CSO-231 Operating systems
End-sem Examination

Enroll
Page No:
Date: / /

Name: Shruti T. Avhad Roll No.:- 20074030
Branch: CSE(IDD) Email: shruti.tavhad.cse20@itbhu.ac.in

Ans 1.) Prepaging is used to reduce a large number of page faults that occur at the process startup. It is basically used to overcome the drawback of demand paging.

Benefits of Prepaging:

* Overcomes the drawback of demand paging.
* Saves time for consecutive address referencing done by a process thus makes easy for OS to guess and load appropriate pages, while the guesses are right for many pages, fewer faults will occur.

Demerits of Prepaging:

* It can be costly if the cost of using prepaging is less than the cost of servicing the corresponding page fault.
* Considerable memory and time loss may occur as a result of unused large numbers of pages bought back in memory.

So, if there are 's' pages that are pre-paged and a fraction 'α' ($0 \leq \alpha \leq 1$) are actually used. Then

$C(\alpha s)$ = cost of saved page fault

$C'((1-\alpha)s)$ = cost of prepaging unnecessary page.

If $C(\alpha s) > C'((1-\alpha)s)$

then prepaging is beneficial

Else it is wasteful.

Enroll
Page No:
Date: / /

So, $s\alpha > s(1-\alpha)$

$$\boxed{\alpha > \frac{1}{2}}$$

i.e when α is close to 1 ($\alpha > 1/2$), prepaging will be beneficial.

Also, prepaging will be useful in prepagin a file, because it is predictable as files are often accessed sequentially.

And, prepaging an executable program as it is difficult because of uncertainity regarding exactly what pages should be bought in the memory.

Enroll
Page No:
Date: / /

Ans2.)

Following are various factors involved in deciding ideal page size -

i) Size of Page Table:
Page size is inversely proportional to the no. of process and size of the page table. since, each ~~pro~~ process must have its own copy of page table.
∴ a larger page size is desirable

ii) Memory Utilization:
During memory allocation to a process, a part of the final page remains unused. Thus, a large page size would result in a large amount of memory being utilised leading to internal fragmentation.
∴ A small page size is desirable.

iii) Time Required to Read/Write a page:
when storage device is an HDD, I/O time is composed of seek, latency and transfer times. Though transfer time increases with increase in page size, the seek and latency times overweigh
∴ A larger page size is desirable to minimize overall I/O time.

iv) I/O time & Memory Allocation:
If we use a larger page size, we need to allocate and transfer not only what is needed but also anything else that is in the page.
∴ a smaller page size results in less I/O & less total allocated memory.

v) Page Faults:
Because the page faults generate large amount of overhead, Hence a larger page size is desirable since it leads to lower no. of pages and lower no. of faults.

vi) Other factors like relationship between page size and sector size on the paging device.

Enroll
Page No:
Date: / /

Ans 3.)

The Dining Philosopher's problem states that n philosophers are seated around a circular table with one chopstick between each pair of philosopher's. A philosopher can only eat if he can pick up the two adjacent chopsticks. One chopstick may be picked up by any one of its adjacent philosophers but not both.

Hence, if all the philosophers first pick their left chopstick and then the right one, a situation may arise when all philosophers have picked their left chopstick at the same time, which leads to a condition of deadlock, since none of the philosophers can eat.

In the odd-even scheme for solving dining philosophers problem, a philosopher at even position picks the right chopstick and then the left chopstick, while a philosopher at odd position picks the left chopstick and then the right chopstick. The best situation with this scheme is that it prevents the condition of deadlock described above, since the 'circular wait' condition for deadlock to occur will not satisfy (When two people refuse to retreat and wait for each other to retreat, so that they can complete their task, is the circular wait.)

However, there are potential bottlenecks/ drawbacks with odd-even scheme solution:-

1. The first bottleneck is minor. The solution is prone to process saturation. for example, if philosopher A is waiting for

a chopstick. Eventually, the owner of the chopstick, philosopher B, adjacent to A, eats and puts the chopstick down, but there is no guarantee that philosopher A will get it if philosopher B wants to eat again before philosopher A picks the chopstick.

2. The more major problem is that philosophers are not equally weighted here. Suppose all the philosophers want to eat at the same time. Philosophers 0 and 1 will have to fight for their first chopstick, as will philosophers 2 and 3. However, philosopher 4 will always get his first chopstick. This gives philosopher 4 an advantage over others, meaning he eats more.
Thus, this solution of odd-even scheme cannot be used if we want to given equal opportunity to all philosophers.

