OS

Lecture Notes

December 24, 2021

Contents

Chapter 1

Intro

1.1 Contents

1. fundamentals
2. Process mgmt. gate imp.
   (a) Process concept
   (b) IPC
   (c) Deadlock gate imp.
3. Memory mgmt.
   (a) partition tech. gate imp.
   (b) virtual Memory gate imp.
   (c) page replacement algo.
4. File system
5. Secondry Memory gate imp.
   (a) logical
   (b) physical

1.2 Books

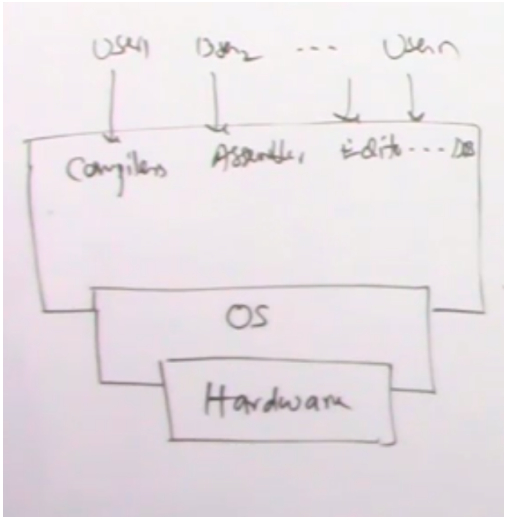
1. galvin
2. tanobann

weightage - 4-6 marks

Chapter 2

Fundamentals

- OS is an interface between user and computer system.
- Purpose of OS is to provide an environment in which user can execute the programs conveniently and efficiently.
- OS creates an abstract machine that provides an easy to use interface for executing and developing the applications.
- position of os in computer systems is as shown in layered view.



- OS is a resource manager (allocation and deallocation)

2.1 Resource and Components of OS

2.1.1 Resources

- There are two types of resource.
  1. Software eg: file, device driver.
  2. Hardware eg: cpu, I/O devices, cpu clocks.

2.1.2 Components

- There are three types of components.
  1. CU
     (a) used to generate timing control signals which intern executes μ-instructions.
  2. ALU
     (a) performs data manipulation operations.
  3. Memory
     (a) classified into two primary eg: RAM,ROM and secondary Memory eg: Hard drives.
     (b) Memory hierarchy and their managers

Components	Access time	Bandwidth(Mbps)	managed by
Registers	0.25 to 1 ns	20,000 to 100,000	compilers
cache	2 to 5 ns	5000 to 10000	hardware
main memory	20 to 100 ns	1000 to 5000	OS
Disk	5 to 10 ms	100 to 500	OS

- OS design principles depends on the architecture, whether it is most basic van–nueman or others like data flow, real time, multi processor, distributed etc.
- User cannot even interact directly with OS a second level of interface CI/CLI is needed.
  - this CI/CLI can be of two types CUI(character user interface) and GUI
  - also the programmer interacts with OS through API and SCI(system call interface)

Chapter 3

Functions and goals of OS

Goals of OS

1. Convenience :Primary goal is to make the computer system convenient to use by or convenient environment to users.

2. Efficient : the secondary goal of OS is to make Efficient use of the computer resources.

3. Reliability & robustness

4. scalability : The OS should be constructed in such a way as to permit the effective development introduction of new system functions without effecting the existing constitutes.

5. Portability : easily moveable to different platforms.

Functions of OS

1. programme execution

2. i/o Operation

3. File system manipulation

4. communication

3.1 Classifications of OS

The introduction of disk technology gave rise to operating systems.
initially there were two different types of OS

1. uniprogramming : OS capable of loading one program at a time into the main memory. CPU is ideal while loading and i/o operation the second programme.
2. multiprogramming/multitasking : OS capable of loading more than one program at a time into the main memory. thus resulting in maximum cpu or resource utilization.

modern classification of OS

1. Time sharing OS
2. Realtime OS : should be more responsive. Further classified as
   (a) hard Realtime systems eg : Air traffic
   (b) soft Realtime systems eg : ATM
3. Distributed OS : A software or collection of independent networked communicating by physically separated computational nodes. each individual node holds a specific software subset of the global aggregate OS.

multiprogramming OS are Further classified on the basis of user

1. Single user
2. multi user

multiprogramming OS can also be classified on the basis of preemption

1. non-preemption : process or program can release the CPU voluntarily either when it requires i/o or it finished executing or when a system call is generated. eg: windows 3.0 - 3.11.
2. preemption : forceful deallocation of the cpu from the process either for another high priority process or time quantum expire. eg: windows 7,8etc , linux etc.

Note : two or more programs can be executed simultaneously with more than one CPU.

Chapter 4

Design of OS

4.1 Introduction

To design a multiprogrammed OS the hardware requires the following

1. Address Translation : Logical Address generated by CPU must be translated into physical Address with the help of hardware called memory management unit implemented with page table or segment table.
2. DMA : used to transfer bulk amt. of data between memory and i/o devices without involving CPU.
3. At least 2 modes of CPU operation : The two modes are
   (a) user mode
   (b) kernel mode (super-user,atomic mode) : instruction executed here are not interrupted i.e executes without preemption.

while program compilation the compiler distinguishes between a user call and system call which leads to the switch to kernel mode.

Chapter 5

system call

They are extended instructions which provides interface between OS and user program.

5.1 Implementation

The following sequence of steps executes in implementing a system call

1. system call replaced with SVC (supervisory call) by the compiler.
2. SVC generates a software interrupt.
3. an interrupt service routine (ISR) is executed. the ISR does two things
   (a) changes the mode of processor to kernal mode.
   (b) lodes the address of required routine from dispatch table to program counter.
4. processor mode is reverted back after executing the routine.

5.2 Classification

system call are of following types

1. process management
2. memory management

3. file system management
4. Device management
5. communication related
6. protection related
7. information maintenance related

A system call may return a 0 ao positive value for success and negative value for failure.eg a fork process system call generally returns a 0 if it is root aor may return PID of its parent process if it is child on success else for failure it may return a negative value.

Chapter 6

Process Management

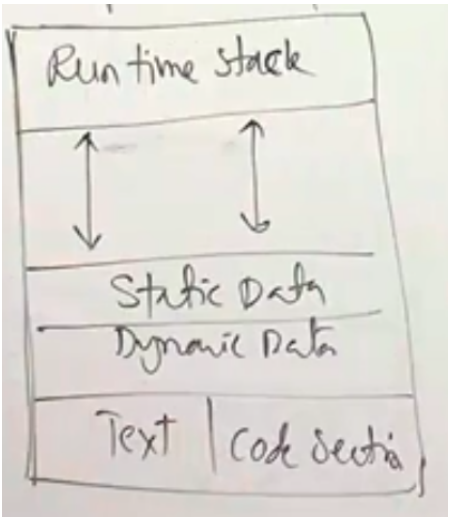
Program cs process :

Program is a finite set of statement,instruction, operation to solve a particular problem.

it is a passive entity

process is animated spirit or a program in/under execution.

6.0.1 Process representation



Runtime stack : it contains activation records.

6.0.2 Operations on process

- create() : resource allocation
- schedule() :
- run() :
- Block() :
- Resume() :
- Terminate() : resource deallocation
- suspend() : sending the process image into secondary memory.

6.0.3 Attributes of a process

- Cpu related
  - pid (process id)
  - ppid (process parent id)
  - pgid (process group id)
- memory related
  - limit register (starting and ending address of memory it is permited to access)
  - memory Management unit
  - size
- I/o related
  - list of i/o devices needed
- file related
  - list of file needed
- Access related
  - permissions required

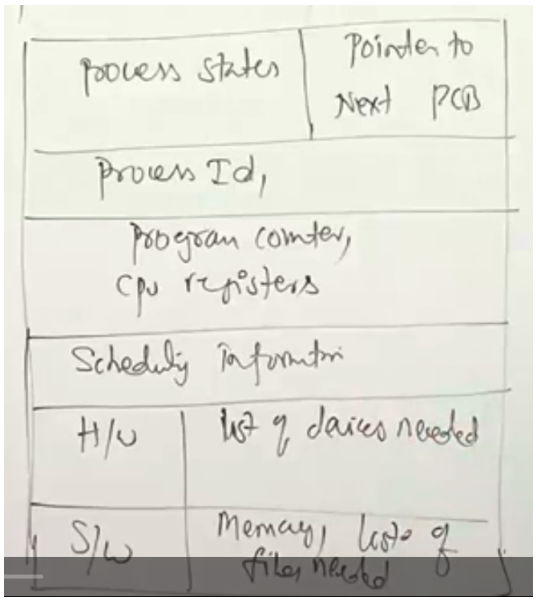
6.0.4 PCB

this list is kept in PCB/TCB (process/task control block), it is a kernal DS,it maintains the attributes of a process. It represents a process in OS. it is like ID card for process. Every process has its own PCB. it represents a process to OS.

