CSCI 3323 (Principles of Operating Systems), Fall 2018
Homework 7

Credit: 30 points.

1 Reading

Be sure you have read, or at least skimmed, Chapter 5.

2 Problems

Answer the following questions. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in one of my mailboxes (outside my office or in the ASO).

1. (10 points) Consider the following two I/O devices. For each device, say whether you think programmed I/O or interrupt-driven I/O makes the most sense, and justify your answer. (*Hint:* Consider the time required for interrupt processing versus the time needed for the actual input/output operation. You will get more credit if you give actual numbers for these times.)

   (a) A printer that prints at a maximum rate of 400 characters per second, connected to a computer system in which writing to the printer's output register takes essentially no time, and using interrupt-driven I/O means that each character printed requires an interrupt that takes a total of 50 microseconds (i.e., 50×10^{-6} seconds) to process.

   (b) A simple memory-mapped video terminal (output only), connected to a system where interrupts take a minimum of 100 nsec to process, copying a byte into the terminal's video RAM takes 10 nsec, and each byte must be copied independently. (It's probably best to think of this as a hypothetical problem, using only the description supplied, rather than trying to extrapolate from what you know or can read about typical actual hardware.)

2. (10 points) The textbook divides the many routines that make up an operating system's I/O software into four layers. In which of these layers should each of the following be done? Why? (Assume that in general functionality should be provided at the highest level at which it makes sense — e.g., in user-level software rather than device-independent software, if that's possible.)

   (a) Converting floating-point numbers to ASCII for printing.

   (b) Computing the track, sector, and head for a disk read operation.

   (c) Writing commands to a printer controller's device registers.

   (d) Detecting that an application program is attempting to write data from an invalid buffer address. (Assume that detecting an invalid buffer address can be done in supervisor mode in some way other than just trying it and possibly generating a no-such-address exception.)

3. (10 points) Suppose at a given point in time a disk driver has in its queue requests to read cylinders 10, 22, 20, 2, 40, 6, and 38, received in that order. If a seek takes 5 milliseconds (i.e., 5×10^{-3} seconds) per cylinder moved, and the arm is initially at cylinder 20, how much seek time is needed to process these requests using each of the three scheduling algorithms discussed (FCFS, SSF, and elevator)? Assume that no other requests arrive while these are being processed and that for the elevator algorithm the initial direction of movement is outward (toward larger cylinder numbers).

3 Programming Problems

Do as many of the following programming problems as you like. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to bmassing@cs.trinity.edu with each file as an attachment. Please use a subject line that mentions the course and the assignment (e.g., "csci 3323 hw 7" or "O/S hw 7"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (Optional; up to 5 extra-credit points each.) The Linux lab machines have special files `/dev/random` and `/dev/urandom` that generate sequences of "random" bytes. (Read the man page for `urandom` for an explanation of the difference between them.) Write a C program that compares the results of generating N integers using one or both of these special files to the results of generating N integers using function `rand()`. The program should take two command-line arguments, the number of "samples" to generate (N) and a seed for `srand()` and should print the generated values and optionally something that would let you compare the different methods.

   For example, I thought it might be somewhat interesting to count, for each method, how many of the generated values are even and how many are odd. Here is a sample of my program's output:

```
using /dev/urandom:
-1700094874
-1054372852
-660700578
779731362
-1895635517
846523869
-1349781274
-82278049
812259054
-1926768141
6 evens and 4 odds

using /dev/random:
-1667308111
-2125439876
-2070173129
1115334449
```

```
1751777993
1997266376
847874132
-398692840
-1656351377
-1885858316
5 evens and 5 odds

using rand() with seed 1
1804289383
846930886
1681692777
1714636915
1957747793
424238335
719885386
1649760492
596516649
1189641421
3 evens and 7 odds
```

You may have a better idea about how to compare!

Hint: You will probably need to use `open` and `read` rather than `fopen` and `fscanf` to read from the special files. `man` pages for these two functions can be found via `man 2 open` and `man 2 read`. To extract an `int` value from one of these files, use `read` to read `sizeof int` bytes.

4 Honor Code Statement

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).[1] 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 `honor-code.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.
- 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.)*

[1] 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 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.)*

5 Essay

Include a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what about the assignment you 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).