Deadlock

Deadlock occurs in a system when two or more processes are unable to proceed because each is waiting for the other to release a resource.

System contain finite number of resource and each resource have no. of instance. Process may use resources but it should not exceed the total no. of resource in the system.

Process utilize resources in following sequences
1) Request 2) use 3) Release.

Deadlock occur : I) Mutual exclusion III) No preemption
II) Hold and wait IV) Circular wait

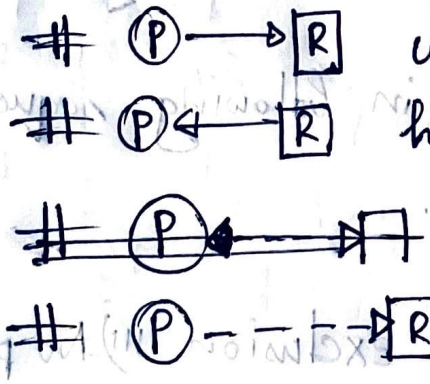
I) Mutual exclusions At least one resource must be held in nonsharable mode, only one resource at a time can use the resource.

II) Hold and waite Process hold resource while waiting for others.

3. No preemption: Resources can not be forcibly taken from process.

4: Circular wait: A circular chain of process exists, each waiting for a resource held by next.

Ⓟ = process R = Resource



Handling / solving deadlocks

i) prevention

ii) Avoidance

iii) Detection and recovery

Banker's algorithm

1) Allocated
2) Max requested
3) Available

(will be given)

Steps:

1) find need (max - allocated)
2) find a process' need that can be fulfilled by available resource.
3) find process sequentially.
4) After process is terminated,
   Available + = Allocated
5) repeat

If a process can not be processed with available resource then it is in unsafe state. and deadlock might occur.

File system

Hard disk architecture ৪ i) Platter v) sector
ii) spindle
iii) Read-write head
iv) Track

Platter
↓
track
↓
sector
↓
data

প্রত্যেক platter এ অনেক track থাকে,
~~এবং~~ এ track এ " sector এ
" sector " " data " .

problem 1৪ 8P, 256 track, 512 sector, 512 kb

∴ disk size = 2×8 × 256×512× (512 kb)

= 2^{40} b

1tb = 10 2^{40} b

∴ number of bit required 40.

Disk accesstime = seek time +
rational latency + transfer time.

transfer rate =

transfer time =

b

problems: Disk access time

seek = 5 ms

600 rpm = 100 rot/sec

surface = 4 × 2

128 track / surface

256 sector / track

256 kb / sector

total capacity = (~~8~~ × 128 × 256 × 256)

transfer rate = total capacity × 8 × 100

Main Memory

Program must brought into main memory and converted to process.

CPU can access main memory (many clock cycle) and register (one clock cycle)

cache sits between memory and cpu.

Protection of memory ensure correct operation.

pair of base and limit register define l.a. space.

base = starting address # limit register = no. of instruction

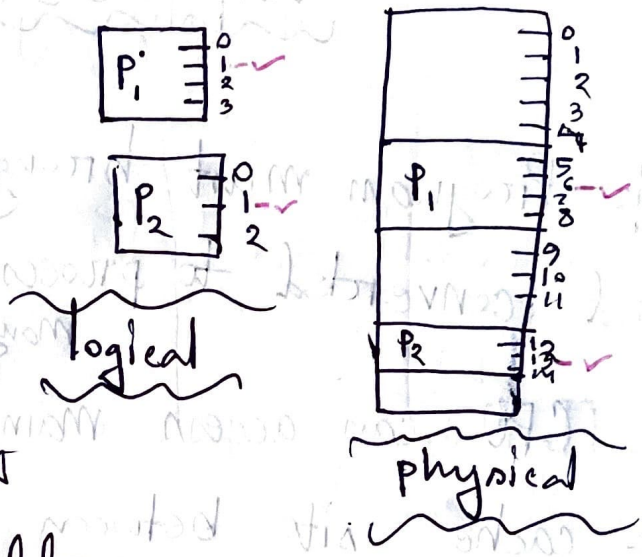
∴ ending address = base + limit

hardware access protection

① address ≥ base
② " < base+limit } yes = access memory
no = send trap to cancel request.

In this

local address is logical address



local address যখন memory তে যায় তখন তাকে physical address.

logical address ⇒ Generated by CPU.

physical address ⇒ address seen by memory unit.

Memory management unit (MMU) ⇒

MMU ⇒ hardware device that map virtual to physical address at run time.

User programs deals with logical address.

