CS433 Operating Systems

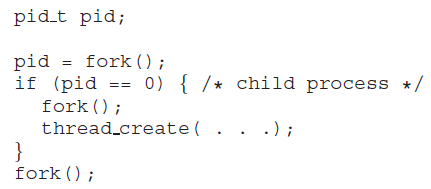
Homework #2

Part II – Process Management

1. Describe the actions taken by a kernel to context-switch between processes.

(3 pts)

1. When a process creates a new process using the *fork( )* operation, which of the following state is shared between the parent process and the child process? (3 pts)
   1. Stack
   2. Heap
   3. Shared memory segments
2. Which of the following components of program state are shared across threads in a multithreaded process? (4 pts)
   1. Register values
   2. Heap memory
   3. Global variables
   4. Stack memory
3. Is it possible to have concurrency but not parallelism? Explain. (4 pts)
4. Consider the following code segment: (2 pts)



* 1. How many unique processes are created?
  2. How many unique threads are created?

1. What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other? (6 pts)
2. Race conditions are possible in many computer systems. Consider a banking system with two methods: deposit(amount) and withdraw(amount). These two methods are passed the amount that is to be deposited or withdrawn from a bank account. Assume that a husband and wife share a bank account and that concurrently the husband calls the withdraw() method and the wife calls deposit(). Describe how a race condition is possible. (6 pts)
3. Explain why spinlocks are not appropriate for single-processor systems yet are used in multiprocessor systems. (4 pts)

.

1. A multithreaded web server wishes to keep track of the number of requests it services (known as **hits**.) Consider the following two strategies to prevent a race condition on the variable hits. The first strategy is to use a basic mutex lock when updating hits: (3 pts)

*int hits;*

*mutex lock hit lock;*

*hit lock.acquire();*

*hits++;*

*hit lock.release();*

A second strategy is to use an atomic integer:

*atomic t hits;*

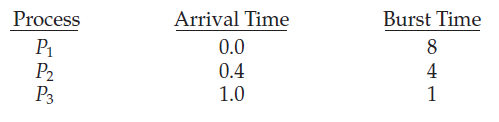
*atomic inc(&hits);*

Explain which of these two strategies is more efficient.

1. Assume that a system has multiple processing cores. For each of the following scenarios, describe which is a better locking mechanism — a spinlock or a mutex lock where waiting processes sleep while waiting for the lock to become available: (3 pts)
   1. The lock is to be held for a short duration.
   2. The lock is to be held for a long duration.
   3. The thread may be put to sleep while holding the lock.
2. Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs? (4 pts)

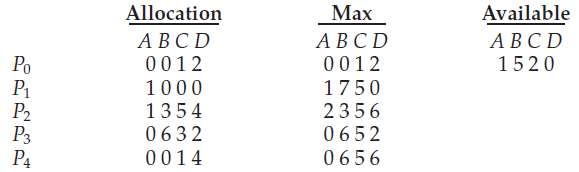
.

1. Explain the difference between preemptive and non-preemptive scheduling. (4 pts)
2. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use non-preemptive scheduling, and base all decisions on the information you have at the time the decision must be made. (4 pts)



* 1. What is the average turnaround time for these processes with the FCFS scheduling algorithm?
  2. What is the average turnaround time for these processes with the SJF scheduling algorithm?

1. Consider two processes, *P*1 and *P*2, where *p*1 = 50, *t*1 = 25, *p*2 = 75, and *t*2 = 30. (8 pts)
   1. Can these two processes be scheduled using rate-monotonic scheduling? Illustrate your answer using a Gantt chart?
   2. Illustrate the scheduling of these two processes using earliest deadline- first (EDF) scheduling.
2. Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock-free. (4 pts)
3. Consider the following snapshot of a system: (6 pts)



Answer the following questions using the banker’s algorithm:

* 1. What is the content of the matrix **Need**?
  2. Is the system in a safe state?

1. Is it possible to have a deadlock involving only one single-threaded process? Explain your answer. (2 pts)