**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**
**First Semester 2003-2004**

**Course Title : OPERATING SYSTEMS** **Course No *CS C372***
**Component : Comprehensive Exam(Regular)** ***Open Book Component***
**Weightage : 38%** **Max Marks: 38** **Date: 07/12/2004**

---

**Note: Attempt all the Questions. Start each answer from a fresh page.**

**Question #1** **(9 Marks)**

Unix provides two mechanisms to link one file to another, hard links and soft links. With hard links, assume that the directory entry for the link maps the link file name to the inode for the file to which it is linked. With soft links, assume that the directory entry for the link maps the link file name to the file name of the file to which it is linked. Consider the situation where we want to create a link "/bin/ls" to the existing file "/sbin/ls". In this case "/bin/ls" is the link file name and "/sbin/ls" is the file to which it is linked. For the questions below assume that each time Unix has to retrieve information from the disk, it takes only one disk read operation. Also assume that all directories are one block in size and the master block is not in memory.

a) Describe (in points) what steps Unix will take, in terms of the disk data structures it will read, in order to resolve the path name "/sbin/ls" so that it can read the first byte of the file. How many disk reads will it require? **(2 Marks)**

b) Consider we use a hard link to link "/bin/ls" to "/sbin/ls". Describe the steps to read the first byte of "/sbin/ls" starting with the link "/bin/ls". How many disk reads will it require? **(2 Mark)**

c) Consider we use a soft link to link "/bin/ls" to "/sbin/ls". Describe the steps to read the first byte of "/sbin/ls" starting with the link "/bin/ls". How many disk reads will it require? **(2 Mark)**

d) Unix maintains a reference count of the number of hard links to the file, and it only removes a file's block on disk when this count reaches zero. If we remove the file "/sbin/ls", is the hard link still valid (Yes/No)? Is the soft link still valid (Yes/No)? **(1 Mark)**

(e) On a Unix file system, how many disk read operations are required to read the first block of the file "/home/solaris/ieng9/csc372/file"? Assume that the master block is in memory, but nothing else. Also assume that all directories are one block in size. **(2 Marks)**

**Question #2** **(6 Marks)**

(a) A program runs on a RISC architecture such that an instruction takes 4 nsec to execute if it does not include a data access, otherwise it takes 10 nsec. Moreover, the program may incur page faults with a probability of 0.0005% at each instruction. A page fault takes about 30 msec to service. Compute the estimated time it takes a program to execute a million instructions if 20% of these require data access. **(2 Marks)**

(b) A computer provides each process with 65,536 bytes of address space divided into pages of 4096 bytes. A particular program has a text size of 32,768 bytes, a data size of 16,386 bytes, and a stack size of 15870 bytes. Will this program fit in the address space? If the page size were 512 bytes, would it fit? Remember that a page may not contain parts of two different segments. **(4 Marks)**

**Question #3** **(3 Marks)**

Consider the file currently consisting of 100 blocks. Assume that the file control block (and the index blocks, in the case of the indexed allocation) is already in the memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single level) allocation strategies, if, for one block, the following conditions hold. In the contiguous allocation case, assume that there is no room to grow in the beginning, but there is room to grow in the end. Assume that the block information that should be added is stored in memory. (Each question carries 0.5 marks that will be awarded only if the 3 entries are right).

a. The block is added at the beginning.
b. The block is added in the middle ($51^{st}$ block)
c. The block is added at the end
d. The block is removed from the beginning
e. The block is removed from the middle ($51^{st}$ block)
f. The block is removed from the end

**Question #4** **(3 Marks)**

Consider following program where N = 1024

```
long  A[N][N], B[N][N], C[N][N] ;
for ( int i = 0; i < N; i ++)
        for ( int j = 0; j < N; j ++)
                for ( int k = 0; k < N; k ++)   C[i][j] += A[i][k] * B[k][j];
```

Suppose that the size of long is 4 and the page size is 4KB. The program starts with only the segments for text and constants in the main memory.

How many page faults will occur if we use an LRU replacement model and a fixed page allocation of 4 pages? Assume that the arrays are stored in row major order and the counting of page faults begins after the text and the constants have been loaded into memory.

