CS 423/523 Midterm

Total: 100 points
Part 1: Process & Thread (25 points)

1.1 Briefly answer the following questions (3 x 3 points each = 9 points)

1.1.1. What is the difference between a process and a thread?

A thread could be looked as a light-weight process. A process may have multiple threads, and these threads share the status and code of this process.

Any reasonable answer is acceptable.

1.1.2. Under what circumstances does a process become an orphan?

If the process’s parent terminates without invoking wait(), the process becomes an orphan.

1.1.3. What is the difference between concurrency and parallelism?

Concurrency is for single CPU-processor system. It supports more than one tasks making progress. Parallelism implies a system with multiple processors and it can perform more than one tasks simultaneously.

1.2 Multiple choice (5 x 2 points each = 10 points)

1.2.1. A CPU scheduler
   a. Controls the degree of multiprogramming
   b. Selects the process that is going to be executed next
   c. Is also called long-term scheduler
   d. Manages memory and disk space

1.2.2. Which of the following is not correct about indirect process communication
   a. The link may be associated with many processes
   b. There exists exactly one link between each pair
   c. Links may be unidirectional
   d. Processes can communicate only if they share a mailbox

1.2.3. Which of the following is not a process state
   a. idle
   b. terminated
   c. running
   d. new

1.2.4. Which of the following is not correct
   a. The program code is also called the text section
   b. The data section contains global variables
   c. The heap contains statically allocated memory
   d. The stack contains temporary data
1.2.5. Which of the following is not correct about interrupts:
   a. When an interrupt occurs, the OS needs to preserve the state of the CPU
   b. When an interrupt occurs, the OS needs to determine the type of interrupt
   c. Hardware interrupts are caused by a system call
   d. An OS is interrupt driven

1.3. A longer question (6 points)

How many lines of ******* are printed when the following program is executed?

```c
#include<stdio.h>
#include<unistd.h>

int main () {
    int i;
    for(i=0; i<4; i++)
        fork();

    printf("*******\n");
    return 0;
}
```

Remember that when a process forks, it creates a copy of itself, after which both processes return from the system call and execute the next instruction.

16 lines

Part 2: Synchronization (25 points)

2.1. Briefly answer the following questions (3 x 3 points each = 9 points)

2.1.1. Given the fact that we already had hardware mechanism to synchronize processes, why we still need software mechanism, e.g., semaphore and mutex lock?

   Hardware mechanism works well, but it is almost impossible for programmers to use. We need to have software mechanism, thus allowing programmers to explicitly control and synchronize the processes.

2.1.2. Please briefly describe the difference between semaphore and mutex lock (just mention one difference).

   Semaphore does not necessarily need busy waiting -- we can use queue to implement the no busy waiting version. In addition, semaphore has counting semaphore which is any value, but mutex lock can only have 1 or 0.
2.1.3. In the basic version of the dining-philosopher problem algorithm, the one mentioned in our class and ppt, what situation can lead to deadlock?

All five philosophers become hungry at the same time and each grabs the left chopstick.

2.2 Multiple choice (5 x 2 points each = 10 points)

2.2.1. Which of the following is not a requirement for a solution to critical-section problem?
   a. Progress
   b. First come first served
   c. Mutual Exclusion
   d. Bounded Waiting

2.2.2. Which of the following is correct about the most basic Peterson's algorithm:
   a. Can handle multiple (more than two) processes case
   b. Based on assumption that the load and store instructions are atomic
   c. Cannot guarantee bounded-waiting
   d. Mutual exclusion is not preserved

2.2.3. Which of the following is not correct about compare_and_swap instruction:
   a. Is hardware based mechanism
   b. Swaps if and only if comparison succeeds
   c. Do not need busy waiting
   d. Executed atomically

2.2.4. Which of the following is not correct about monitor?
   a. Only one process may be active within the monitor at a time
   b. Internal variables are only accessible by code within the procedure
   c. Do not provide condition variables
   d. Is a high-level abstraction for process synchronization

2.2.5. Which of the following is not correct?
   a. Semaphore contains binary semaphore and counting semaphore
   b. Semaphore is a software-based solution for synchronization
   c. In no busy waiting semaphore implementation, with each semaphore, there is an associated waiting queue
   d. Semaphore may not be an integer variable

2.3 A longer question (6 points)

The following algorithm shows a solution for bounded buffer problem (also called producer-and-consumer problem). Assume the buffer size is infinite. Please answer whether such a solution has synchronization problem? Why?
// producer
do {
    ...
    /* produce an item */
    ...
    wait(empty);

    // add the produced item to the next available buffer
    buffer[in] = produced_item;
in += 1;
counter += 1;

    signal(full);
} while (true);

// consumer
do {
    wait(full);

    // remove an item from the buffer
    while (counter == 0)
    {
        // do nothing
        next_consumed = buffer[out];
    out += 1;
counter -= 1;

    signal(empty);

    ...
    /* consume the item */
    ...
} while (true);

This solution has synchronization problem. Since it does not have mutex for put/remove produced/consumed items, multiple producers/consumers may put/remove the item at/from the same buffer slot.

Part 3: CPU Scheduling (25 points)

3.1. Briefly answer the following questions (3 x 3 points each = 9 points)

3.1.1. What is the difference between preemptive and non-preemptive CPU scheduling?

Non-preemptive CPU scheduling is once a CPU is assigned to a certain process, the process keeps this CPU until it releases the CPU, but preemptive CPU scheduling is contrast -- a CPU can be taken away from another process at any time.
3.1.2. Please list two scheduling algorithms you know (just the name).

