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 schedule**r = 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.

SCHDULING:

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 i**s 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 in unaware of them.
65. The two steps of a process execution are : **CPU & I/O Burst.**
66. **non – preemptive scheduling occur**s : 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 system**s.
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 imp**lies 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

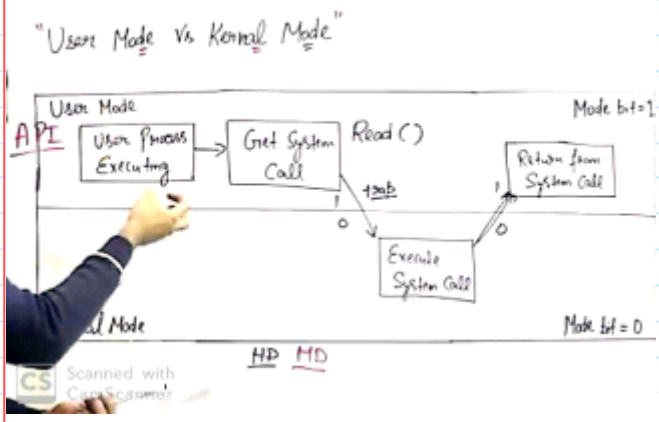
- **I/O Devices:**
  1. Each device controller has a local buffer.
  2. CPU moves data from/to main memory to/from local buffers.
  3. I/O is from the device to local buffer of controller.
  4. Device controller informs CPU that it has finished its operation by causing an **interrupt.**

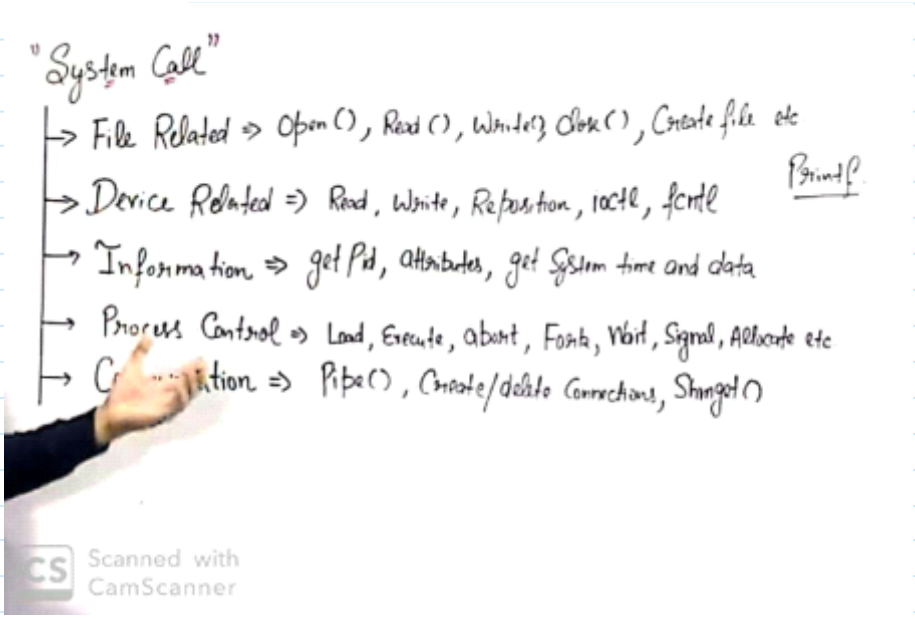
- **Common Functions of Interrupts:**
  1. Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines.
  2. Interrupt architecture must save the address of the interrupted instruction.
  3. Incoming interrupts 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





2. Threads

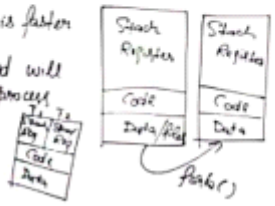
20 जून 2019 18:02

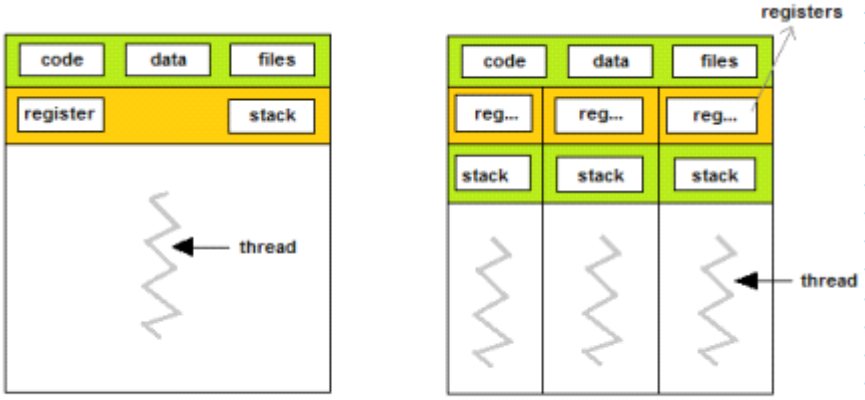
Process

1) System Calls involved in Process
2) OS treats different processes differently
3) Different process have different Copies of Data, files, Code
4) Content 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) Content switching is faster
5) Blocking a thread will block entire process
6) Interdependent



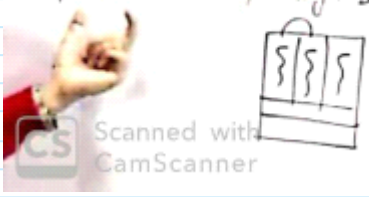


"User level Thread"

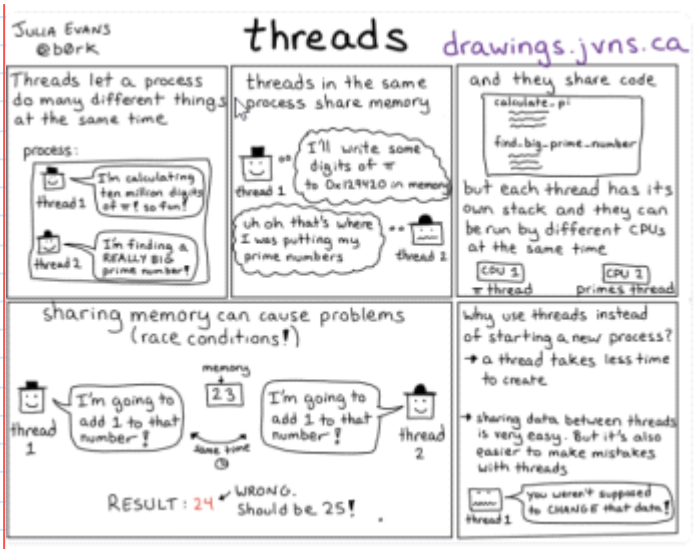
1) User level threads are managed by User level library
2) User level threads are typically fast
3) Context Switching is faster
4) If one user level threads perform blocking operation then entire process got blocked

"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



Process > KLT > ULT


Julia Evans
@b0rk
threads
drawings.jvns.ca
Threads let a process do many different things at the same time
process:
thread 1
I'm calculating ten million digits of π so fun!
thread 2
I'm finding a REALLY BIG prime number!
threads in the same process share memory
thread 1
I'll write some digits of π to 0x129420 in memory
uh oh that's where I was putting my prime numbers
thread 2
and they share code
calculate_pi
find_big_prime_number
but each thread has its own stack and they can be run by different CPUs at the same time
CPU 1
π thread
CPU 2
primes thread
sharing memory can cause problems (race conditions!)
memory
23
thread 1
I'm going to add 1 to that number!
same time
I'm going to add 1 to that number!
thread 2
RESULT: 24 ← WRONG. Should be 25!
Why use threads instead of starting a new process?
→ a thread takes less time to create
→ sharing data between threads is very easy. But it's also easier to make mistakes with threads
thread 1
you weren't supposed to CHANGE that data!


What's a thread?
Julia Evans
@b0rk
drawings.jvns.ca
a process can have lots of threads
process id
1888
1888
1888
1888
thread id
1888
1892
1893
2007
threads share memory
thread 1
I'm going to write "2" to address 2477886
I can overwrite that if I want!
thread 2
but they can run different code
thread 1
I'm doing a calculation!
I'm waiting for a network request to finish!
thread 2
sharing memory can cause problems
"race conditions"
memory
23
thread 1
I'm going to add 1 to that number!
same time
I'm going to add 1 to that number!
thread 2
RESULT: 24 ← WRONG (should be 25)
if you have 8 CPU cores, you can run code for 8 threads at the SAME TIME
1 5
2 6
3 7
4 8
SO BUSY
CPU cores

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

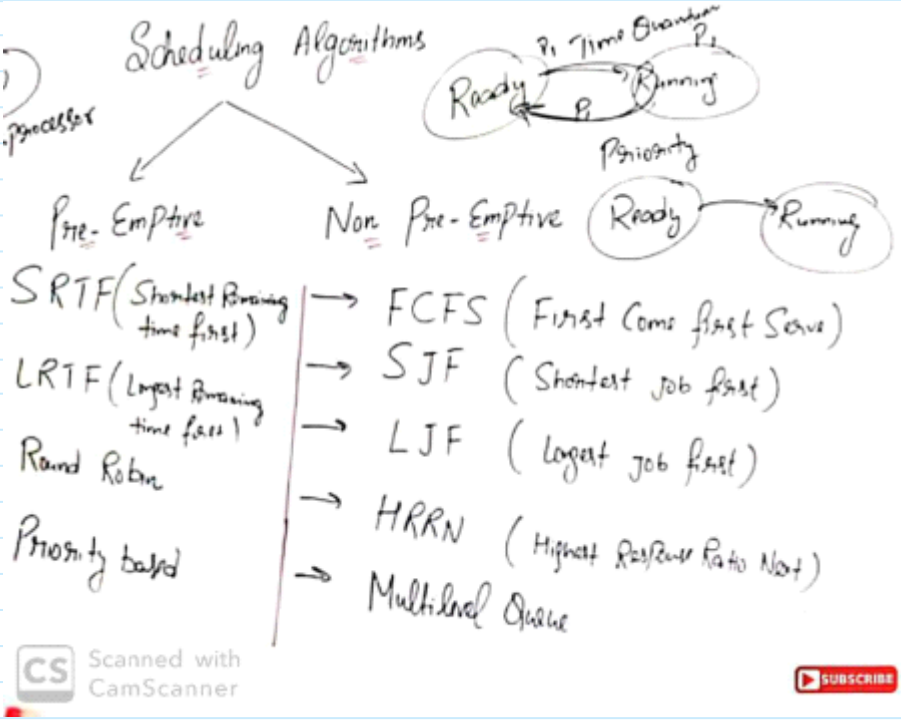
1. FCFS: arrival time
2. SJF: time burst; if arrival time is also same then process id
3. LJF
4. HRRN
5. Multilevel que: problem of starvation can happen.
6. Multilevel feedback que:

