Operating System.

Introduction.

• **What is OS:**
→ Interface b/w h/w and user.
→ Resource manager.
→ set of utilities.

• **Von nuemann Arch:**
→ stored program concept, ie the program must be first Loaded into main mem, then only CPU can execute instructions.

• **Real Time OS**
→ System should generate response in restricted amount of time.
(a) Soft Real Time OS
(b) Hard Real Time OS

• **Uniprogrammed OS.**
→ Only one process can reside inside main memory at a time
→ disadvantage
(a) CPU remains IDLE during I/O.
(b) inefficient utilization.

• **Functions / Goals.**
→ CPU scheduling
→ Security
→ Memory management
→ File management.

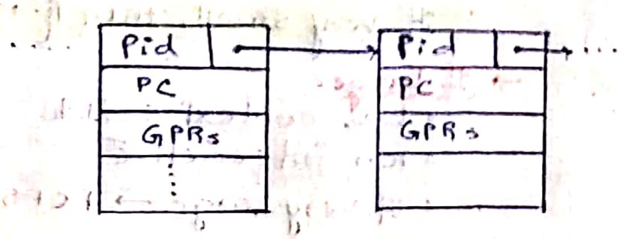
• **Goals:**
→ Convinience
→ efficient
→ Portability
→ Scalability
→ Reliability
→ Robustness.

• **Multiprogrammed OS.**
→ Multiple process can reside inside main memory location at a time.
→ Objectives
(a) Max utilization.
(b) Max throughput.
→ Types
(a) Preemptive
(b) non-preemptive.

Process Management:

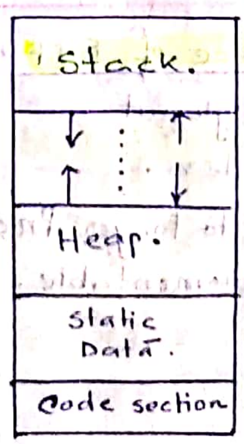
• **Program**
→ Binary file stored in Disk.
→ Dead set of instruction and data.
→ Program → Instruction, Data

• **Process Control Block. (PCB)**
→ its the meta data of process.
→ Pid, PC, GPRs, files in use, size, priority etc. are stored.



• **Process.**
→ Program under execution
→ Instance of a program

• **Process Structure**



• fork( ) system Call.

→ Create child process (duplicate)

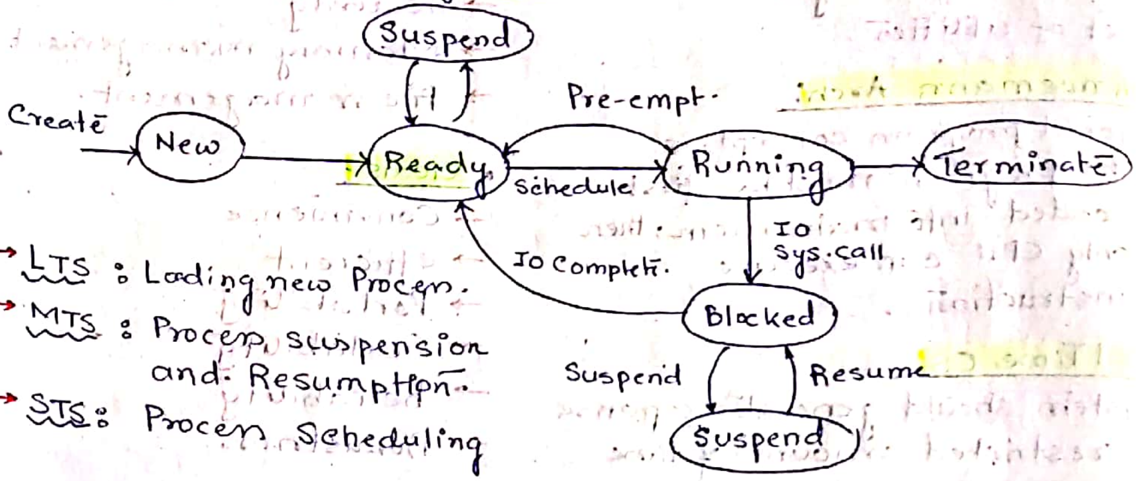
→ 0 ← Returned to child.

→ ≠0 ← Returned to parent.

→ # Children = $2^n - 1$

→ # Processes = 2^n

• State Transition Diagram.



→ LTS : Loading new Process.

→ MTS : Process suspension and Resumption.

→ STS : Process Scheduling

• Scheduling Formulas:

→ AT_i = Arrival time ; → CT_i = Completion time

→ x_i = BT ; y_i = IOBT.

