# INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

## Computer Science and Engineering

Course: CS341 (Operating System), End-Semester Exam

Date: 21st Nov 2014 Timing: 1.00PM-4.00PM Full Marks: 60

**1. [Lock, Monitor and Deadlock: 5 Lectures (=26%)]** **16 Marks [3+2+2+3+6]**

a) [**3 Marks**] What are the possible ways to handle deadlock in Dining-Philosophers problem (Five philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers). [May be using lock/monitor/semaphore you can handle this, but you need to say : how to break at least one of the deadlock conditions]

Ans: *(1) Allow at most 4 philosophers to be sitting simultaneously at the table. : 5 fork, 4 people, so one person can have two at a time (2) Allow a philosopher to pick up the forks only if both are available (picking must be done in a critical section : No hold and wait (3) Use an asymmetric solution -- an odd-numbered philosopher picks up first the left chopstick and then the right chopstick. Even-numbered philosopher picks up first the right chopstick and then the left chopstick.*

b) [ **2 Marks**] Show that, if both the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated.

Ans: *Wai()t is Counter Increment and Signal() is Counter Decrement, we know if both counter increment and decrement in not executed atomically then they may interleave, it may produce wrong result. In general counter increment/decrement get implemented using three instructions LOAD R1 CTR, INC/DEC R1, STR R1 CTR. If we are not executing atomically then, we cannot say counter increment (or decrement) is a single event. To make the counter increment (or decrement) as single event, we need to execute atomically. Atomically means execute all statements of the function or null.*

c) [ **2 Marks**] Write a monitor that implements a counting semaphore for a resource with 25 instances. The monitor should have three public functions: init_ctr(), wait_ctr() and signal_ctr();

Ans:

```
class Monitor_CTR_SEMAPHORE {
        condition Block; int count;
          public wait(){  count--;  if (count<0) cond_wait(block); }
          public signal(){ count++;  if (count<=0) cond_signal(block); }
          public init(){count=25;}
};
```

d) [**3 Marks**] A system has 10 processes and three identical resources. Each process needs a maximum of two resources to complete. Is deadlock possible? Explain your answer. [*No credit for answer YES/ NO, justification carry marks*]

*Ans: Deadlock is possible (The system is not deadlock free). Three identical resources can be assumed as a resource with three instances. Suppose that three processes hold one resource instance each and requested for one. Then they will be in deadlock.*

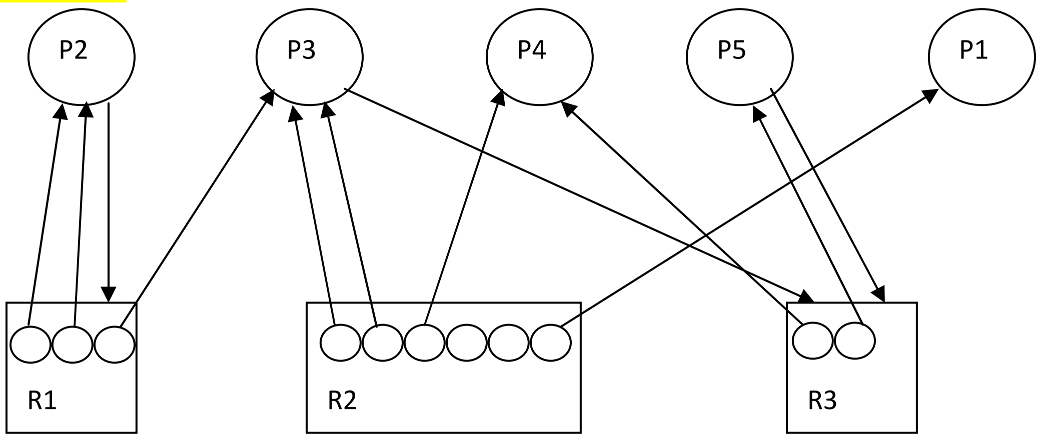
e) [**6 Marks:2+2+1+1**] Draw Resource Allocation Graph (RAG) for the system described in the following Table. Annotate edges (with X symbol) for reduction of the resource allocation graph. Is system in a safe or unsafe state? Is the system deadlocked? [*You will not get any credit of last two questions unless you draw the RAG correctly and reduce*]

