Scan or click here for more resources

Last Minute Notes – Operating Systems

Real Os:

- **Hard Real-Time Systems:**

  These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of any accident. Virtual memory is rarely found in these systems.Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

  - Flight Control Systems
  - Missile Guidance Systems
  - Weapons Defense System
  - Medical System
  - Inkjet printer system
  - Railway signalling system
  - Air traffic control systems
  - Nuclear reactor control systems
  - Anti-missile system
  - Chemical plant control
  - Autopilot System in Plane

- **Soft Real-Time Systems:**

  These OSs are for applications where for time-constraint is less strict.Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

  - Personal computer
  - Audio and video systems
  - Set-top boxes
  - DVD Players
  - Weather Monitoring Systems
  - Electronic games
  - Multimedia system
  - Web browsing
  - Online transaction systems
  - Telephone switches
  - Virtual reality

Synchronization Tools:

A **Semaphore** is an integer variable that is accessed only through two atomic operations, wait () and signal (). An atomic operation is executed in a single CPU time slice without any pre-emption. Semaphores are of two types:

1. **Counting Semaphore** – A counting semaphore is an integer variable whose value can range over an unrestricted domain.
2. **Mutex** – A mutex provides mutual exclusion, either producer or consumer can have the key (mutex) and proceed with their work. As long as the buffer is filled by producer, the consumer needs to wait, and vice versa.

At any point of time, only one thread can work with the entire buffer. The concept can be generalized using semaphore.

3.

Misconception:

There is an ambiguity between binary semaphore and mutex. We might have come across that a mutex is binary semaphore. But they are not! The purpose of mutex and semaphore are different. May be, due to similarity in their implementation a mutex would be referred as binary semaphore.

4.

Deadlock:

A situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Deadlock can arise if following four conditions hold simultaneously (Necessary Conditions):

1. **Mutual Exclusion** – One or more than one resource are non-sharable (Only one process can use at a time).
2. **Hold and Wait** – A process is holding at least one resource and waiting for resources.
3. **No Preemption** – A resource cannot be taken from a process unless the process releases the resource.
4. **Circular Wait** – A set of processes are waiting for each other in circular form.

Methods for handling deadlock: There are three ways to handle deadlock

1. **Deadlock prevention or avoidance**: The idea is to not let the system into deadlock state.
2. **Deadlock detection and recovery** : Let deadlock occur, then do preemption to handle it once occurred.
3. **Ignore the problem all together** – : If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

Banker's Algorithm:

This algorithm handles multiple instances of the same resource.

Memory Management:

These techniques allow the memory to be shared among multiple processes.

- **Overlays** – The memory should contain only those instructions and data that are required at a given time.

- **Swapping** – In multiprogramming, the instructions that have used the time slice are swapped out from the memory.

Memory Management Techniques:

(a) Single Partition Allocation Schemes –

The memory is divided into two parts. One part is kept to be used by the OS and the other is kept to be used by the users.

(b) Multiple Partition Schemes –

1. **Fixed Partition** – The memory is divided into fixed size partitions.
2. **Variable Partition** – The memory is divided into variable sized partitions.

Variable partition allocation schemes:

1. **First Fit** – The arriving process is allotted the first hole of memory in which it fits completely.
2. **Best Fit** – The arriving process is allotted the hole of memory in which it fits the best by leaving the minimum memory empty.
3. **Worst Fit** – The arriving process is allotted the hole of memory in which it leaves the maximum gap.

Note:

- Best fit does not necessarily give the best results for memory allocation.
- The cause of external fragmentation is the condition in Fixed partitioning and Variable partitioning saying that entire process should be allocated in a contiguous memory location.Therefore **Paging** is used.

1. **Paging** –
   The physical memory is divided into equal sized frames. The main memory is divided into fixed size pages. The size of a physical memory frame is equal to the size of a virtual memory frame.
2. **Segmentation** –
   Segmentation is implemented to give users view of memory. The logical address space is a collection of segments. Segmentation can be implemented with or without the use of paging.

Page Fault:

A page fault is a type of interrupt, raised by the hardware when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Page Replacement Algorithms:

1. **First In First Out (FIFO)** –
   This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

For example, consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots. Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> 3 Page Faults. When 3 comes, it is already in memory so —> 0 Page Faults. Then 5 comes, it is not available in memory

so it replaces the oldest page slot i.e 1. —> 1 Page Fault. Finally, 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —> 1 Page Fault.

2.

Belady's anomaly:

Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference string 3 2 1 0 3 2 4 3 2 1 0 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

3.

4. **Optimal Page replacement –**

   In this algorithm, pages are replaced which are not used for the longest duration of time in the future.

Let us consider page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 and 4 page slots. Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults. 0 is already there so —> 0 Page fault. When 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—> 1 Page fault. 0 is already there so —> 0 Page fault. 4 will takes place of 1 —> 1 Page Fault. Now for the further page reference string —> 0 Page fault because they are already available in the memory.

5.

Optimal page replacement is perfect, but not possible in practice as an operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

6.

7. **Least Recently Used (LRU) –**

   In this algorithm, the page will be replaced which is least recently used.

Let say the page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 . Initially, we have 4-page slots empty. Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults. 0 is already their so —> 0 Page fault. When 3 came it will take the place of 7 because it is least recently used —> 1 Page fault. 0 is already in memory so —> 0 Page fault. 4 will takes place of 1 —> 1 Page Fault. Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

8.

9.

File System: A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities.

10.

File Directories: Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system.

11.

1. **SINGLE-LEVEL DIRECTORY**: In this a single directory is maintained for all the users
2. **TWO-LEVEL DIRECTORY**: Due to two levels there is a path name for every file to locate that file.
3. **TREE-STRUCTURED DIRECTORY** : Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability.

File Allocation Methods:

1. **Continuous Allocation**: A single continuous set of blocks is allocated to a file at the time of file creation.
2. **Linked Allocation(Non-contiguous allocation)**: Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain.
3. **Indexed Allocation** : It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file

Disk Scheduling:

Disk scheduling is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling.

1. **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write.
2. **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.
3. **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
4. **Disk Access Time:** Seek Time + Rotational Latency + Transfer Time
5. **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of the all requests.

Disk Scheduling Algorithms:

1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.
2. **SSTF:** In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in a queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first.
3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence also known as elevator algorithm.

4. **CSCAN:** In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.
5. **LOOK:** It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.
6. **CLOOK:** As LOOK is similar to SCAN algorithm, in a similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.