COMP 252 Principles of Systems Software

Fall Semester 2000

Midterm Examination

Date: October 21, Time 5pm : 7pm

Name:___SOLUTION_____Student ID:_____Email:_____

Instructions:

1. This is a closed book/notes exam. Don't discuss the exam questions with your colleagues during the exam.
2. This examination paper consists of 7 pages and 6 questions.
3. Please write your name, student ID and Email on this page.
4. For each subsequent page, please write your student ID at the top of the page in the space provided.
5. Please answer all the questions within the space provided on the examination paper. You may use the back of the pages for your rough work.
6. Please read each question very carefully and answer the question clearly and to the point. Make sure that your answers are neatly written, readable and legible.
7. Show all the steps you use in deriving your answer, where ever appropriate.
8. For each of the questions assume that the concepts are known to the graders. Concentrate on answering to the point what is asked. Do not define or describe the concepts.

Question	Points	Score
1	15	
2	20	
3	15	
4	10	
5	20	
6	20	
TOTAL	100	

1. a) What does SPOOLING stand for ? Describe the SPOOLING operation and the advantage of doing this. (5 points)

Answer: SPOOLING is Simultaneous Peripheral Operation On-line. The disk is used as a large buffer for reading ahead on input devices and for storing output files until the output devices are able to accept them. SPOOLING allows overlap of the I/O of one job with the computation of other jobs and improve system throughput.

b) What does DMA stand for ? Describe the DMA operation and the advantage of doing so. (5 points)

Answer: DMA stands for Direct Memory Access. Data are directly transferred between main memory and the input or output device one block at a time. The CPU is interrupted only after the entire block has been transferred instead of one character at a time. It reduces the overhead of the CPU in handling I/O interrupts so it can spend more time on computation.

c) What is a zombie process ? (5 points)

Answer: A zombie process is a child process that has exited but not yet acted upon by its parent process with the WAIT system call. Its state is maintained by the operating system but it is considered dead otherwise (i.e., cannot run or compete for system resources).

2. a) In the Client-Server Model of operating system structure, most of the traditional operating system functions are implemented as servers running in the user mode.
   i) name two advantages of doing so; (6 points)

Answer: - implementing each function as a separate server allows modularity and is easier to do as it enables the implementor to focus on one function at a time.
- debugging in the user mode is easier as one can use the existing debugging tools
- an error in a server in the user mode may not bring down the entire system, while a kernel error is more serious

   ii) what is the major function of the kernel in this model ? (4 points)

Answer: The main function is to enable communications between the clients and the server modules.

b) What are the main advantages and disadvantages of using threads instead of processes ? (6 points)

Answer: Advantages: multiple threads can execute in a single address space so that sharing of resources and data is easy. Creation and deletion of threads are much less expensive.
Disadvantage: threads running in the same address space may interfere with one another if there is not properly synchronized.

c) Name four reasons why a process may leave the running state. (4 points)

Answer: - process termination
- time quantum expiration
- waiting for I/O
- preempted by another process

3. Consider the following code which represents dining philosopher application with 3 philosopher threads and three chopsticks. Assume that all the objects are properly created and that variables are properly initialized. Find and state the major problem in this code and correct it by adding some lines to the code. (15 points)

```
Lock            *MyLock;
Condition       *chopstick[3];
Thread          *Phil[3];
bool        chopstick_busy[3];

void    Philosopher(int x)
{       int phil_id = x;

1.  for (int num =0; num < 3; num ++) {
2.      MyLock->Acquire();
3.        while  (chopstick_busy[phil_id])            chopstick[phil_id]->Wait(MyLock);
4.        while  (chopstick_busy[(phil_id +1) % 3])  chopstick[(phil_id + 1) % 3]->Wait(MyLock);
5.        chopstick_busy[phil_id] = TRUE;
6.        chopstick_busy[(phil_id +1) % 3] = TRUE;
7.      MyLock->Release();

8.      cout << "Philosopher " << phil_id << " is eating\n";

9.
10.       chopstick[phil_id]->Signal(MyLock);
11.       chopstick[(phil_id + 1) % 3]->Signal(MyLock);
12.       chopstick_busy[phil_id] = FALSE;
13.       chopstick_busy[(phil_id +1) % 3] = FALSE;
14.

15.     cout << "Philosopher " << phil_id << " is thinking\n";

  }
}
```

