

IMP POINTS:

27 जून 2019 21:32

1. The main function of the command interpreter is = to get and execute the next user-specified command.
2. Operating system, the resource management can be done via = both time and space division multiplexing
3. Dtrace: Facility which dynamically adds probes to a running system, both in user processes and in the kernel.
4. The OS X has hybrid kernel.



Processes

5. Uni-processing systems: The systems which allows only one process execution at a time.
6. In operating system, each process has its own address space and global variables + open files + pending alarms, signals and signal handlers.
7. FORK: system call creates to the new process in UNIX.
8. Ready state of a process: when process is scheduled to run after some execution.
9. A process stack does not contain: PID of child process.
10. Which system call returns the process identifier of a terminated child : wait.
11. The address of the next instruction to be executed by the current process is provided by the: Program counter
12. A Process Control Block(PCB) does contains: **Code + Stack + data**.
13. The Process Control Block is: Data Structure.
14. The entry of all the PCBs of the current processes is in: Process Table.
15. The degree of multiprogramming is: the number of processes in memory.



Process Scheduling Queues

16. When the process issues an I/O request : It is placed in an I/O queue.
17. What is a long-term scheduler= It selects which process has to be brought into the **ready queue**.
18. If all processes I/O bound, the ready queue will almost always be **EMPTY** and the Short term Scheduler will have a **LITTLE** to do.
19. What is a **medium-term scheduler** = It selects which process to remove from memory by **swapping**.
20. What is a short-term scheduler : It selects which process has to be executed next and allocates CPU.
21. The primary distinction between the short term scheduler and the long term scheduler is : The frequency of their execution.
22. The only state transition that is initiated by the user process itself is : **block**.
23. In a time-sharing operating system, when the time slot given to a process is completed, the process goes from the running state to the : Ready state.
24. In a multiprogramming environment : more than one process resides in the memory.
25. Suppose that a process is in "Blocked" state waiting for some I/O service. When the service is completed, it goes to the : Ready state.
26. The context of a process in the PCB of a process does not contain : context switch time.
27. Which of the following does not interrupt a running process ? Scheduler process.



Process Synchronization:

28. Which process can be affected by other processes executing in the system? cooperating process.
29. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called race condition.
30. If a process is executing in its critical section, then no other processes can be executing in their critical section. This condition is called mutual exclusion.
31. A semaphore is a shared integer variable that can not drop below zero.
32. Mutual exclusion can be provided by the both mutex locks and binary semaphores.
33. Priority inversion = When high priority task is indirectly preempted by medium priority task effectively inverting the relative priority of the two tasks.
34. Process synchronization can be done on both hardware and software level.
35. A **monitor** is a module that encapsulates = shared data structures + procedures that operate on shared data structure + synchronization between concurrent procedure invocation.
36. To enable a process to wait within the monitor, a condition variable must be declared as condition.
37. In UNIX, the return value for the fork system call is ZERO for the child process and NON ZERO INTEGER for the parent process.
38. The child process completes execution, but the parent keeps executing, then the child process is known as : Zombie.
39. Inter process communication : allows processes to communicate and synchronize their actions **without using the same address space**.
40. Message passing system allows processes to : communicate with one another without resorting to shared data.
41. An IPC facility provides at least two operations receive & send message.
42. Messages sent by a process : can be fixed or variable sized.
43. The link between two processes P and Q to send and receive messages is called : communication link.
44. In indirect communication between processes P and Q : there is a mailbox to help communication between P and Q.
45. In the non-blocking send : the sending process sends the message and resumes operation.
46. In the Zero capacity queue : the sender blocks until the receiver receives the message.
47. The Zero Capacity queue : is referred to as a message system with no buffering.
48. Bounded capacity and Unbounded capacity queues are referred to as : **Automatic buffering**.

49. **remote method invocation** : allows a thread to invoke a method on a remote object.
 50. The initial program that is run when the computer is powered up is called : **bootstrap program**.
 51. How does the software trigger an interrupt ? Executing a special operation called **system call**.
 52. What is a **trap/exception** ? software generated interrupt caused by an error.
 53. **An interrupt vector** = is an address that is indexed to an interrupt handler.
 54. In a **memory mapped input/output** : the CPU writes one data byte to the data register and sets a bit in control register to show that a byte is available.
 55. In a **programmed input/output (PIO)** : the CPU uses **polling** to watch the control bit constantly, **looping** to see if device is ready.
 56. In an **interrupt driven input/output** : the CPU receives an interrupt when the device is ready for the next byte.
 57. How does the **Hardware trigger an interrupt** ? Sending signals to CPU through system bus.
 58. Operation is performed by an **interrupt handler** ? Saving the current state of the system + Loading the interrupt handling code and executing it + Once done handling, bringing back the system to the original state it was before the interrupt occurred.
- ✓ **SCHEDULING:**
59. Which module gives control of the CPU to the process selected by the short-term scheduler? **Dispatcher**.
 60. The interval from the time of submission of a process to the time of completion is termed as **turnaround time**.
 61. In priority scheduling algorithm, when a process arrives at the ready queue, its priority is compared with the priority of currently running process.
 62. **Time quantum** is defined in **round robin** scheduling algorithm.
 63. Process are classified into different groups in **multilevel queue scheduling algorithm**.
 64. **User level threads** are managed by thread library and the kernel is unaware of them.
 65. The two steps of a process execution are : **CPU & I/O Burst**.
 66. **non – preemptive scheduling occurs** : When a process goes from the running state to the waiting state.
 67. The switching of the CPU from one process or thread to another is called : process switch + task switch + context switch.
 68. **Dispatch latency** is : the time to stop one process and start running another one.
 69. Waiting time is : the total time spent in the ready queue.
 70. Round robin scheduling falls under the category of : **Preemptive scheduling**.
 71. With round robin scheduling algorithm in a time shared system = using very large time slices converts it into **First come First served scheduling** algorithm.
 72. The portion of the process scheduler in an operating system that dispatches processes is concerned with : assigning ready processes to CPU.
 73. Which of the following algorithms tends to minimize the process flow time ? **Shortest Job First**.
 74. Under multiprogramming, turnaround time for short jobs is usually _____ and that for long jobs is slightly _____ = Shortened; Lengthened.
 75. Which of the following statements are true ? (GATE 2010) = ALL
 - I. Shortest remaining time first scheduling may cause starvation
 - II. Preemptive scheduling may cause starvation
 - III. Round robin is better than FCFS in terms of response time
 76. The most optimal scheduling algorithm is : **SJF – Shortest Job First**.
 77. The real difficulty with **SJF** in short term scheduling is : knowing the length of the next CPU request.
 78. The FCFS algorithm is particularly troublesome for **multiprogramming systems**.
 79. Preemptive Shortest Job First scheduling is sometimes called : **SRTN scheduling – Shortest Remaining Time Next**.
 80. '**Aging**' is : increasing the priority of jobs to ensure termination in a finite time.
 81. Which of the following scheduling algorithms gives minimum average waiting time ? **SJF**.
 82. The segment of code in which the process may change common variables, update tables, write into files is known as : **critical section**.
 83. The following three conditions must be satisfied to solve the critical section problem : **Mutual Exclusion + Progress + Bounded Waiting**.
 84. **Mutual exclusion implies** that : if a process is executing in its critical section, then no other process must be executing in their critical sections.
 85. **Bounded waiting implies** that there exists a bound on the number of times a process is allowed to enter its critical section : after a process has made a request to enter its critical section and before the request is granted.
 86. A minimum of **2** variable(s) is/are required to be shared between processes to solve the critical section problem.
 87. In the **bakery algorithm** to solve the critical section problem : each process receives a number (may or may not be unique) and the one with the lowest number is served next
 88. An un-interruptible unit is known as : **atomic**
 89. The **TestAndSet** instruction is executed : atomically.
 90. The two atomic operations permissible on semaphores are : **wait and signal**.
 91. **Spinlocks** are : CPU cycles wasting locks over critical sections of programs + Locks that avoid time wastage in context switches + Locks that work better on multiprocessor systems.
 92. The main disadvantage of spinlocks is that : **they require busy waiting**.
 93. The wait operation of the semaphore basically works on the basic **block()** system call.
 94. The signal operation of the semaphore basically works on the basic **wakeup()** system call.
 95. The code that changes the value of the semaphore is : critical section code.
 96. What will happen if a non-recursive mutex is locked more than once ? Deadlock.
 97. A semaphore : can be accessed from multiple processes.
 98. The two kinds of semaphores are : binary & counting.
 99. A mutex : must be accessed from only one process.