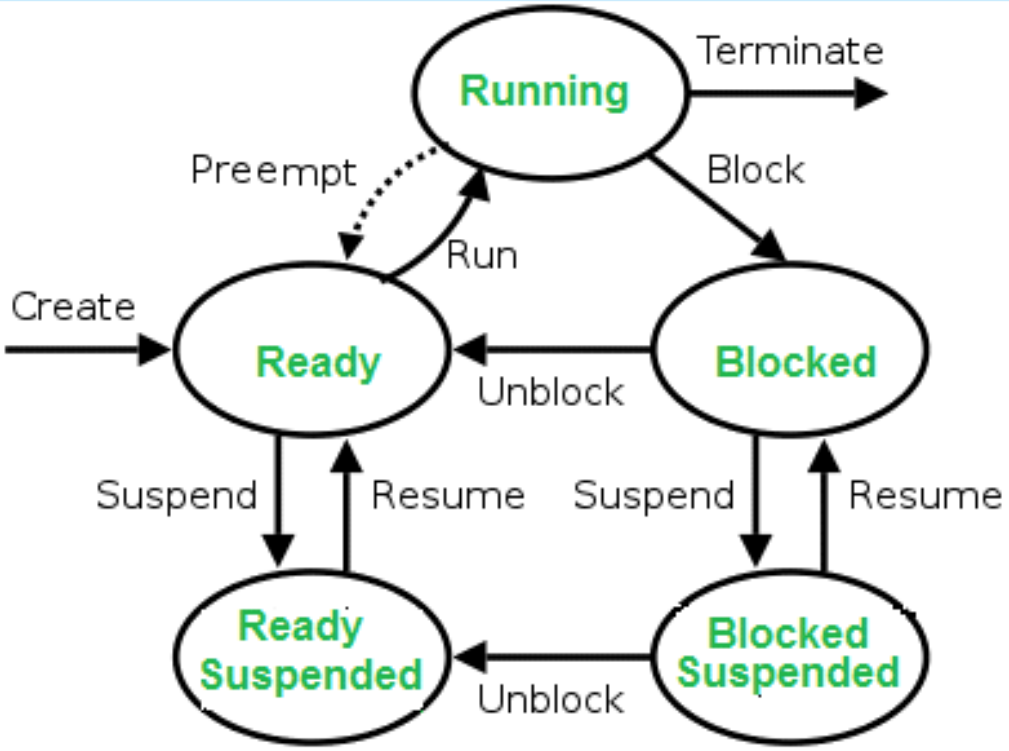
Interrupt: has the highest priority among all processes

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

PARAMENTER	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


Scheduling Algorithms
Pre-Emptive
Non Pre-Emptive
Ready
Running
Priority
SRTF (Shortest Burning time first)
LRTF (Longest Burning time first)
Rand Robin
Priority based
FCFS (First Come first Serve)
SJF (Shortest job first)
LJF (Longest job first)
HRRN (Highest Response Ratio Next)
Multilevel Queue
Scanned with CamScanner
SUBSCRIBE


Running
Terminate
Preempt
Block
Run
Create
Ready
Blocked
Unblock
Suspend
Resume
Suspend
Resume
Ready Suspended
Blocked Suspended
Unblock

"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. {Completion time − Arrival time}

→ Waiting time. {Turn Around time − Burst time}

→ Response time. {(The time at which a process get CPU first time) − (Arrival time)}

Scanned with CamScanner

SUBSCRIBE

2A. PROCESS SCHEDULING

21 जून 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
~~P_1~~	1	3	6	5	2	2
P_2	2	4	10	8	4	4
~~P_3~~	1	2	3	2	0	0
~~P_4~~	4	4	14	10	6	6

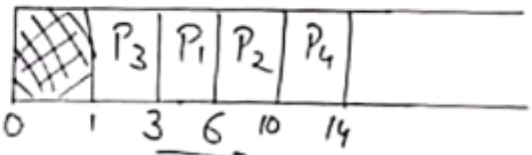
Criteria: "Burst Time"

Mode: "Non-Preemptive"

TAT = CT − AT

WT = TAT − BT

Gantt Chart



P_2 P_4

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~~ 4 ~~3~~ 2 1 0	9	9	4	0
P_2	1	~~3~~ 2 1 0	4	3	0	0
P_3	2	4 0	13	11	7	7
P_4	4	1 0	5	1	0	0

Criteria: Burst Time

Mode: Preemptive

TAT = CT − AT

WT = TAT − BT

RT = { CPU first time − AT }

Gantt Chart

P_1	P_2	P_2	P_2	P_4	P_1	P_1	P_1	P_1	P_3

0 1 2 3 4 5 6 7 8 9 13

Time ⟶

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

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

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

Scanned with CamScanner

2D. FCFS

22 जून 2019 11:16

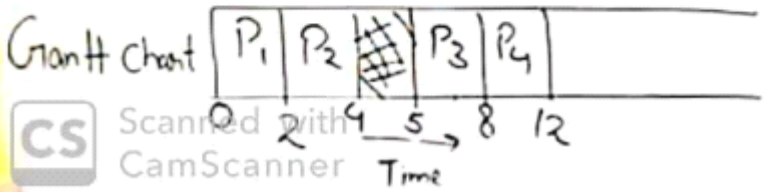
Process No	Arrival Time	Execution Burst Time	Completion Time	TAT	WT	RT
~~P_1~~	0	2	2	2	0	0
~~P_2~~	1	2	4	3	1	1
~~P_3~~	5	3	8	3	0	0
~~P_4~~	6	4	12	6	2	2

Criteria: "Arrival Time"

Mode: "Non-Preemptive"

CT − AT = TAT

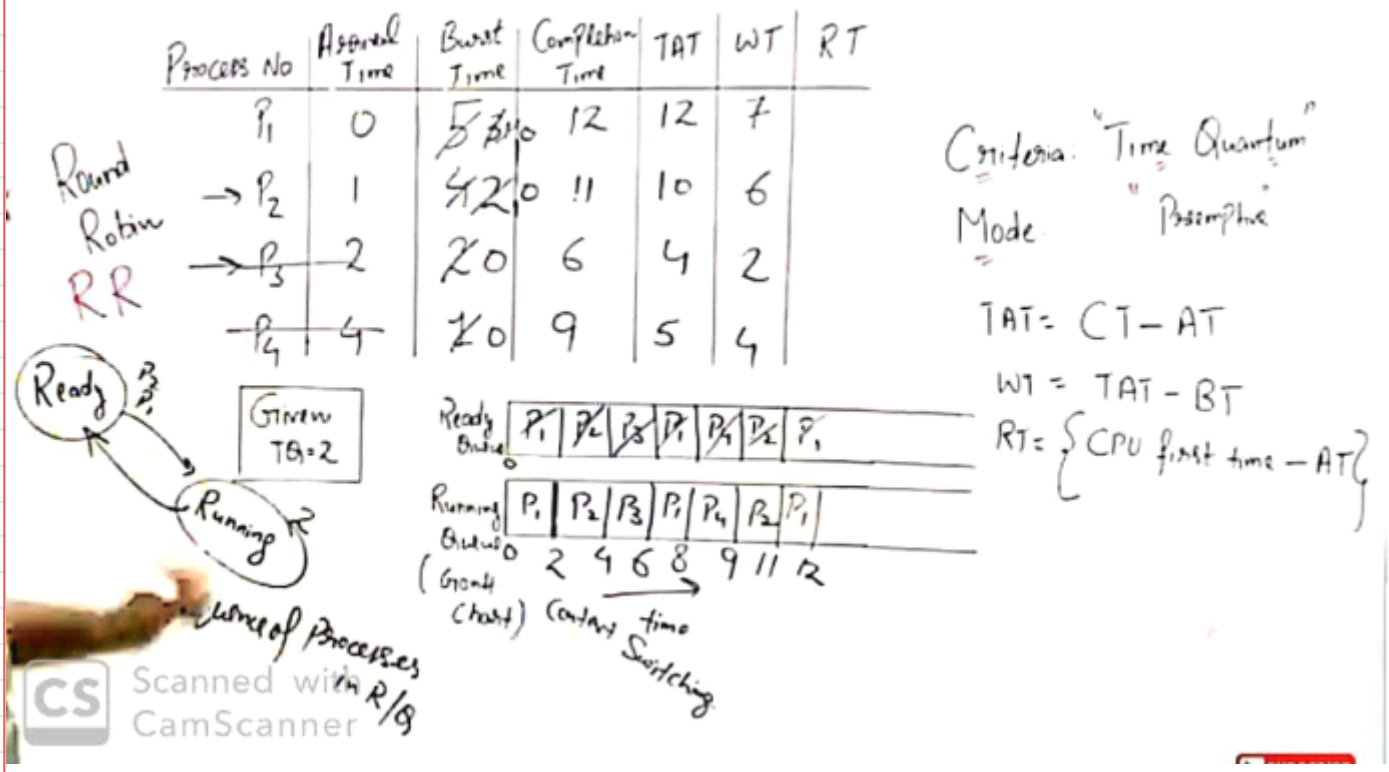
TAT − BT = WT



Scanned with CamScanner

2E. RR

22 जून 2019 11:17

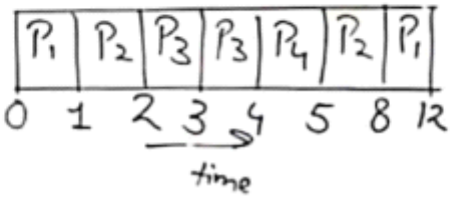


2F. PRIORITY SCHEDULING

22 जून 2019 11:18

Priority	Process No	Arrival Time	Burst Time	Completion Time	TAT	WT
10	P_1	0	5 4	12	12	7
20	P_2	1	4 3	8	7	3
30	P_3	2	2 1 0	4	2	0
40	P_4	4	1 0	5	1	0

Higher the no. higher the Priority



Scanned with CamScanner

Priority Scheduling

Criteria: "Priority"

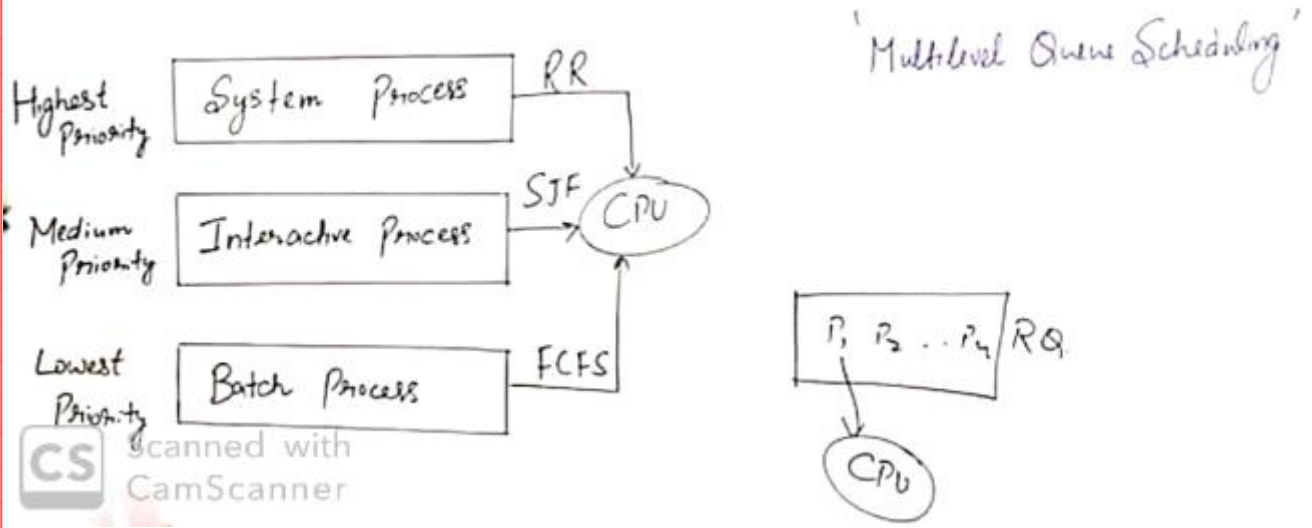
Mode: "Preemptive"