| | Current Allocation | | | Outstanding Allocation | | | Maximum Allocation | | | Resource Available | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | |
| P2 | 2 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 1 | 0 | 2 | 0 |
| P3 | 1 | 2 | 0 | 0 | 0 | 1 | 2 | 5 | 2 | | | |
| P4 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 4 | 2 | | | |
| P5 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 0 | 1 | | | |

Ans:

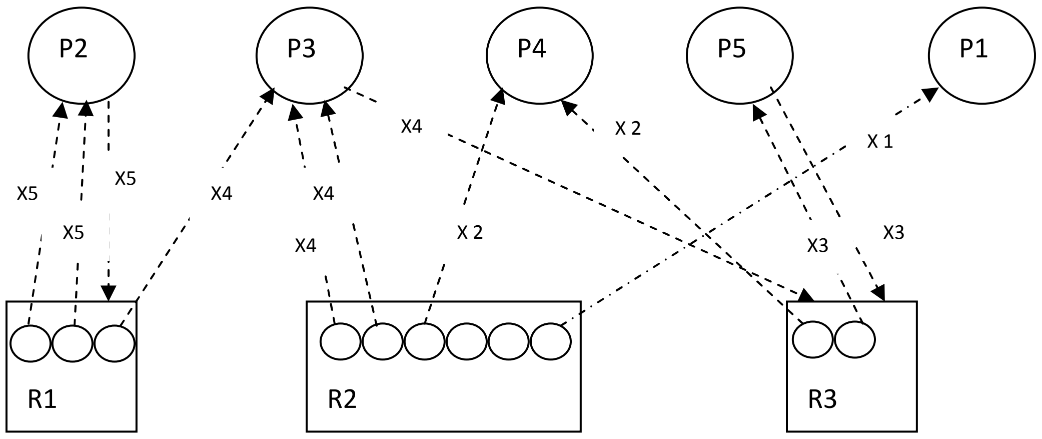
Total number of instances of a resources = current allocation + available, Total instances of R1=3, R2=6 and R3=2.

RAG is shown below:



Is system is safe or Unsafe:
As maximum allocation is greater than current allocation plus the request or outstanding, so safety algorithm abort unconditionally will not work. Everyone will get 1 mark with condition that person have drawn the RAG.

Reduced RAG is shown below: No process left to execute after reduction, so system in not in deadlock state.



2. **[Main Memory and Virtual Memory: 8 Lectures (=42%)]** **25Marks [4+2+3+3+4+6+5]**

a) [ **2+2 Marks** ] What is the cause of thrashing? How can you model thrashing based on working set of processes?

*Ans: In a virtual memory system, thrashing is a condition in which excessive paging operations (page replacement or swapping in and out) are taking place. Thrashing is computer activity that makes little or no progress, usually because memory have become exhausted or too limited to perform needed operations.*

*Suppose a system, have m frames in memory. $WSS_i$ is working set of process $P_i$ and that is total number of pages referenced in the most recent $\Delta$ (varies in time). The total demand frames of all the running process is $D = \Sigma WSS_i$, (which is approximation of locality of all the running process in the system). If $D > m$ then it implies there is a Thrashing problem.*

b) [**2 Marks**] Prove that optimal page replacement algorithm will not suffer Belady's anomalies.

*Ans: An optimal algorithm will not suffer from Belady's anomaly because by definition an optimal algorithm replaces the page that will not be used for the longest time. Belady's anomaly occurs when a page replacement algorithm evicts a page that will be needed in the immediate future. An optimal algorithm would not have selected such a page.*

*c)* [**3 Marks**] Segmentation is same as paging but uses variable sized pages. Define LRU segment replacement algorithm for demand segmentation.