- 100. Semaphores are mostly used to implement : IPC mechanisms.
- 101. Spinlocks are intended to provide only =Bounded Waiting

102.

0. OS INTRO + FUNCTIONS

20 जून 2019 19:47

☐ **I/O Devices:**

1. Each device controller has a local buffer.
2. CPU moves data from/to main memory to/from local buffers.
3. I/O is from the device to local buffer of controller.
4. Device controller informs CPU that it has finished its operation by causing an **interrupt**.

☐ **Common Functions of Interrupts:**

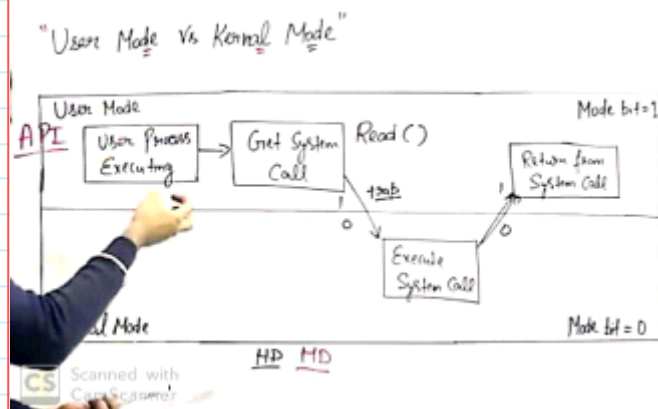
1. Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines.
2. Interrupt architecture must save the address of the interrupted instruction.
3. Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
4. A trap is a software-generated interrupt caused either by an error or a user request.
5. An operating system is interrupt driven

☐ **Interrupt Handling:**

1. The operating system preserves the state of the CPU by storing registers and the program counter.
2. Determines which type of interrupt has occurred:
 - a. **Polling.**
 - b. **Vectored interrupt system**
3. Separate segments of code determine what action should be taken for each type of interrupt.

1. SYSTEM CALLS

20 जून 2019 19:48



"System Call"

- File Related \Rightarrow `Open()`, `Read()`, `Write()`, `Close()`, `Create file` etc
 - Device Related \Rightarrow `Read`, `Write`, `Reposition`, `ioctl`, `fcntl` Printf
 - Information \Rightarrow `get Pid`, `Attributes`, `get System time and data`
 - Process Control \Rightarrow `Load`, `Execute`, `abort`, `Forst`, `Wait`, `Signal`, `Allocate` etc
 - Communication \Rightarrow `Pipe()`, `Create/delete connections`, `Shmget()`
- Scanned with CamScanner

2. Threads

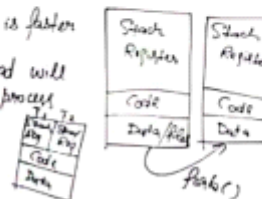
20 जून 2019 18:02

Process

- 1) System Calls involved in Process
- 2) OS treats different processes differently
- 3) Different process have different Copies of Data, files, Code
- 4) Context switching is slower
- 5) Blocking a process will not block another
- 6) Independent

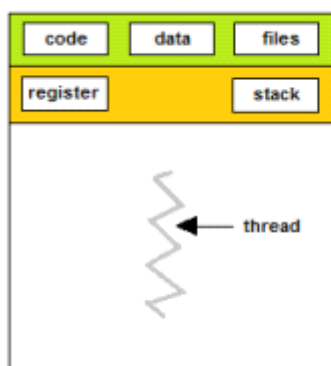
Threads

- 1) There is no system Call involved
- 2) All User level threads treated as single task for OS
- 3) Threads share same copy of Code and data
- 4) Context switching is faster
- 5) Blocking a thread will block entire process
- 6) Interdependent

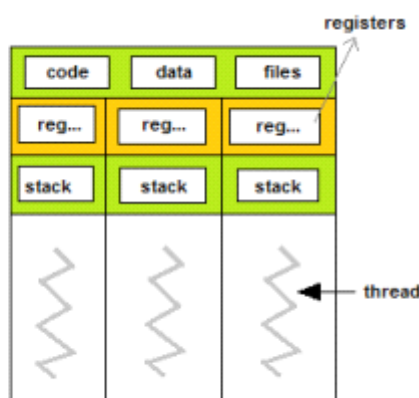


CS Scanned with

SUBSCRIBE



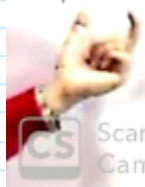
single-threaded process



multithreaded process

"User level Thread"

- 1) User level threads are managed by User level library
- 2) User level threads are typically fast
- 3) Context Switching is faster
- 4) If one user level threads perform blocking operation then entire process get blocked



CS Scanned with CamScanner

"Kernel Level Thread"

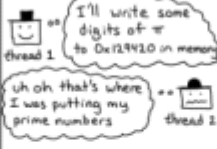
- 1) Kernel level threads are managed by OS System Calls
- 2) Kernel level threads are slower than User level
- 3) Context Switching is slower
- 4) If one kernel level thread blocked, No affect on others

$$\frac{\text{Process}}{\text{CT}} > \text{KLT} > \text{ULT}$$

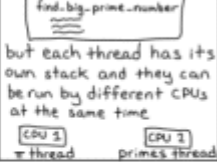
Threads let a process do many different things at the same time



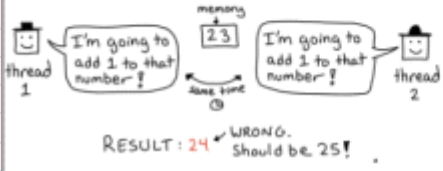
threads in the same process share memory



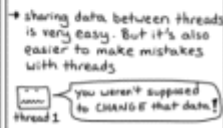
and they share code



sharing memory can cause problems (race conditions!)