Enroll
Page No:
Date: / /

Ans 4)

The sleeping barber problem has the following conditions and assumptions:

- 1 Barber
- 1 Barber chair
- A waiting room with n chairs for waiting customers.

If there are no customers, the barber will sleep in the barber chair. A customer must wake the barber if he is asleep. If a customer arrives while the barber is working, the customer sits and waits in an empty chair if it's available and leaves the shop if all chairs are preoccupied. When the barber finishes a haircut, he inspects the waiting room to see if there are any waiting customers and falls asleep if there are none.

Solution

To solve the sleeping Barber problem, we will be using three semaphores.

i) The first one is for Barber: This semaphore is used to tell whether the barber is idle or working. The barber acquires it before checking for customers and releases it when they begin either to sleep or cut hair.

ii) The second one is for customer: The customer aquires this semaphore before entering the shop and release it once they are sitting in waiting room or barber chair or when the leave the shop because no seats are available.

Enroll
Page No:
Date: / /

iii) The third one is used to provide mutual exclusion: which is required for the process to execute start

- Barber Process blocks customer semaphore (which is initially 0.)
- Barber goes to sleep ~~with~~ until a customer comes up.
- Customer arrives → Customer Process acquires customer semaphore.
- Customer Semaphore → Wakes up the Barber → Barber process acquires barber semaphore.
- If meanwhile a second customer arrives → checks if any free seats are available in waiting room → if not then leaves the shop → releases waiting seat semaphore.

Conditions of failure

i) A race condition might arise wherein the barber sleeps while the customer waits for the barber to get haircut, because actions like - checking the waiting room, entering the shop, taking a waiting room chair - take a certain amount of time.

ii) Another problem is when two customers arrive at the same time and there is only one empty seat in the waiting room and both try to sit. Only one person gets to sit in that case.

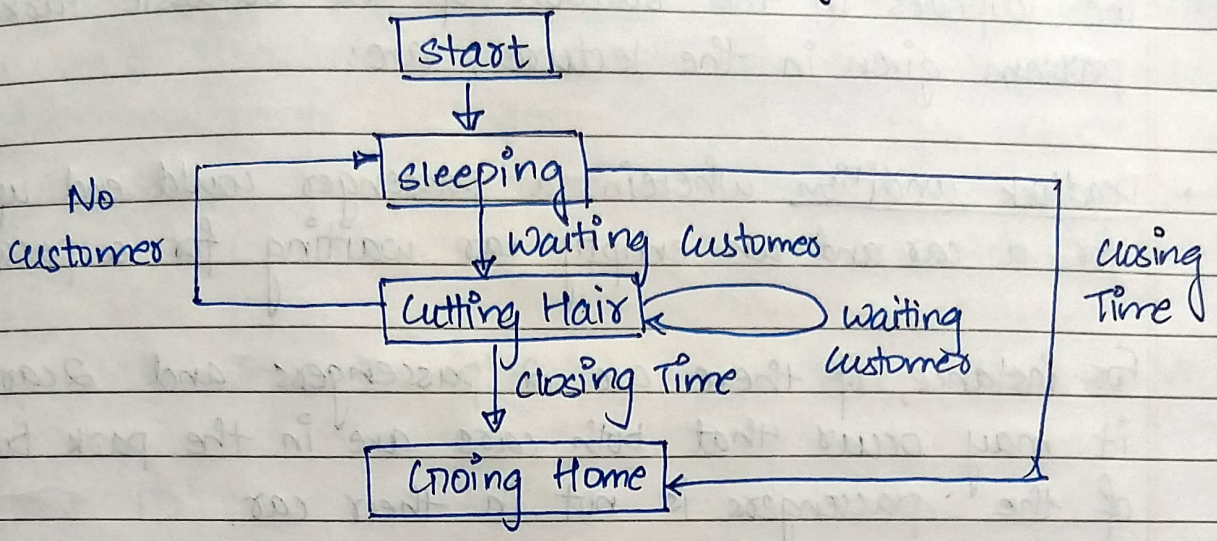
iii) The given solution may lead to starvation condition as particular customer in the waiting room might end up waiting indefinitely. This can be solved with a first-in

Enroll

Page No:

Date: / /

First-out (FIFO) queue for waiting room.



