Operating Systems, Spring 2020

Homework Assignment #2

Due midnight Thursday, April 30, 2020

Instructions

1. If any question is unclear, please ask for a clarification.
2. You are required to do all the homework assignments on Linux.
3. You are required to give your TA a demo of your program.
4. For the program that you write, you are required to include a Makefile so that your TA can compile your program by issuing the command "`make clean dep all`" from a shell. Otherwise, the program part of your homework will not be graded—meaning that you will receive zero marks.
5. Unless stated otherwise, you are required to work on the homework assignment individually.
6. Neither late nor copied homework will be accepted.

Part I (50%)

1. (10%) Consider a computer that does not have a TEST AND SET LOCK instruction but does have an instruction to swap the contents of a register and a memory word in a single indivisible action. Can that be used to write a routine *enter_region* such as the one found in Fig. 2–12.
2. (20%) Measurements of a certain system have shown that the average process runs for a time T before blocking on I/O. A process switch requires a time S, which is effectively wasted (overhead). For round-robin scheduling with quantum Q, give a formula for the CPU efficiency (i.e., the useful CPU time divided by the total CPU time) for each of the following:
   (a) $Q = \infty$
   (b) $Q > T$
   (c) $S < Q < T$
   (d) $Q = S$
   (e) Q nearly 0
3. (10%) Consider the interprocess-communication scheme where mailboxes are used. Suppose a process P wants to wait for two messages, one from mailbox A and one from mailbox B. What sequence of `send` and `receive` should it execute so that the messages can be received in any order?

4. (10%) Consider the following program that uses the Pthreads API. What would be the output of the program? (*Note that the line numbers are for references only.*)

Listing 1: pthread.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <pthread.h>
5  #include <sys/types.h>
6
7  int value = 1;
8
9  static void *runner(void *param);
10
11 int main(int argc, char **argv)
12 {
13     pid_t pid = fork();
14     if (pid > 0) {
15         printf("A = %d\n", value);
16     }
17     else if (pid == 0) {
18         pid_t pid = fork();
19         if (pid > 0) {
20             printf("B = %d\n", value);
21         }
22         else if (pid == 0) {
23             pid_t pid = fork();
24             pthread_t tid;
25             pthread_attr_t attr;
26             pthread_attr_init(&attr);
27             pthread_create(&tid, &attr, runner, NULL);
28             pthread_join(tid, NULL);
29             if (pid > 0)
30                 printf("C = %d\n", value);
31             else
32                 printf("D = %d\n", value);
33         }
34         else {
35             exit(1);
36         }
37     }
38     else {
39         exit(1);
40     }
41
42     return 0;
43 }
44
45 static void *runner(void *param)
46 {
47     value += 1;
48     pthread_exit(0);
49 }
```

Part II (50%)

Write a program to simulate the dining philosopher problem mentioned in the textbook using the Pthreads API on Linux. Make sure that your implementation is able to handle 5 philosophers and is free of race condition.

Gentle Reminder

Once again, as mentioned in the instructions, neither late nor copied homework will be accepted.