Operating Systems
Mid-Term

Name:________________ Spring 2017

ID:___________________

The exam is 180 minutes long. The total score is 115pts. Please read questions carefully.

1. Multiple Choice (24pts)

(1). A ____ can be used to prevent a user program from never returning control to the operating system.

A. Portal
B. program counter
C. firewall
D. timer

(2). What are some other terms for kernel mode?

A. supervisor mode
B. system mode
C. privileged mode
D. All of the above

(3). The ________ provides a portion of the system call interface for UNIX and Linux.

A. POSIX
B. Java
C. Standard C library
D. Standard API

(4). Microkernels use _____ for communication.

A. message passing
B. shared memory
C. system calls
D. virtualization

(5). A(n) ______________ allows several unrelated processes to use the pipe for communication.

A. named pipe
B. anonymous pipe
C. LIFO
D. ordinary pipe

(6). Which of the following statements is true?

A. Shared memory is typically faster than message passing.
B. Message passing is typically faster than shared memory.
C. Message passing is most useful for exchanging large amounts of data.
D. Shared memory is far more common in operating systems than message passing.

(7). Thread-local storage is data that ____.

A. is not associated with any process
B. has been modified by the thread, but not yet updated to the parent process
C. is generated by the thread independent of the thread's process
D. is unique to each thread

(8). According to Amdahl's Law, what is the speedup gain for an application that is 60% parallel and we run it on a machine with 4 processing cores?

A. 1.43
B. 0.7
C. 0.55
D. 1.82

(9). Which of the following is true of cooperative scheduling?

A. It requires a timer.
B. A process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
C. It incurs a cost associated with access to shared data.
D. A process switches from the running state to the ready state when an interrupt occurs.

(10). Which of the following scheduling algorithms must be nonpreemptive?

A. SJF
B. RR
C. FCFS
D. priority algorithms

(11). A race condition _____.

A. results when several threads try to access the same data concurrently
B. results when several threads try to access and modify the same data concurrently
C. will result only if the outcome of execution does not depend on the order in which instructions are executed
D. None of the above

(12). A counting semaphore _____.

A. is essentially an integer variable
B. is accessed through only one standard operation
C. can be modified simultaneously by multiple threads
D. cannot be used to control access to a thread's critical sections

2. Terminologies (24pts)

(1). Virtual machine

Virtual machine software provides an interface that is identical to the underlying bare hardware

(2). Socket

An endpoint for communication identified by an IP address concatenated with a port number.

(3). Pull migration (Hint: Multiprocessor Scheduling)

An idle processor pulls a waiting task from a busy processor.

(4). Bounded waiting (A requirement of a critical section solution)

A waiting process only waits for a bounded number of processes to enter their critical sections.

(5). Cache coherency
Cache coherency problems can arise when more than one processors refer to the same data. Coherency defines what value is returned on a read.

(6). DMA
DMA is to release CPU from handling excessive interrupts. Once its drive sets up DMA registers, CPU is interrupted when all jobs are done.

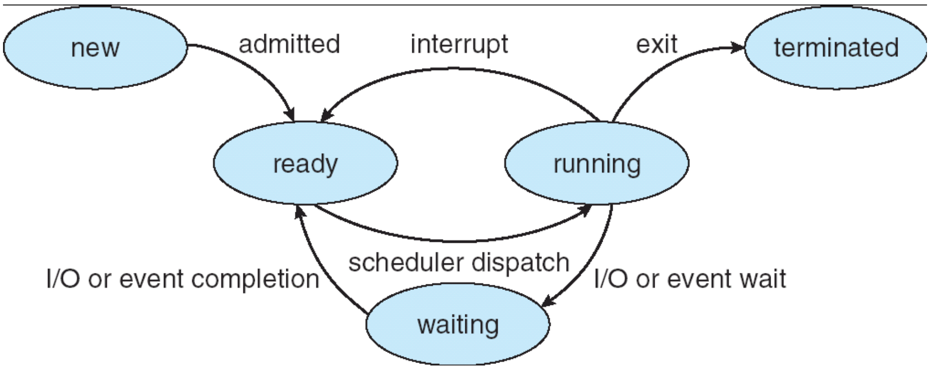
(7). Performance tuning
A procedure that seeks to improve performance by removing bottlenecks.