TAT = CT − AT

WT = TAT − BT

2G. MULTILEVEL QUE

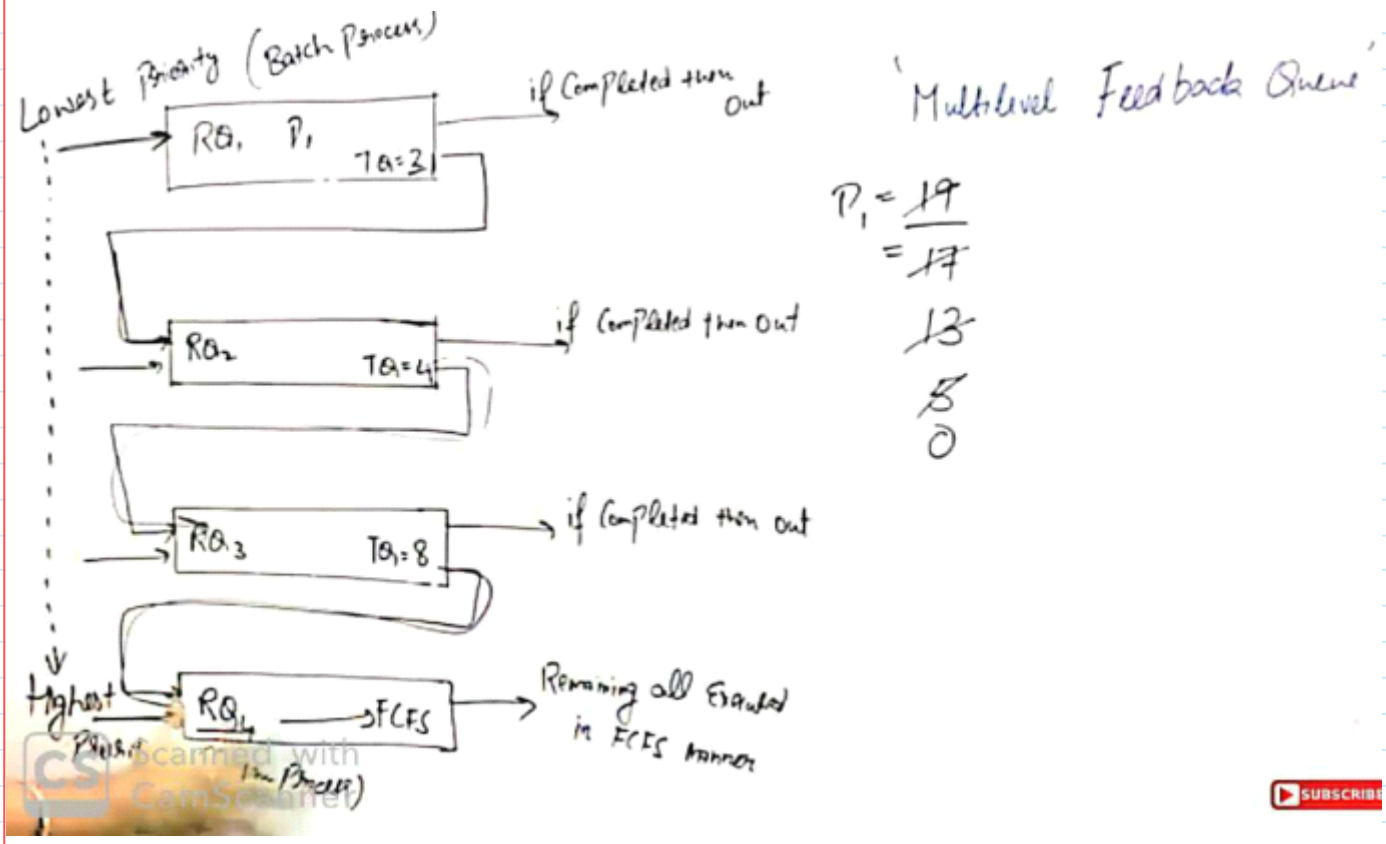
22 जून 2019 11:18



Scanned with CamScanner

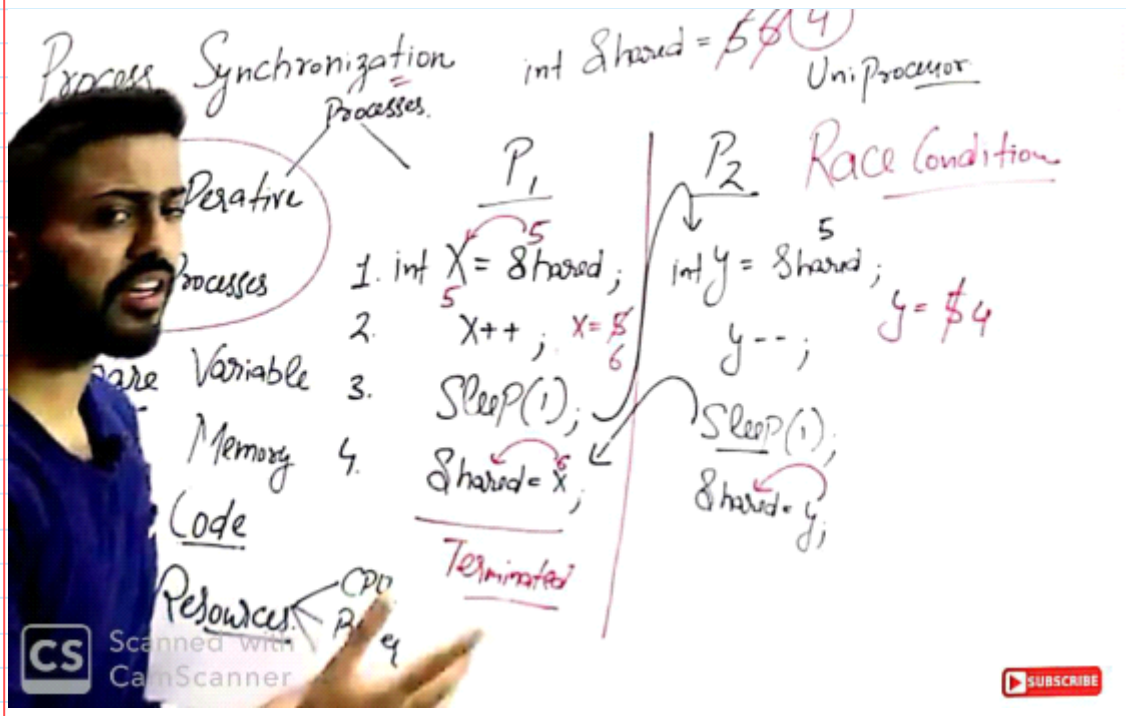
2H. MULTILEVEL FEEDBACK

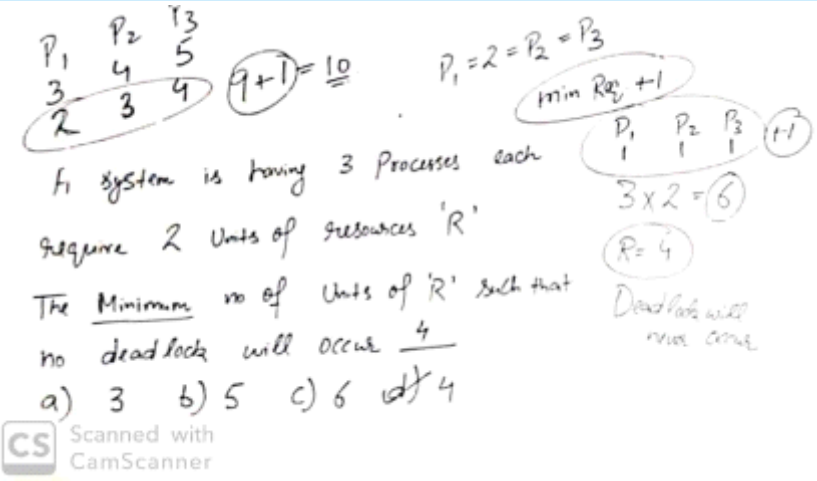
22 जून 2019 11:19

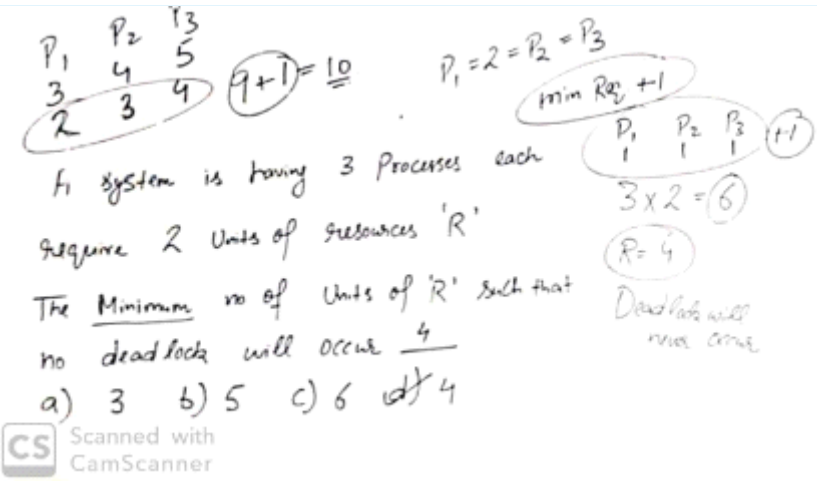


3. Process Synchronization

20 जून 2019 19:36




P1 P2 P3
3 4 5
2 3 4
(1+1) = 10
P1 = 2 = P2 = P3
min Req +1
P1 P2 P3
1 1 1
+1
3 x 2 = 6
R = 4
Deadlock will never occur
If 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
(1+1) = 10
P1 = 2 = P2 = P3
min Req +1
P1 P2 P3
1 1 1
+1
3 x 2 = 6
R = 4
Deadlock will never occur
If 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 =< 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		

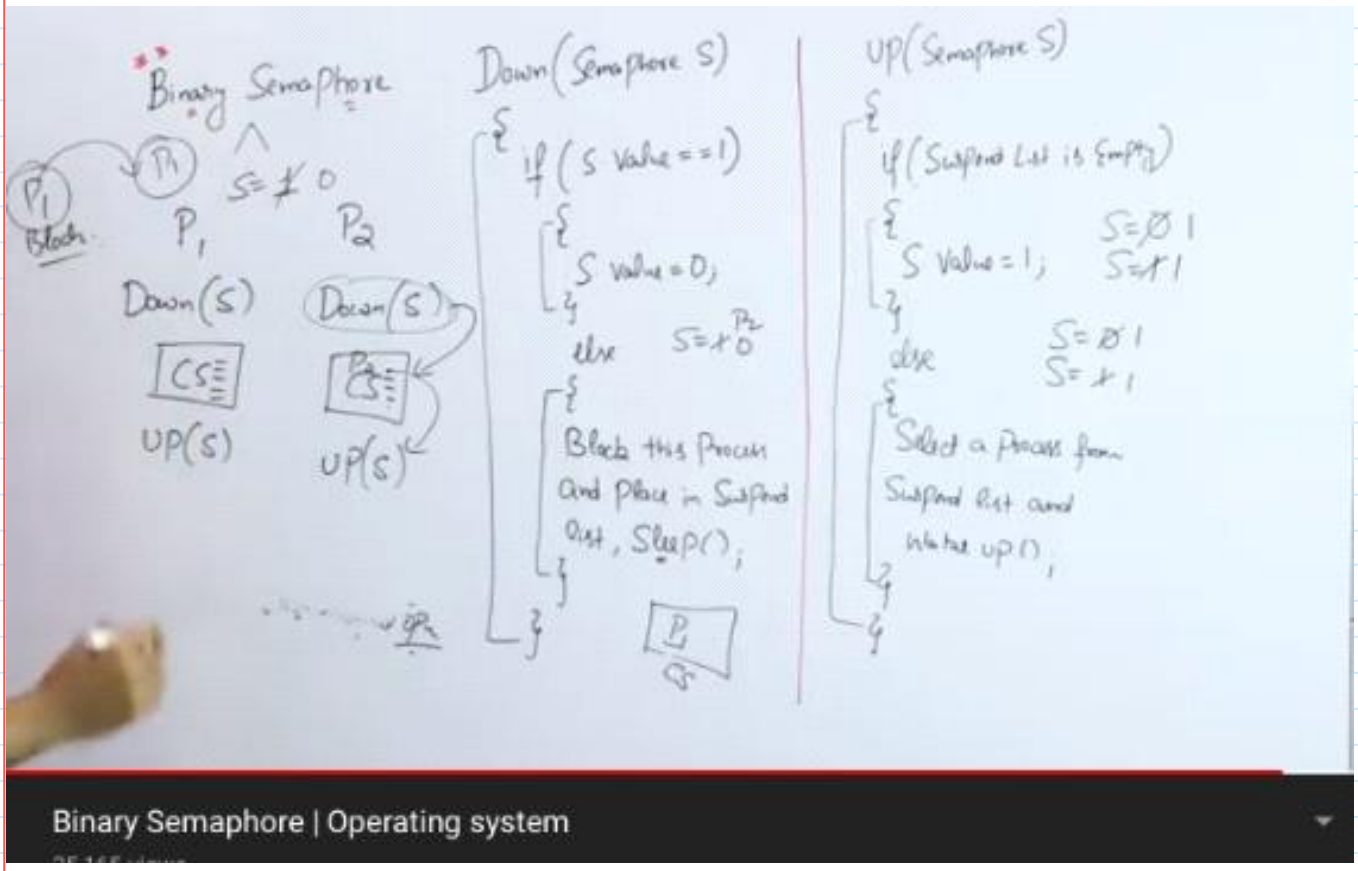
- If S = 0; means there is no process in block list. Means there is no need to wakeup any processes.
- If S = -4, means there are 4 process in block list.
- If S = 10, means 10 processes can be brought into Critical Section (CS) = means 10 successful operation = means process is in CS.