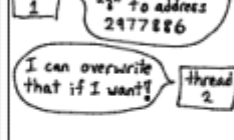
Why use threads instead of starting a new process?
→ a thread takes less time to create



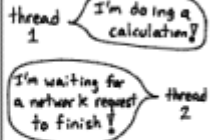
a process can have lots of threads

process id	thread id
1888	1888
1888	1892
1888	1893
1888	2007

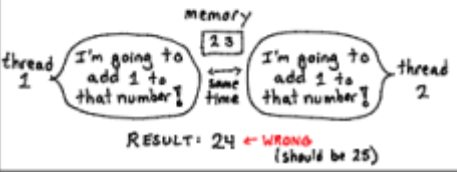
threads share memory



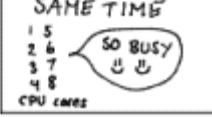
but they can run different code



Sharing memory can cause problems "race conditions"



if you have 8 CPU cores, you can run code for 8 threads at the SAME TIME



2.0. PROCESS STATES

20 जून 2019 19:43

2.1 SCHEDULING

20 जून 2019 18:09

Scheduling Algorithm

Pre-emptive Processor

1. SRTF: time burst(it is SJF with pre-emptive)
2. LRTF
3. ROUND ROBIN: ready que, time quantum, context switching saved in PCB
4. PRIORITY: priotiry criteria; in case of conflit use FCFS

Non Pre emptive

No need to find RT BCOZ RT = WT. Avg TAT; Avg WT

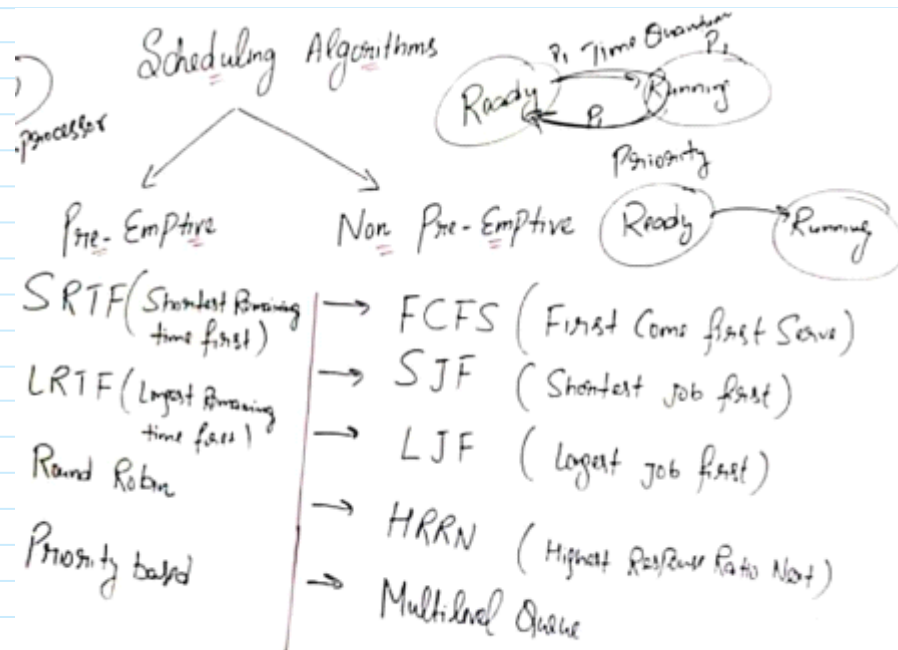
1. FCFS: arrival time
2. SJF: time burst; if arrival time is also same then process id
3. LJF
4. HRRN
5. Multilevel que: problem of starvation can happen.
6. Multilevel feedback que:

★ **Interrupt:** has the highest priority among all processes

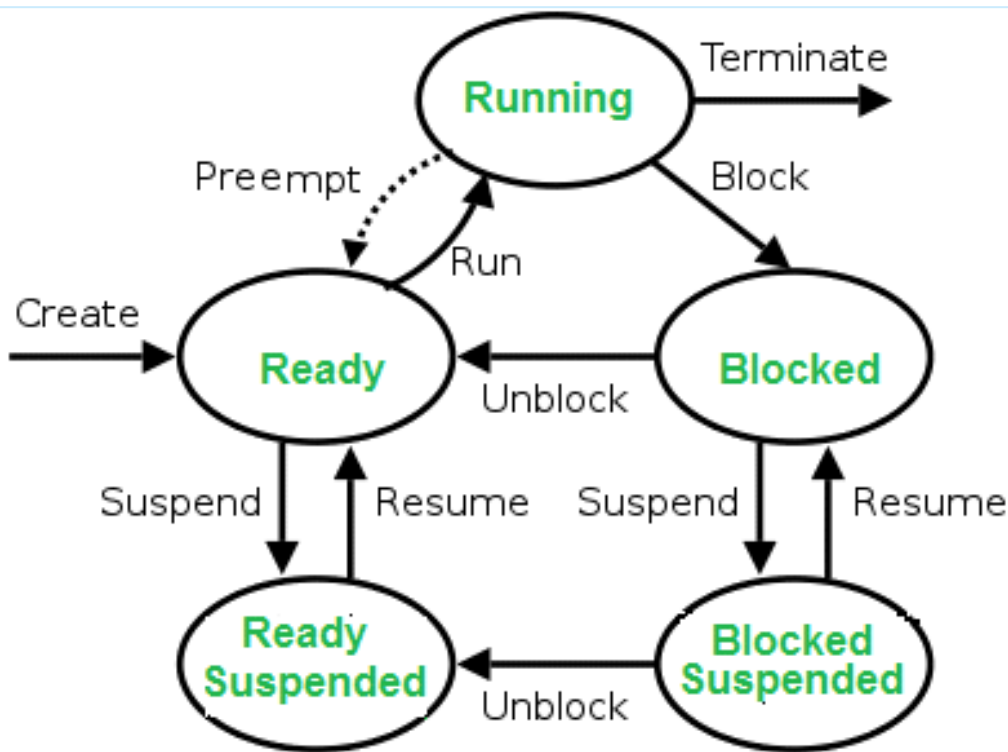
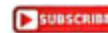
★ **Whenever we want to design Scheduling algorithms , we have to take care of 5 things:**

1. Maximum CPU Utilization
2. Minimum TAT(Turn Around Time)
3. Minimum WT(Waiting Time)
4. Minimum RT(Response Time)
5. Maximum Throughput

PARAM ENTER	PREEMPTIVE SCHEDULING	NON-PREEMPTIVE SCHEDULING
Basic	In this resources(CPU Cycle) are allocated to a process for a limited time.	Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process can not be interrupted untill it terminates itself or its time is up.
Starvation	If a process having high priority frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then later coming process with less CPU burst time may starve.
Overhead	It has overheads of scheduling the processes.	It does not have overheads.
Flexibility	flexible	rigid
Cost	cost associated	no cost associated



Scanned with
CamScanner



"CPU Scheduling"

→ Arrival time. The time at which process enter the Ready Queue or State

→ Burst time. Time required by a process to get execute on CPU

→ Completion time. The time at which process complete its execution

→ Turn Around time. $\{ \text{Completion time} - \text{Arrival time} \}$

→ Waiting time. $\{ \text{Turn Around time} - \text{Burst time} \}$

→ Response time. $\left(\text{The time at which a process get CPU first time} - (\text{Arrival time}) \right)$



Scanned with
CamScanner



2A. PROCESS SCHEDULING

21 जन 2019 21:02

- ☐ **Process scheduling:** Removal of the running process from the CPU + the selection of another process on the basis of a particular strategy.