*Ans: Since segments are not same size, the segment that is chosen to be replaced may not be big enough to leave consecutive locations for the needed segment. We may have two strategies where segments cannot be relocated and they can.*

- *Based on assumption segment cannot be relocated, we need to find a LRU segment which size plus nearby free fragment near that segment is bigger than the demanded one that definite the fit.*
- *Based on assumption segment cannot be relocated: we need to find a LRU segment which size plus available fragmented free space is bigger than the demanded one.*

d) [**1.5+1.5 Marks**] What kind of hardware support OS need to implement TLB? Is it possible to increase TLB size of a computer by upgrading or updating the OS?

*Ans: TLB itself is hardware and OS simply use this. A translation look aside buffer (TLB) is a **hardware cache** that **memory management hardware** uses to improve virtual address translation speed. The majority of desktop, laptop, and server processors include one or more TLBs in the memory management hardware, and it is nearly always present in any hardware that utilizes paged or segmented virtual memory. The TLB is implemented as content-addressable memory (CAM). The CAM search key is the virtual address and the search result is a physical address, this search is **done parallel and it is quick**.*

*There is no way to increase TLB size of a machine/computer by upgrading or updating the OS.*

e) [**2+2 Marks**] What should be the structure of TLB in global frame allocation and local frame allocation? Suggest some efficient approach to organize TLB in non-uniform Memory Access (NUMA) multiprocessor.

*Ans: Irrespective of the frame allocation policy (global or local), TLB structure is same. Every TLB entry has 4 fields such as <ProcessID>, <Virtual Page Num>, <Frame Num> and <ValidBit>.*

*In NUMA multiprocessor, each processor or core keeps their own private TLB and there may be common shred last level TLB. So in a Quad core i5/i7 processor there are 4 private TLBs and one shared TLB on chip.*

f) [**6 Marks**] Suppose you have modified the demand paging fetch policy to a new one, where you pre-fetch the next logical page when access to the current logical page. Calculate number of page fault occurs in executing the following program, assuming page size=**400byte**, number of frame=**10**, page transfer time **25ns**, statement inside the loop takes **1ns** to execute, and size of integer is 4byte.

```
void main () {
     int i, A[100000], B[100000], C[100000];
     for(i=0;i<100000;i++)
          A[i]=B[i]+C[i];
}
```

*Ans: As discussed in the class, all the data access (access for A, B and C) for the above code are stream access, data access are not repeated and data is access one after another in a sequence.*

*Number of Frames = 10, Page size = 400byte, each page can store 100 integers.*

*If we compile the above code, we get executable. Code size will be around 700byte. So it will require at most 2 pages.*
*Data size will be = 1000 pages for A, 1000 pages for B and 1000 pages for C;*

*To execute the statement A[0]=B[0]+C[0], we require 3 memory access.* ***We will three page faults*** *and it will load the A[0-99], B[0-99] and C[0-99] to memory.*

*Suppose p, q and r be the virtual page number for A[0-99], B[0-99] and C[0-99]. As A, B, C are continuous memory location and at the time of accessing A[0], B[0] and C[0], it will start pre-fetch the next pages p+1, q+1 and r+1. To pre-fertch all three pages require 25*3=75ns. As if write to page A, so we need to swap out to HDD and to swap out a page of A require another 25ns. So in total it requires 100ns.*

*Accessing A[0], B[0] and C[0] will trigger pre-fetch to A[100-199], B[100-199] and C[100-199] respectively and 100th data of A, B and C will be in memory after 100ns. Similarly accessing A[100], B[100] and C[100] will trigger pre-fetch to A[200-299], B[200-299] and C[200-299] respectively and this process continue so on. As both pre-fetching swap in and out time is 100ns and it is equal to execution time for i=0 to 100 of the loop statement. So there will be any page fault further.*