→ $TAT_i = CT_i - AT_i$ → $WT_i = TAT_i - (x_i + y_i)$.

→ Schedule Length (L) = Max (CT_i) − min (AT_i).

→ Throughput = $\frac{n}{L}$

• Scheduling Algorithm:

→ FCFS → Round Robin

→ SJF → Multilevel Queue

→ SRTF → Multilevel feedback

→ LRTF

→ HRRN

→ Priority

• Response Ratio $= \frac{WT + BT}{BT} \simeq \frac{WT}{BT}$

• Exponential Averaging

→ used to Predict the next BT of the Process

→ t_i → Actual

→ τ_i → Predicted.

$$\tau_{n+1} = \alpha \cdot t_n + (1-\alpha)\tau_n$$

* Bias is on actual.

• Performance of SJF.

→ Max throughput

→ Avg WT is less.

→ Starvation to longer Processes.

→ Non-implementable.

• Effects of time Quantum.

→ If v. small:

- more Context switch
- more interactive.
- If very small, then $\eta \to 0$.

→ If Large.

- Less context switch
- less interactive
- If very Large → FCFS.

Process Synchronization.

• Critical Section (C.S.)
→ Part of the program where shared resources are accessed. (Read OR write).

• Race Condition
→ The outcome of a program depends on seq. of execution.

• Preemption.
→ Forcefully stop execution of currently running Process.

• Mutual Exclusion.
→ No 2 or more process in the C.S. concurrently.

• Requirement for S.M. < b_2 b_1 b_0 >
→ Mutual Exclusion (b_2)
→ Progress. (b_1)
→ Bounded Waiting. (b_0)

Synchronization Mechanism

Software.
→ Lock Var (0 1 0)
→ Strict Alteration (1 0 1)
→ Petersons (1 1 1)
→ Backery (1 1 1)
→ Dekkers. (1 1 1)

Hardware
→ TSL (1 1 0)
→ SWAP (1 1 0)

Kernel Based
→ Semaphore (~)
→ monitors. (~)

⊛ Peterson's soln. is 2 process soln. only.
+ Deadlock possible because of priority inversion problem.

NOTE Priority inversion problem may occur when busy wait S.M is used along w/ Preemptive Priority scheduling.

Semaphore
- Counting
- Binary / Mutex.

• Counting Semaphore.
→ UP & DOWN : atomic
→ UP is ALWAYS successful.
→ -ve value of counting semaphore indicate size of queue iff initial value was +ve.

• Binary Semaphore.
→ if there are blocked process in queue, then value must be 0.
→ if value is 0, then there may or maynot be blocked process.
→ If val = 1, then nothing blocked.

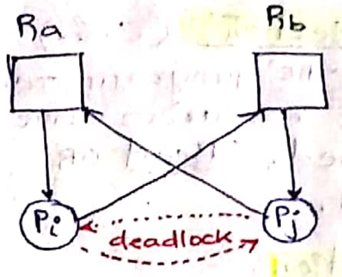
• First Reader writer Problem.
→ If Readers are reading, then subsequent readers are also allowed, even if writer are waiting.

• 2nd Reader writer Problem:
t_0 : R_1 ✓
t_1 : R_2 ✓
t_2 : W_1 ⟳
t_3 : R_3 ⟳] → wait untill all queued writers are done.

• Dining Philosophers
→ (N-1) Phy : L → R
1 Phy : R → L
→ odd, Phy → L → R
even Phy : R → L.

Deadlocks

- **Condition for Deadlock.**
  - → Mutual Exclusion of Rsc.
  - → Hold & wait
  - → No preemption
  - → Circular wait



Deadlock Handling

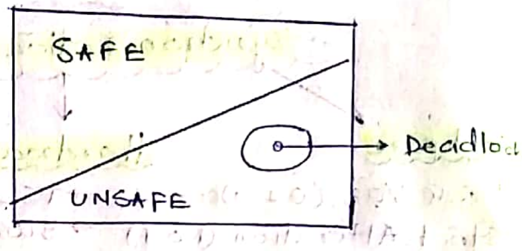
Type 1
- → Prevention
- → Avoidance (Bankers).

Type 2
- → Detection and Recovery

Type 3
- → Ostrich Algorithm

$\boxed{n(K-1)+1 \le R}$

give one less to each process & still have on left extra to prevent deadlock.



Memory Management

- **Functions.**
  1. Allocation of mem.
  2. Protection
  3. Free space mgmt.
  4. Deallocation.