★ **Multiplexing:** A multiprogramming technique used- by OS to allow more than one process to be loaded into the executable memory at a time and the loaded process to shares the CPU.

Process Scheduling Queues

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

- ☐ **Imp Process Scheduling queue of OS:**

- **Job queue** – This queue keeps all the processes in the system.
- **Ready queue** – This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** – The processes which are blocked due to unavailability of an I/O device constitute this queue.

- ☐ **Process States:**

1. **Running:**
2. **Not Running:** Not running process are kept in queue, waiting for their turn. Dispatcher is used to select a process and get it executed by CPU. Context Switches uses dispatcher.

- ☐ **Schedulers:** Special system software + to select jobs to be submitted into system (Job Queue) and which process will run. 3 types are:

1. **Long-Term Scheduler (LTS):** Job scheduler. Admits programs to system (From Job que to Ready que i.e Memory) for CPU Scheduling. **Primary Objective:** Balanced mix jobs - I/O bound + Processor Bounds. **Time- Sharing OS do not have LTS.**
When it is needed: When a process changes the state from new to ready.
2. **Short-Term Scheduler: Or Dispatchers: CPU Scheduler.** Objective: Increase system performance. It changes state from ready to running. It allocates CPU to process which is ready. **Dispatchers= make the decision of which process to execute next.** Short-term schedulers are faster than long-term schedulers.
3. **Medium-Term Scheduler:** Swapping. Removes process from memory. Reduces degree of multiprogramming. MTS handles swapped out process. Part of time sharing system.
Swapping: A running process may become suspended if it makes an I/O request. So, Suspended process moved into secondary storage

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	Process swapping scheduler.
2	Speed < STS	Speed = fastest	Speed = b/w LTS & STS
3	It controls the (DoM)	It provides lesser control over DoM	Reduces DoM Deg of Multiprogrm.
4	Absent or minimal in time sharing system	Minimal in time sharing system	Present in Time sharing systems.
5	Selects process from Job queue to put in Ready queue.	Select process to execute from ready queue. Dispatcher.	It can re-introduce the process into memory and execution can be continued.

- ☐ **Context Switch:**

★ A mechanism to **store and restore the state or context of a CPU in Process Control block PCB**, so that a process execution can be resumed from the same point at a later time. It enables multiple processes to share a single CPU. Its an essential part of a multitasking operating system features.

★ When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.

2B SJF

22 जून 2019 11:14

SJF

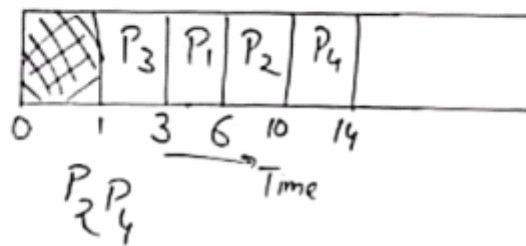
Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
$\rightarrow P_1$	1	3	6	5	2	2
$\checkmark P_2$	2	4	10	8	4	4
$\checkmark P_3$	1	2	3	2	0	0
$\rightarrow P_4$	4	4	14	10	6	6

Criteria: "Burst Time"
Mode: "Non-Preemptive"

$$TAT = CT - AT$$

$$WT = TAT - BT$$

Gantt Chart

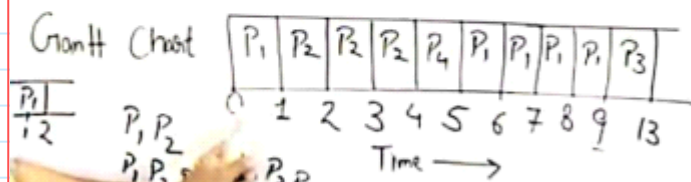


Scanned with
CamScanner

2C. SRTF

22 जून 2019 11:15

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P_1	0	5	9	9	4	0
P_2	1	2	4	3	0	0
P_3	2	4	13	11	7	7
P_4	4	1	5	1	0	0



Scanned with CamScanner

Criteria: Burst Time
Mode: Preemptive

$$TAT = CT - AT$$

$$WT = TAT - BT$$

$$RT = \int \text{CPU first time} - AT$$

$$\text{Avg TAT} = \frac{24}{4} = 6$$

$$\text{Avg WT} = \frac{11}{4} = 2.75$$

$$\text{Avg RT} = \frac{7}{4} = 1.75$$

2D. FCFS

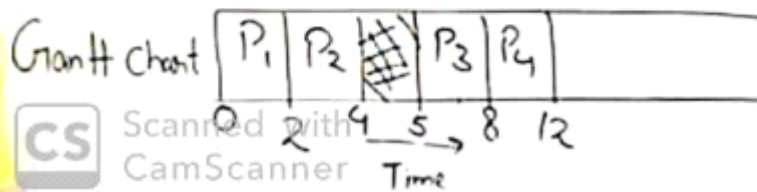
22 जून 2019 11:16

Process No	Arrival Time	^{Execution} Burst Time	Completion Time	TAT	WT	RT
P ₁	0	2	2	2	0	0
P ₂	1	2	4	3	1	1
P ₃	5	3	8	3	0	0
P ₄	6	4	12	6	2	2

Criteria: "Arrival Time"
Mode: "Non-Presumptive"

$$CT - AT = TAT$$

$$TAT - BT = WT$$



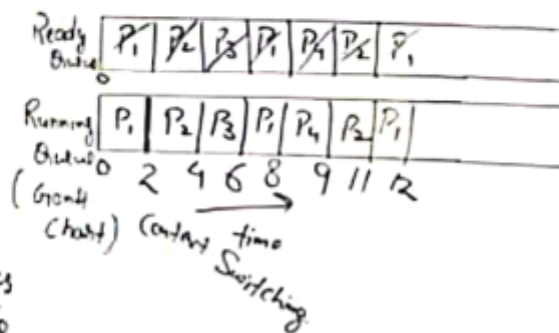
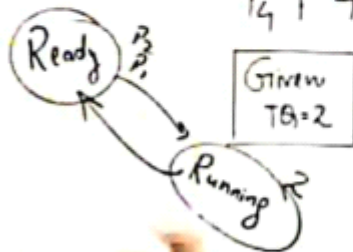
2E. RR

22 जून 2019

11:17

Round Robin
RR

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P ₁	0	5	12	12	7	
→ P ₂	1	4	11	10	6	
→ P ₃	2	2	6	4	2	
→ P ₄	4	2	9	5	4	



Criteria: "Time Quantum"
Mode: "Preemptive"

$$TAT = CT - AT$$

$$WT = TAT - BT$$

$$RT = \{ \text{CPU first time} - AT \}$$

Scanned with R/B
CamScanner

2F. PRIORITY SCHEDULING

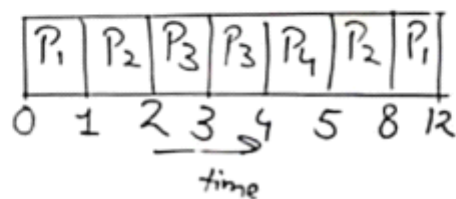
22 जून 2019 11:18

Priority	Process No	Arrival Time	Burst Time	Completion Time	TAT	WT
10	✓ P ₁	0	5	12	12	7
20	✓ P ₂	1	4	8	7	3
30	✓ P ₃	2	2	4	2	0
40	✓ P ₄	4	1	5	1	0