*Irrespective any page replacement algorithm, it require not more than 2 pages for B, 2 pages for C and 3 pages for A.*

*So total number of page fault will be 3 (data) +2 (for code) =**5**.*

g) [**5 Marks**] Given the average ~~page~~ size of a process is ***p***, the page size *s* and the size of a page table entry is ***e byte***, what page size minimizes wasted space due to internal fragmentation and page table.
*Ans: Since the average number of pages required per process will be **p/s** and the amount of space required by the page table will be **(p/s).e.** The amount of space lost due to internal fragmentation is **s/2**. So total space wastage is*

*Waste = (p/s).e + s/2*

*To find the value of s that yields the minimal values, take first derivative with respect to s and set the resulting equation to zero.*

$$0 = - p.e/s^2 + 1/2 \quad === > s = \mathrm{sqrt}(2 p e).$$

**3. [Storage, File System and I/O device driver : 6 Lectures (=31%)] 19 Marks [2+3+4+5+5]**

a) [**2 Marks**] What are the differences between logical partitioning and physical partitioning of disk?

*Ans: With physical partitions, a disk is subdivided in to partitions, and somehow the disk driver knows about the subdivision scheme. Earlier days a disk can have at most 4 physical partitions and it was limited by disk MBR. MBR-based partition table schemes insert the partitioning information for (usually) four "primary" partitions in the master boot record (MBR) but now a day HDD uses UUID based partition table.*

*On other hand the logical partition is extended to either (a) partition a physical partition into many logical partitions or (b) combined many physical disks to make logical volume or partition. We can assume virtual FS as logical volume which combine many disks or storages.*

b) [**3 Marks=1+2**] As compared to non-RAID system, which RAID level offer best performance in term of bandwidth and capacity? Explain how that RAID level provides the best performance.

*Ans: RAID 0*

*Multiple disk drives may provide better throughput and performance via interleaving /stripping. Striping distributes contents of files roughly equally among all disks in the set, which makes concurrent read or write operations on the multiple disks almost inevitable. Concurrent operations make the throughput of most read and write operations equal to the throughput of one disk multiplied by the number of disks.*

*Increased throughput is the big benefit of RAID 0 versus spanned volume. Capacity of a RAID 0 volume is sum of the capacities of the disks in the set which is same as with a spanned volume.*

***Other RAID uses Mirroring or ECC or Parity, so capacity reduces.** But RAID 0 offers the best performance and capacity, but with no reliability and error check.*

c) [**4 Marks**] Suppose a file F of size 16KB is **shared by 100 processes**, each process read the entire F 10 times sequentially, disk uses **linked allocation policy** to store both Inode and data of the file with block size 512byte. Calculate the number of disk block access related to access the file F in executing all the 100 processes. (Write all the assumptions if you are assuming at the time of calculation)

*Ans: As disk uses Linked allocation for storing blocks, no extra blocks for FAT. It needs to access Inode table from the disk. And there are ceil(16KB/(512-4)B)=33 spanned block need to access to load the whole file F sequentially. So total number of block read/access to read the whole file is 33+1 (access the Inode table). Once the file loaded to memory by any one process, we do not need to access hard disk block. As the file F is shared by other 99 processes. So total number of disk block access will be 34.*

*(Assumption is 4byte is required to store the pointer to the next block, so amount of data stored in a block is 512-4=508 byte).*

*This is same as libc.so shared library of linux system, most of the process access (shared read-only) this libc.so. Once a process read this file from Disk, other processes simply use that read data.*

d) [**5 Marks= 2+1+1+1**] Write general structure of

i. I/O hardware and Timer.

