All questions must be done by yourself without outside assistance, with the exception of Question 1. You can have up to two collaborators on this first question, but you must identify each of them, and turn in the question either as a group or as identical answers on your respective assignments. You can hand in your solutions either as hard copy during class on April 14, or by email if submitted by 5 PM on April 15. Send your solutions to me (butler@cs.uoregon.edu) with the subject header “CIS 415: Assignment 1 Solutions”. Anything else might get eaten by my spam filter. If submitting your solution through email, please send it as a PDF (generated using Word, OpenOffice, LaTeX, etc.) or a plain text file. I will not accept attachments in their source format (i.e., don’t send me your Word file). If handwritten, the solutions should be neat and legible. Make sure to answer the question, but remember that brevity is the soul of wit: be concise rather than rambling. If we can’t understand your answer or it doesn’t make sense, you will lose marks.

1. **Interrupts** (can be done as a group) (20pts)
   Suppose that you are developing a new computer and OS with significant resource constraints. The hardware is only allowed to support either clock interrupts or I/O interrupts, but not both.
   (a) Discuss how your system would support multiprogramming without clock interrupts, ensuring that all processes can perform I/O, that prevents any process from monopolizing the CPU or going into an infinite loop, and that supports time sharing by switching quickly amongst processes.
   (b) Do the same but for a system without I/O interrupts.
   (c) Argue which you would choose with the goal of providing the best user performance for a low price. What makes your solution more economical than the alternative? Under what solutions would your OS perform well? Under what conditions would it respond poorly? Note that your design is for a uniprocessor system and that user processes and the OS must take turns with the one CPU.

   There are many possible answers. Be sure that your solution is consistent with multiprogrammed uniprocessor systems (what we’ve talked about so far in class) and interrupts. Do not provide answers where the OS acts as a monitor watching over actions and intervening (e.g., an OS noticing a PC instruction run 1000 times and switching processes is a wrong answer). You are welcome to use diagrams for clarification if necessary. A good answer will be a page or less single-spaced.

2. **Textbook Questions** (20 pts)
   Answer questions 2.8, 2.19, 3.6, 3.7, 4.2 in Silberschatz 8/e. The numbers may be different in 7/e; if you’re using that edition, ask a friend to look at their textbook and do the cross-referencing, or check the copy on reserve in the library or the two in room 100. (If anyone figures out the cross-reference, email me and I’ll I’ll send it out on the mailing list.) You only need a few sentences to answer each question.

3. **Process Analysis** (20 pts)
   Design an experiment using proc to extract as much information as you can about a running process, without actually having the source code of its program. This can be done most easily using Solaris: you should have access to ix.cs.uoregon.edu. Read the Solaris man page (section 1) for the proc family of commands. The experiment may begin something like this:
% sleep 600 &
[1] 1363
% psig 1363
1363: sleep 600
HUP default
INT default
...

You can also do this on Linux by examining the /proc filesystem and showing how you’d extract the requisite information. For either OS, explain which piece of information each step of the experiment is designed to apply, and how that related to the program, process, and operating system. In the above example, psig lists the signal handlers that are current installed and perhaps additional information. In Solaris, “LWP” means “light-weight process”, which is similar to a thread.

4. System Call Implementation (20 pts)
Add a system call to the Linux kernel. You will need root access on your machine to do this, which you should have if you’re running either a copy on your local machine or within a VM. Follow the instructions given at http://appusajeev.wordpress.com/2010/11/13/implementing-a-system-call-in-linux-kernel-2-6-35/
but implement a different system call than the one shown. It is your choice as to what to implement. Describe what your system call does, what you had to do to implement it, and show the source code of your test program and your results of the test code execution using the script command.

Note: Like all assignments in this class you are prohibited from copying any content from the Internet or sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class, except where noted. Consulting online sources is acceptable, but under no circumstances should anything be copied. Failure to abide by this requirement will result in sanctions ranging from zero on the assignment to dismissal from the class.