# Explanation for some questions

1. CPU-scheduling decisions may take place under the following four circumstances:
2. When a process switches from the waiting state to the ready state (for example, at completion of I/O)
3. When a process terminates
4. When a process switches from the running state to the waiting state (for example, as the result of an I/O request or an invocation of wait for the termination of one of the child processes)
5. When a process switches from the running state to the ready state (for example, when an interrupt occurs)

**WRONG**.

1. Two processes enter into critical zone by using semaphores ***mutex***, at first ***mutex*** = 1, when ***mutex*** = 0 means **\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

A）none process has entered the critical zone

B）one process has entered the critical zone

C）two process have entered the critical zone

D）one process has entered the critical zone, another one is waiting

1. Which page replacement algorithm can cause Balady’s Anomaly. （ ）

A）LRU. B）FIFO.

C）Working Set. D）Optimal

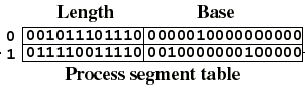


Figure 3.

1. Figure 3 is the segment table of a process. When given the logical address <0001, 001011110000>, its physical address is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

A. 001011101110001011110000 B. 011110011110001011110000

C. 0000010000000000001011110000 D. 0010000000100000001011110000

1. A multiprogramming system has one CPU and two IO devices – IO1 and IO2. The competition for CPU follows the preemptive priority scheme, which means the job with higher priority could interrupt the running job with lower priority and get the CPU. While, the competition for IO devices is non-preemptive (Why? ☺ ). Now there are three jobs – J1, J2 and J3 – stored in memory, and they arrive at same time. J1 has the highest priority and J3 has the lowest priority. Their ordered requests for CPU and IO devices are listed as follows:

**J1: IO2(30ms), CPU(10ms), IO1(30ms), CPU(10ms).**

**J2: IO1(20ms), CPU(20ms), IO2(40ms).**

**J3: CPU(30ms), IO1(20ms).**Please draw the **Gantt diagram** and answer the following questions:

1. Compute the turnaround time of those three jobs;
2. Compute the CPU utilization rate after all three jobs are finished;
3. Compute the utilization rate of IO1 after all three jobs are finished.
4. Consider the following segment table:

Segment Base Length

0 219 600

1 2300 14

2 90 100

3 1952 96

What are the physical addresses for the following logical addresses [(x,y): “x” is the segment, and “y” corresponds to the relative address]?

a. 0,430

b. 1,10

c. 2,500

d. 3,112

1. An OS uses demand paging system in memory management. Assume the capacity of the main memory which a process could be allocated to is 300 byte, which is divided into 3 frames. The process will access the following Logical address byte series: 115, 228, 120, 88, 446, 102, 321, 432, 260, 167（Attention: 115B is only equivalent to 1 page）. Please Answer：

(a) Writing down the page-reference string.

(b) Analyze the page replacement situation and calculate the page fault frequency when LRU and FIFO algorithm is used

1. Assume that the main memory is organized using pure paging. The page table (uncompleted) is shown as Figure 2.

|  |  |
| --- | --- |
| Page table | |
| 0 | 4 |
| 1 |  |
| 2 |  |
| 3 | 3 |
| 4 | 1 |

Figure 2

For each of the following decimal logical addresses, the **physical addresses** according to them are shown in Figure 3.

|  |  |
| --- | --- |
| logical addresses | physical addresses |
| 0 | 2048 |
| 600 | 88 |
| 1024 | 0 |

Figure 3

(1) Compute the page size.

(2) Fulfill the page table.

(3) For each of the following decimal logical addresses, compute the **physical addresses** according to the page table.

(3-1) 1600

(3-2) 2500

1. Readers-Writers Problem is a quite popular synchronization problem in OS. One of its typical sub-problems is the First Readers-Writers Problem, in which readers have priority over writers. That is, unless a writer has permission to access the object, any reader requesting access to the object will get it. (**Note this may result in a writer waiting indefinitely to access the object, AKA starvation.**) Following is an unfinished pseudo code for this problem, please complete it. (5 scores)

**Begin**

readcountmutex, wmutex: semaphore; // 两个互斥信号量

readcount: Integer; // 面向Reader的计数器

readcountmutex = wmutex = 1;

rcount = 0;

**Cobegin**

**Process procedure Reader**

**begin**

**repeat**

// *do something*

P(**readcountmutex**);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

if (readcount ==1) then P(**wmutex**);

V(**readcountmutex**);

***perform read operations***;

P(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

readcount:= readcount - 1;

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) then V(\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

V(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

// *do something else*

**until** false;

**end**

**Process procedure Writer**

**begin**

**repeat**

P(**wmutex**);

***perform write operations***;

V(**wmutex**);

**until** false

**end**

**CoEnd**

**End**

1. Figure 1 illustrates a bridge, and the arrows show the directions of the corresponding cars. Only one car is allowed on the bridge at any time, but several cars are allowed to pass the bridge one by one if they are for the same direction. You are required to fill the blanks in following code which is used to cope with this synchronization problem using P and V operations.

