BITS, Pilani – Dubai
Dubai International Academic City, Dubai

III Year (Computer Science)
First Semester, 2010-2011

Comprehensive Examination

Course No: CS C321 | **Course Title: Operating Systems**
Date: 28th Dec 2010 | **Weightage: 40%**
Duration: 3 Hours | **Max. Marks. 80**

(Answer Parts A and B on separate answer books.)
(Answer the questions in the sequential order.)
(Answer all the parts of a question together.)

PART – A

1. a) Explain the different parameters in the Disk I/O Transfer timing, with diagram. **[4M]**
b) A disk scheduler receives the following sequence of request for tracks. 27, 129, 110, 186, 147, 41, 10, 64, 120. Find out the total seek time for both FIFO (First In First Out) and SSTF (Shortest Seek Time First) scheduling policies, assuming a seek time of 1ms for every track. (Initially request queue is empty, and head is on track no 0, which is inner most track and track 200 is the outer most track). **[6M]**

2. a) Clearly bring out the distinction between deadlock avoidance and deadlock detection strategies used in OS. **[4M]**
b) With respect to the following resource allocations for the processes P1, P2, and P3 , Resources R1, R2, R3 and R4, determine using the deadlock detection strategy to check for deadlocks. Show the steps clearly. **[6M]**

	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

Request Matrix

	R1	R2	R3	R4
P1	0	0	1	1
P2	2	0	0	1
P3	0	1	2	0

Allocation Matrix

Resource Vector			
R1	R2	R3	R4
4	2	3	2

2	1	0	0

Available Vector

3. a) Explain clearly what memory compaction means and its need in Dynamic Memory Allocations in OS. **[4M]**
b) With clear diagram explain the paging hardware mechanism and its working. In Virtual Memory Management? **[6M]**

[PTO]

4. a) What is page fault. Explain the steps taken by the OS to handle page fault. **[4M]**
b) Shown below is the code for reader-writer problem. Explain clearly why the first reader thread alone calls wait (w_or_r). Why there is a need for using semaphore mutex before updating readcount? **[6M]**

```
int read_count = 0;
semaphore mutex = 1;
semaphore w_or_r = 1;

writer {
    wait ( w_or_r );
    write();
    signal ( w_or_r );
}
```

```
reader() {
   wait (mutex);
   readcount += 1;
   If (readcount == 1)  wait (w_or_r);
   signal (mutex);
   read();
   wait(mutex);
   readcount -= 1;
   If ( readcount == 0 ) signal (w_or_r);
   signal (mutex);
}
```

PART – B

5. Following is a code of a multithreaded java program. 4m

```
public class Demo implements
Runnable {
  private int x, y, z;

  public Demo() {
    this.z=0;
    Thread newthread = new
Thread(this);
    newthread.start();
    int Number = 5*7*8*9;
    int myId = 1;
    try {
      Thread.sleep(150);
      this.y=Number;
    } catch (Exception e) {}
    this.z=1;
  } // End of Demo()
```

```
public void run(){
  int denom = 2*3*4;
  this.x=denom;
  int myId = 2;
  try{
    Thread.sleep(90);
  } catch (Exception e) {}
  int temp=this.z;
  while (temp < 1){
    temp= this.z;
  }
  System.out.println("Id : " +
myId +  " final result ="+
this.y/this.x) ;
} // End of run()

} // End of class Demo
```

With respect to the above code.
a) What are the threads created and mentions their ids. **[2M]**
b) What is the requirement for the while loop in the run method. **[4M]**
c)outline how the threads are scheduled in the above example ? **[4M]**

6. a) Show the monitor based solution fot the producer-consumer problem using condition variables. **[7M]**
b) What is the difference between condition variables (used in monitors) and semaphore, **[3M]**

[PTO]

7. a) Explain what is meant by bootstrapping in OS ? **[5M]**

b) Explain the advantages of interrupt driven I/O data as compared to polling based data transfer. **[5M]**