Higher the no.
higher the Priority



Scanned with
CamScanner



Priority Scheduling

Criteria: "Priority"

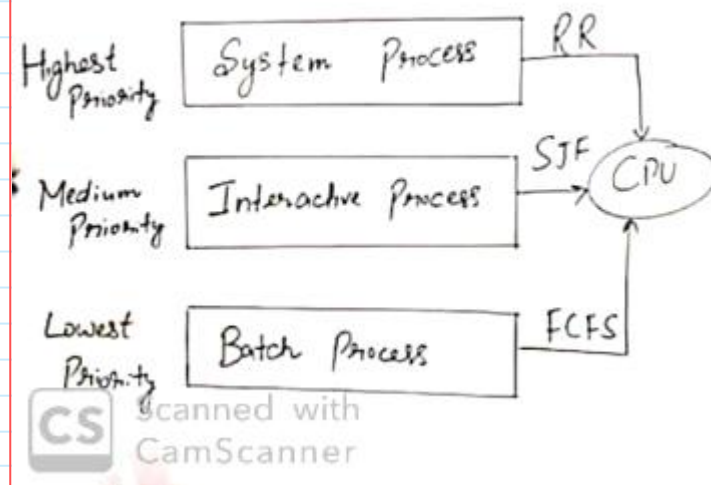
Mode: "Preemptive"

$$TAT = CT - AT$$

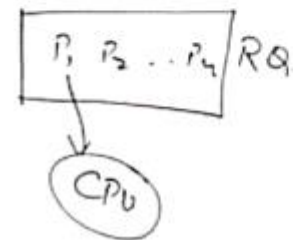
$$WT = TAT - BT$$

2G. MULTILEVEL QUE

22 जून 2019 11:18

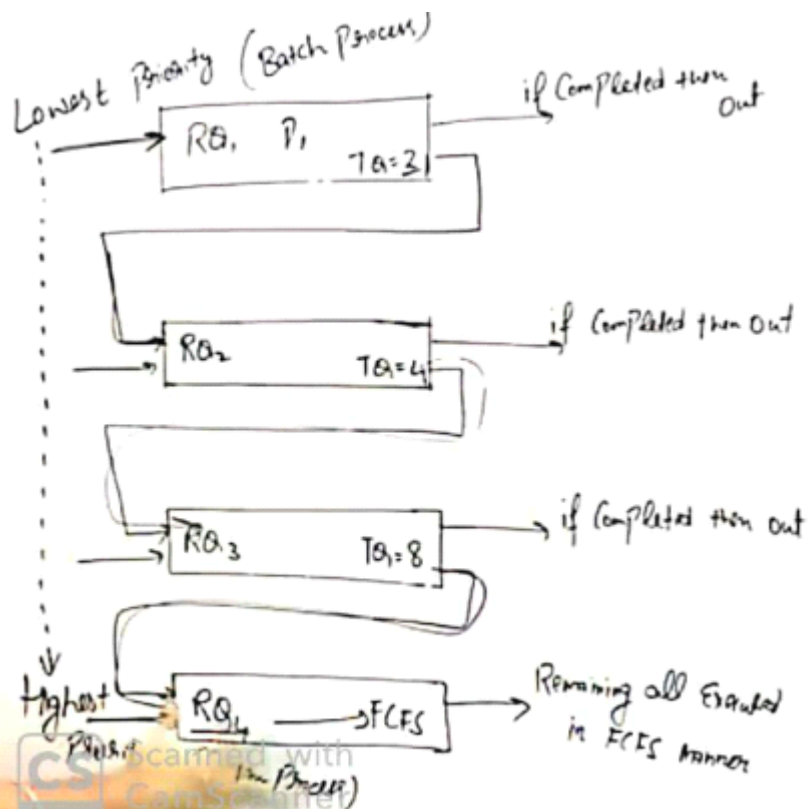


'Multilevel Queue Scheduling'



2H. MULTILEVEL FEEDBACK

22 जून 2019 11:19



'Multilevel Feedback Queue'

$$P_1 = \frac{19}{17}$$
$$13$$
$$5$$
$$0$$

SUBSCRIBE

3. Process Synchronization

20 जून 2019 19:36

Process Synchronization

Processes

Perative Processes

Shared Variable

Memory

Code

Resources

CPU

Process

int Shared = 5

UniProcessor

Race Condition

P₁

1. int X = Shared;

2. X++ ; X = 6

3. Sleep(1);

4. Shared = X;

Terminated

P₂

int Y = Shared;

Y--;

Sleep(1);

Shared = Y;

y = 4

Scanned with CamScanner

SUBSCRIBE

P₁ P₂ P₃

3 4 5

2 3 4

(9+1) = 10

P₁ = 2 = P₂ = P₃

min Req + 1

P₁ P₂ P₃

1 1 1

(1+1)

3 x 2 = 6

R = 4

Dead lock will not come

F₁ system is having 3 Processes each require 2 Units of resources 'R'

The Minimum no of Units of 'R' such that no deadlock will occur 4

a) 3 b) 5 c) 6 d) 4

Scanned with CamScanner

P₁ P₂ P₃

3 4 5

2 3 4

(9+1) = 10

P₁ = 2 = P₂ = P₃

min Req + 1

P₁ P₂ P₃

1 1 1

(1+1)

3 x 2 = 6

R = 4

Dead lock will not come

F₁ system is having 3 Processes each require 2 Units of resources 'R'

The Minimum no of Units of 'R' such that no deadlock will occur 4

a) 3 b) 5 c) 6 d) 4

Scanned with CamScanner

4. CRITICAL SECTION PROBLEM

20 जून 2019 19:36

Dispatcher

20 जून 2019 19:45

5. SEMAPHORES

20 जून 2019 18:08

☐ What is it?

★ Integer Variable which is used in **mutual exclusive manner** for various **concurrent cooperative process** to achieve synchronization.

☐ Why it is used?

★ **To Prevent the Race condition.** Race condition occurs when = we sync different process OR Run multiple process simultaneously, specially cooperative process (processes which have something in common).

☐ Type of Semaphores:

1. **Counting:** integer value - infinity to + infinity
2. **Binary:** integer value - 0 to 1

☐ Operations used in Entry/Exit:

1. **Entry code: P() ; down; wait:**
Entry: When s value reaches 0, it'll be last value to be Put into CS.
If $S = -4$ means already 4 Process are in Block List
2. **Exit Code: V(): UP ; Signal + Post Release**
 $S = S + 1$;
If $S < 0$, then select a process from Block list and wake up (means ready list) based on FIFO (First in First out). Means it can now again Try to go into CS - its in ready que.

P1,P1..Pn		
Entry	$S = S - 1$; If $S < 0$, then P will be in suspended list or Block	Run = Entry code: P(); down; wait
CS		
Exit	$S = S + 1$; If $S \leq 0$, then select a process from Block list and wake up (means ready list) based on FIFO (First in First out).	Run = Exit Code: V(): UP ; Signal + Post + Release
Terminate		

★ If $S = 0$; means there is no process in block list. Means there is no need to wakeup any processes.