- **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.

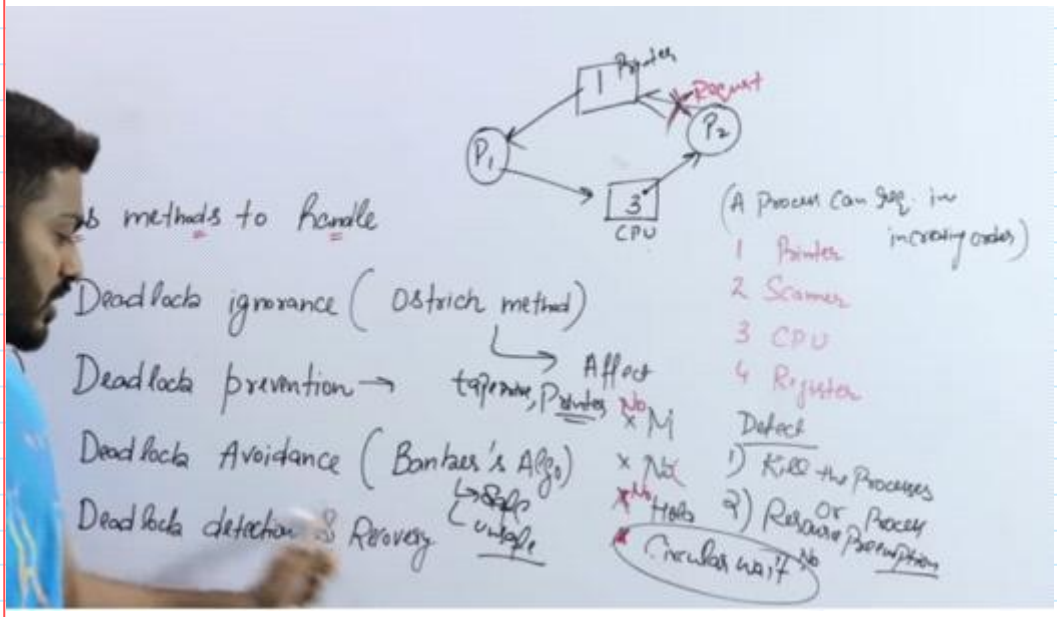
BINARY SEMAPHORES

24 जून 2019 15:39



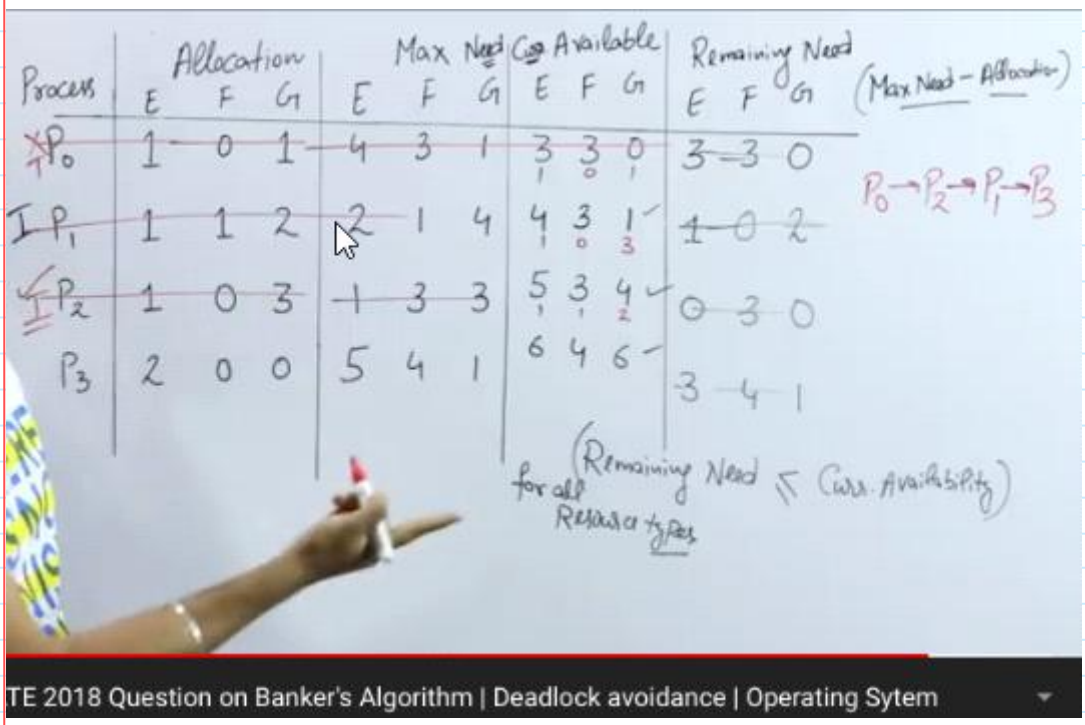
6. DEADLOCK

20 जून 2019 19:37



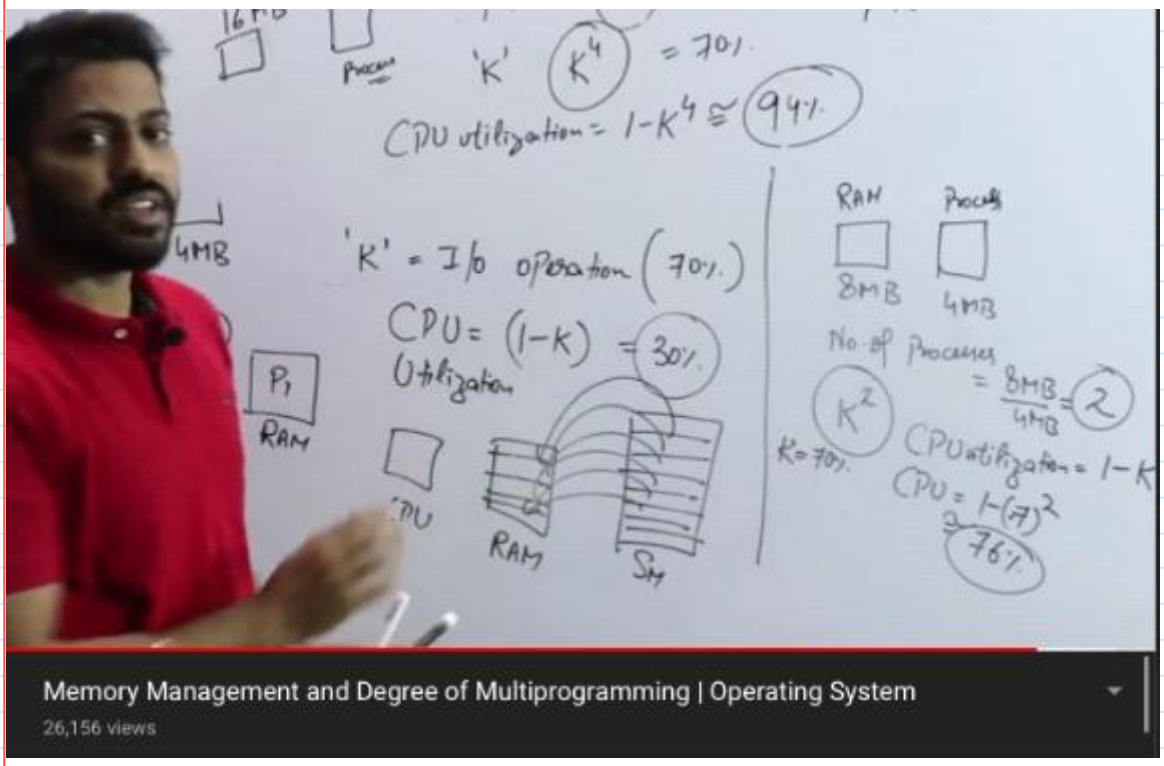
BANKERS ALGORITHM

24 जून 2019 15:31



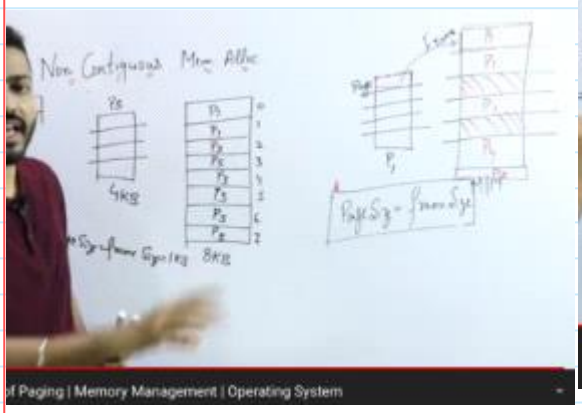
7. MEMORY MANAGEMENT

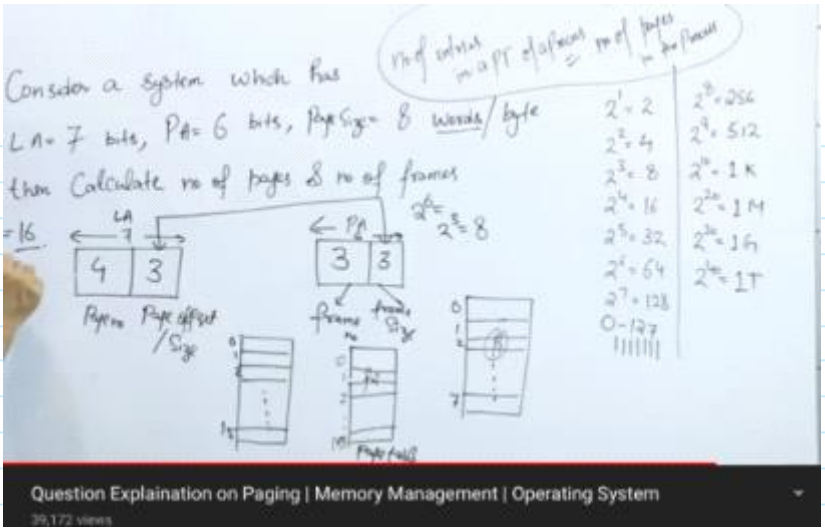
