ECS150
WQ2016
Programming Assignment 3
I/O, Process Manager (memory management)
Due: Monday, March 14, 2016, 9 AM

These problems ask you to make very small modifications to Minix tasks or servers. You will be Modifying, recompiling and assembling them using MAKE to link the various executable files, and then produce a new MINIX. All of this is (hopefully) quite straightforward.

Each of these programs requires only minor modifications to MINIX. The key is locating the current source code that should be modified; there are numerous acceptable solutions

To receive credit for you solutions, you will demonstrate your updated MINIX to me, Apoorva, Yang or Parisa on Monday, March 14. We will give you the opportunity to make an appointment. During the demonstration you should be prepared to run programs you have previously written that demonstrate that your updated MINIX works properly.

.

1. Modify the terminal driver to backspace space n characters in the sequence of entered characters. You might chose control-k (or another control character) followed by some number n (one digit only) as the way to backspace. The characters you are backspacing over are to be deleted but also saved in a buffer you need to declare as part of tty.h. If n exceeds the number of entered characters, backspace to the beginning of the line.

   Once the cursor has been back-spaced, your new tty-driver should allow you to insert new characters, moving characters ahead of the cursor to the right one at a time and to be displayed. Any at any point you should be able to "paste" in the previously deleted sequence of characters, using a different control character (say control-y).

   Demonstrate your solution on characters you will enter at the MINIX "login" prompt.

2. Modify the PM so it allocates memory *worst fit,* instead of *first fit,* which is the current implementation. Also, pick a function key (not the one that displays the process table or the memory map) to display the holes in main memory, including the base address and length for each hole.

   Recall that worst-fit allocation finds the largest hole that holds a new request for memory. The fix is not hard but it is a bit of a challenge to demonstrate that it works. First of all, you have to identify the function key that prints out the memory map for each process; you might modify the action of the key so it just prints the memory map for user processes. Next you have to run some programs. I suggest that you create a number of processes, say 2 or more, each of which runs a *small* program that is an infinite loop or just puts the process to sleep for a long time. Now kill off one or more of these programs to create one or more "holes" in memory. You will want to show the locations of these holes in your demo. Now run a new program that **could** fit into one of these holes, say the first newly created hole, but have your new process manager, which, again, does worst-fit, not pick the first hole but instead picks the largest hole, perhaps the big hole at the upper end of main memory. Please give us a convincing demo.