Contiguous allocation

Main memory divided into two partitions :

1] OS usually held in low memory with interrupt vector
2] User processes held in high memory.
3] Each processes contained in single contiguous memory.

Fixed sized partitions : This refers to dividing the computers memory into equal size fixed section where only one program/process can stay.

Variable partition size : memory divided into various sizes to a given process needs.

hole : block available memory.

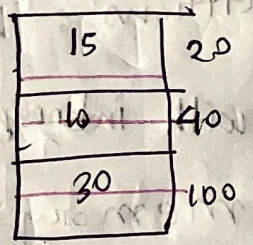
First fit : Allocate the first ~~fit~~ hole that is big enough.

best fit : Allocate the smallest leftover hole.

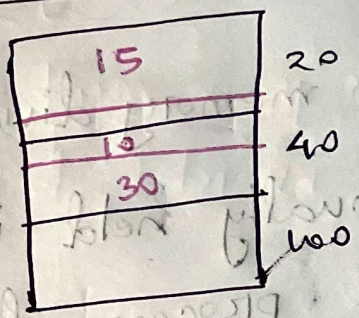
Worst fit : " " largest hole.

Fixed size - first fit

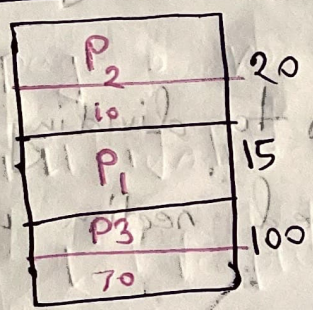
$P_1 = 15$, $P_2 = 10$, $P_3 = 30$



Variable size - b.f



V.s - b.f



External fragmentation: Total memory space exists to satisfy a req, but it is not contiguous.

Internal fragmentation: When a program is allocated memory, it might get little extra memory which is unused. This wasted space is internal fragmentation.

Worst case fragmentation = 1 frame - 1 byte

On avg = $\frac{1}{2}$ frame

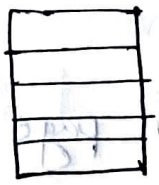
Reduce e. fragmentation by compaction 6

1. Shuffle memory and place all the free memory together
2. Compaction is possible only if memory is dynamic.

Paging

each piece of process is page

page table



physical/main memory

each piece of main memory is called frame.

Page no = ~~a page~~ কত no. page এ আছে।

Page offset = কত এ page এর কত no. instruction

page table translate logical address to physical address.

page number, $p = (m-n)$

n offset, $d = n$

no. of address space = m

logical address

1| page no. represent করে bit এর data লাগবে।

2| ~~off~~ same for offset.

Example:

~~no page~~ no of page = 5, per page এ 4টা instruction.

∴ to represent 5, we need atleast 3 bit.

" " 4, " " 2 bit

∴ 2P 1 no. instruction

0	1	0	0	1

logical → physical address

1| no. of content in page table

2| find highest frame no. in the page table.

(frame no * page size) + offset

Example: l.a = 1010

page table	
0	2
1	5
2	0
3	7

∴ page table content = 4

= 2^2 = 4

no, offset is 2.

10 | 10

↓2

highest frame no. = 8 7

to represent 8 ~~bits~~, we need 3 bit.

∴ P.a = | 0 | 0 | 0 | 1 | 0 |

frame | offset

---x---

Practice problems:

2] offset = 1 [page size = 2^1]

page no. of process 1 = 5 [no we need 3 bit to represent 5]

∴ 0 | 1 | 0 | 0 | 1 | : logical address

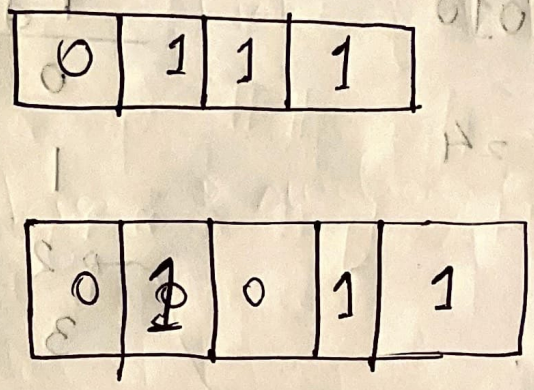
offset

frame num = $\frac{32}{2}$ = 16 = 2^4

| 0 | 1 | 0 | 0 | 1 | - physical address

offset

c)



Implementation of Page table

page table is kept in main memory.

page table base register (PTBR) points to page table.

PTLR indicate size of page table

Associative memory is cache memory.