20 जून 2019 19:38

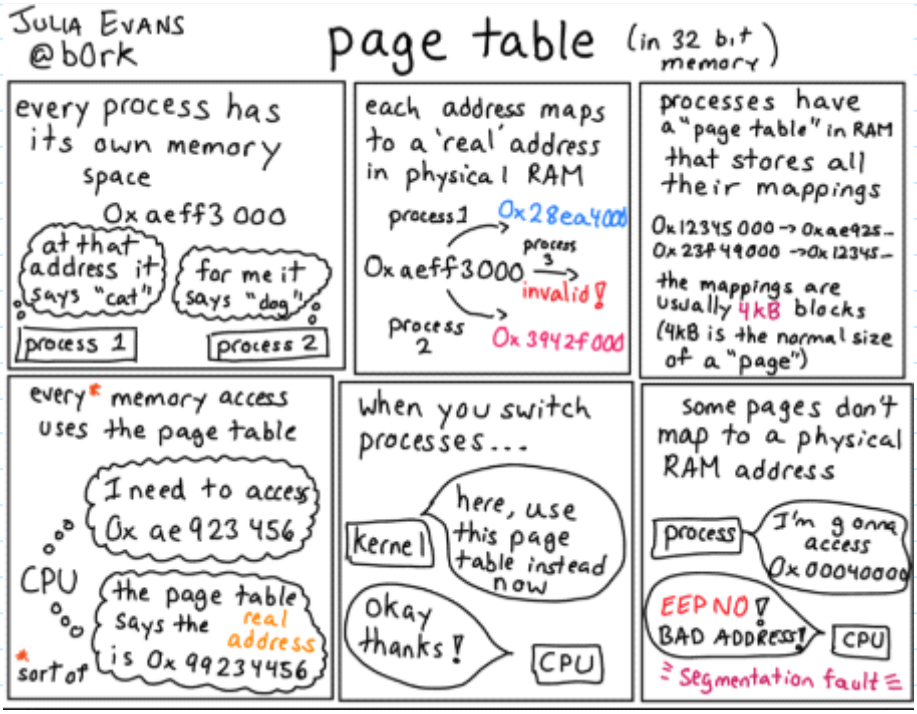


9. PAGING

20 जून 2019 19:39

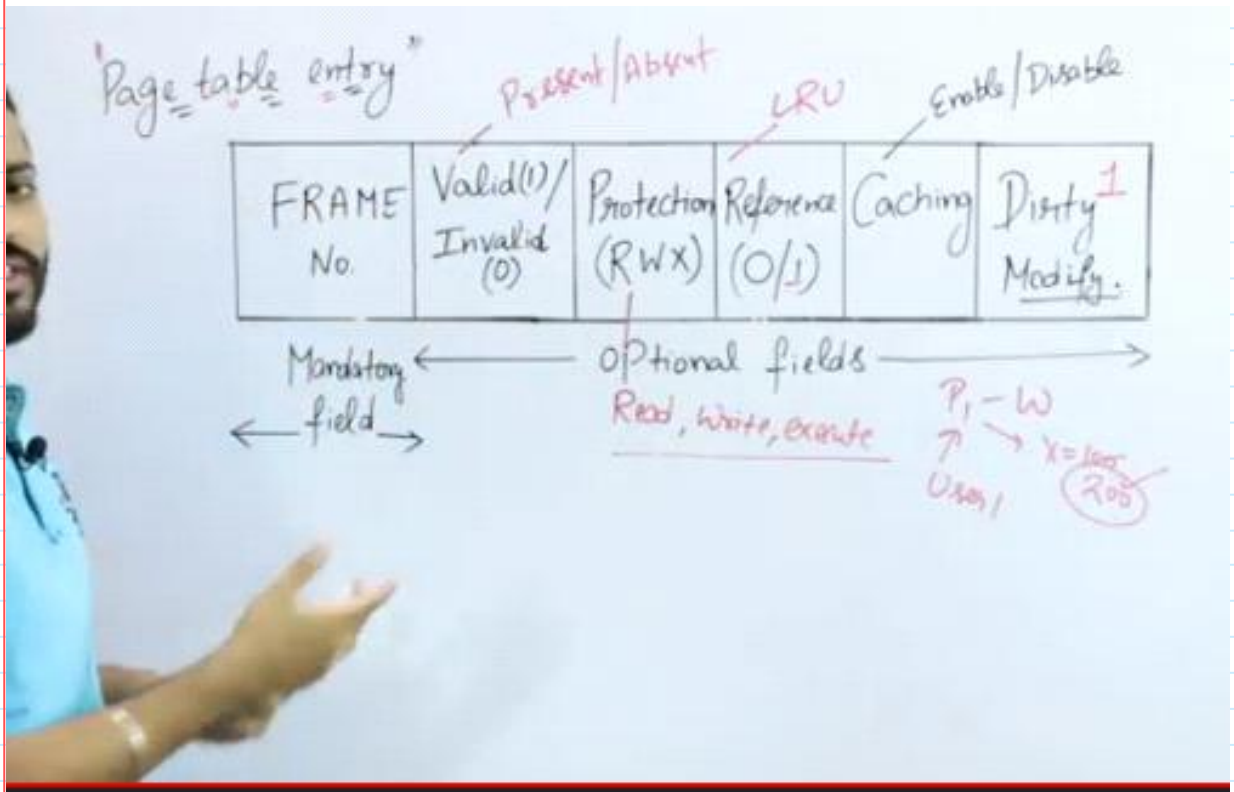






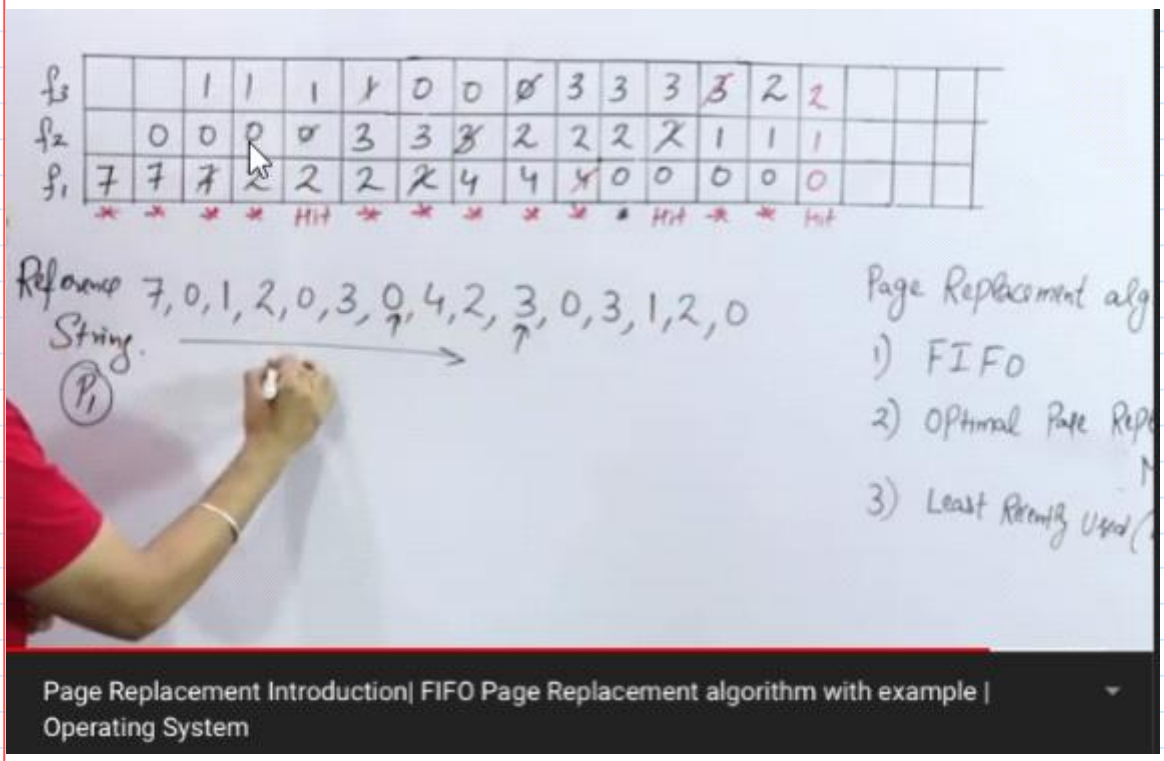
PAGE TABLE

24 जून 2019 15:42



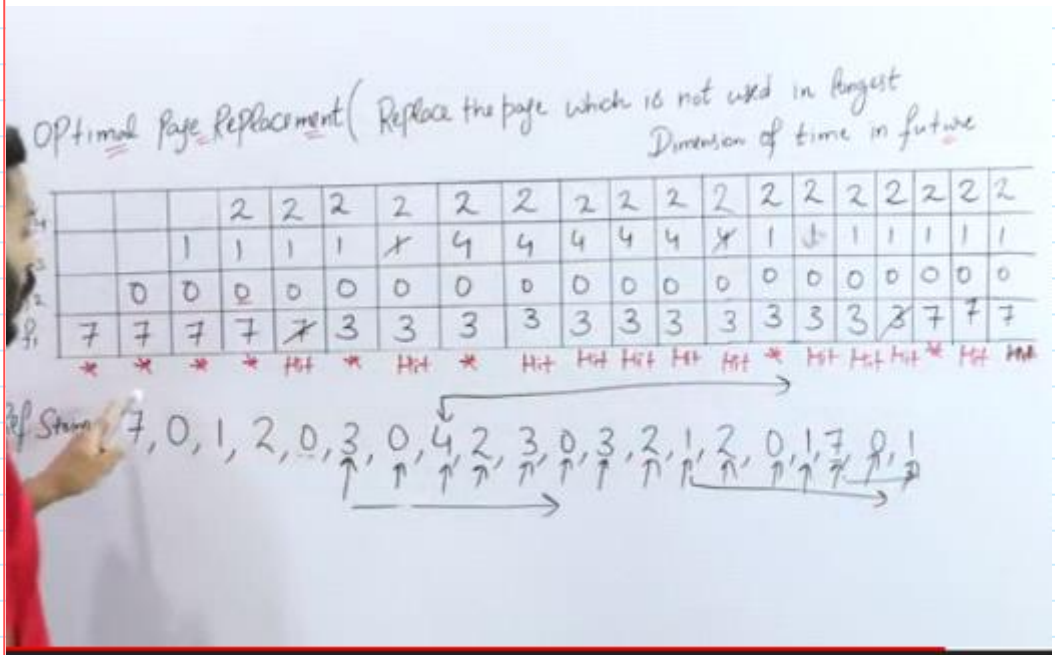
PAGE REPLACEMENT FIFO

24 जून 2019 15:35



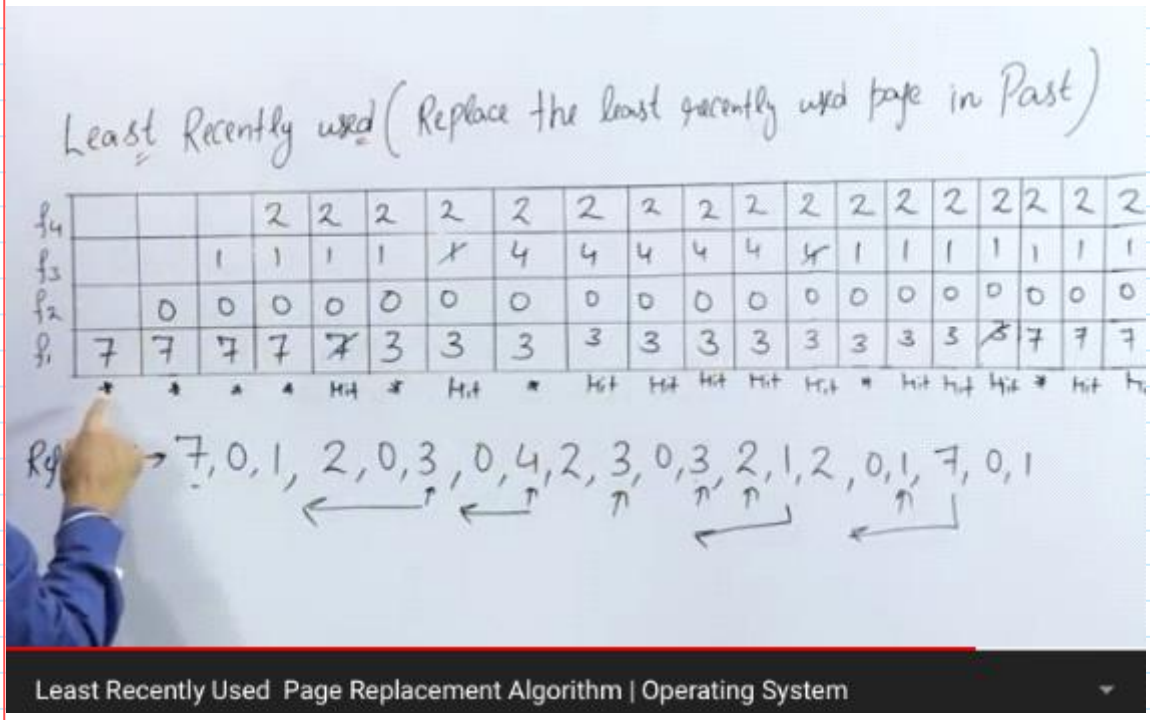
OPTIMAL PAGE EPLACEMENT

24 जून 2019 15:36



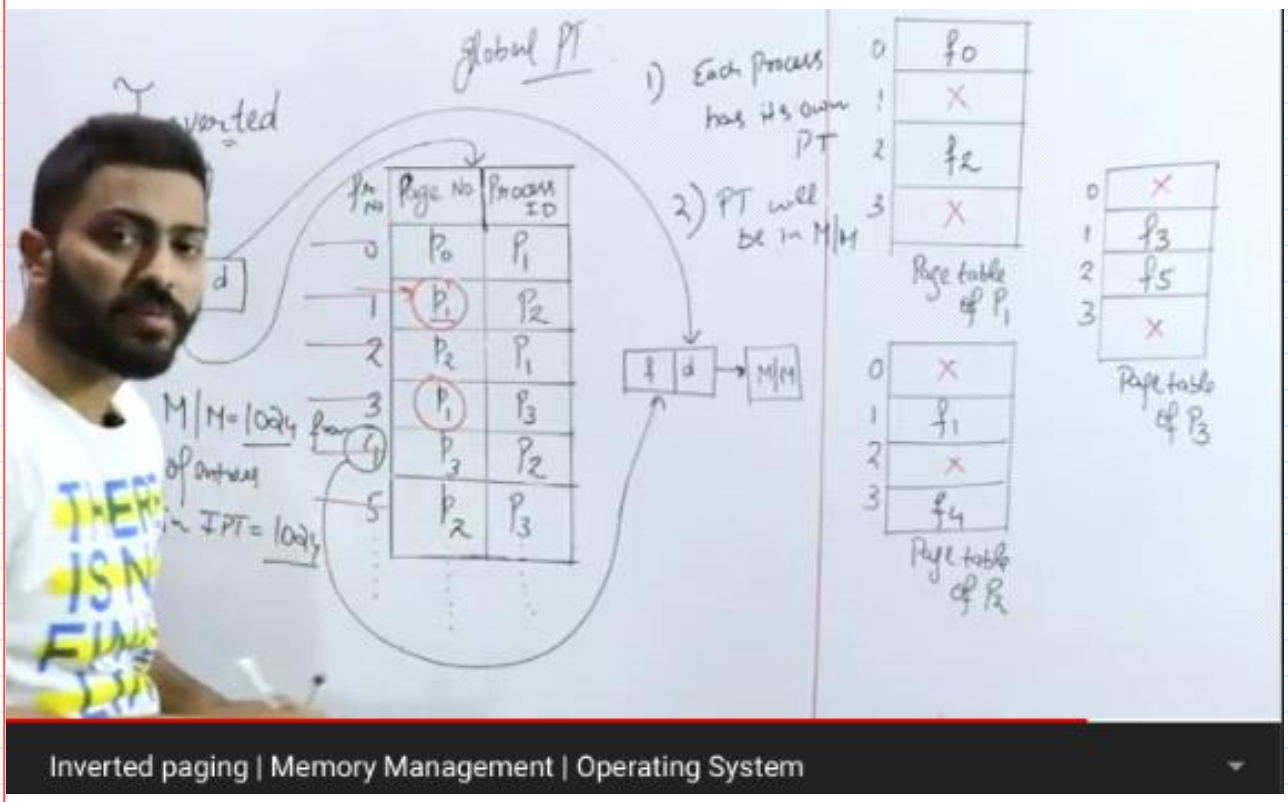