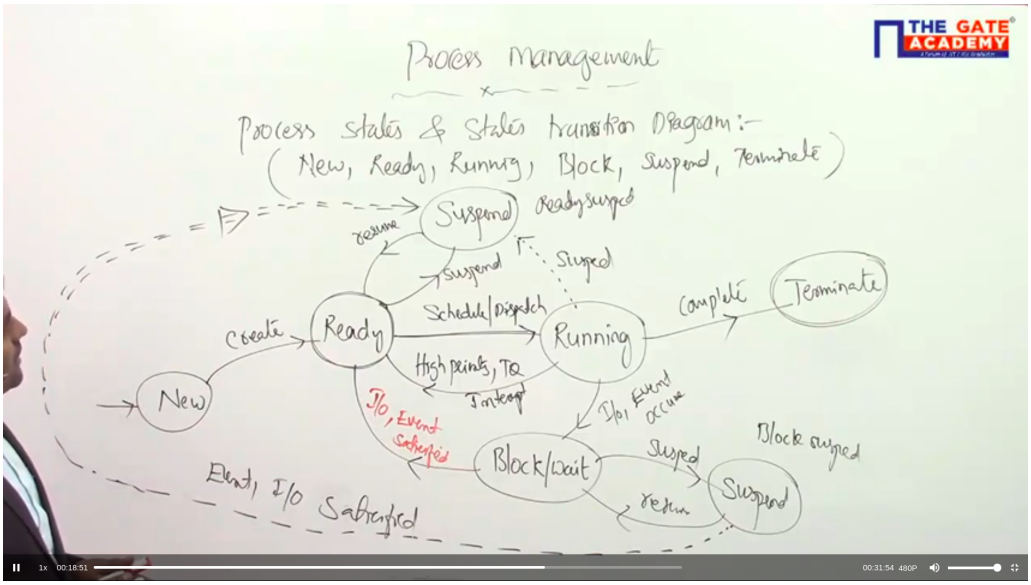
It contains the following

- process state
- program counter
- CPU registers
- CPU scheduling information{priority,time limit,scheduling queue}
- including the list above.

representation of PCB



6.1 Process state and state transition diagram



ready suspend is in secondary memory and **block suspend** is in primary memory.

6.2 Schedular/dispatcher

responsible for selecting the process to be executed. schedular is responsible for selecting process for scheduling or operating with the resource from the queue servicing the resource.

three kind of schedular

1. LTS (long term schedular)

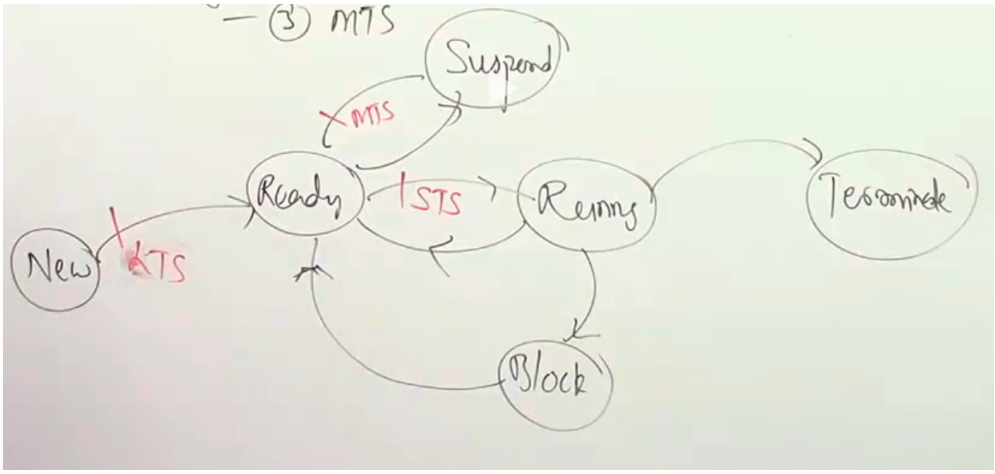
   - selects a process from disk and load them to main memory for execution
   - execution frequency of this is less then others.
   - has to select good mix of cpu and io bound process.
   - controls the degree of multi programing.
   - it keeps some balance between process creation and termination so as to keep the degree of multiprogramming stable avg. rate of process creation = avg. rate of process termination.

2. STS (short term schedular/cpu schedular)

   - selects the process from ready queue and assign to running state
   - more busy than LTS
   - it is invoked each time the cpu requires new process for execution.

3. MTS (medium term schedular)

4. - suspends the process to secondary memory.
   - it is used to decrease the load on the cpu.
   - it requires swaping (swapin ,swapout)



6.3 Dispatcher

it is a component of cpu schedular,its function is context switching.

6.3.1 Context Switching

It is task of switching the cpu from one process to another saving the state of hold process and loading the saved state od queued process.

it highly depends on hardware support
context switching/Dispatch latency is the time required for switching.
During CS processor is ideal.
two kinds of CS

1. partial CS loading PCB
2. full CS saving old process and loading new PCB

cases where these two are used

- process completed (PCS)
- need i/o (FCS)
- Timed out (FCS)

6.4 CPU scheduling

when there are more than one process ready to be executed with the CPU a selection decision needs to be made to pick a process for execution from among the ready processes this activity is called process or CPU scheduling.

Scheduling the fundamental function of OS every resource is scheduled before use

6.4.1 Objective of CPU scheduling

1. fairness
2. maximize the CPU utilization
3. maximize throughput
4. minimize the response time
5. minimize the turnaround time
6. minimize the waiting time

throughput : it measure the work being done it no. of process/job/task completed per unit time.

CPU utilization : how busy the processor is

turnaround time : time between the submission time of process to its completion time.

Response time : time between process submission of request to response is generated.

6.5 Process Time

Arrival time :
Submission time :
Bus/service time :

1. i/o bus time
2. cpu bud time

waiting time :
completion time :
Turnaround time :
Decline time :
Scheduling length :
Queue :

- Job q
- ready q
- Device q
- Event q
- suspend q

6.5.1 Notaions

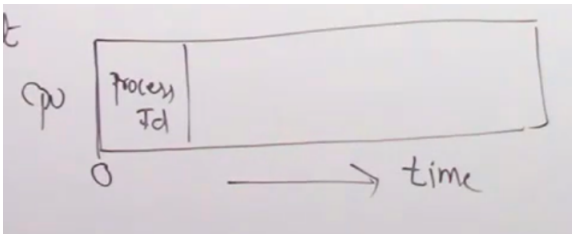
- $AT(p_i) = A_i$
- $BT(p_i) = X_i$
- $WT(p_i) = W_i = TAT_i - X_i$
- $CT(p_i) = C_i$
- $ST(p_i) = S_i$
- $TAT(p_i) = C_i - A_i$
- $Avg.WT = \frac{1}{n}\sum_{i=1}^{n} TAT_i - X_i$
- $Avg.TAT = \frac{1}{n}\sum_{i=1}^{n} C_i - A_i$
- $Schedulelength(SL) = max(CT_i) - min(A_i)$
- $throughput = \frac{n}{SL}$

6.6 CPU Scheduling Algorithms

1. FCFS
2. SJF
3. SRTF
4. priority Scheduling Algorithms
5. HRRN
6. Round Robin (used in time sharing)

6.6.1 FCFS

criteria : based on arrival time mode:non-preemption Implementation: fill RQ with FIFO Q uses Gantt chart



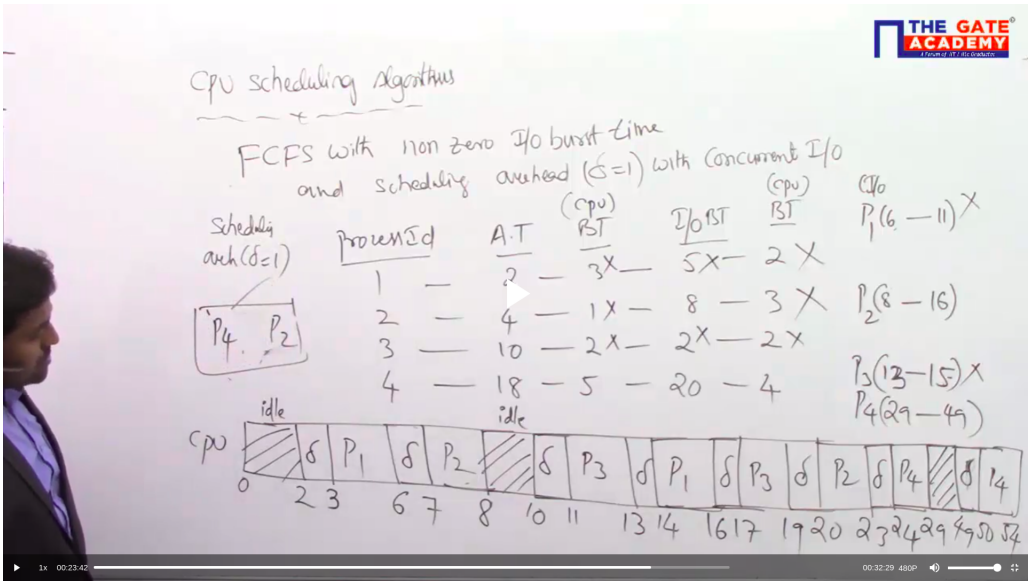
6.6.2 FCFS with non zero io burst

scheduling overhead $\delta = 1$ with concurrent I/o

scheduling overhead - time required to schedule.