★ If $S = -4$, means there are 4 process in block list.

★ If $S = 10$, means 10 processes can be brought into Critical Section (CS) = means 10 successful operation = means process is in CS.

☐ Advantages of Semaphores:

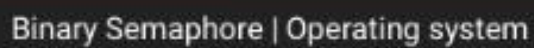
1. **Semaphores allow only one process into the critical section.** They follow the mutual exclusion principle strictly and are much more efficient than some other methods of synchronization.
2. There is **no resource wastage** because of busy waiting in semaphores as processor time is not wasted unnecessarily to check if a condition is fulfilled to allow a process to access the critical section.
3. Semaphores are **implemented in the machine independent code** of the microkernel. So they are **machine independent**

☐ Disadvantages of Semaphores:

1. Semaphores are complicated so the **wait and signal operations must be implemented in the correct order to prevent deadlocks.**
2. **Semaphores are impractical for last scale use as their use leads to loss of modularity.** This happens because the wait and signal operations prevent the creation of a structured layout for the system.
3. Semaphores **may lead to a priority inversion** where low priority processes may access the critical section first and high priority processes later.

24 जून 2019 15:39

24 जून 2019 15:39



6. DEADLOCK

20 जून 2019 19:37

6 methods to handle

Deadlock ignorance (Ostrich method)

Deadlock prevention → Affected resources, P_1 $\times M$

Deadlock Avoidance (Banker's Algo) → Safe state

Deadlock detection & Recovery → Unsafe state

(A process can get in increasing order)

- 1 Printer
- 2 Scanner
- 3 CPU
- 4 Register

Defect

- 1) Kill the processes
- 2) Release process or process preemption

Circular wait

```
graph LR
    P1((P1)) -- holds --> Printer[Printer]
    P1 -- requests --> CPU[CPU]
    P2((P2)) -- holds --> CPU
    P2 -- requests --> Printer
    Printer --> P1
    CPU --> P2
```

BANKERS ALGORITHM

24 जून 2019 15:31

Process	Allocation			Max Need			C _{urr} Available			Remaining Need			(Max Need - Allocation)
	E	F	G	E	F	G	E	F	G	E	F	G	
P₀	1	0	1	4	3	1	3	3	0	3	3	0	
P₁	1	1	2	2	1	4	4	3	1	1	0	2	
P₂	1	0	3	1	3	3	5	3	4	0	3	0	
P ₃	2	0	0	5	4	1	6	4	6	3	4	1	

$P_0 \rightarrow P_2 \rightarrow P_1 \rightarrow P_3$
 (Remaining Need \leq C_{urr}. Availability)
 for all Resource types

TE 2018 Question on Banker's Algorithm | Deadlock avoidance | Operating Sytem

7. MEMORY MANAGEMENT

20 जून 2019 19:38

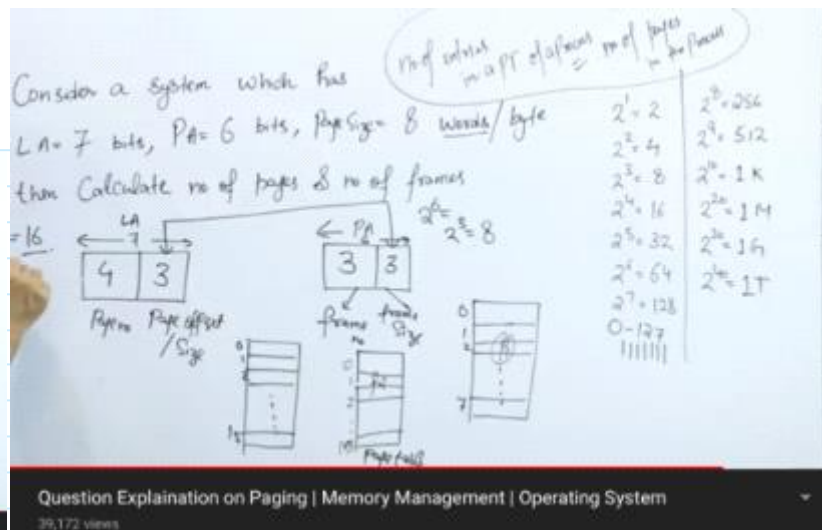
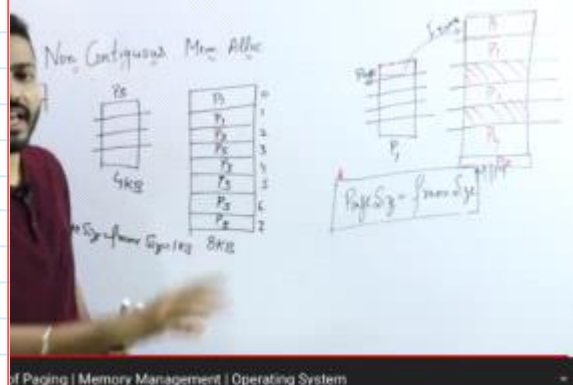
The whiteboard contains the following handwritten notes and diagrams:

- Top left: $16MB$ and $4MB$ with a small square box.
- Top center: $'K' = 70\%$ and $K^4 = 70\%$.
- Top right: $CPU\ utilization = 1 - K^4 \approx 94\%$.
- Middle left: $'K' = I/O\ operation\ (70\%)$.
- Middle center: $CPU = (1 - K) = 30\%$ and $CPU\ Utilization$.
- Middle right: P_1 RAM, $'PU'$, RAM , and SH with a diagram showing a stack of memory blocks and a pointer.
- Bottom left: RAM and $Process$ with a diagram showing a stack of memory blocks and a pointer.
- Bottom center: $No.\ of\ Processes = \frac{8MB}{4MB} = 2$ and K^2 .
- Bottom right: $K = 70\%$, $CPU\ utilization = 1 - K^2$, and $CPU = 1 - (70\%)^2 = 76\%$.

Memory Management and Degree of Multiprogramming | Operating System
26,156 views

9. PAGING

20 जून 2019 19:39



JULIA EVANS
@b0rk

page table (in 32 bit memory)

every process has its own memory space

$0x\ aeff3\ 000$

at that address it says "cat"

for me it says "dog"

process 1

process 2

each address maps to a 'real' address in physical RAM

process 1 $\rightarrow 0x28ea4000$

process 2 $\rightarrow 0x3942f000$

$0x\ aeff3000$ is invalid!

processes have a "page table" in RAM that stores all their mappings

$0x12345000 \rightarrow 0xae925$

$0x23f49000 \rightarrow 0x12345$

the mappings are usually 4kB blocks (4kB is the normal size of a "page")

every memory access uses the page table

I need to access $0x\ ae923\ 456$

CPU: the page table says the real address is $0x\ 9923456$

sort of

When you switch processes...

kernel: here, use this page table instead now

CPU: Okay thanks!

some pages don't map to a physical RAM address

process: I'm gonna access $0x\ 00040000$

CPU: EEP NO! BAD ADDRESS!