LEAST RECENTLY USED PAGE

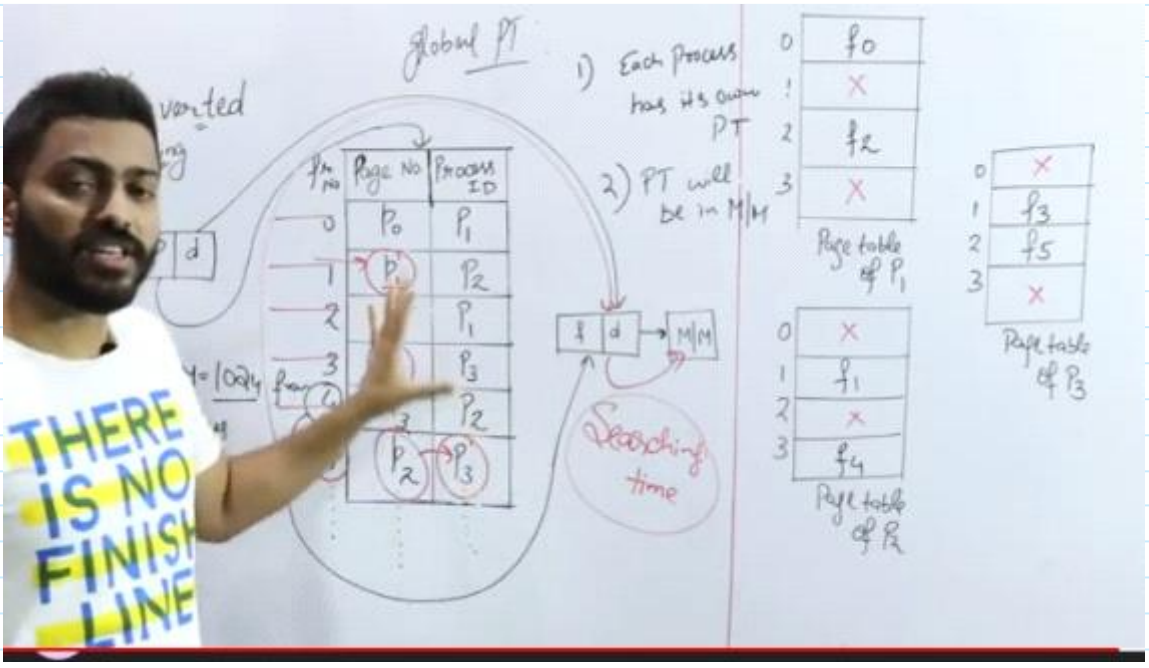
24 जून 2019 15:37



INVERTED PAGING

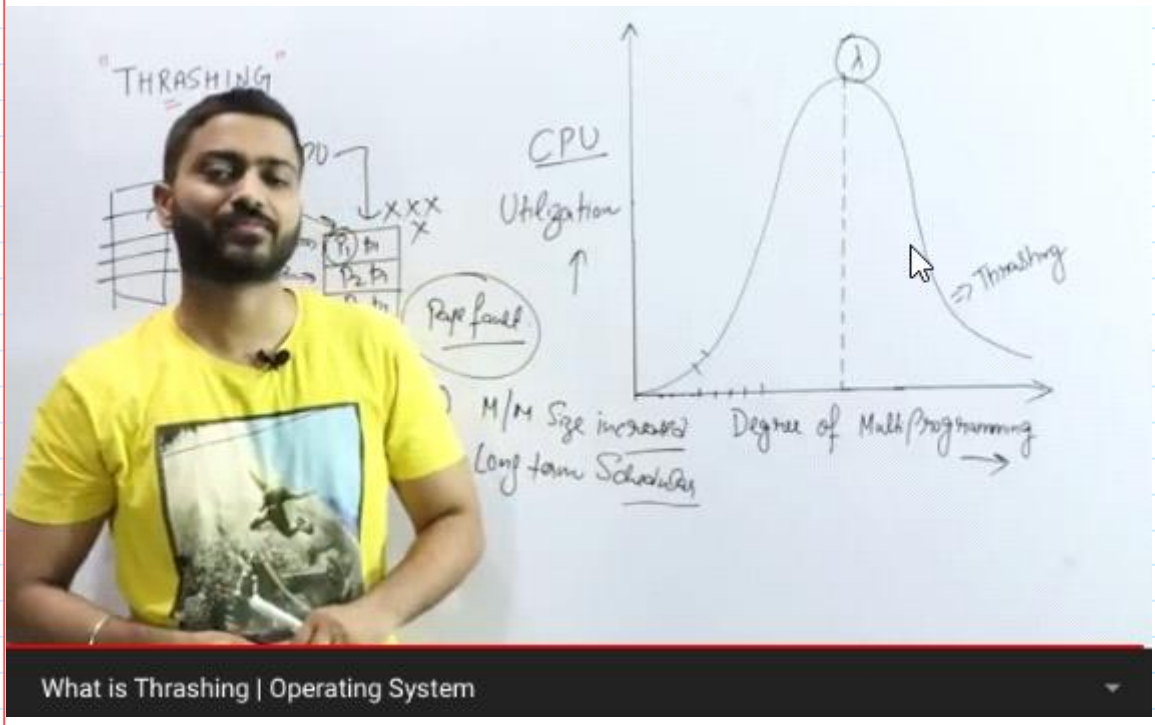
24 जून 2019 15:41





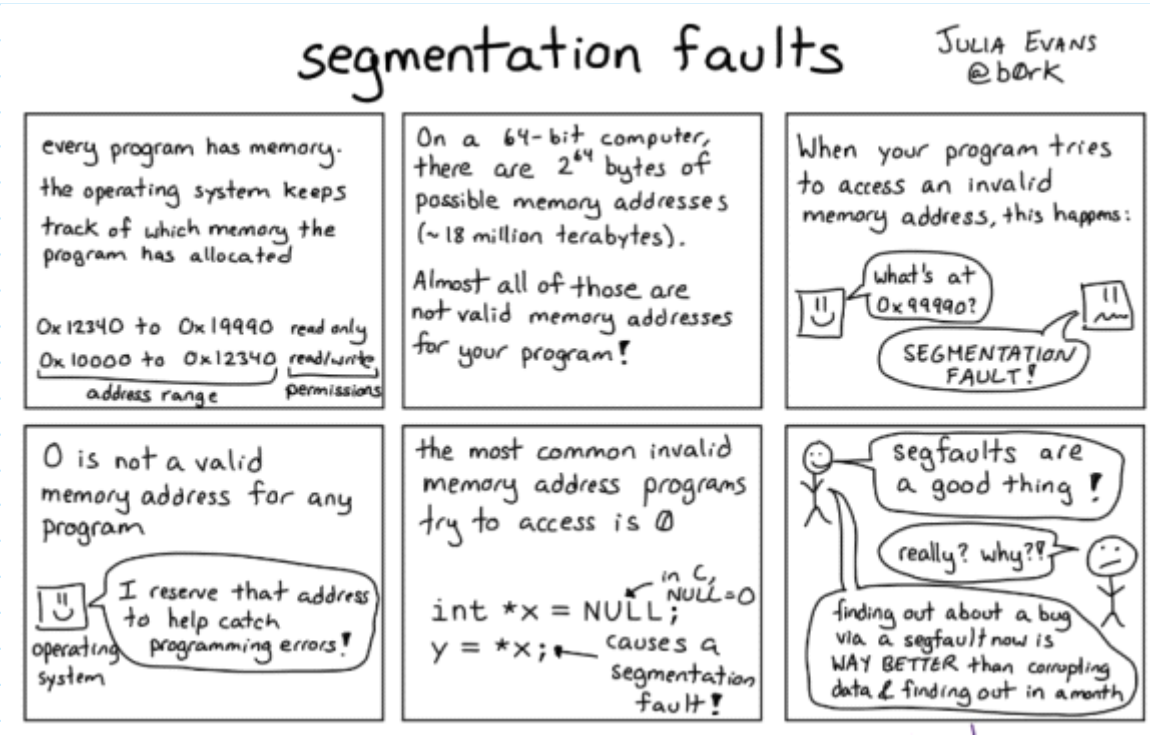
8. THRASHING

20 जून 2019 19:39



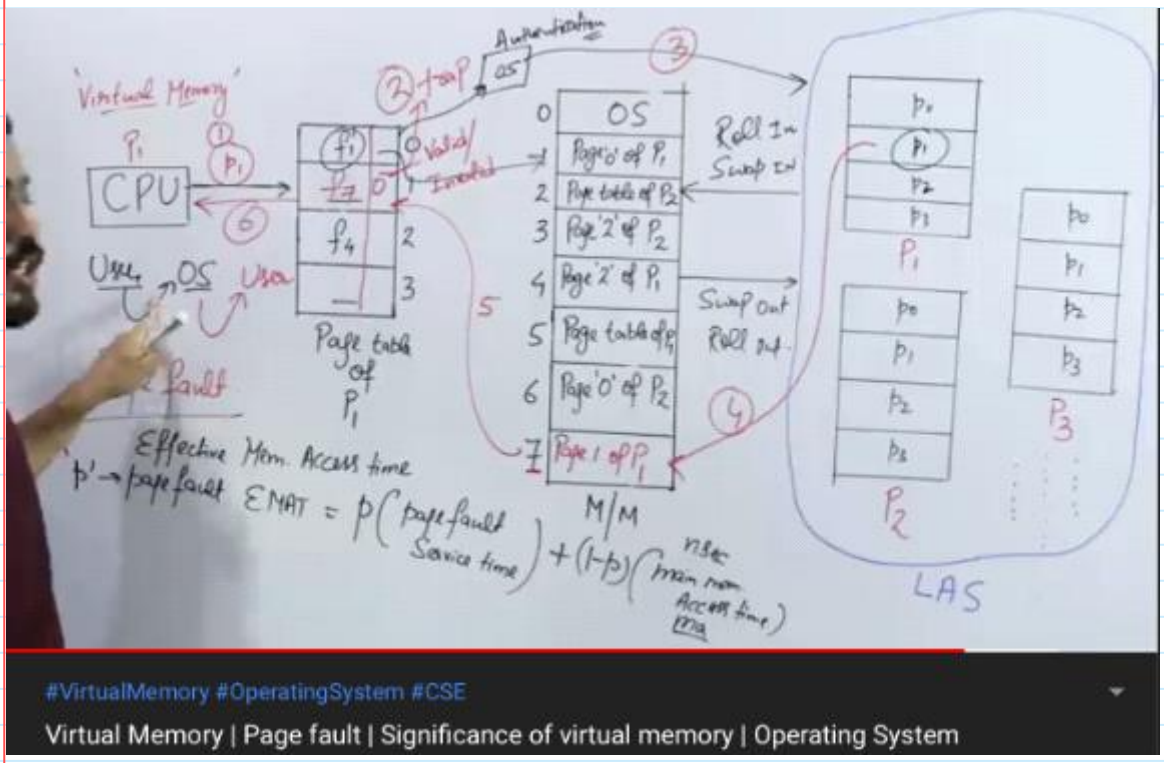
10. SEGMENTATION

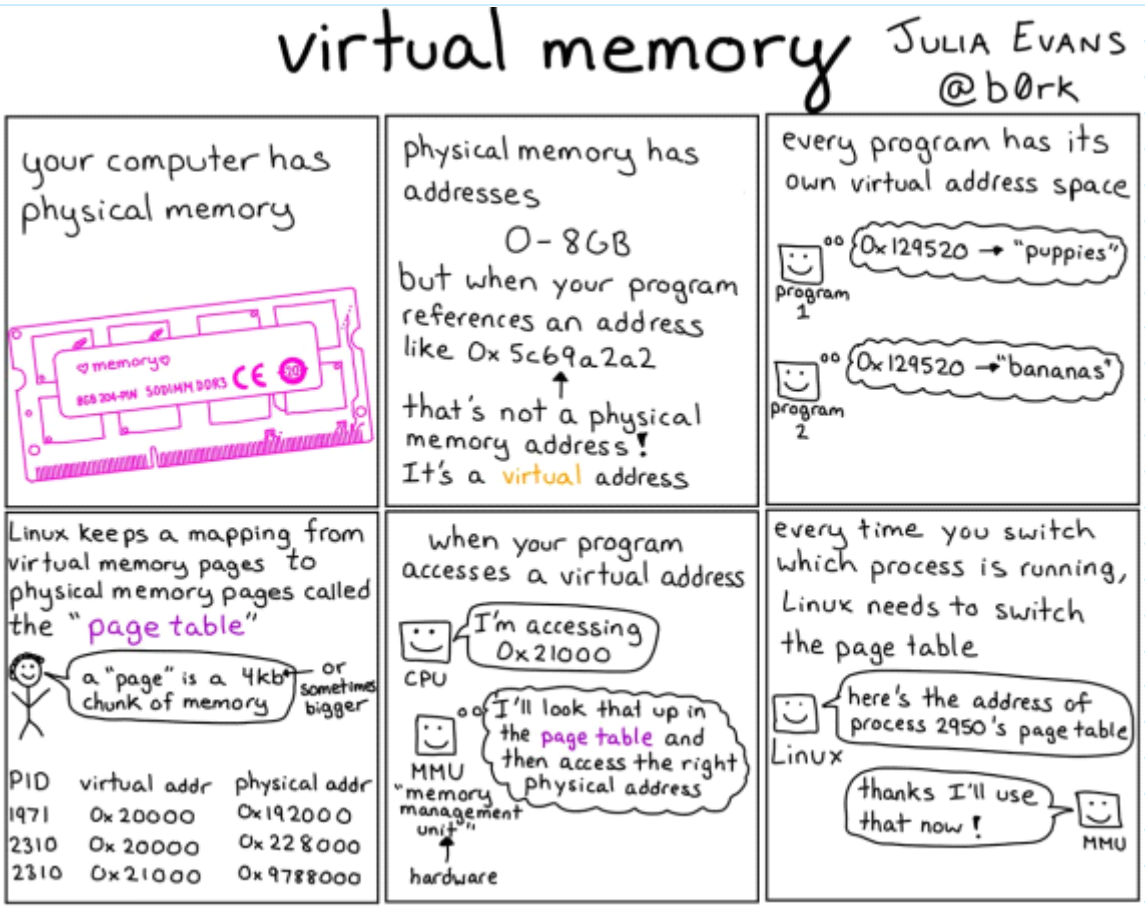
20 जून 2019 19:39



11. PAGE FAULTS | VIRTUAL MEMORY

