CSCI 3323 (Principles of Operating Systems), Fall 2020

Homework 1a

Credit: 25 points.

1 Reading

Be sure you have read, or at least skimmed, Chapter 1.

2 Problems

Answer the following questions. You may write out your answers by hand and scan them, or you may use a word processor or other program, but please submit a PDF or plain text via e-mail to my TMail address. (No links to shared files on Google Drive please.) Please use a subject line that mentions the course and the assignment (e.g., "csci 3323 hw 1a" or "O/S hw 1a").

1. (10 points) For each of the following instructions, say whether it should be executed only in kernel (i.e., supervisor) mode and briefly explain why.

   (*Hint*: In general, user programs should not be allowed to execute instructions that might interfere with the operating system's control of the machine. The most reasonable way to keep them from doing so is to allow such instructions only in supervisor mode. Note that this question refers to machine-level *instructions*, not necessarily functionality. An operating system could make the functionality of some of these instructions available to user programs by wrapping them in system calls, and possibly requiring user programs to supply a password to (successfully) execute these calls.)

   (a) Disable all interrupts.
   (b) Read the time-of-day clock.
   (c) Change whatever registers are used to determine which part of memory the current process has access to.
   (d) Set the time-of-day clock.
   (e) Switch from user mode to supervisor mode.

2. (5 points) We've talked some in class about the benefits of having an operating system (providing a virtual machine, managing system resources). Can you think of circumstances in which it would be advantageous not to have one? If so, what?

3. (10 points) Most UNIX systems include some command that allows you to trace all system calls made by a process or command. Under Linux, this command is `strace`. For example, to trace all the system calls made during execution of the command `ls -l` and record the output in `OUT`, you would type

```
strace -o OUT ls -l
```

Your mission for this problem is to run `strace` for a command of your choice, capture the output, and then describe what some of it means. (*Note* that I think this may be most easily done on a Linux system. If you're planning to use Cygwin, I'd recommend doing this one step — running `strace` — on the Linux virtual desktop. You can then transfer the output to your own machine to mark up.)

Specifically, I want you to pick at least four lines of the output using different system calls and briefly explain each of these lines, describing in general terms what the system call is supposed to do and what the parameters and return value mean. (So, you will turn in a printout of (part of) the output of `strace` with your homework. You might want to mark it up with numbers and then refer to these numbers in your explanation.)

The `man` page for `strace` explains the general format of the output. To find out what the individual system calls do, you will need to read their `man` pages. Some of these are easy to find — e.g., the first call is usually to `execve`, and `man execve` will tell you about it. Some are a little harder to track down — e.g., `man write` produces information about a `write` command rather than a system call — but `man` with a section number of 2[1] (e.g., `man 2 write`) should show you the `man` page for the `write` system call.

As an example of what I have in mind, here is a line from a trace of the command `ls /users/cs4320` with commentary. (You should choose system calls other than `execve`.)

```
execve("/usr/bin/ls",
```

$$"ls","-l","/users/cs4320"$$

```
, 0x7ffda8fecae0 /* 73 vars */) = 0
```

The call to `execve` creates a new process to run the command. Parameters are the command to execute, the arguments to pass to it, and an array of environment variables (quite a lot of them, apparently!). The return value of 0 probably doesn't mean anything, since the `man` page for `execve` says that the function doesn't return if the call is successful.

3 Pledge

Include the Honor Code pledge or just the word "pledged", plus *at least one of the following* about collaboration and help (as many as apply).[2] Text *in italics* is explanatory or something for you to fill in. For programming assignments, this should go in the body of the e-mail or in a plain-text file `pledge.txt` (no word-processor files please).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- I worked with *names of other students* on this assignment.

[1] `man` pages are organized into "sections" — one for commands, one for system calls, one for library functions, etc.

[2] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

4 Essay

Include a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what if anything you think you learned from the assignment, and what if anything you found found interesting, difficult, or otherwise noteworthy. For programming assignments, it should go in the body of the e-mail or in a plain-text file `essay.txt` (no word-processor files please).