First-come first-served, shortest-job-first, shortest-remaining-time-first, and priority scheduling

3.1.3. What is the difference between multilevel queue scheduling and multilevel feedback queue scheduling?

Multilevel queue scheduling is processes are permanently assigned to a queue, but multilevel feedback queue scheduling allows a process to move between queues.

3.2 Multiple choice (5 x 2 points each = 10 points)

3.2.1. Which of the following is not a valid process state transition?
   a. running to waiting
   b. running to ready
   c. waiting to ready
   d. ready to waiting

3.2.2. Which of the following is not correct about Shortest-Job-First scheduling?
   a. It achieves minimum average waiting time for a given set of processes
   b. The difficulty is knowing the length of the next CPU request
   c. It is not priority scheduling
   d. The algorithm associates with each process the length of its next CPU burst

3.2.3. Which of the following is correct about Round-Robin scheduling?
   a. Timer interrupts every quantum to schedule next process
   b. Can provide the optimal waiting time
   c. Each process is not preempted
   d. It is impossible to draw a Gantt chart for Round-Robin scheduling process

3.2.4. Which of the following is correct about Shortest-remaining-time-first scheduling?
   a. It concentrates on response time
   b. It can be looked as a preemptive version of shortest job first (SJF) scheduling
   c. It has the optimal average waiting time
   d. It does not need preemption

3.2.5. Which of the following is not a CPU scheduling criteria?
   a. Disk throughput
   b. CPU utilization
   c. Amount of time a process has been waiting in the ready queue
   d. Amount of time to execute a particular process

3.3 A longer question (2 x 3 points each = 6 points)
<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
</tr>
<tr>
<td>P2</td>
<td>9</td>
</tr>
<tr>
<td>P3</td>
<td>7</td>
</tr>
<tr>
<td>P4</td>
<td>2</td>
</tr>
</tbody>
</table>

Provide a Gantt chart for the above processes using shortest-job-first (SJF) scheduling.

P4, P1, P3, P2

Please compute average waiting time (All the processes arrive at the same time).

\[
\frac{0 + 2 + 7 + 14}{4} = \frac{23}{4} = 5.75
\]

**Part 4: OS Structures and Deadlocks (25 points)**

### 4.1. Briefly answer the following questions (3 x 3 points each = 9 points)

**4.1.1.** What is the difference between deadlock and starvation?

Deadlock is multiple processes wait for or block each other in a loop order, but starvation means a certain process cannot get resources it needs.

**4.1.2.** What is the difference between microkernel and kernel?

Microkernel only contains the minimal OS functions, e.g., memory management and CPU scheduling, and move many functions, e.g., file management and device driver, to user space. In other words, microkernel tries the best to make kernel “thinner”.

**4.1.3.** What is the difference between real-time OS and traditional OS?

Real-time OS requires the tasks have to be finished within the time constraints.

### 4.2. Multiple choice (5 x 2 points each = 10 points)

**4.2.1.** Which of the following is not correct about system boot?

a. Execution starts at a fixed memory location after power on
b. Bootstrap loader is needed to load and start the kernel
c. Sometimes bootstrap can be a two-step process where bootstrap loader is loaded by ROM code
d. Initial boot code can also be held in RAM.

**4.2.2.** Which of the following is the scope of OS security?

a. Denial-of-Service (DoS) defense
b. Worm attack
c. Identity theft
d. All of the above

4.2.3. Which is not correct about system call?
a. Typically a number is associated with each system call for identification
b. Parameters can be passed in registers or in memory
c. The programmer has to know the API as well as its underlying implementation
d. System calls are mostly accessed by programs via a high-level API rather than
direct system call use

4.2.4. Which of the following is correct about Banker’s algorithm?
a. Each process must a priori claim maximum use
b. A process does not need to return its resources after they’re granted
c. It works only if the number of processes is less than the number of resource types
d. It works only if there’re more than one instances for each type of resource

4.2.5. Which of the following is not correct about resource-allocation graph?
a. It contains cycles if there is deadlock
b. It contains no cycle if there is no deadlock
c. A vertex represents either a process or a resource type
d. An edge represents either a request or an assignment

4.3 A longer question (6 points)

The following table shows snapshot at time T. There are 5 processes, 3 resource types: A (10 instances), B (5 instances), and C (7 instances).

<table>
<thead>
<tr>
<th></th>
<th>Allocation</th>
<th>Max</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A  B  C</td>
<td>A  B  C</td>
<td>A  B  C</td>
</tr>
<tr>
<td>P0</td>
<td>0  1  0</td>
<td>7  5  5</td>
<td>3  3  2</td>
</tr>
<tr>
<td>P1</td>
<td>2  0  0</td>
<td>3  2  2</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>3  0  2</td>
<td>9  5  2</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>2  1  1</td>
<td>2  2  2</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>0  0  2</td>
<td>4  3  3</td>
<td></td>
</tr>
</tbody>
</table>

Please use Banker’s algorithm to check whether this system is in a safe state? If not, please explain why; if it is in a safe state, please give a sequence that satisfies safety criteria.

This is in a safe state, and the sequence is <P1, P3, P4, P0, P2>