*Ans: Every I/O hardware usually have registers where device driver places commands, addresses, and data to write, or read data from registers after command execution. So every I/O device has (a) data-in register, (b) data-out register, (c) status register and (d) control register.*

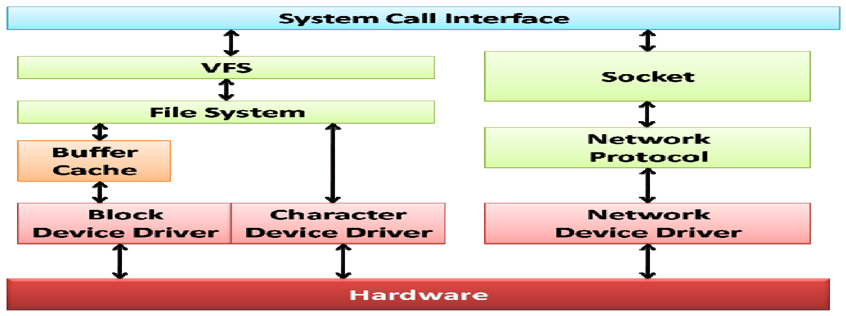
*Times have data-in register to store (alarm value), status register and control register. Timers usually have 4-6 alarms, it automatically do count time and interrupt to CPU.*

ii. A Linux kernel module.

*Ans: A linux kernel module have two required administrative modules/functions and some optional appropriate '**module service' functions**. Administrative modules are **init()** and **exit()/cleanup()**. Also module uses 'printk()' instead of 'printf()', includes the <linux/module.h> header-file and specifies a legal software license ("GPL").*

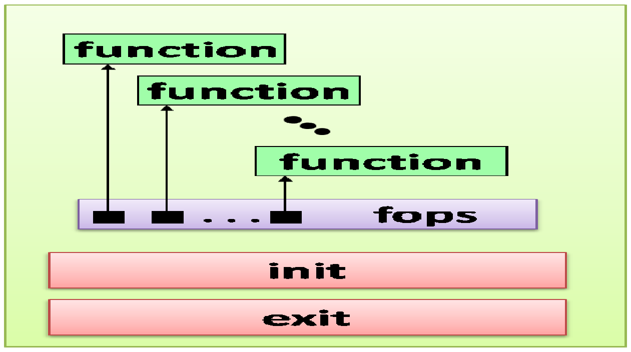
iii. Linux/Unix device driver architecture

*Ans: infer from the bellow given picture*



iv. A character device driver

*Ans: infer from the bellow given picture. It is kernel module with many service functions, which includes the standard I/O functions read(), write(), seek(), open() and close(). All the service functions are packages with fops array which points to these functions.*



e) [**Marks=2+1.5+1.5**] In Linux OS, file system uses Inode data structure to store the attribute of file.

i. What is difference between Inode Table and File Allocation Table?

Ans: *Inode Table or Master File Table (MFT) holds information about all the files and directories on the disk drive including file type, size, access rights, pointers to associated data blocks and other attributes.*

*When we access a file, it first access the IT (or MFT) to get the pointer to first block of file or indexed block of file which contain pointers to all the block of the file.*

*File Allocation Table is an array to store pointers to all the blocks of a file. In general, file number indexed in IT (MFT) points to FAT.*

*In earlier, FAT16, FAT32, VFAT file system, they used to call MFT as FAT, which store start and end block for all the files and directory.*

ii. Where Inode of a file get stored in the disk? Where the Inode Table get stored in the disk? Where the FAT get stored in the disk?

Ans:

*Inode get stored in area where the Inode table (or superblock) or Master File Table get stored.*

*IT (or MFT) is array of Inode (or FCB: file control block). MFT/IT occupy around 1% disks, this area of Disk is a special area called MFT area. Inode store the pointer to FAT or fast block of file.*

*FAT of file generally regarded as virtually first indexed block of data, it get stored in HDD and pointer to FAT get stored in Inode. FAT do not get stored inside the MFT (and MBR area).*

iii. How can you design a fault tolerant Inode table and FAT of a file system? Give an example of such file system.

*Ans: if we store multiple copies of Inode table (or MFT), then there will be redundant. Also for FAT of file can be stored in multiple places along with data.*

*NTFS is a very good example, where MFT get stored at multiple location.*