(8). Mid-term scheduler
Swap processes in and out memory to control the degree of multiprogramming.

3. Please answer the following questions for process scheduling. Explanation is needed to receive any credit. (15pts)
   (1). Given 3 processes P1, P2 and P3 with CPU burst time 9, 2 and 4 respectively. Suppose that the 3 processes arrive at time 0, and P1 and P3 are the first and last processes in the ready queue respectively. What is the average waiting time in running the 3 processes under the Round-Robin Scheduling with the time slice = 3. (5pts)
   (2). Consider FCFS and Round-Robin Scheduling. If processes are only of CPU burst and all arrive at time 0, do FCFS and Round-Robin scheduling with time slice = 1 always have the same total waiting time in running all processes? (5pts)
   (3). Suppose that the variance of the turnaround time is the criterion in process scheduling. Shall we have a small time slice on Round-Robin Scheduling for a better average turnaround time when all processes arrive at time 0? (5pts)

Ans: (1) ((5+1)+3+(5+3)) / 3 = 17/3; (2) No, give one example, such as processes of CPU bursts 7, 1 and 1 respectively. (3) Yes. Give an explanation.

4. Consider process states: new, ready, running, waiting and terminated. Please explain how a state makes a transition to another state, where there is only one processor. (12pts)



5. Please answer the following questions regarding the designs of operating systems: (15pts)
   (1). What is the difference between multiprogramming and time sharing? (5pts)
   (2). Which one of the following memory unit is managed by the operating systems: Cache main memory and disks? (5pts)
   (3). Operating system services include user interfaces. UNIX shells, including the Bourne shell and C shell, provide command interpreters. Consider UNIX shells, please give me two commands that are implemented by system programs. (5pts)

   Ans: (1) Time sharing is a logical extension of multiprogramming, where CPU services each of ready tasks in a way that every task receives CPU time in an interactive fashion; (2) Main memory and disk; (3) System programs: rm, ls

6. How can deferred cancellation ensure that thread termination occurs in an orderly manner as compared to asynchronous cancellation? (10pts)

   Ans:
   Asynchronous cancellation: the thread is immediately cancelled in response to a cancellation request. There is no insurance that it did not quit in the middle of a data update or other potentially dangerous situation.
   Deferred cancellation: the thread polls whether or not it should terminate. This way, the thread can be made to cancel at a convenient time.

7. Consider the following code which is developed to solve the critical-section problem. The two processes, P0 and P1, share the following variables:

```
boolean flag[2]; /* initially false */
int turn;
```

The structure of process Pi (i == 0 or 1) is shown as follows; the other process is Pj (j == 1 or 0). Prove that the algorithm satisfies all three requirements for the critical-section problem. (15pts)

```
do{
  flag[i] = true;
  while (flag[j]){
      if (turn==j){
          flag[i] = false;
          while(turn == j)
              ; // do nothing
          flag[i] = true;
      }
  }
          /* Critical Section */
  turn = j;
  flag[i] = false;
          /* Remainder Section */
}while(true);
```

Ans:

(1). Mutual exclusion is ensured through the use of the flag and turn variables. If both processes set their flag to true, only one will succeed, namely, the process whose turn it is. The waiting process can only enter its critical section when the other process updates the value of turn.

(2). Progress is provided, again through the flag and turn variables. This algorithm does not provide strict alternation. Rather, if a process wishes to access their critical section, it can set their flag variable to true and enter their critical section. It sets turn to the value of the other process only upon exiting its critical section. If this process wishes to enter its critical section again—before the other process—it repeats the process of entering its critical section and setting turn to the other process upon exiting.

(3). Bounded waiting is preserved through the use of the TTturn variable. Assume two processes wish to enter their respective critical sections. They both set their value of flag to true; however, only the thread whose turn it is can proceed; the other thread waits. If bounded waiting were not preserved, it would therefore be possible that the waiting process would have to wait indefinitely while the first process repeatedly entered—and exited—its critical section. However, Dekker's algorithm has a process set the value of turn to the other process, thereby ensuring that the other process will enter its critical section next.