i/o burst - time require to service io

see lecture sec 2 6/10

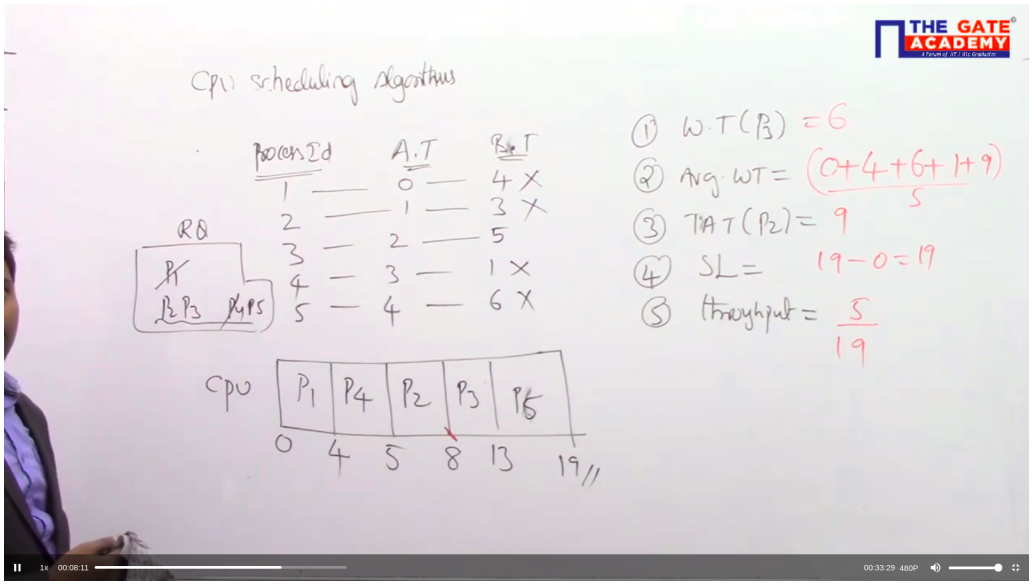


6.7 Shortest Job First or Shortest process next

criteria : based on burst time

mode : non preemption

allocate the cpu for process with smallest CPU burst. with process with 2 cpu burst FCFS is used to break the tie.



6.8 Shortest remaining time first

criteria : based on burst time

mode : preemption

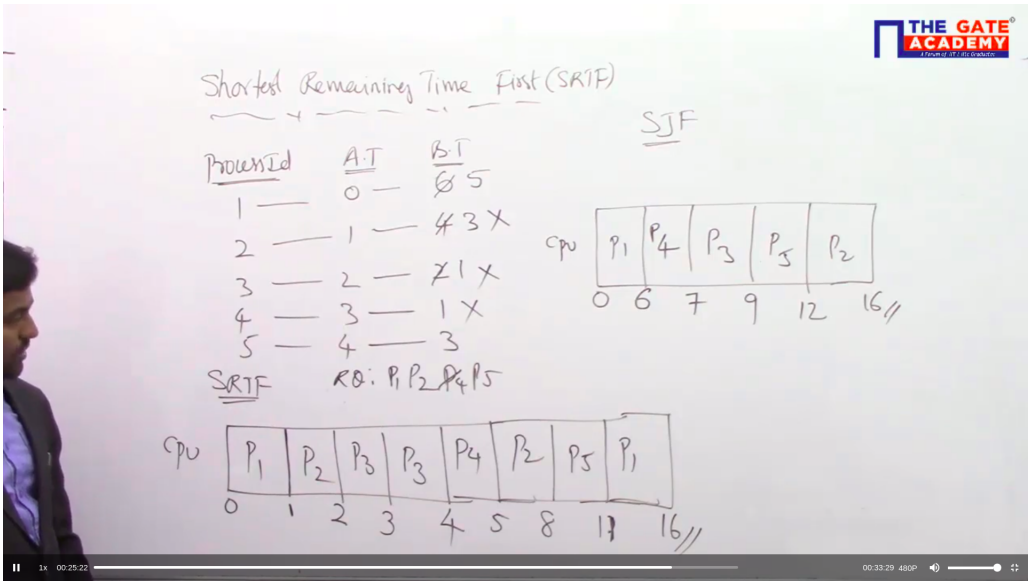
it is SJF with preemption. preempt currently executing process if newly arrived process has shortest CPU burst time in comparison to executing one preempted process is kept in front of the ready queue.

Advantages

- fairer to shorter jobs.
- increases throughput of system.
- it minimizes turn around time,waiting time.
- this is the optimal algorithm.

limitations

- it starves the longer jobs
- knowing the length of the next CPU burst time is difficult hence not practical.
- it is used a benchmark to mesure the performance of other algorithms.



6.8.1 Predicted techniques of burst time of a process

SJF can be implemented by predicting burst time as shown -

let p_i be a process t_i be the original burst time and T_i be the predicted burst time.

prediction based on-

- Size

  if p1 of size 130kb takes 30ms then p2 of size 128kb may take 30ms

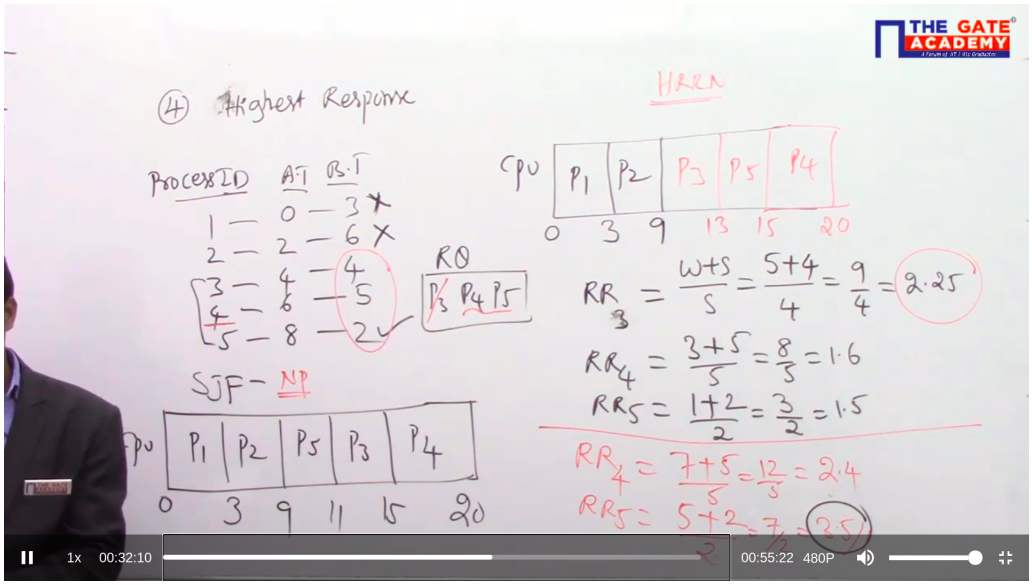
- types
  - OS process (5-10)
  - user interactive process (10-15)
  - foreground process (20-30)
  - background process (30-50)
- new burst time average of previous burst times of that process.
- Exponential averaging $T_{n+1} = \alpha t_n + (1 - \alpha)T_n$ where $0 < \alpha < 1$

6.9 Highest Response Ratio Next

It is not only favours the shorter jobs but also limits the waiting time of longer jobs.

criteria : response ratio(RR)

$RR = \frac{w+s}{s}$ w= waiting time s = burst time



6.10 Priority Based Scheduling Algorithm

criteria: based on priority (integer value)

