CS 273 (Operating Systems), Fall 2020, R. Brown

Homework 11 Due Monday, 11-2-20

A. **Input/Output layers**

The six layers of I/O software are:

(A) User-level software

(B) Device-independent OS software

(C) Device driver

(D) Interrupt handler

(E) Device controller

(F) Device

Which of the six layers of I/O software **must be** used in the following operations, and why?

1. A single character is read from terminal standard input
2. A PDF file is displayed on the screen by PDF viewer software
3. A file is opened for writing
4. An open file descriptor is duplicated, using the `dup()` system call
5. An `lseek()` operation is performed on a file open for reading.
6. A page is swapped out due to a page fault
7. A memory reference is satisfied in cache
8. A `fork()` call succeeds
9. An `execve()` call succeeds

To submit this by-hand part, you can use the page
`https://www.stolaf.edu/people/rab/os/asgt/hw11+.html`

B. **Input/Output**

1. p.430 14, 16, 37

Note: Problem 14 is an example exam-like problem.

To submit this by-hand part, you can use the page
`https://www.stolaf.edu/people/rab/os/asgt/hw11+.html`

C. **Plans for system call**

Submit your current plans for adding system calls that satisfy the basic assignment of the kernel project.

- This should be a 1-page progress report of your work so far on Step 4 of the assignment sheet `https://www.stolaf.edu/people/rab/os/pp-syscall-asgt.html`

Submit these plans in PDF format using the project submission form

`https://www.stolaf.edu/people/rab/os/proj_submit.html?deliv=hw11`

D. **Review**

Note: This is an example exam-like problem (except for length).

Consider the following code with semaphores for the producer-consumer problem discussed in the text. This represents the same algorithm as in the text.

```
#define N 100        /* number of slots in the buffer */
init_sem(mutex, 1);   /* controls access to critical sections */
init_sem(empty, N);   /* counts empty buffer slots */
init_sem(full, 0);    /* counts full buffer slots */

void producer(void) {
  int item;

  while (1) {
    item = produce_item();
    down(&empty);
    down(&mutex);
    insert_item(item);
    up(&mutex);
    up(&full);
  }
}

void consumer(void) {
  int item;

  while (1) {
    down(&full);
    down(&mutex);
    item = remove_item();
    up(&mutex);
    up(&empty);
    consume_item(item);
  }
}
```

1. Explain why this code satisfies each of the four goals for correct IPC.
   a) Mutual exclusion
   b) Bounded wait
   c) Independence of speed
   d) Progress
2. Would the algorithm satisfy the four goals for correct IPC if the order of the `down()` calls were switched in `producer()`?
   - If yes, explain why the modified code satisfies two of the four IPC goals.
   - If no, explain why the modified code does *not* satisfy one of the four IPC goals.
3. Would the algorithm satisfy the four goals for correct IPC if the order of the `up()` calls were switched in `producer()`?
   - If yes, explain why the modified code satisfies two of the four IPC goals.
   - If no, explain why the modified code does *not* satisfy one of the four IPC goals.

To submit this by-hand part, you can use the page
`https://www.stolaf.edu/people/rab/os/asgt/hw11+.html`