\equiv Segmentation fault \equiv

PAGE TABLE

24 जून 2019 15:42

Page table entry

FRAME No.	Valid(1)/Invalid(0)	Protection (RWX)	Reference (0/1)	Caching	Dirty Modify.
-----------	---------------------	------------------	-----------------	---------	---------------

← Mandatory field →

Optional fields →

Present/Absent

LRU

Enable/Disable

Read, write, execute

P, W
User, X = 1000
200

Format of Page Table Entry | Memory Management | Operating System

PAGE REPLACEMENT FIFO

24 जून 2019 15:35

Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0

Page Replacement alg

- 1) FIFO
- 2) Optimal Page Replacement
- 3) Least Recently Used

Page Replacement Introduction | FIFO Page Replacement algorithm with example | Operating System

OPTIMAL PAGE REPLACEMENT

24 जून 2019 15:36

Optimal Page Replacement (Replace the page which is not used in longest Dimension of time in future)

			2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		1	1	1	1	X	4	4	4	4	4	X	1	↓	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7
	*	*	*	*	Hit	*	Hit	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit

Ref. Strm: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Arrows indicating replacements: 7 → 0, 0 → 1, 1 → 2, 2 → 0, 0 → 3, 3 → 0, 0 → 4, 4 → 2, 2 → 3, 3 → 0, 0 → 3, 3 → 2, 2 → 1, 1 → 2, 2 → 0, 0 → 1, 1 → 7, 7 → 0, 0 → 1.

Optimal Page Replacement algorithm | Operating System

LEAST RECENTLY USED PAGE

24 जून 2019 15:37

Least Recently used (Replace the least recently used page in Past)

f ₄				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2		
f ₃			1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1		
f ₂		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
f ₁	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	3	7	7	7	
	*	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit	Hit

Rq → 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Arrows indicating replacements:

- 7 → 0 (at index 2)
- 0 → 1 (at index 4)
- 1 → 2 (at index 6)
- 2 → 3 (at index 8)
- 3 → 0 (at index 10)
- 0 → 4 (at index 12)
- 4 → 2 (at index 14)
- 2 → 3 (at index 16)
- 3 → 1 (at index 18)
- 1 → 2 (at index 20)
- 2 → 0 (at index 22)
- 0 → 1 (at index 24)
- 1 → 7 (at index 26)
- 7 → 0 (at index 28)

Least Recently Used Page Replacement Algorithm | Operating System

INVERTED PAGING

24 जून 2019 15:41

Inverted

Global PT

1) Each Process has its own PT

2) PT will be in M/M

$M/M = 1024$ for P_1

$IPT = 1024$

Page No	Process ID
0	P_0
1	P_1
2	P_2
3	P_1
4	P_3
5	P_2

Page table of P_1

0	f_0
1	X
2	f_2
3	X

Page table of P_2

0	X
1	f_1
2	X
3	f_4

Page table of P_3

0	X
1	f_3
2	f_5
3	X

Inverted paging | Memory Management | Operating System

Inverted

Global PT

1) Each Process has its own PT

2) PT will be in M/M

$M/M = 1024$ for P_1

$IPT = 1024$

Page No	Process ID
0	P_0
1	P_1
2	P_1
3	P_3
4	P_2
5	P_3

Page table of P_1

0	f_0
1	X
2	f_2
3	X

Page table of P_2

0	X
1	f_1
2	X
3	f_4

Page table of P_3

0	X
1	f_3
2	f_5
3	X

Searching time

8. THRASHING

20 जून 2019 19:39

The video frame shows a man in a yellow shirt standing in front of a whiteboard. The whiteboard contains handwritten notes and a graph illustrating thrashing. The graph plots CPU Utilization (Y-axis) against the Degree of Multiprogramming (X-axis). The curve rises to a peak labeled 'λ' and then falls, with the downward slope labeled 'Thrashing'. Handwritten notes include 'THRASHING', 'CPU Utilization ↑', 'Page fault', 'M/M Size increased', and 'Long term Scheduler'.

What is Thrashing | Operating System

10. SEGMENTATION

20 जून 2019 19:39

segmentation faults

JULIA EVANS
@b0rk

every program has memory.
the operating system keeps
track of which memory the
program has allocated

0x12340 to 0x19990 read only
0x10000 to 0x12340 read/write
address range permissions

On a 64-bit computer,
there are 2^{64} bytes of
possible memory addresses
(~18 million terabytes).

Almost all of those are
not valid memory addresses
for your program!

When your program tries
to access an invalid
memory address, this happens:

what's at
0x99990?
SEGMENTATION
FAULT!

0 is not a valid
memory address for any
program

I reserve that address
to help catch
programming errors!
operating
system

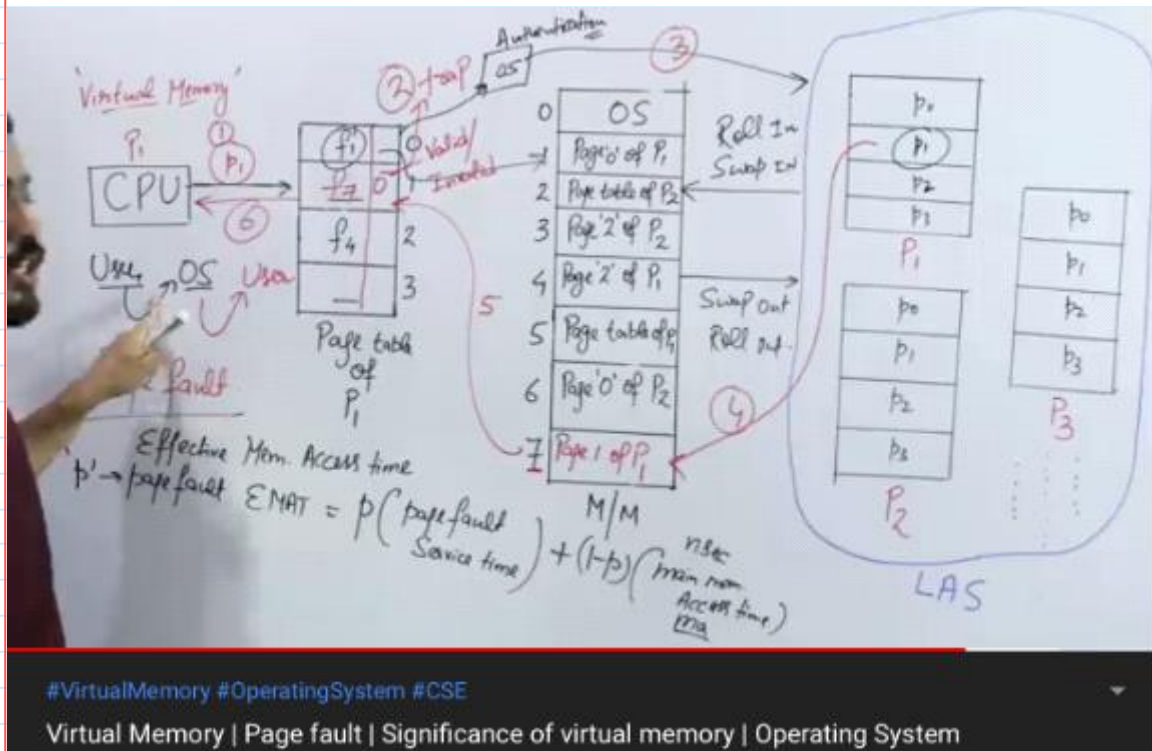
the most common invalid
memory address programs
try to access is 0

int *x = NULL;
y = *x; ← in C, NULL=0
causes a
segmentation
fault!

