Operating Systems, Spring 2016

Midterm

2:10pm ∼ 3:50pm, Tuesday, April 22, 2016

INSTRUCTIONS:

1. This is a *closed-book* exam.
2. Try to solve all of the problems.
3. Try to give short answers. (Hint: An answer need not always be longer than the question.)
4. No cheating.
5. Please hand in both the exam sheet and the answer sheet.
6. Please note that unless otherwise stated, all the line numbers for the program listings are for reference only.

1. (20%) What would be the output of the following C program that uses the Pthreads API?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <pthread.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7
8  static void *runner(void *param)
9  {
10     ++(* (int*) param);
11     pthread_exit(0);
12 }
13
14 int main(int argc, char **argv)
15 {
16     int status;
17     int value = 100;
18     pid_t pid = fork();
19     if (pid > 0) {
20         waitpid(-1, &status, 0);
21         printf("A = %d\n", ++value);
22     }
23     else if (pid == 0) {
24         pid_t pid = fork();
25         if (pid > 0) {
26             waitpid(-1, &status, 0);
27             printf("B = %d\n", value++);
28         }
29         else if (pid == 0) {
30             pid_t pid = fork();
31             pthread_t tid;
32             pthread_create(&tid, NULL, runner, &value);
33             pthread_join(tid, NULL);
34             if (pid > 0) {
35                 waitpid(-1, &status, 0);
36                 printf("C = %d\n", --value);
37             }
38             else {
39                 printf("D = %d\n", value--);
40             }
41         }
42         else {
43             return 1;
44         }
45     }
46     else {
47         return 1;
48     }
49     return 0;
50 }
```

2. (20%) What would be the output of the following C program? Why?

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main()
{
    int status, fd[2];
    pipe(fd);
    pid_t pid = fork();
    if (pid > 0) {
        close(fd[1]);
        close(0);
        dup(fd[0]);
        close(fd[0]);
        waitpid(-1, &status, 0);
    }
    else if (pid == 0) {
        close(fd[0]);
        close(1);
        dup(fd[1]);
        close(fd[1]);
        execl("/bin/echo", "echo", "welcome", "to", "nsysu", (void*) 0);
    }
    else {
        return 1;
    }
    return 0;
}
```

3. (20%) What would be the output of the following C program? Why?

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main()
{
    int status, fd[2];
    pipe(fd);
    pid_t pid = fork();
    if (pid > 0) {
        close(fd[0]);
        close(1);
        dup(fd[1]);
        close(fd[1]);
        waitpid(-1, &status, 0);
    }
    else if (pid == 0) {
        close(fd[1]);
        close(0);
        dup(fd[0]);
        close(fd[0]);
        execl("/bin/echo", "echo", "welcome", "to", "nsysu", (void*) 0);
    }
    else {
        return 1;
    }
    return 0;
}
```

4. (20%) Suppose that two processes, P_1 and P_2, are running in a uniprocessor system. P_1 has two threads. P_2 has three threads. All threads in both processes are CPU-intensive; that is, they never block for I/O. The operating system uses simple round-robin scheduling.

   (a) Suppose that all of the threads are user-level threads, and that user-level threads are implemented using a single kernel thread per process. What percentage of the processor's time will be spent running P_1's threads?

   (b) Suppose instead that all of the threads are kernel threads. What percentage of the processor's time will be spent running P_1's threads?

5. (20%) 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?