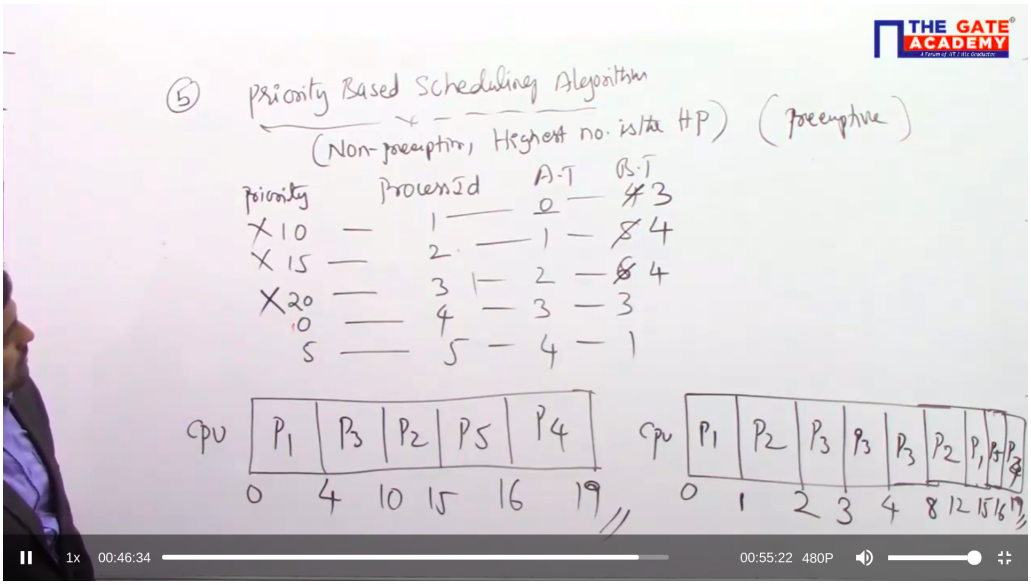
mode : non-preemptive/preemptive

implementation : priority queue

Process are kept in the ready queue in the order of their priority values irrespective of arrival time.

FCFS is used to break the tie.

- once priority is assigned it cannot change during execution.
- lower priority may get infinite block or starved
- to prevent this problem aging mechanisms(dynamically increasing the priority of a process with time) is used

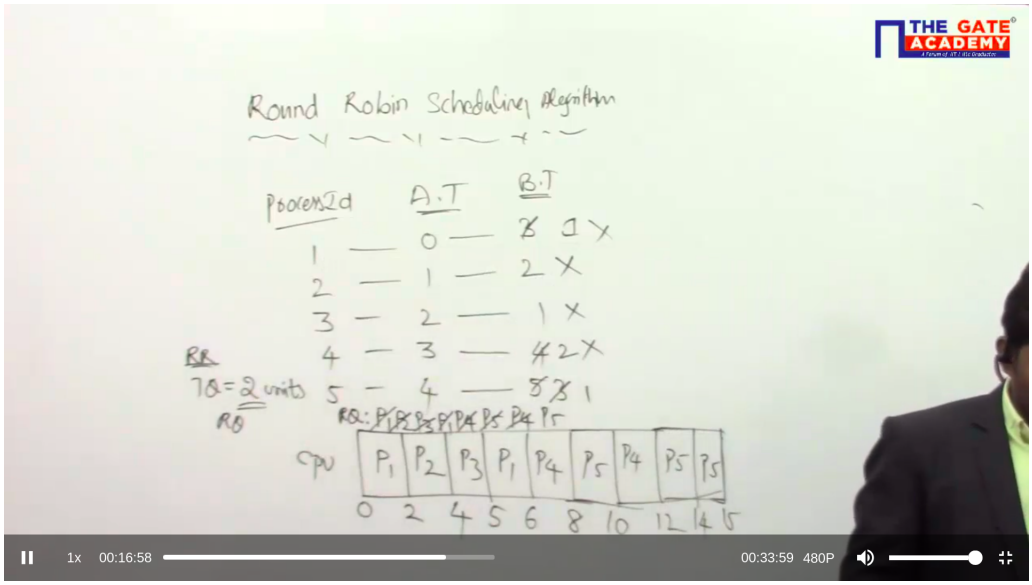


6.11 Round Robin

used in time sharing system

criteria: based on AT + time quantum/time slice
implementation: RQ = circular queue
round robin is a FCFS with preemption

- give n process in rr and time quantum q each process get 1/n of cpu time in time quantum q, each process waits not more than (n-1)q time units.
- turn around time depends on time quantum.
- performance of rr depends on time quantum.
- if q is small it is less efficient.
- if q is mid max context switches.
- if q is large less context switches.
- if q is very large it becomes FCFS.


THE GATE ACADEMY
Round Robin Scheduling Algorithm
Process Id
A.T
B.T
1 — 0 — 8 1 X
2 — 1 — 2 X
3 — 2 — 1 X
4 — 3 — 4 2 X
5 — 4 — 5 3 1
RR
TQ = 2 units
RQ
RQ: P1 P2 P3 P1 P4 P5 P4 P5
CPU
P1 P2 P3 P1 P4 P5 P4 P5 P5
0 2 4 5 6 8 10 12 14 15
1x
00:16:58
00:33:59
480P

Chapter 7

Inter Process Communication & synchronization

Purpose of IPC

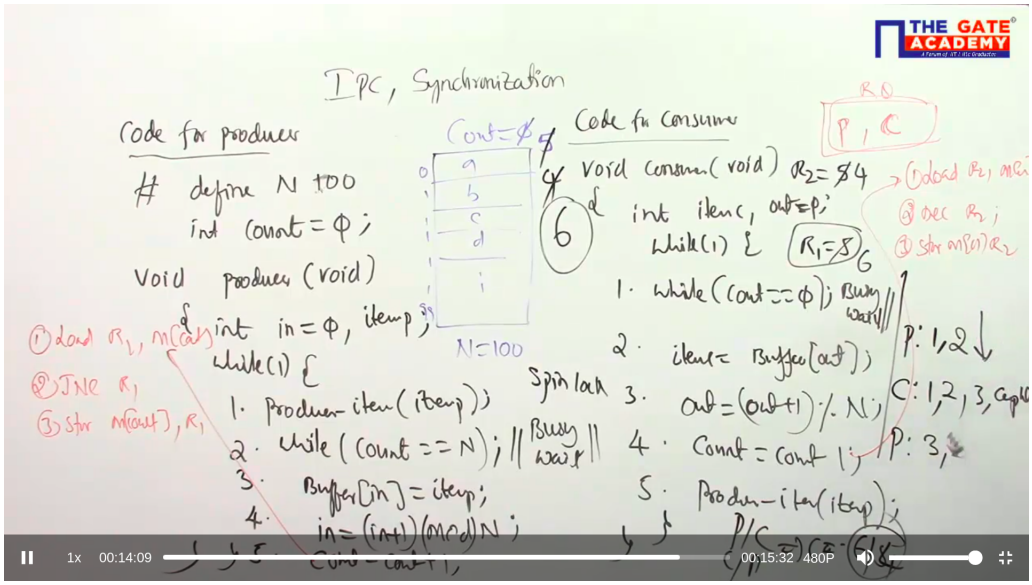
1. Data sharing
2. Data transfer
3. Resource sharing

to establish communication between any entity we need media.
Lack of synchronization

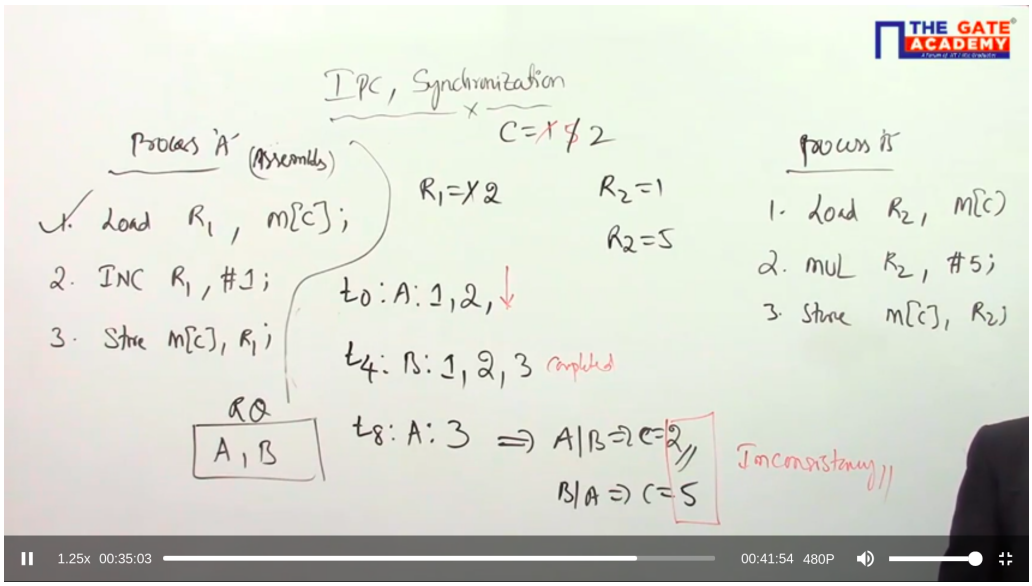
- misunderstanding (wrong results)
- Loss of data
- Deadlock

types of synchronization

1. cooperating sync process Two or more process are said to be in cooperating synchronization if and only if they get effected by each other eg: producer consumer problem



2. competing sync process two are more process are said to be under competing synchronization if and only if they compete to access the resource simultaneously. eg : let A to process A and B try to access shared variable c with initial value 1 A wants to increment c by 1 B want to multiply c by 5. the final value should be either 6 or 10 but



Necessary condition for synchronization problem

- critical section : portion of programme where shared resources are accessed. non critical section is the portion of the program that does not access any shared resources.
- Race condition : Process must be racing to access critical section simultaneously.
- preemption : scheduling algorithm must be preemptive.