segfaults are
a good thing!
really? why??
finding out about a bug
via a segfault now is
WAY BETTER than corrupting
data & finding out in a month

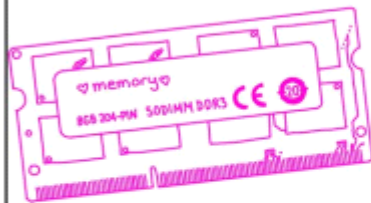
11. PAGE FAULTS | VIRTUAL MEMORY

20 जून 2019 19:41



virtual memory JULIA EVANS @b0rk

your computer has physical memory



physical memory has addresses

0-8GB

but when your program references an address like 0x5c69a2a2

that's not a physical memory address!
It's a **virtual** address

every program has its own virtual address space

program 1: 0x129520 → "puppies"

program 2: 0x129520 → "bananas"

Linux keeps a mapping from virtual memory pages to physical memory pages called the "page table"

a "page" is a 4kb chunk of memory or sometimes bigger

PID	virtual addr	physical addr
1971	0x20000	0x192000
2310	0x20000	0x228000
2310	0x21000	0x9788000

when your program accesses a virtual address

CPU: I'm accessing 0x21000

MMU "memory management unit": I'll look that up in the page table and then access the right physical address

hardware

every time you switch which process is running, Linux needs to switch the page table

Linux: here's the address of process 2950's page table

thanks I'll use that now! MMU

14. Fragmentation

20 जून 2019 19:46

Fragmentation:

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Fragmentation is of two types –

S.N.	<u>Fragmentation & Description</u>
1	<p><u>External fragmentation</u> Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.</p> <p>External fragmentation can be reduced by <u>compaction or shuffle memory contents</u> to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.</p>
2	<p><u>Internal fragmentation</u> Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.</p> <p>The internal fragmentation can be reduced by <u>effectively assigning the smallest partition</u> but large enough for the process.</p>

15. Unix /Linux/(Based General Ques)

20 जून 2019 19:46

DISK SCHEDULING ALGO

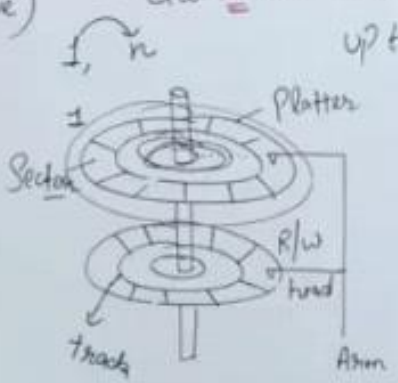
24 जून 2019 15:43

"Disk Scheduling Algorithms"

Goal: To minimize the Seek time

Seek time = time taken to reach up to desired track

- FCFS (First Come first Serve)
- SSTF (Shortest Seek time first)
- SCAN
- LOOK
- CSCAN (Circular SCAN)
- CLOOK (Circular Look)



The diagram illustrates the physical structure of a disk drive. It shows two platters stacked vertically. Each platter is divided into concentric circles representing tracks. The tracks are further divided into sectors. A read/write head assembly is shown positioned between the platters, with a label 'R/W head' pointing to it. The head is connected to an 'Arm'. Labels include 'Platter', 'Sector', 'Track', 'R/W head', and 'Arm'. Arrows indicate the direction of data flow or head movement.

Platter
↓
Surface
↓
Track
↓
Sector

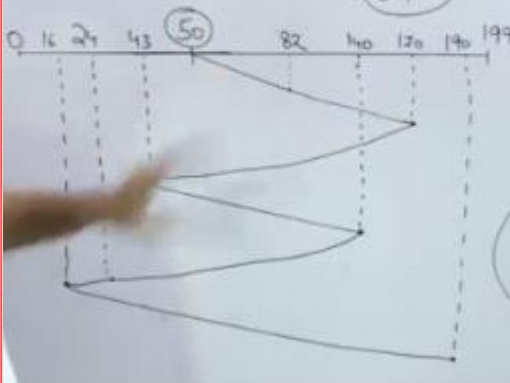
FIFO DISK

24 जून 2019 15:44

"Disk Scheduling Algorithms"

→ FCFS (First Come first Serve)

$= 642$



A disk contains 200 tracks (0-199)
Request queue contains track no.
82, 170, 43, 140, 24, 16, 190 respectively
Current position of R/W head = 50
Calculate total no of tracks movement by R/W head.

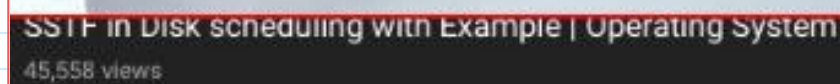
$$(82-50) + (170-82) + (170-43) + (140-43) + (140-24) + (24-16) + (190-16)$$
$$(170-50) + (170-43) + (140-43) + (140-16) + (190-16)$$

FCFS in Disk scheduling with Example | Operating System

7,816 views

24 जून 2019 15:45

24 जून 2019 15:45



SCAN AND C-SCAN

24 जून 2019 15:45

Ques: A disk Contains 200 tracks
Request queue Contains tracks
82, 170, 43, 140, 24, 16, 190
Current position of R/W head = 50
→ Calculate total no of tracks movement by head using **SCAN**?
→ If R/W head takes 1ms to move from one track to another then total time taken —?

Diagram showing disk tracks (0 to 199) and head movement path for SCAN algorithm. The head starts at 50, moves to 199, and then jumps back to 16. The total movement is calculated as $(199 - 50) + (199 - 16) = 332$.

SCAN Algorithm in Disk scheduling with Example | Operating System

72,124 views

Ques: A disk Contains 200 tracks (0 to 199)
Request queue Contains tracks no.
82, 170, 43, 140, 24, 16, 190
Current position of R/W head = 50
→ Calculate total no of tracks movement by R/W head using **C-SCAN**?
→ Direction is towards large value?

Diagram showing disk tracks (0 to 199) and head movement path for C-SCAN algorithm. The head starts at 50, moves to 199, and then jumps back to 43. The total movement is calculated as $(199 - 50) + (199 - 0) + (43 - 0) = 391$.

C-SCAN Algorithm in Disk scheduling with Example | Operating System

57,288 views

LOOK AND C-LOOK

24 जून 2019 15:46

Ques: A disk contains 200 tracks (0-199).
Request queue contains tracks no. 82, 170, 43, 140, 24, 16, 190 respectively.
Current position of R/W head = 50
→ Calculate total no. of tracks movement by R/W head using **LOOK**?
→ If R/W head takes 1ms to move from one track to another then total time taken — ?

Direction is towards large value?
 $(50 - 16) + (190 - 50) = 34$

LOOK Algorithm in Disk scheduling with Example | Operating System
45,737 views

Ques: A disk contains 200 tracks.
Request queue contains tracks 82, 170, 43, 140, 24, 16, 190 respectively.
Current position of R/W head = 50
→ Calculate total no. of tracks movement by R/W head using **C-LOOK**?
→ If R/W head takes 1ms to move from one track to another then total time taken — ?

Direction is towards large value?
 $(190 - 50) + (190 - 16) + (43 - 16) = 341$