8. Shown below is the code for acquire() and release() implementation for locks

```
struct lock {
  int held = 0;
}

void acquire (lock) {
    while (lock->held) ;
    lock->held = 1;
}

void release (lock) {
    lock->held = 0;
}
```

a) Explain the problem with the above code. Describe two techniques (show implementations as required), by which the problem can be resolved. **[4M]**

b) Justify whether we can use "disable interrupts" for lock implementation in multiprocessor systems. **[6M]**

BITS, Pilani – Dubai

Dubai International Academic City, Dubai

III Year - CS
First Semester 2010-2011

COURSE NO. : CS C372 — COURSE TITLE : Operating Systems

Test 2 (Open Book) — **Date : 11-12-2010.**

Total marks=40, Weightage=20%

Class notes and Text book alone are permitted.

Answer all the questions

1. Shown below is the implementation of reader-writer problem. Two reader threads (R1, R2) are created at t=1.5ms, t=2.5ms and each thread takes 5ms to complete the read() method. A writer thread is started at 3.5ms. Assume other delays are negligible.

```
int read_count = 0;
semaphore mutex = 1;
semaphore w_or_r = 1;

writer {
    wait ( w_or_r );
    write();
    signal ( w_or_r );
}
```

```
reader() {
    wait (mutex);
    readcount += 1;
    If (readcount == 1)  wait (w_or_r);
    signal (mutex);
    read();
    wait(mutex);
    readcount -= 1;
    If ( readcount == 0 ) signal (w_or_r);
    signal (mutex);
}
```

a) Specify, the state (count and queue) of w_or_r semaphore, and the value of readcount, at times 2ms, 3ms, 4ms, 6ms, 7ms and 8ms **[6M]**

b) Why and at what time does the writer thread enter the critical section? **[4M]**

2. Resource allocations for the processes P1, P2, and P3 using resources R1, R2, and R3 are given below. Consider each of the following requests independently and find out whether they can be granted, using deadlock avoidance algorithm

	R1	R2	R3
P1	2	1	2
P2	3	2	4
P3	4	2	1

Claim Matrix

R1	R2	R3
1	0	1
0	0	1
1	1	1

Allocation Matrix

Resource Vector		
R1	R2	R3
4	2	5
Available Vector		
2	1	2

a) P3 requests 1 0 0? **[5M]**

b) P2 requests 2 0 0? **[5M]**

PTO

3. a) Less error prone as the compiler implements mutual exclusion to ensure that only one process./thread is allowed to enter monitor. Because its tthe compiler, not the programmer is arranging for mutual exclusion it is much less likely that something will go wrong. In semaphore, exchanging order of calls to wait and signal can cause deadlock.

Provides an object oriented construct, so data is can be accessed by procedures in the monitor. Inadvertent manipulation of data does not happen. **[5M]**

b) Assume that the line **csignal(notempty);** is removed and that No producer has yet executed. Now if a consumer enters take, it would be queue on non empty, waiting for a producer to signal notempty. However this would not happen in the producer as the signal is missing. So the consumer would be hung for ever. **[5M]**

4.

```
semaphore num_ip_addr = 5 ; // counting semaphore
//client code
get_ip_and_do_work() {
  wait(num_ip_addr);
  get_ip_addr();
  do_work();
  surrender_ip_addr();
  signal(num_ip_addr);
}
```

when more that 5 clients call get_ip_and_do work, the semaphore num_ip_addr would hae been decremented to 0. So the next client calling get_ip_and_do_work(), would be queued on num_ip_addr. This will be awakened when one of the clients surrenders_ip_add and signals num_ip_addr. Thus it is ensured that only 5 clients can connect concurrently to the internet. **[10M]**

BITS, PILANI – DUBAI
FIRST SEMESTER 2010 – 2011
THIRD YEAR (CS)

A

Course Code: CS C372
Course Title: Operating Systems
Duration : 20 minutes

Date: 09.12.10
Max Marks: 14
Weightage: 7%

Quiz 2(Closed book)

Name: . . . . . ID No: . . . . . Sec / Prog: . . . . . .

