Operating Systems, Spring 2012

Midterm

2:10pm ∼ 3:50pm, Tuesday, April 17, 2012

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%) Draw a picture to show how a C program consisting of n files named $P_1, P_2, \cdots, P_n$ gets compiled and linked step by step as we discussed in the classroom.

2. (20%) Draw a picture to show how a process looks alike as we discussed in the classroom.

3. (15%) 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) $Q > T$

   (b) $S < Q < T$

   (c) $Q = S$

   To simplify the answers, you may assume Q divides T evenly.

4. (15%) 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?

5. (15%) What would be the output of the following C program that uses the Pthreads API? (*Note that the line numbers are for references only.*)

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/wait.h>

static void *runner(void *param)
{
    (* (int*) param)--;
    pthread_exit(0);
}

int main(int argc, char **argv)
{
    int status;
    int value = 101;
    pid_t pid = fork();
    if (pid > 0) {
        printf("A = %d\n", --value);
        waitpid(-1, &status, 0);
    }
    else if (pid == 0) {
        pid_t pid = fork();
        if (pid > 0) {
            printf("B = %d\n", --value);
            waitpid(-1, &status, 0);
        }
        else if (pid == 0) {
            pid_t pid = fork();
            pthread_t tid;
            pthread_attr_t attr;
            pthread_attr_init(&attr);
            pthread_create(&tid, &attr, runner, &value);
            pthread_join(tid, NULL);
            if (pid > 0) {
                printf("C = %d\n", --value);
                waitpid(-1, &status, 0);
            }
            else
                printf("D = %d\n", --value);
        }
        else {
            return 1;
        }
    }
    else {
        return 1;
    }
    return 0;
}
```

6. (15%) What would be the output of the following C program? (*Note that the line numbers are for references only.*)

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main()
{
    int status;
    int fd[2];
    pipe(fd);
    pid_t pid = fork();
    if (pid > 0) {
        close(fd[0]);
        close(1);
        dup(fd[1]);
        close(fd[1]);
        wait(&status);
    }
    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;
}
```