Name: ID#: X.500:

CS 4061: Practice Exam 2

Summer 2020

University of Minnesota

Exam period: 30 minutes

Points available: 40

Background: Sigblo C'Ker runs an application called `coordinated_changer` which makes changes to a single file in a safe way. According to the documentation for the code, any number of such processes can be run and they will be coordinated using a semaphore so no data will be lost. While running the program Sigblo accidentally hits the keystroke `Ctrl-c` and finds that `coordinated_changer` closes immediately but on trying to re-run it, Sigblo finds that he cannot get any more instances to run: all seem to "hang" immediately on starting. Looking at the source code for `coordinated_changer`, Sigblo would like to alter it so that `Ctrl-c` will kill `coordinated_changer` safely.

```
1  // rough code for coordinated_changer.c
2  int main(){
3    sem_t *file_lock = sem_open(..);
4
5    perform_setup();
6
7    sem_wait(file_lock);
8    modify_file_for_a_while();
9    sem_post(file_lock);
10
11   perform_cleanup();
12   return 0;
13 }
```

Problem 1 (5 pts): Based on the provided source code, explain why killing one instance of `coordinated_changer` at the wrong time causes all others to stall.

Problem 2 (10 pts): Advise Sigblo on what changes should be made to prevent deadlock in `coordinated_changer`.

Problem 3 (5 pts): Pam Elif is writing a small database system. She would like to support multiple client programs reading and writing the database system simultaneously so is thinking of using a shared memory segment such as is provided by POSIX `shm_open()`. She also would like the database to be backed up by a disk file which a daemon process will occasionally copy from shared memory to disk but is finding the whole arrangement to seem overly complex.
Suggest a simpler mechanism that Pam can use which allows multiple processes to share memory that is automatically written to disk periodically.

Problem 4 (10 pts): Contrast FIFOs and POSIX Shared Memory as means for inter-process communication. Describe at least 3 aspects that are similar or different between them (e.g. 1 similarity / 2 differences or 2 similarities / 1 difference).

Background: Below are two blocks of code associated with a recent lab/HW which demonstrated the `runner_sem1` and `runner_sem2` programs. These two both attempted to accomplish the same goal but had some differences which are explored in this problem.

```
1  // runner_sem1.c main loop
2  while(file_pos < size){
3    sem_wait(sem);
4    char status, command[MAXLINE];
5    sscanf(file_chars+file_pos,
6           "%c %1024[^\n]",
7           &status, command);
8
9    if(status == '-'){
10     file_chars[file_pos] = 'R';
11     sem_post(sem);
12     printf("%03d: %d RUN '%s'\n",
13            line_num,getpid(),command);
14     fflush(stdout);
15     char call[1024];
16     sprintf(call,"%s > /dev/null",command);
17     system(call);
18     file_chars[file_pos] = 'D';
19   }
20   else{
21     sem_post(sem);
22   }
23   file_pos += strlen(command)+3;
24   line_num++;
25 }
```

```
1  // runner_sem2.c main loop
2  while(file_pos < size){
3    sem_wait(sem);
4
5    char status, command[MAXLINE];
6    sscanf(file_chars+file_pos,
7           "%c %1024[^\n]",
8           &status, command);
9    if(status == '-'){
10     file_chars[file_pos] = 'R';
11     printf("%03d: %d RUN '%s'\n",
12            line_num,getpid(),command);
13     fflush(stdout);
14     char call[1024];
15     sprintf(call,"%s > /dev/null",command);
16     system(call);
17     file_chars[file_pos] = 'D';
18   }
19   sem_post(sem);
20   file_pos += strlen(command)+3;
21   line_num++;
22 }
```

Problem 5 (5 pts): Discuss the placement of the semaphore locking/unlocking between the two codes. Describe what period of time each of the codes keeps the shared semaphore locked and what happens during that time.

Problem 6 (5 pts): Based on the locking scheme above, which of the two approaches do you expect/observe is more efficient when multiple `runner` programs are working together? Describe which version will result in completing jobs faster and why.