CIE - 1: 1st unit + unit - 2 till threading
galvin & silvershacks : reference TB

UNIT-1 : INTRO

Abstraction : hiding inner lying complexities of a system & only portrays whatever is needed

OS takes care of / manages :-

1. process / task / program / thread
2. memory
3. files
4. resources

Program : inactive state (or entity), once its opened, typed in it comes to active state

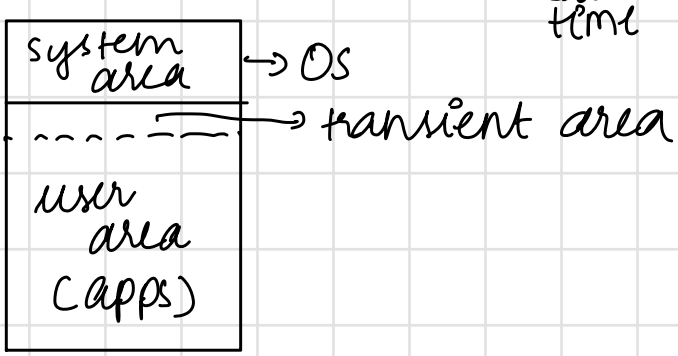
task : used for real-time OS
& task & process used interchangibly
process : active entity (list of instructs to perform)

real time OS, every task ka response time is very very less (microseconds)
eg : in cars, airbag deployment

OS runs in the CPU first before other processes do

OS definition : set of programs that manages process, memory, files and resources
juggles b/w user convineance & CPU efficiency

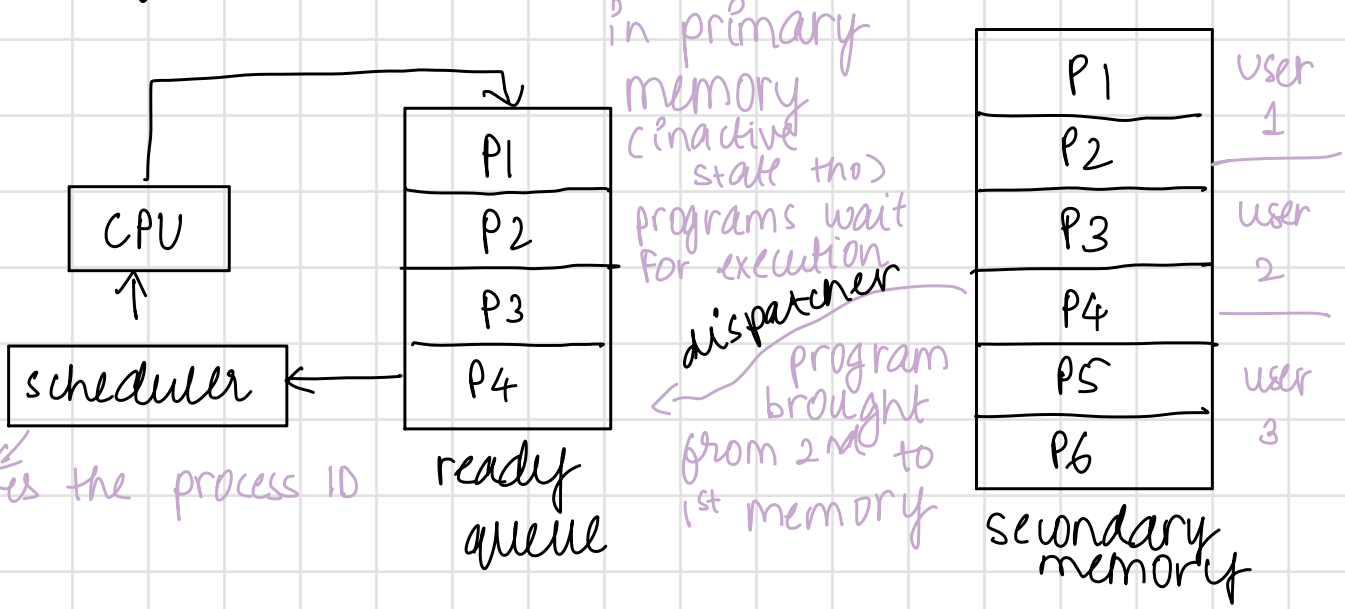
Process management : mom & kids & 1 toy or buy multiple toys (multi core)
↳ First come first serve, priority, time sharing
↳ real time ↳ round robin



all programs majorly reside in secondary memory

process scheduling : which process should execute at what point of time in the CPU it has a scheduling algorithm (FCFS , priority , time sharing AKA round robin)

dispatcher : it'll take a process from one queue to another queue or in and out of CPU



process ID: a unique ID is given to each process
OS identifies processes using this

swapping: movement of processes in and out of memory or in & out of CPU or 2° memory

context & context switching → part of process mgmt
↳ at any point of time how far has the program executed (state of a process)
→ switching between 2 different processes
context: visualize as a screenshot of a video at some random time

resource: any entity outside CPU like memory, i/p & o/p devices
→ CPU itself is a resource

shareable resource: eg pdf file many ppl can use it
non shareable: eg memory or printer

transient memory: its a buffer where as programs can be brought, but it can't be used by user programs, only used by OS.