20 जून 2019 19:41





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 –

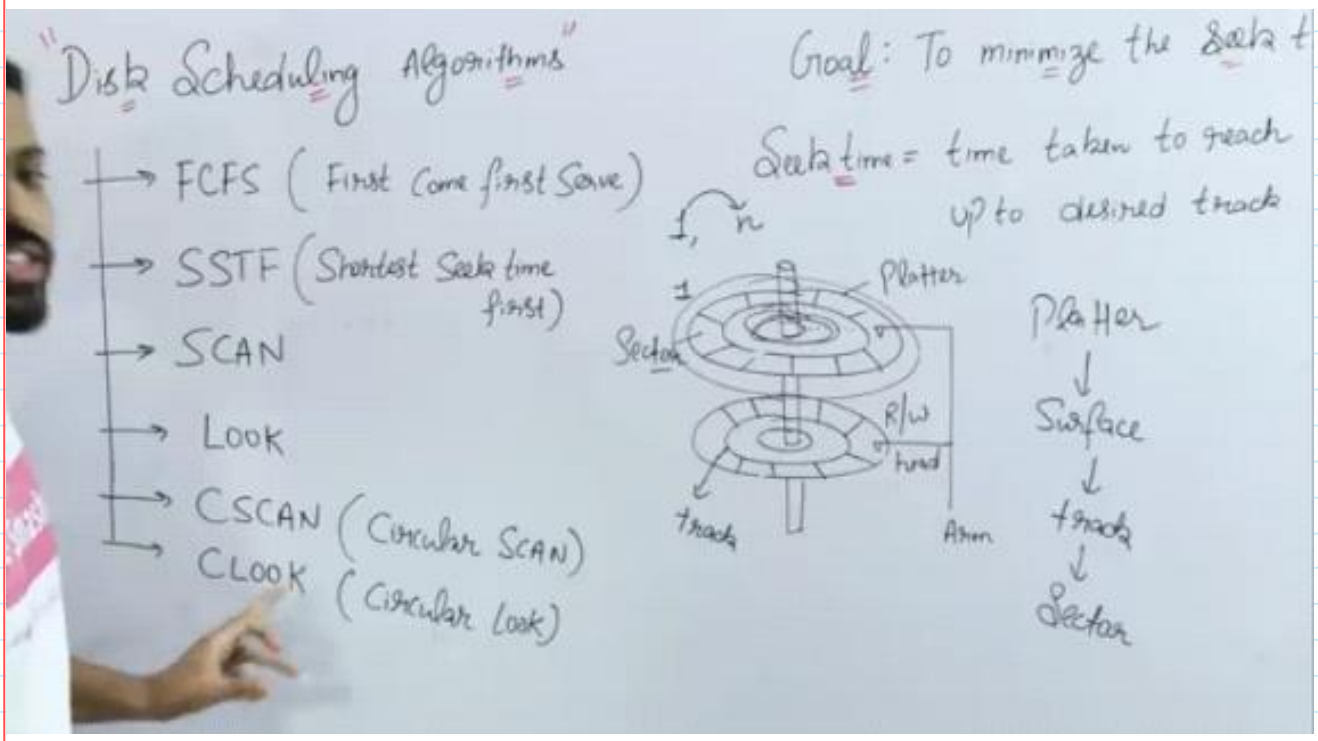
S.N.	Fragmentation & Description
1	**External fragmentation** Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used. External fragmentation can be reduced by **compaction or shuffle memory contents** to place all free memory together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

