screen up. Do this by overwriting the first line of characters with the contents of the second line, overwrite the second line of characters with the contents of the third line, and so on.