Instructions: (if any) Answer all the Question

1. For the producer-consumer code shown below, justify the effect of interchanging the statements in Consumer code shown in **bold** (i.e. **wait(full); wait(mutex);** → **wait(mutex),wait(full);**), Justify? **[2M]**

Semaphore mutex = 1; Semaphore empty = N; Semahore full =0;

```
Producer() {
  while(1) {
     Produce new resource;
     wait(empty);
     wait(mutex);
     Add resource to an empty buffer;
     signal(mutex);
     signal(full);
  }
}
```

```
Consumer() {
  while(1) {
     wait(full);
     wait(mutex);
     Remove resource from a full buffer;
     signal(mutex);
     signal(empty);
     consume new resource;
  }
}
```

2. In the code shown in Q1.,justify how the consumer is prevented from consuming an item when the buffer is just empty? **[2M]**

3. Write briefly on the difference between conditional wait in monitors and semaphores? **[2M]**

PTO

A

4. For the reader-writer code shown below, justify the need for **w_or_r** semaphore? **[2M]**

```
int  read_count = 0;
semaphore mutex = 1;
semaphore w_or_r = 1;

writer() {
      wait ( w_or_r );
      write();
      signal ( w_or_r );
}
```

```
reader() {
    wait (mutex);
    read_count += 1;
    If (read_count == 1)  wait (w_or_r);
    signal (mutex);
    read();
    wait(mutex);
    read_count -= 1;
    If ( read_count == 0 ) signal (w_or_r);
    signal (mutex);
}
```

5. In Q4, justify the need for **mutex** semaphore? **[2M]**

6. Describe briefly the situations where mutex and counting semaphores are used? **[2M]**

7. In the code shown below, explain what will happen, if the line **if (count==k) cwait(notfull);** is removed? **[2M]**

Monitor boundedbuffer:
buffer: array[0..k-1] of items; count:=0: integer; notfull, notempty: condition;

Append(v):	Take(v):
if (count==k) cwait(notfull);	if (count==0) cwait(notempty);
ADD ITEM	REMOVE ITEM
count++;	count--;
csignal(notempty);	csignal(notfull);

BITS, PILANI – DUBAI

A

FIRST SEMESTER 2010 – 2011

Course Code: CS C372 | **THIRD YEAR (CS)** | Date: 24.11.10

Course Title: Operating Systems | **QUIZ 1** | Max Marks: 16

Duration : 20 minutes | Weightage: 8%

Name: . . . . . ID No: . . . . . Sec / Prog: . . . . . .

Instructions: (if any) Closed book- Answer all the Questions

1. Shown below is the implementation of acquire() and release() for locks. What is the design flaw? **[2M]**

```
struct lock {
  int held = 0;
}

void acquire (lock) {
    while (lock->held) ;
    lock->held = 1;
}

void release (lock) {
    lock->held = 0;
}
```

2. In Q1, give a solution by which the flaw can be removed? **[2M]**

3. "When a thread of a process blocks, the whole process gets blocked". The statement is **TRUE** for **[2M]**
   i) Kernel Level Threads
   ii) User Level Threads
   iii) Both Kernel Level Threads and User Level Threads
   iv) None of the above

4. What is the minimum information to be saved and restored during a thread switch? **[2M]**

P.T.O

A

5. Which kind of threading can utilize multiple CPUs and justify the reason for the same. **[2M]**

6. Which of the following is TRUE for mutual exclusion? **[2M]**
   i) Only one thread at a time can execute in the critical section
   ii) All other threads are forced to wait on entry
   iii) When a thread leaves a critical section, another can enter
   iv) All of the above

7. Thread switching time in User Level Threads is less than that in Kernel Level Threads. Why ? **[2M]**

8. Give an implementation of non-spinning type of lock? **[2M]**