15. Unix /Linux/(Based General Ques)

20 जून 2019 19:46

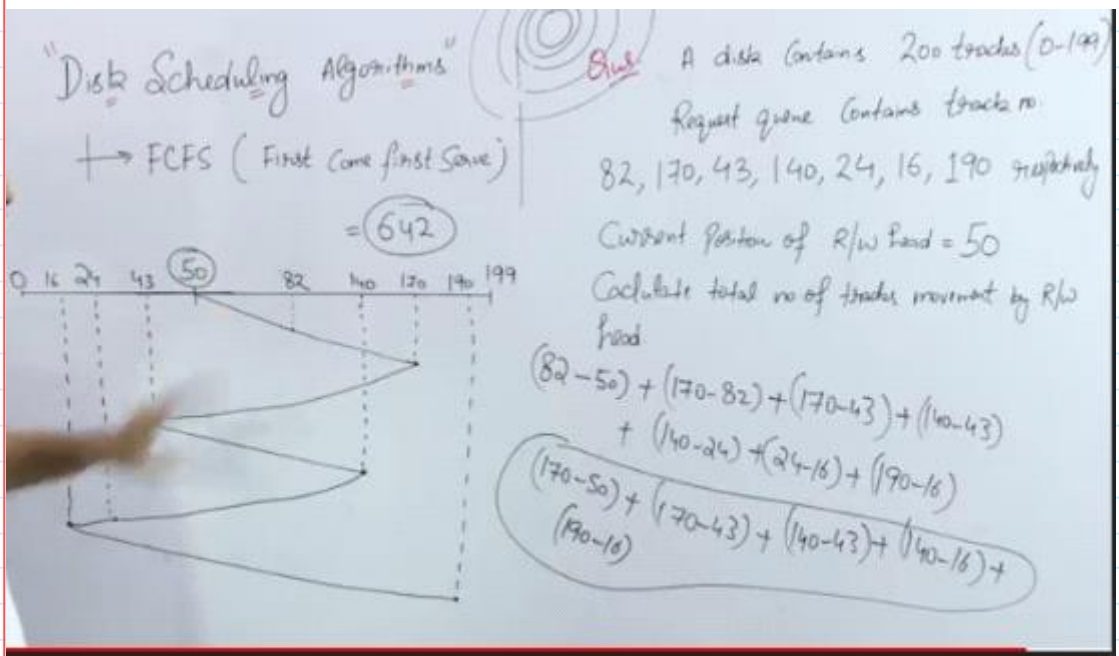
DISK SCHEDULING ALGO

24 जून 2019 15:43



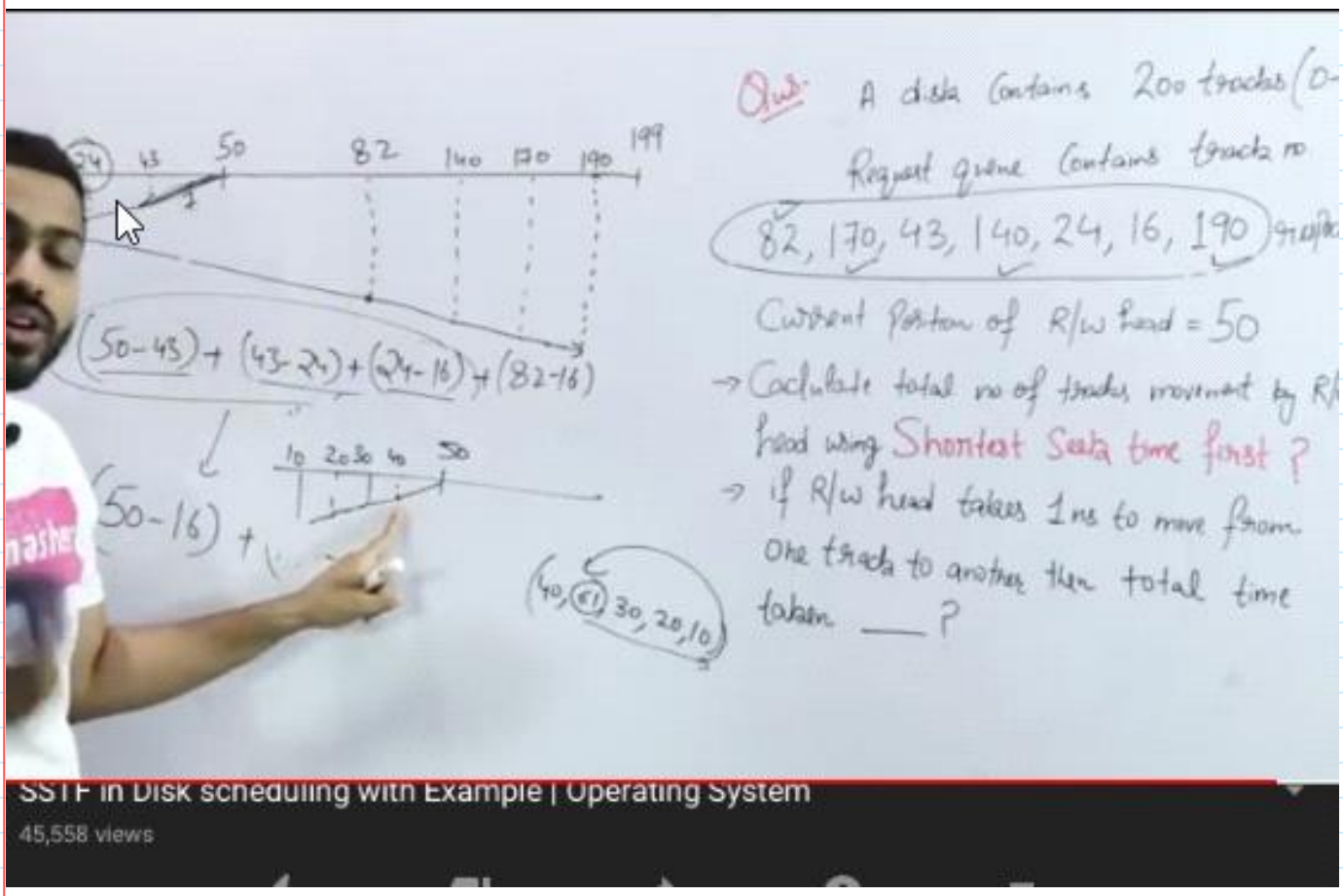
FIFO DISK

24 जून 2019 15:44



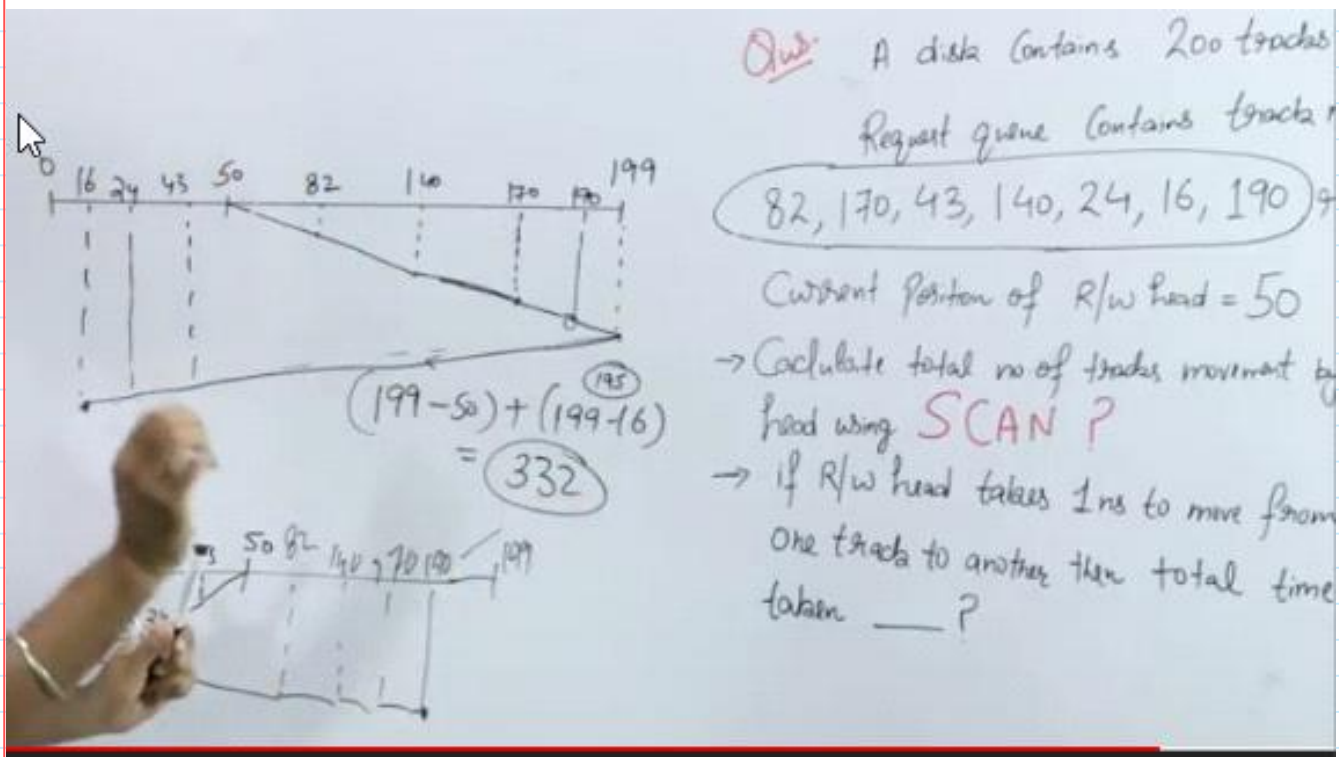
SSTF

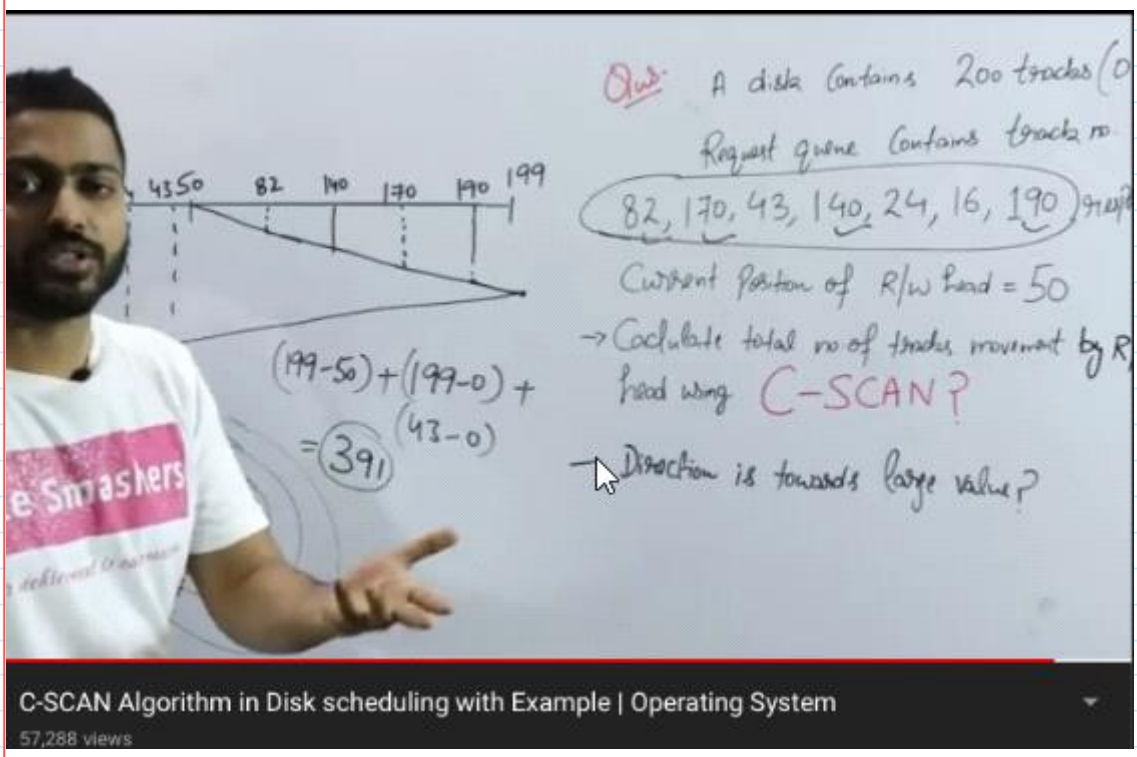
24 जून 2019 15:45



SCAN AND C-SCAN

24 जून 2019 15:45





LOOK AND C-LOOK

24 जून 2019 15:46