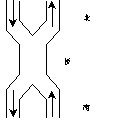


Figure 1: A bridge (Direction for top is north)

Var integer mutex, availn, avails;

// availn (for North) and avails (for South) are semaphores to synchronize the passing cars

availn = m; // **when it’s m, it means only the cars from this direction could pass the bridge**

avails =0; // **when it’s 0, it means only the cars from this direction are forbidden to pass the bridge**

mutex = 1; // mutual exclusion for accessing the bridge

**COBEGIN**

**Car for South**:

BEGIN

P(\_\_\_\_\_\_\_\_\_\_\_\_);

P(\_\_\_\_\_\_\_\_\_\_\_\_);

Cross the bridge;

V(\_\_\_\_\_\_\_\_\_\_\_\_);

V(\_\_\_\_\_\_\_\_\_\_\_\_);

END;

**Car for North**:

BEGIN

P(\_\_\_\_\_\_\_\_\_\_);

P(\_\_\_\_\_\_\_\_\_\_);

Cross the bridge;

V(\_\_\_\_\_\_\_\_\_\_);

V(\_\_\_\_\_\_\_\_\_\_);

END;

**COEND**;

1. A hard disk has 40G, Its each block size is 1K，and each table entry of FAT needs 20 bits，then Its FAT (File Allocation Table) need （ ）memory space

A）100M B）120M C）140M D）160M

1. A disk has 10 cylinders, each cylinder has 20 tracks, each track was divided into 16 sectors. Bit Map was used managing the disk ，if the word length is 16-bit ，then the Bitmap need （ ）words

A）200 B）128 C）256 D）100

1. Here is a file system, which adopts multi-level index structure to support the search some records in a file. The block size is 512 bytes, and 3 bytes are used to represent the block number. If the cost to the logic block number in a physical block is not considered, please figure out the largest size of a file when using 2-level and 3-level index structures.
2. Given a hard disk whose size is 500GB (We use 1GB=230B here), and the block size is 8KB. 32 bytes are used to locate one block. Please answer following questions: (10 points)

a. If bit map (bit vector) is used to manage the hard disk which should be also stored in hard disk, what’s the number of blocks used to store that bit map?

b. Now FAT (File Allocation Table) is used to the blocks of the hard disk, and the pointer size to locate a block is 32 bytes. If the blocks for the FAT itself are connected following linked list idea as Fig. 1 (each pointer uses 32 bytes), how many blocks should be used to store the FAT?

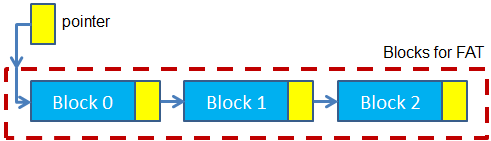


Fig. Linked list idea to organize the data blocks for a file

c. Now we use 2-level index model to organize the data of a file as Fig. 2, and each index table (inner or outer) can only occupy one block at most. What’s the largest size of a file following this kind of organization?

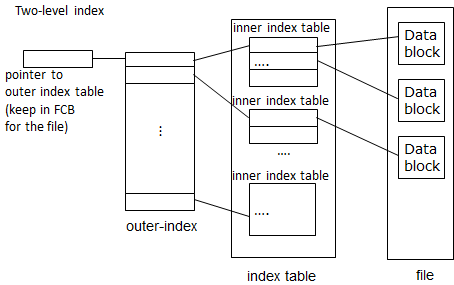


Fig. 2 level indexed structure to organized the data blocks for a file

d. When using i-node (shown in Fig. 3) to organize the data or a file, we use all the bytes of one block to store an i-node. In that i-node, 5KB is used to represent the attributes of the file, and the rest bytes are used for indirect pointers. If the rest bytes are evenly assigned to those 3 indirect pointers (namely, for instance, if we have 12\*32 bytes, single indirect type uses 4\*32 bytes, double indirect type uses 4\*32 bytes, and triple indirect types uses 4\*32 bytes), What’s the largest size of a file following this kind of organization?

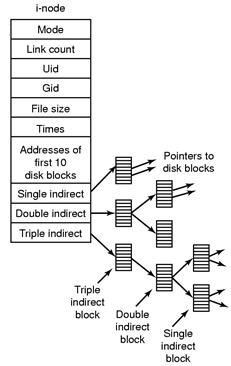


Fig. Demonstration of an i-node