Project 3: A Serial Console

Consult the submit server for deadline date and time

1 Overview

You will alter GeekOS to run a shell on serial ports 1 and 2 (COM1 and COM2), just as it runs a shell on the screen and keyboard. To do so, you will provide a serial port driver for GeekOS and modify the process data structures to remember which console is to be used.

2 Goal

The primary goal of this assignment is to develop an understanding of hardware devices in an operating system, how processes can wait for input from devices, and a bit about initializing hardware in a driver.

3 Reference

A reasonably detailed reference on programming the 8250 UART chip is at: `http://en.wikibooks.org/wiki/Serial_Programming/8250_UART_Programming` The 8250 is archaic, but plenty sufficient for this project. You may also want to look at the disk blockdev.c driver or the keyboard.c driver for GeekOS-specific hints.

4 The 16550

You're welcome to use 16550 features if you like, but Qemu appears to emulate a sufficiently swift serial port that you will not be able to overrun the one-character buffer enough to need the 16 character buffer of the 16550.

5 On boot

Print `"Welcome to the serial console!\r\n"` to the serial port.

6 Rules

Cite any and all additional sources at the top of serial.c.

Enable interrupts and read data from the serial port in response to the interrupt. Do not poll.

7 Hints

Alter Sys_PrintString to print to the serial console for the serial shell and all its descendant processes.

Alter Sys_GetKey to fetch a character from the serial port, blocking as needed until a character is available. Note that blocking can be challenging.

Modify thread structures as appropriate to track the processes that use this alternative console.

8 Notes

Invoking "exit" from the serial console shall not halt the OS. (Unlike exiting the init shell, which will.)

You need not handle backspace. Backspace requires a notion of cursor position, and this is closely tied to the screen.c implementation in geekos. I believe it could be done and would be more friendly to the user accessing the shell, but it is not obviously educational.

If two processes call GetKey at the same time, they may retrieve keystrokes in any order.

Ordinarily, an init process would create both shells, and the terminal console and serial console devices would have a similar interface. It is fine to create both shells from inside the kernel for this assignment, otherwise new system calls or device files would be necessary. That is, adding a serial console in the same way a "real" OS does it is beyond the scope of this project.