In the previous scheme every data require two memory access &

1) page table 2) data / instruction

It can be solved by fast lookup hardware cache called translation look aside buffer (TLB) / associative memory

Effective access time:

Associative lookup = ε time unit

Hit ratio = α [percentage of time that a page no. is found in associative register]

Example:

$\alpha = 80\%$, $\varepsilon = 20$ns for TLB search, 100 ns for mem

$\therefore$ EAT = $(.8 \times 120) + (0.2 \times 220) = 140$ns

Shared code

One copy of read only code shared among processes.

similar to multiple thread sharing same process

private code and data

Each process keeps a copy of code and data

This private code and space can be anywhere in the l.a.s.

Disk architecture

Platters → surface → tracks → sector → data

Disk access time = (ST + Rotational latency + TT)

Seek time = Time taken by R/W head to reach desired track.

Rotation time = time taken for one full rotation (360°)

Rotational latency = time taken to reach to desired section (half of rotation time).

$$\text{Transfer time} = \frac{\text{Data to be transfer}}{\text{transfer rate}}$$

$$\text{transfer rate} = \left(\text{no. of head (surface)} \times \text{capacity of one track} \times \frac{\text{no of rotation}}{\text{in sec}}\right)$$

Capacity of one track = sector × data

practice problems

1] platter = 4, track = 64, Sector = 128, Data = 128kb

no. of bit required = $4 \times 64 \times 128 \times 128kb$

$= 2^2 \times 2^1 \times 2^6 \times 2^7 \times 2^7 \times 2^{10} b$

$= 2^{33} b$

Disk size $= 2^3 \times 2^{30} b$

$= 8 GB$

2] seek time, st = 20ms

disk rotation = 1000 rpm

∴ rotation in 1 sec = 16.67

∴ 1 n time = 0.06 sec = 6ms

platter = 8
track = 128
sector = 64
data = 128

transfer ~~time~~ rate = $16 \times (64 \times 128) \times$ ~~6ms~~ 16.67

$= 16 \times (64 \times 128 \times 2^{10}) \times 16.67$

$= 2^4 \times 2^6 \times 2^7 \times 2^{10} \times 16.67$

$= 2^{27} \times 16.67$

transfer rate = $\frac{1024}{2^{27} \times 16.67}$

$= 4.58 \times 10^{-7}$ byte/s

rotational latency $= \frac{1}{2} \times 60 = 30$ ms

access time $= 20\text{ ms} + 30 + 4.58 \times 10^{-7}$

$= 50.00000046$

$= 50.00046$

3) total no. of pointer in one block $= \frac{\text{size of data block}}{\text{each address size}}$

$= \frac{32 \text{ bit}}{4}$

$= 8$

maximum file size $= (8 + 8 \times 2 + 2 \times 8^2 + 2 \times 8^3) \times 32\text{b}$

$= 1176 \times 32\text{b}$

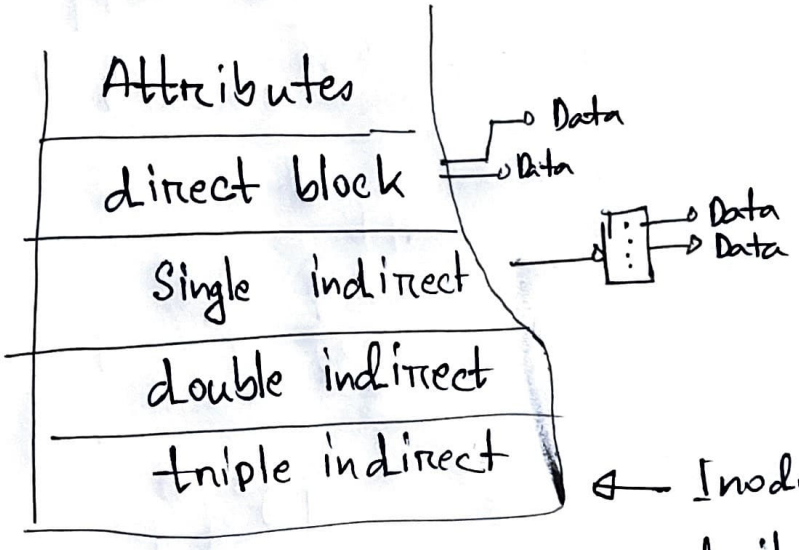
$= 37632\text{ b}$

$= 36.75\text{ kb}$

UNIX ~~Str~~ Inode structure

I means index and node means block.

#



In direct block we store pointers and it directly points data block. Suppose we had a file and stored it in a disk block and it's address ~~it~~ stored it in direct block

In single indirect, it will give us pointers and # ↑ will give us data.

data block consist of