**Lab 4: Minix Scavenger Hunt II**

Name: Tyler Holland (tyhollan)

Date: 5/14/10

**Task 1: Build and test a modified kernel**

**Approach:**

I had to figure out where to put a print statement in the kernel code in order to have Minix print an @ symbol on the cursor whenever it was idle. In order to figure out how to do this, I looked at our textbook. I figured this had something to do with scheduling, since I was only printing when the system was idle. On page 185, the book describes that inside the proc.c file, the pick\_proc function sets up the next functions to run. I saw that in that code, there was a loop that went through the entire priority queue in order to run the highest priority processes first. I saw in the proc.h file that the IDLE\_Q priority was 15. So, the final code looked like this:

**In proc.c in the pick\_proc function:**

/\* Check each of the scheduling queues for ready processes. The number of

\* queues is defined in proc.h, and priorities are set in the task table.

\* The lowest queue contains IDLE, which is always ready.

\*/

for (q=0; q < NR\_SCHED\_QUEUES; q++) {

if ( (rp = rdy\_head[q]) != NIL\_PROC) {

next\_ptr = rp; /\* run process 'rp' next \*/

**if(q == IDLE\_Q) /\* Tyler Holland's added lab 4 code \*/**

**{**

**kprintf("@\b"); /\* Print @ on top of cursor when idle \*/**

**}**

if (priv(rp)->s\_flags & BILLABLE)

bill\_ptr = rp; /\* bill for system time \*/

return;

}

}

I added the bold code. I figured that when q was equal to IDLE\_Q, that would mean that there are no other processes to run before it, and that the system was indeed idle. That is when I printed the @ symbol.

I then did as the lab said and cd'd into tools, did a make image, copied the image into the boot folder, and restarted Minix. I was happy to see that upon reboot, the cursor had an @ on top of it and nothing was going wrong.

In order to test my code, I wrote a simple program to be a continuously running process:

**int main void()**

**{**

**while(1) {}**

**return 0;**

**}**

I then ran this program, and the cursor appeared without an @ on top of it while it was running. After I ctrl+C'd the program, the @ returned. I then knew that the edit worked properly, and I was finished without any problems.

**Problems Encountered:** At first I tried using the sleep command to create a process that would make the @ go away, but realized that sleep would not do that. While sleep is running, the @ stays.

**Solutions:** I wrote my own program that would for sure be a running process. When I ran it, my program worked. Problem solved.

**Lessons Learned:** It is a lot easier to read and think about the programming problems in Minix, especially when adding/editing things in existing code, rather than just trying to brute force the solution. Because I thought about the problem a lot before hand, my solution worked perfectly the first time around. It was also interesting looking into the source code for Minix, I am glad that it is very well documented, it makes it easy to understand what is going on in the code.