7.0.1 Requirements of synchronization mechanisms

- Primary
  - mutual exclusion : no two process may be present in the critical section in same time
  - progress : no process running outside critical section should interrupt any other process to access critical section if it is free.
- Secondary
  - bounded wait : no process should wait forever to enter critical section
  - architectural neutral

7.1 Synchronization Mechanisms

- mutual exclusion with busy wait (spin lock)
- mutual exclusion without busy wait (blocking)

this two provides solution for both user and kerna

- Disabling interrupts -busy wait
- lock variable -busy wait
- strict alternative -busy wait

- peterson solution -busy wait
- TSL instruction -busy wait
- sleep and wake up -blocking
- seamaphore -blocking
- manitoss -oops

7.1.1 Disabling inturrepts

this mechanism disables inturrepts just before entring the critical section and then enable it back after exiting it.

used for kernal mode

7.1.2 Lock variable

mutual exclusion with busy wait

solution for user mode

used for multi-process system

here a lock variable is used when it is 0 that means CS is free and when it is 1 CS is busy.

does not guarantee mutual exclusion

7.1.3 strict alternative

mutual exclusion with busy wait

software solution (user mode)

2-process solution

maintains a variable called turn if turn is 0 p0 will access if turn is 1 p1 will access and only p0 will change turn to 1 and p1 will change turn 2 0.

progress is not guaranteed here

7.1.4 Peterson solution

software solution at user mode

2-process solution

mutual exclusion with busy wait

it takes into account both whose turn it is and who is interested
guarantees progress and bounded wait

7.1.5 TSL instruction/ Test Set lock instruction

mutual exclusion with busy wait
multi-process solution
implementable in user mode with hardware
TSL instruction is executed in kernel mode
here the setting of lock variable takes place in kernel mode so no preemption.
guarantees progress and bounded wait
when ever a CS algorithims comes in conflict with scheuling algorithm there is a deadlock eg: a low priority process p_l is in CS and the scheduling algorithm tries to preempt it to run high priority process p_h, even if p_l preempts p_h does nothing but wait in CS.

7.1.6 sleep and wake

blocking mechanism
here we suspend the process which will wait in above mechanisms and the later wake it up this saves cpu time.
this creates deadlock for eg. in producer consumer problem if consumer starts then both of the process may sleep simultaneously.

7.1.7 Semaphores

Dijkstra listed using an integer to count the number of wakeups saved for future use it is non busy wait constraint blocking mechanism.
mutualy exculsion with blocking
multiple process
user mode application
Semaphore is variable (type of semaphore itself)
with the property that it takes an integer value c
two kinds of semaphore

- primary (takes 0,1)
- counting (takes from $-\infty to + \infty$)

operations on Semaphore

- Down/P/wait decrementing Semaphore if after performing down operation if the value of Semaphore is positive then it i successfull operation other wise unsuccessful those PCB for which the operation id unsuccessful is kept in queuetypt L.
- Up/V/Signal incrementing Semaphore if after performing up operation if the value of Semaphore is ¿=0 then it selects a process from queue and wakes it up directly to CS.

Semaphores are OS resources implemented in kernel mode.
binary Semaphores is similar to mutex lock

7.1.8 Counting semaphore

Struct Semaphore{ int value; //current value of Semaphore
queuetype L; // list of PCB of those process that get blocked while performing down operations. }

the negative value of semaphore indicates those many no. of process are blocked.

posetive value of counting Semaphore indicates those many down operation can be carried out.

7.1.9 Binary semaphore

operations on Semaphore

- Down/P/wait

  if(val==1){val=0;return}

  else{put this process into s.l and block it}
- Up/V/Signal

  if(s.l is empty){s.val=1;}

  else{select a process from s.l and wake up}

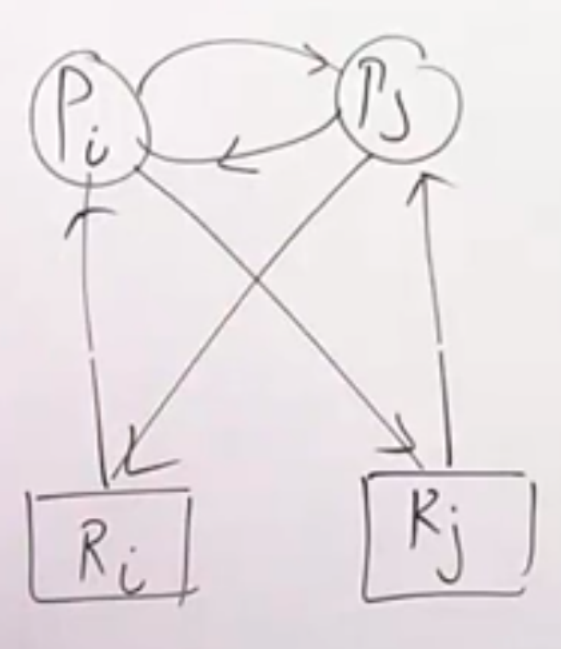
it is not possible to know how many blocked process are there with binary semaphore possible with counting semaphore.

if there is a process in CS or if there are blocked process in queue then binary semaphore has to be 0.

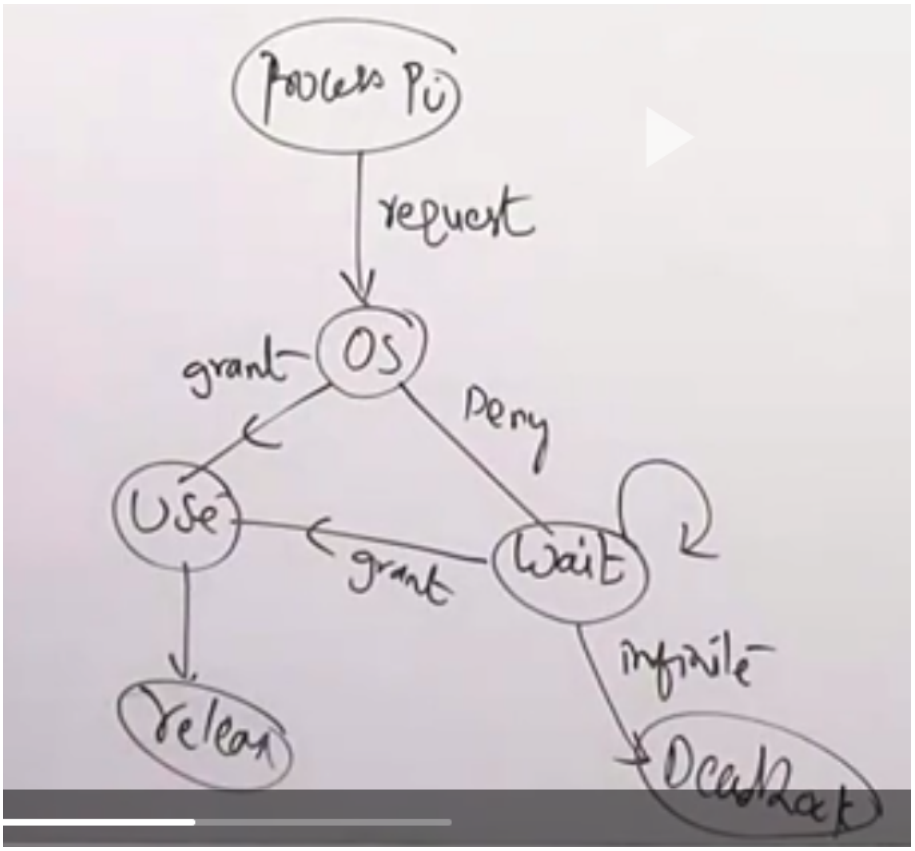
Chapter 8

deadlock

when a process p_1 is holding one resource and requesting other resource held by process p_2 and p_2 is intern requesting a resource held by the process p_1 this situation creates a deadlock.



a set of blocked process each holding resource and waiting to aquire resource held by another process in that set or permanent blocking of set of processes that either compete for system resource or communicating with each other.