Enroll

Page No:

Date: / /

Ans5.)

The loopholes in the solution of the Jurassic Park problem given in the lecture, are:

- Deadlock condition wherein a passenger could end up waiting for a car and an empty car waiting for a passenger.

  For instance, if there are 2 passengers and 2 cars then it may occur that both cars are in the park but one of the passengers is not in their car.

- Starvation condition where one passenger may end up waiting for a car for a long time as other passengers may take the car for ride continuously leaving that one passenger waiting.

Enroll

Page No:

Date: / /

Ans 6.)

In UNIX / LINUX systems, an inode (also called index node) is a data structure for storing the file system metadata - with pointers to its data. When a file is created on the system, a file name and inode number is assigned to it.

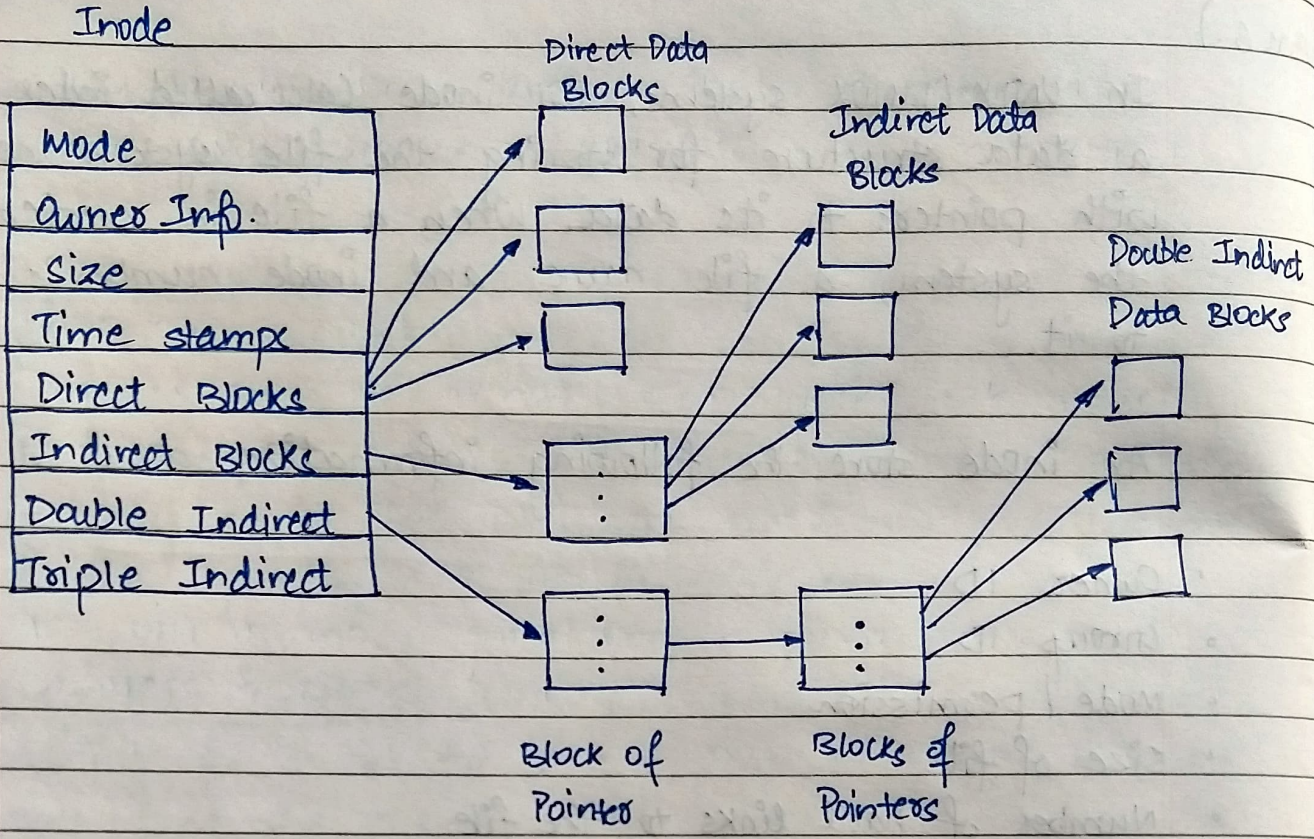
An inode store the following information about a file:

- Owner ID
- Group ID
- Mode / permission
- Size of file
- Number of hard links to the file
- Time last accessed
- Time last modified
- Time inode last modified
- File protection flags
- Pointers to the blocks storing file's contents.

