CSE 120: Homework 1

Merrick Qiu

1. If we didn't have privileged instructions, then users could run code that could halt the whole machine and do many other actions that they should not be able to do.

   If we didn't have memory protection, then users could modify and delete memory from anywhere in the machine, which could break the normal functioning of the computer or be used for malicious purposes. There would be no privacy between applications as well.

   If we didn't have timer interrupts the OS would not be able to gain control of non-cooperative processes, and a program with an infinite loop could make the computer get stuck.

2. Calling exit is useful to signal to the parent process whether or not the program ran successfully or if some sort of error occured.

3. (a) Setting the value of the timer should be priviledged since a program could maliciously prevent timer interrupts.

   (b) The clock value is not private, so reading it is not an issue and it shouldn't be a privileged operation.

   (c) Clearing memory can be dangerous, so it should be priviledged.

   (d) Turning off interrupts should be priviledged since a program could use it maliciously to avoid handing control to the OS.

   (e) Switching to kernel mode should be priviledged since it could be maliciously used by a program to take control of the machine.

4. open can fail if permission to read the file is denied. read can fail if it is given a file descriptor to a directory. fork can fail if there is not enough memory for the child. exec can fail if there is not enough memory for the new process. unlink can fail if the file is being used by another program.

5. (a) If a program gets stuck in an infinite loop and cannot exit, SIGKILL might be the only way to terminate the program. If it could be caught, then it would be possible to write programs that cannot ever be terminated.

   (b) As long as the state of the program is saved while it is paused, the programmed can be continued later at any time without any issues.

6. The interval timer could be set to 1ms, and everytime there is a timer interrupt, the OS could update a variable that kept track of the number of miliseconds that have elapsed in order to keep track of the time.

7. I could create an exception for each system call and execute the trap instruction during the exception handling.

8. (a) There are 6 total processes.
   (b) /bin/ls runs 2 times.