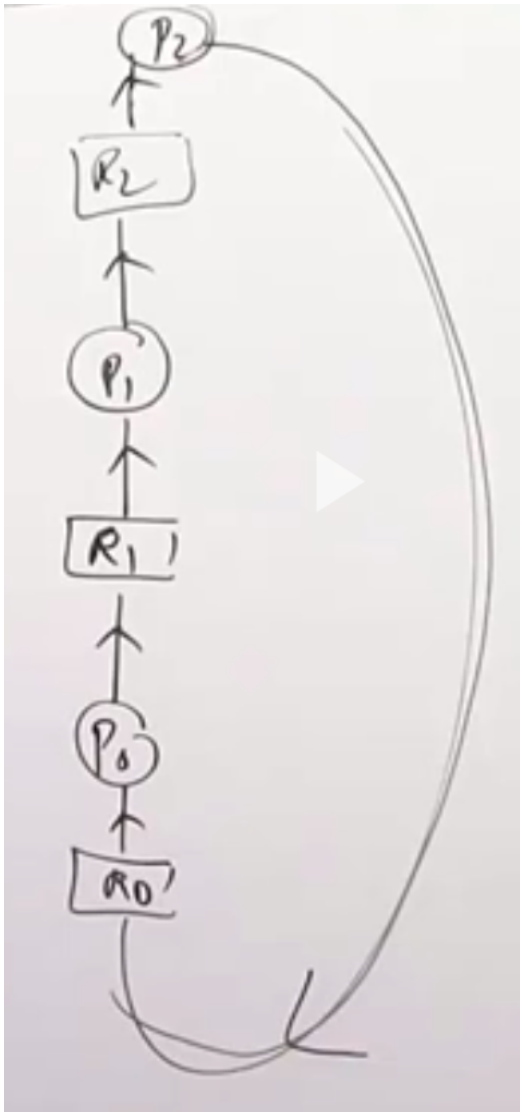


8.0.1 Necessary conditions for deadlock

- mutual exclusion of resource

- hold and wait : if the process holding atleast one resource and is waiting to aquire additional resource held by another process. every hold and wait does not result in deadlock.

- circular wait : there exist a closed chain of processes such that each process holds atleast one resource needed by next process in chain.



- no preemption of resource : a resource can only released only by process holding it.

if all the above conditions occur simultaneously then deadlock occurs.

8.0.2 Resource allocation graph

it depicts the state of system of resources and processes.

the vertices of the graph are either resource in square or process in circle.

resources are two types

single instance (a dot in square)

multiple instance (multiple dots in a square)

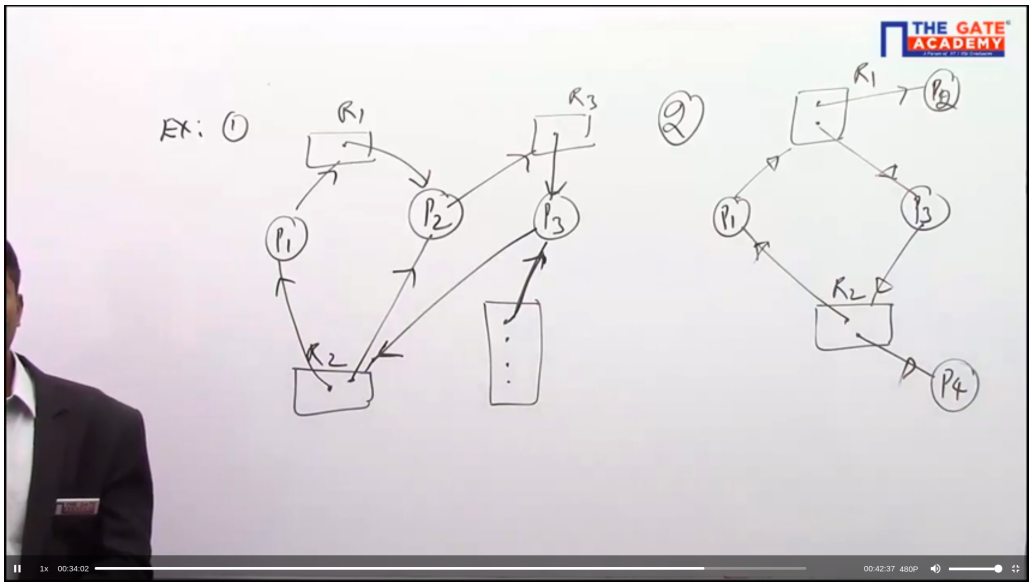
Edges are of three types

claim (dotted arrow)

request (solid arrow from process to resource)

assigned (solid arrow from resource to process)

examples



imp.points

- if graph contains no cycle no deadlock
- else
  - if there is only one instance/process (Necessary sufficien condn.)
  - if there are several instances/ resource then there is a possibility of deadlock.

8.1 Methods for handling deadlock

four Methods two avoid deadlock

- prevention (ensure that the system will never enter deadlock)
- avoidance (check each time a process request for resource for possible deadlock)
- recovery (allow the system to enter in deadlock and then recover it)
- ignorance (ignore the problem)

8.1.1 Prevention

Dis-satisfies one of four condition of deadlock

- mutual exclusion (it will never dis-satisfy this)
- hold and wait (do not request a resource by holding a resource, a new request can only be made by releasing all held resource)
- no-preemption (allow preemption)
- circular wait (all the resources uniquely identified never allow a process to request a lower member resource than the last one requested)

8.1.2 Avoidance

it is the simplest and most useful model it need prior information about process request it works based on the concept of system state (safe Unsafe) it require that each process maximum no. of resource of each type.

Safe state: a state is safe if the system can allocate resources to each (upto its maximum) in some order and still avoid a deadlock or if their exist a safe sequence for bankers algorithm if a system is in safe state means no deadlock if a system is unsafe means there is a possibility of deadlock

deadlock avoidance Methods ensure that system will never enter deadlock
resources allocation graph is used for single instances
bankers algorithm used for multiple instances.

8.1.3 Recource Allocation graph

- a graph is constructed where the nodes are recources (square) and processes(circle)
- **request edge** : a solid edge from process to recource signifieng that the process requests that resource.
- **Allocated edge** : a solid edge from recource to process signifieng that the resource is allocated to that process.
- **claim edge** : a dashed edge from process to resource signifieng that a proces will request for that resource at some future point.
- if their is a cycle in such a graph then the system is in usafe state.

8.1.4 Bankers algorithm

got through lecture 3/5

Chapter 9

Memory Management

9.1 Intro

the ability to load multiple programs in main memory is called multi programming

OS is responsible for loading programmes, allocating and deallocating main memory etc.

9.2 Memory management techniques

9.2.1 Functions of memory manager

- Allocation
- protection
- Address translation
- Free space management
- de Allocation

9.2.2 Goals of memory manager

- efficient utilization of memory space (minimizing the wastage of resource)
- run larger programmes with small memory

9.3 Memory management techniques

Two types

- Contagious Allocation implemented with
  - fixed partition
  - variable partition
- Non-Contagious Allocation implemented with
  - simple paging
  - segmentation
  - segmented paging
  - virtual mem. paging

9.4 Partition

main memory is divided into two partitions

- os partitions
- user space partition- which is further division takes place through
  - fixed partition
  - variable partition

9.4.1 Fixed Partition

used in multiprogramming with fixed no. of takes

user space is divided into equal or unequal partitions statically not at run time

partition isto programm is 1:1 (no process interfears each other)

partition is not shareable

each partition may have it on scheduling queue or one single queue may be used for all partitions

os has some thing called partition controll block

os gets a request from programm with its size and the os then allocates it as per various Allocation methods

- first fit select: first sufficient empty partition
- best fit: a selection which minimizes space wastage
- worst fit: selects largest partition allways
- next fit: selects the next sufficient available no need for searching

after allocating memory here there may be a lot of space wasted here and there this is called **Internal fragmentation**.