**Question #5** **(4 Marks)**

Consider the following snap shot of the system of five processes P0 to P4 and 4 resource types, A, B, C, D. The total instances of A, B, C and D in the system are 3, 12, 15 and 11 respectively.

| Processes | Maximum Claim | | | | Need | | | |
|---|---|---|---|---|---|---|---|---|
| | *A* | *B* | *C* | *D* | *A* | *B* | *C* | *D* |
| **P0** | **3** | **10** | **15** | **7** | **2** | **7** | **10** | **5** |
| **P1** | **2** | **7** | **10** | **5** | **1** | **4** | **5** | **3** |
| **P2** | **3** | **5** | **10** | **8** | **3** | **5** | **10** | **8** |
| **P3** | **1** | **4** | **5** | **3** | **0** | **1** | **0** | **1** |
| **P4** | **2** | **10** | **10** | **6** | **2** | **10** | **10** | **6** |

(a) Determine if the system is in the safe state. If yes, give the safe sequence.
(b) If process P2 requests (0,X,0,Y), then determine the maximum values of X and Y which can be granted without making the state unsafe.
(c) Assume that at time T0, P4 made a request (0,3,0,4) and this request was granted. Determine if the system is in safe state and give the safe sequence if any.

**Question #6** **(3 Marks)**

For each of the following code segments, where the variables are initialized as shown, and thread A and B can run concurrently, what are the possible ending values of x for Process A, Process B and Process C after both threads are completed?

| Process A | Process B | Process C |
|---|---|---|
| **int x=2;**<br>**semaphore m = 1;**<br>**semaphore s = 0;**<br><br>**thread A {**<br>**x = x + 4 ;**<br>**x = x + 1;**<br>**}**<br><br>**thread B {**<br>**x = 2 * x ;**<br>**}** | **int x=2;**<br>**semaphore m = 1;**<br>**semaphore s = 0;**<br><br>**thread A {**<br>**x= x + 4 ;**<br>**wait(m) ;**<br>**x = x + 1;**<br>**signal(m) ;**<br>**}**<br><br>**thread B {**<br>**wait(m) ;**<br>**x = 2 * x ;**<br>**signal(m) ;**<br>**}** | **int x=2;**<br>**semaphore m = 1;**<br>**semaphore s = 0;**<br><br>**thread A {**<br>**x = x + 4 ;**<br>**wait(m) ;**<br>**x = x + 1;**<br>**signal(s) ;**<br>**}**<br><br>**thread B {**<br>**wait(s) ;**<br>**x = 2 * x ;**<br>**signal(m) ;**<br>**}** |

**Question #7** **(5 Marks)**

```
int i=3, sum =0;
        main()
        {
                        while( i > 0)  {
                                if (  fork ( ) )
                                        sum = sum + (--i) ;
                                else {
                                        sum = sum + i;
                                        if(i = = 1)  execlp("/bin/ls", "ls",NULL);
                                        i-- ;  }
                                if (  fork ( ) )   {
                                        sum = sum + i + 2 ;
                                        i-- ;  }
                                else {
                                        sum = sum + 1 – i ;
                                        if(i = = 1)  execlp("/bin/ls", "ls",NULL);
                                        i-- ; }
                        }
        }
```

How many new processes will be created? What will be the largest and smallest value of sum if all the parents are waiting for its children to finish execution?

**Question #8** **(5 Marks)**

A multilevel feedback queue algorithm works with the following condition

**Number of queues** 3 (Q1,Q2 and Q3)

**Scheduling algorithm** Q1 R R, Q2 preemptive priority Q3 FCFS

**Method used to upgrade a process** No upgrading among queues

**Method to demote a process** Q1→After 2 units of time, Q2→ When the Priority becomes 0 (Priority of the process reduces by 1 for every 1 units of execution in CPU from Q2), one process in Q2 can preempt running process in Q2 only if the former one got higher priority than the later. While transferring control from any other Queue to Q2 if more than one process have same priority, then the first process who came in the Queue will get the first chance. Q3→The Process who came first to the queue will get the first chance.

**Method used to determine which queue a process will enter**

P1, P3 & P5 are entering through Queue #1 (Q1)

P2 & P4 are entering through Queue #2 (Q2)

| Process | T.CPU burst | CPU burst | I/O burst | Priority | Arrival time |
|---|---|---|---|---|---|
| **P1** | **10** | **5** | **2** | **2** | **0** |
| **P2** | **9** | **6** | **1** | **4** | **1** |
| **P3** | **5** | **3** | **5** | **5** | **3** |
| **P4** | **9** | **5** | **3** | **3** | **6** |
| **P5** | **6** | **4** | **2** | **3** | **9** |

Draw the Gantt chart and calculate Average waiting time, average turn around time, and CPU utilization.