Answer: Signaling on conditional variables is done without acquiring the lock.
Line 9. MyLock->Acquire();
Line 14. MyLock->Release();

4. Consider the following solution to the too much milk problem. Explain step by step, what is the problem with this solution? (10 points)

```
 Thread A                    Thread B
Leave Note A;                Leave Note B;
if (no Note B) {             if (no Note A) {
   if (no Milk) {               if (no Milk) {
  buy Milk;                    buy Milk;
   }                            }
}                            }
Remove Note A;               Remove Note B;
```

Answer: Consider the following situation:

- Thread A executes Leave Note A and gets interrupted.
- Then, thread B executes Leave Note B and gets interrupted.
- Thread A resumes, and the if (no Note B) condition fails. So thread A does not buy milk. It then gets interrupted.
- Thread B resumes, and the if (no Note A) condition fails. So thread B does not buy milk. It then removes Note B.
- Thread A resumes and removes Note A

In this case neither thread buys milk, thus failing the progress requirement.

5. In the Physical Education (PE) lesson a number of PE teachers are teaching students to play table tennis. There is only one table for the whole class. PE teachers wait until two students come and are ready to take a short lesson. When a pair of students are ready, one free PE teacher takes them to the table and if the table is free he starts giving a lesson. Table must be used exclusively for one short lesson at a time. The students have to wait until their lesson is finished before leaving. Once the lesson is completed the PE teacher goes back and waits for the next couple of students. This operation can be described using multiple threads representing the PE teachers and students. Implement the proper coordination mechanism between the PE teachers and students. For your convenience a skeleton structure for the threads is given below and you have to insert the synchronization actions in proper places. (20 points)

//Global variables and their initial values

Semaphore Student_Ready; (initial value=0)
Semaphore Student_Leave; (initial value =0)
Semaphore Table_Free ; (initial value =1);

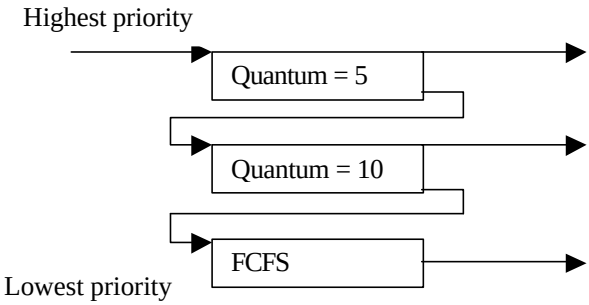
PE_Teacher_Thread() {	Student_Thread() {
While(TRUE) {	// Indicate student ready
// Wait until two students are ready	Student_Ready->V();
Student_Ready->P();	
Student_Ready->P();	// wait until the lesson is over
// Access the table	Student_Leave->P();
Table_Free->P();	
// give a lesson in table tennis	}
// release the table	
Table_Free->V();	
// complete the lesson and let the students go	
Student_Leave->V();	
Student_Leave->V();	
} // end while	
}	

6. Consider a 3-level feedback queue with Round-Robin Scheduling used for the two high-priority queues and FCFS used for the lowest priority queue as shown in the figure below: (20 points)

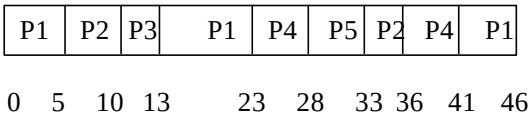
a) Construct a Gantt chart to display the sequence of process executions. Note that if the process from the lower priority queue is preempted with the process from the higher priority queue, it will go to the end of the same queue and it will be scheduled again until it spends the remaining time quantum. (15 points)

b) What is the average waiting time of the following sequence of process executions constructed in part a) (5 points).

Process	P1	P2	P3	P4	P5
CPU-burst	20	8	3	10	5
Arrival time	0	2	5	23	25



a) Gant chart



c) waiting time

W1=(0-0)+(13-5)+(41-23)=26

W2= (5-2)+(33-10)=26

W3=(10-5)=5

W4= (23-23) + (36-28) = 8

W5 = (28-25) = 3

W= 68/5=13.6