9.5 performance

- Internal fragmentation present
- External fragmentation not poresent
- Ristricted degree of multi programing
- no. of process $\propto$ no. of partition
- maximum process size = partition size

9.6 Variable Partition

A process is allocated memory as much as it requires. the no. and size of partition is variable.

if enough memory is available for a process but is scattered in different portions it is called **External fragmentation**

9.7 solving External Fragmentation

The problem of external fragmentation can be solved in two ways

- compaction
- Non-contagious memory allocation

9.7.1 Paging

paging is implemented by organizing the following

- organisation of logical address space
- organisation of physical address space
- memory management unit
- translation of logical address to physical address

9.7.2 organisation of logical address space

logical address is divided into equal no of page

so that a particular program may take many pages to accomodate

logicall address is divided into page index and offset so first 3 or 4 bits represent page and rest represent offset

9.7.3 organisation of physical address space

the physical address space is divided into equal sized frames

the frame size is kept equal to page size from logical address

physical address is divided into frame index and offset so first 3 or 4 bits represent frame and rest represent offset

9.7.4 memory management unit

the MMU is called page table or page translation table

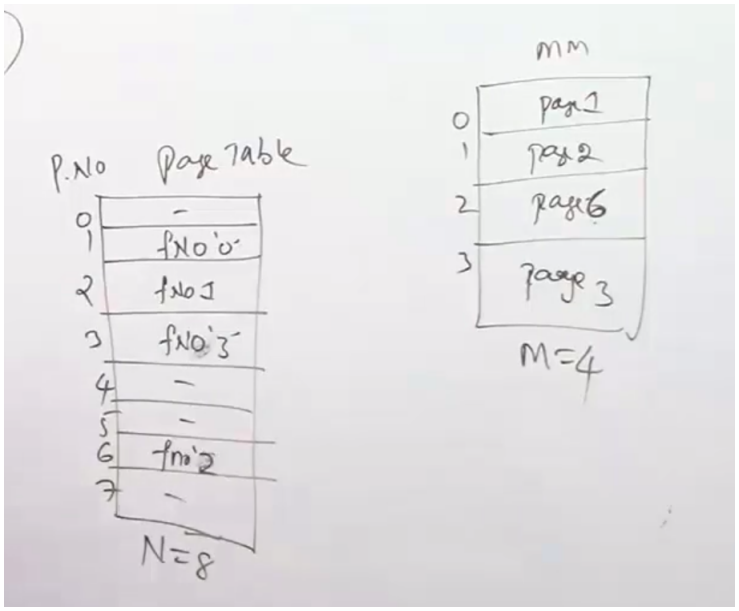
each process is associate its own page table which wre kept in main memory

page table organised into series of entries where no. of entries in page table = no. of entries in logical address space(page table size)

page table entro contains essentially frame no. of physical address space in which the page referred is present

page table size is given by n*e bytes e= page table entry & n= no. of pages

page table of a process is not shareable to other processes



9.7.5 Address translation (LA tp PA)

- CPU generate logical address
- the page part of the address is translated to 2 bit frame part of physical address through page table
- the offset of logical address is kept same as of physical address

9.7.6 Simple Paging

if program contains n pages then before the start of its execution all the n pages are loded into n frames of main Memory

performance

- temporal : the time to access the main memory becomes twice because of read time for page table but introduction of a cache system which contains the table of logical vs physical address called translation look aside buffer (TLB) we can reduce the time to be almost same as main memory access time bcoz read time for cache is insignificant. hence the efective memory acces time **EMAT=x(c+m)+(1-x)(c+2m)** x= hit rate of TLB c= acces time of TLB m= acces time Main Memory **overhead with TLB = xc+(1-x)(c+m)**

- spatial :
  - internal frag. : to decrease the IF we need to decrease page size but it increases the page table size
  - page table size : to decrease page table size we need to increase page size but it increases internal frag.
  - so we neet to find an optimal soln $\mathbf{p}=\sqrt{2Se}$ s= LAS.

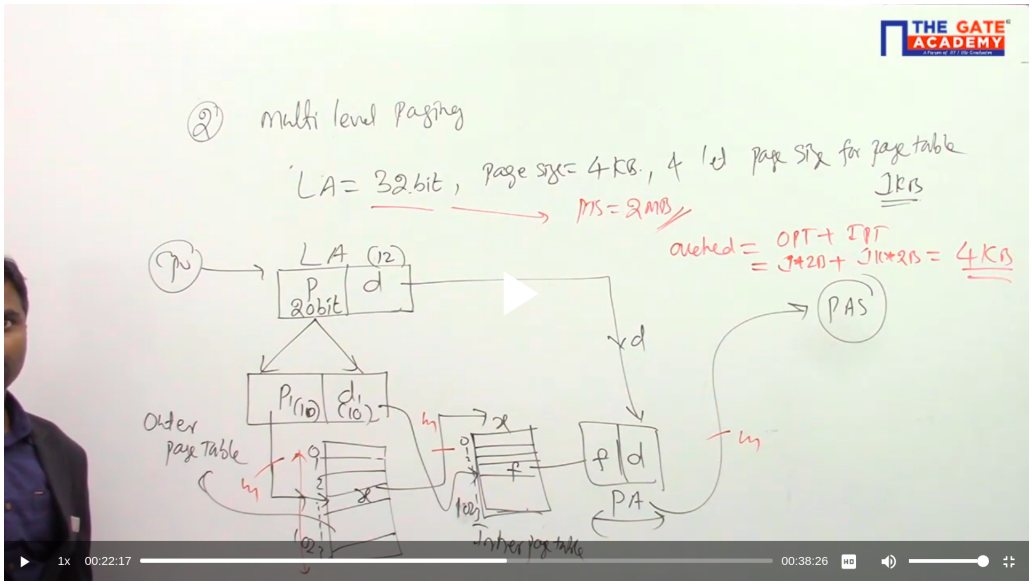
  solution of the overheads are to set an optimal page size and multi level paging i.e applying paging on page table (lecture 7)

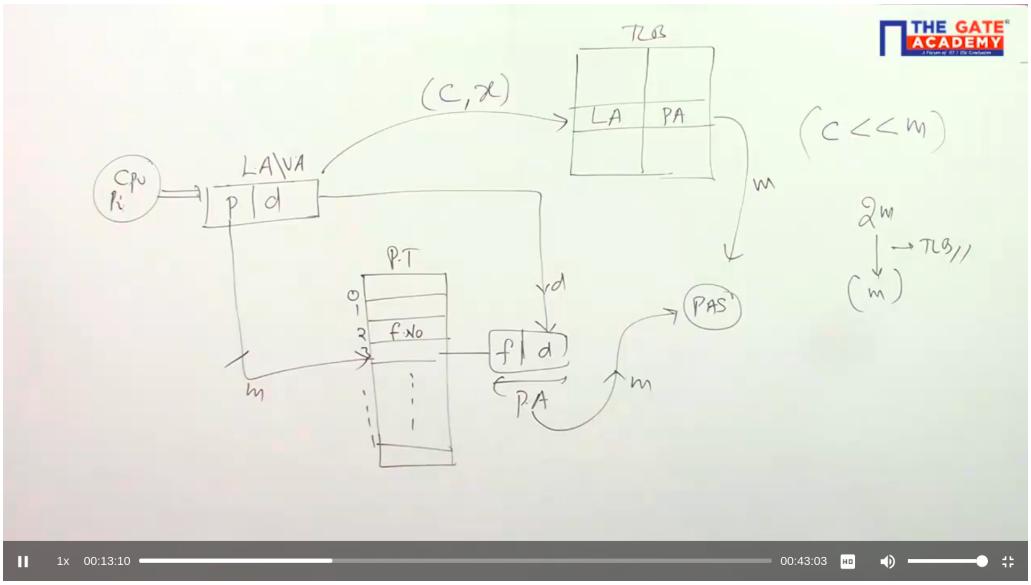
9.8 Multi-level Paging

it is paging applied on page table

all the inner page table is kept in secondary storage only the outer page table is to be kept in maim Memory

while accesing a location one of the total page table is loaded into main memory





9.9 Translation look-aside buffer TLB

to reduce the effective acces time of mrmory TLB is introduced

effective memory access time = hit rate of TLB(tlb acces time + memory acces time)+miss rate of TLB(tlb acces time + 2*memory acces time)

9.10 Segmentation

paging does not preserve user view of memory allocation to the programme as per user view, programme is divided into logical checks that represents functions or procedures or data structures known as segments

this segments which may be different sizes which are stored physical address space

memory reference consist of segment no. and offset.

segmentation preserve the user view of the programme

advantage of segmentation is it simplifies the handling of growing data structure

it allows the programme to alter and recombine independently

similar to page table in paging we maintain a segment table in segmentation this table stores base vs limit.

the problem of internal fragmentation is solved but externa fragmentation occurs

external fragmentation can be overcome by segmented-paging (i.e each segment is then divided into pages) but it has internal frag.

9.11 Segmented Paging

user address space is broken up into no. of segments at the discreetion of programmer each segment in tern is broken up into no. of fixed size pages. which are equal in length to main memory frames.

from the programmar point of view logical address still consist of segment no. and segment offset but from the system point of view segment offset is viewed as page no. and page offset for a page within a specified segment

9.12 Virtual Memory

virtual memory gives an illusion to the programmer that huge amount of memory is available for executing programmes greater than the size of available physical memory

OS helps us to swap in and swap out of programme during execution

virtual memory is implemented using demand paging

memory refrences are dynamically translated into physical addresses at run time

a process may be proken up into pieces(pages) that do not need to locate contagiously in main memory

all pieces or pages of a process do not need to be loade into main memory at a time that is pages of a process are loaded into main memory from the secondary memory on demand basis is known as demand paging.

