# Operating Systems, Spring 2010

Midterm

2:10pm $\sim$ 4:10pm, Tuesday, April 20, 2010

---

**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. (15%) Explain why `fork` returns twice and what are returned as we discussed in the classroom?

2. (20%) Consider the following preemptive priority-scheduling algorithm based on dynamically changing priority. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate $\alpha$; when it is running, its priority changes at a rate $\beta$. All processes are given a priority of 0 when they enter the ready queue. The parameters $\alpha$ and $\beta$ can be set to give many different scheduling algorithms.

   (a) What is the algorithm that results from $\beta > \alpha > 0$?

   (b) What is the algorithm that results from $\alpha < \beta < 0$?

3. (20%) Suppose that two processes, $P_1$ and $P_2$, are running in a uniprocessor system. $P_1$ has three threads. $P_2$ has two 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?

4. (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.*

5. (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 without from being blocked by each other?

6. (15%) Consider the following C program that uses the Pthreads API. What would be the output of the program?

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

int value = 5;

static void *runner(void *param);

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

static void *runner(void *param)
{
    value++;
    pthread_exit(0);
}
```