- **Goals.**
  - → Effective utilization (min fragmentation)
  - → Manage execution of Larger program in smaller memory. [virtual memory.]

Mem Allocation methods

Contiguous
- → Partitioning
- → Overlays
- → Buddy system

Non-contiguous.
- → Paging
- → Segmentation
- → V.M

NOTE Paging v/s segmentation

	IF	EF
Paging	✓	✗
Seg.	✗	✓

- **Fixed Partitioning**

→ No of Partitions are fixed but not size of Partition

→ Partioned @ Start-up, then deleted @ reboot.

→ Best fit (BF) works best.

→ 1 Partition = 1 Process.

→ internal fragmentation

→ Deg. of M.P ∝ # Partitions.

→ Max size ∝ Largest Partition

- **Variable Partitioning.**

→ Create Partition whenever required

→ Partition Table maintained in OS area of memory

→ Worst fit (WF) give the best results.

→ No IF, but EF.

→ Deg of M.P. ∝ not limited

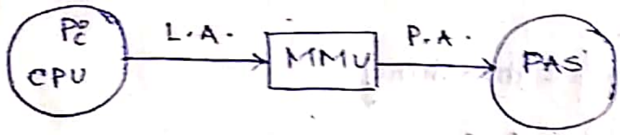
→ Max size ∝ Largest-free hole size

NOTE Soln for EF.

→ Compaction

→ Non contiguous

- **Non Contiguous Allocation.**



- **Paging**

→ # pages = $\frac{LAS}{PS}$ = N.

→ $p = \log_2 N$

→ $d = \log_2 PS$

→ LA: | p | d |

→ PT. entry = e

→ FS = PS

→ # Frames = $\frac{PAS}{FS}$ = M

→ $f = \log_2 M$

→ PA: | f | d |

→ PT. size = #Pages * e.

$$\boxed{EMAT(TLB/S.P.) = x(c+m) + (1-x)(c+2m).}$$

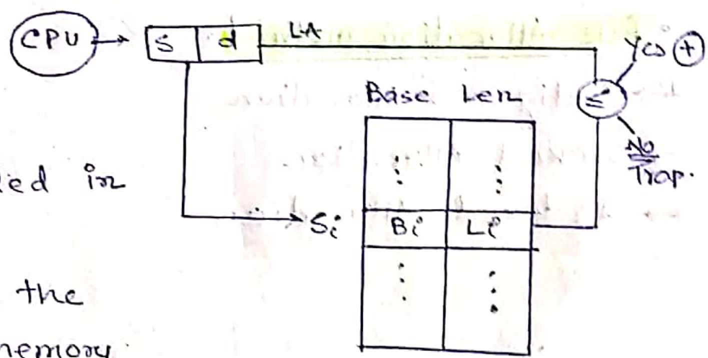
(*) To differentiate b/w entries, store the Pid also.

- **Segmentation**

→ LAS is divided into diff size segments.

→ Any seg. can be loaded in any free hole

→ Segmentation preserves the physical view of the memory



(*) TLB can also be used for segmentation.

Demand Paging:

→ If page to be replaced is modified, 2 disk access is Req.
  → 1 to Write back
  → 1 to Read new.
→ Otherwise, only 1 disk access is required.

$$\boxed{\text{EMAT}(\text{demand paging}) = p \times pfst + (1-p) \times m.}$$

Page Replacement Algorithm:

→ FIFO
→ Optimal
→ LRU
→ MFU
→ MRU
→ etc

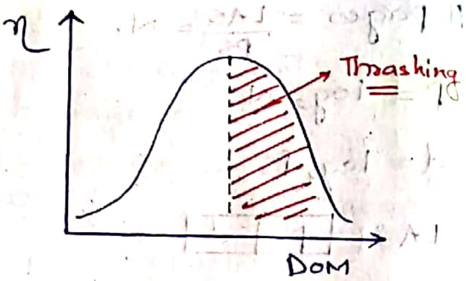
* FIFO suffers from Belady's Anomaly.
* Optimal replacement gives the least.

Thrashing.

→ It implies very high or excessive Paging activity.
→ {PS ↓ ≃ Refstring ↑ ≡ PF↑ } ≡ thrashing
→ {Frames ↓ ≃ PF ↑ ≃ Thrashing ↑ }.

Reasons

→ Lack of frames
→ DoM ↑ ≅ PF ↑ ≅ Thrashing
→ PS ↓
→ Replacement algorithm



File System.

→ Access time = ST + RL + TT.
→ RL = R/2 [R → 1 Rotation time]

File Allocation method.

→ Contiguous Allocation
→ Linked Allocation
→ Indexed Allocation