9.12.1 advantages

provides faster response

needs less memory

supports more no. of process

it improves the cpu utilization and through put

9.12.2 Performance of virtual memory

- mainmemory acces time m
- page fault service time s
- page fault rate P
- hit rate (1-p)
- effective memory acces time = (1-p)m+ps

9.12.3 Page replacement algorithms

in case when no free frame to bring requested bage we use replacement algo

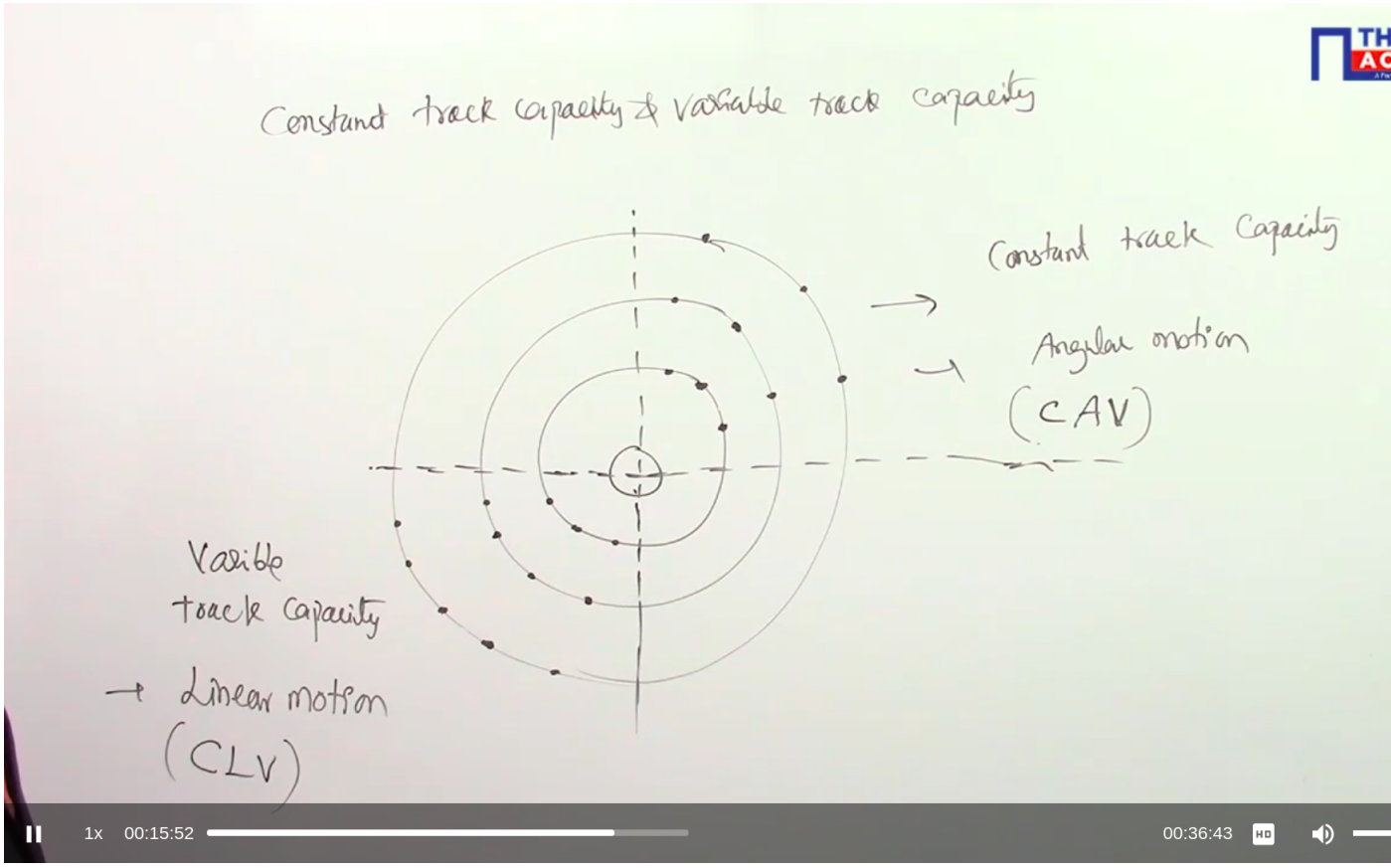
- bring requested page from disk
- find empty frame and allocate it
- if no empty frame in memory
  - select a page frame to replace it into secondary memory by replacement algo
- load the requested page in main memory
- update the page table

Page reference string : set of succesively and uniquely pages reffered in the given list of logical address space

9.12.4 Replacement algo

- FIFO: belody's anamoly with increased no of frames to proces the page fault raye logically should decrease but some reference string with increase in no. of frames page fault also increase only fifo based algo suffers
- Optimal replacement : replace the page that will not used for longest period of time : practically not immplementable future info required

- LRU (least recently used):
- counting algorithms
  - least frequently used
  - most frequently used



9.13 Secondary memory (magnatic disk)

a disk is semi random access memory where the placing the r/w head on the track is random access and getting the desired sector of a trach is serial access

information is arranged in the disk in the form of tracks and sector.
the smallest unit of transfer from the disk is sector cluster(group of sector)
the disk constructon may result

- constant track capacity
- variable track capacity

the disk capacity is given by T*S*b ehere t is no.of tracks s is the no of sector per track bis size of sector in bytes
placing the controlinfoon every sector is formating.
the formated diskcapacity is given by T*S*control bytes
time to acces info from the disk system is given by $T_{avg.} = seektime + avgRotationalLatency + datatransfertime$ seek time = time needed to move the head to desired track= no. of track * seektime avg.RL = time for one rotation/2

9.14 Cyllinder

logical collection of simmilar position tracks occurs on surfaces
such cyllenders may be addressed by two kind of addresses

- logical address : ¡cyllinder surface sector¿
- linear address : c*m*n+h*m+s m=no of surfaces, n= no of sectors per surface

Chapter 10

File System

10.1 Introduction

visible part of the OS and implementable secondary storage device is file system

FS provides mechanism for online storage and accsess to programme and data of the OS

FS consist of

- collection of files
- directory structure : information about all files in FS

File : file is collection of logically realted set of records
it is nothing but data structure wchich has

- defination
- Representation
  - Flat (series of bytes)
  - recurse - hirachical form(B,B+ trees)
- Operations
  - create
  - open

  - close/delete
  - seek
  - read/write/append
  - copy/rename
  - truncate

- Attribute
  - file name
  - file id
  - file extension
  - file type
  - size
  - owners
  - permission
  - mode
  - date/time creation modification
  - link
  - location
  - protection
  - extended file attributes (checksum,character encoding)

above all attributes of file are keptn FCB (file controll block)
typically directory entry may caontain file name and id
directory structure is classified into

- single level
- two level
- tree structure
- Acyclic graph directory structure

10.1.1 Single level

simplest directory structure

all files contained in same directory which is easy to support and understand

limitations

- limited no of file
- name conflict
- serching becomes difficult when no. of file increases

10.2 Two level

a solution to create separate directory for every user

each user has an user file directory

it solves the name conflict problems

it effectively isolate users

limitation

- local user file sharing ristricted

10.3 Tree structure / Multi level / Hirarchichal

it is extended to two level directory

it allows the user to create there more subdirectories and organise their file accordingly

a tree structure is most common directory

sharing of file is possible with path name(location attribute)

10.4 Acyclic graph sirectory structure

dierect file sharing is possible with link attribute

10.5 Disk Space Allocation/File allocation

the disk is divided into no. of blocks

each block is addressable through address

file allocation method

- Contagious allocation
- linked allocation[non- contagious]
- Indexed allocation [mixed of above 2]

limitation

- IF
- EF
- file size cant be increased(always)
- sequential and random acces time possible(advantage)

10.6 linked allocation

allocation is done on the basis of individual blocks data can be store in any block which is free each block containd pointer which point to next block

with block is n then only n-1 units of data are stored 1 ius for pointer

limitation

- IF
- sequential acces possible only

10.7 Indexed allocation

it is a combination of contagious and non contagious allocation method here the block called indexxed block which contains addresses of all the data block used by the file

if a file require n blocks n+1 blocks are used where first block is indexed block

the index block contains address of all data block

limitation

- IF
- sequential acces possible(advantage)
- index blocks need space

10.8 Extended indexed allocation

used in unix/linux based systems

it has something called I-node simillar to index block in simple indexed mode

apart from otrher attributes of file it has multilevel indirect DBA's a direct dba is a field which contains the address to data blocks directly a two level indirect DBA contains the address of a block which in turn contains address to data blocks simillarly in 3 level Indirect DBA we have two levels of block which contains address and the last level contains data blocks

unix/linux has direct 2 level and 3 level DBA's

10.9 Disk Sceduling Algorithms

algorithms

- FCFS
- SSTF(shortest seek time first or nearest track next)
- SCAN (elavator)
- LOOK
- C-SCAN
- C-LOOK

Chapter 11

Endnotes

Spooling: Simultaneous Peripheral Operation On-Line : this overlaps I/O of one job with processing of other job. It uses disk as a huge buffer for reading as far ahead as possible on the input devices and for storing output file until the output devices are able to accept them.