Enroll

Page No:

Date: / /



In UNIX-based file systems, the files inode contains the first 15 pointers of the index block.

i) The first 12 of these pointers point to Direct Blocks. They contain addresses of blocks that contains data of the file. Usually, the data of small files is accessed by this Direct Blocks and thus, small files do not need a separate index block.

ii) The next 3 pointers point to Indirect Blocks

a) single Indirect Block

- This points to an index block that contains the addresses of blocks that contain data.

Enroll
Page No:
Date: / /

b) Double Indirect Block

- This points to a block which contains address of a block that contains the addresses of blocks that contains the addresses of blocks containing data.

Usually medium-sized files used single Indirect Block whereas Double And Triple indirect Blocks are used by large files.

Ans 7.)

Deadlock Ignorance is the most commonly used approach in the domain of deadlock handling. This strategy assumes that deadlocks never (or very rarely) occur and thus, can simply be ignored.

There is always a tradeoff between correctness and performance. Most commercial OS like window and Linux, which are best suited for single end user, mainly focus on performance. Usage of deadlock handling mechanism hampers the performance of the system, provided that deadlock happens very rarely and it is unnecessary to use deadlock handling mechanism all the time.

Hence, the most commercial OS do not adopt any solution for handling deadlock.

Enroll
Page No:
Date: / /

Ans 8.)

Following are the parameters that are taken into consideration when we evaluate various management schemes:

i) Hardware support:
Different kinds of memory management schemes require different kind of physical memory space arrangement.
- for single/multiple partition schemes, base register or base-limit register is used.
- for paging and segmentation, mapping tables to define address map are used.

ii) Fragmentation:
A multiprogrammed system will effectively utilise the high level of multiprogramming if the process are compactly packed into memory and there is little/minimum memory waste or fragmentation.

- systems with fixed-sized allocation units - internal fragmentation (single-partition scheme, paging).
- systems with variable-sized allocation units - external fragmentation (multiple-partition scheme, segmentation)

iii) Performance:
As the complexity of memory management scheme increases, the time required to map a logical address to a physical address and it also increases which affects the performance.

iv) Relocation:
External fragmentation can be minimised by compaction., i.e, shifting a program in memory by relocating their logical addresses dynamically at execution time.

v) Swapping:
Swapping involves copying processes from main memory to a backing store and later copying them back to main memory. This allows more and more processes to be run, than can be fit into memory at one time.

vi) Protection:
Data protection is provided by different permission modes (execute-only, read-only, red-write) when code or data is shared and also provides a simple run-time checks for programming errors.

vii) Sharing:
It is a means of running many processes with a limited amount of memory by sharing a code and data among different users, which further increases the level of multiprogramming.

Enroll
Page No:
Date: / /

Ans 9.)

The algorithm best suited for accessing when density of data is relatively high on a secondary disk is the direct access (or relative access), which is based on a disk model of a file containg data.

The direct access method is based on a disk mode of file, since disks allow access to any file block. Here, a file is made up of a fixed-length logical records that allow programs to read and write records rapidly in no particular sequence.

Direct access files are of great use for immediate access to large amounts of information. since, for direct access, the file is viewed as a numbered sequence of blocks or records, therefore, there are no restrictions on the order of reading or writing any block/record.

Ans 10.)

Various factors that are taken into consideration are:-

1. Monetary cost of additional disk-storage requirement.
2. Needs for performance in terms of number of input-output operations.
3. Measuring the performance in case of a disk failure.
4. Measuring the performance when data is rebuilt.

Enroll
Page No:
Date: / /

Following are the cases where certain specific RAID level has higher preference over other:

1.) high-performance applications with minimal data safety and protection - RAID level 0
2.) Rebuilding the data - RAID level 1.
3.) Good data transfer rates for large transfers and few disks usage for small data transfers - RAID level 5
4.) Reliability, data safety and security - RAID level 6.
5.) Best write performance - RAID level 1.
6.) Low storage overhead and read-only applications - RAID level 5.
7.) High Input-Output requirements - RAID level 5.
8.) Moderate storage requirements for the data - RAID level 1.