OS execution is an overhead to the CPU efficiency

kernel → heart of OS : Those OS programs which are basic in nature which need to be active all the time in CPU

Types of OS:

1. batch processing system - FCFS
2. Time sharing system - round robin
3. multi-programming system - priority based
4. Real time OS - priority based scheduling

I] Batch processing system

1. turnaround time: time taken from creation of batch until the time it gives output & finishes execution

batch of CY 2028 passing out together

Batch
↓
jobs
↓
job step
↓
processes (set of instructions)

batch of 2028
↓
semesters
↓
courses
↓
CIEs, quizes, ELs

2. It always tries to focus on CPU efficiency by reducing overhead

II] Multiprogramming system:

1. context switching is used
2. concept of priority came
2 types of processes:
→ CPU bound processes: more CPU instructions
→ I/O bound processes: more i/p & o/p
3. fair usage of CPU & user

↳ shopkeeper scanning products and o/p of cost

4. priority imp to bring in parallelism
5. High priority is to be given to I/O processes
if high priority was given to CPU then I/O processes is starved

CPU starvation & Resource starvation

interrupt → kernel mode is activated : any stoppage in execution of CPU
→ hardware : insert pendrive
→ software : eg printing, div by zero, ∞ loops, int overflow
↳ system call : sub request which user places to OS for a particular service
AKA trap
↳ identified by a status flag (combination of bits & each bit has some significance eg: parity, overflow)

OS during start :
→ bootloader : initializes registers inside CPU

OS functions
1. list of users (active)
2. authentication
3. loads progs specific to that user
4. OS scans thru ports & maintains a list of resources

resource allocation table : list of resources with IDs and allocates accordingly to user

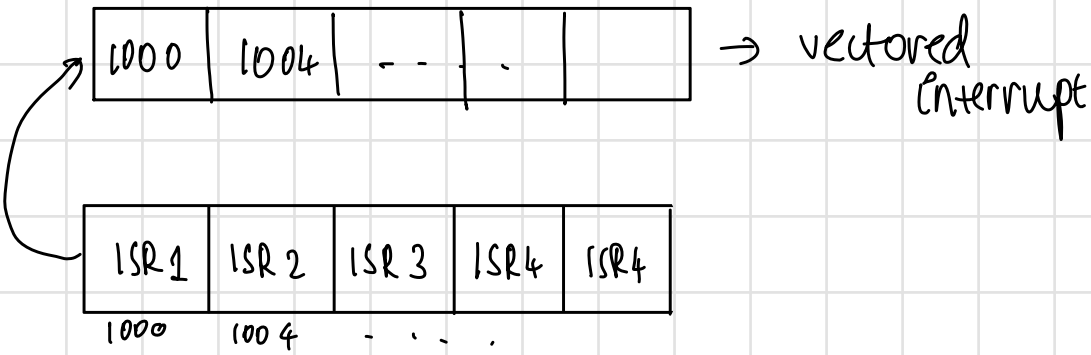
resources → ID → allocation

if no resource is free, the process must wait for the resource to be relinquished so that it can be used.

interrupt service routine: change in instruction for some particular interrupt → custom ISR
eg: if someone pinches you'll react or have a default ISR which can be to do nothing

system call is an interrupt
interrupt → OS invoked → checks status flag → calls appropriate ISR or default ISR to the CPU

Interrupt > priority than process
ie a process will be pre-empted
like the process stops at its current state & ISR stuff happens and it goes back & continues execution from previous point.
vectored interrupt: stores address of corresponding ISRs



dynamic allocation: paired sharing & allocation of resources (overhead is more) break head more
resource allocation table needs to be executed

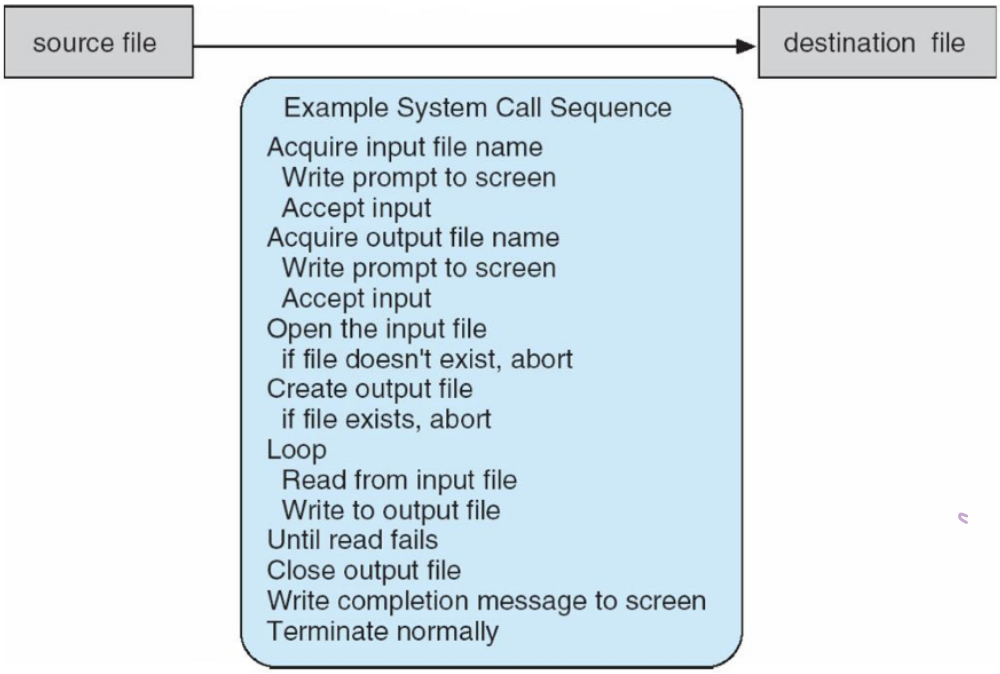
Mode bit: a single bit which tells whether a user program is being executed or a kernel prog
mode 0 → kernel prog
mode 1 → user prog

Process synchronisation : when 2 processes want to access same resource or file, an order must be specified to access them. This part of resource mgmt is called process synchronisation.

Protection & security : Protect files of one user from being accessed by another

protection : safety within the system b/w one user & another

security : protection from threat



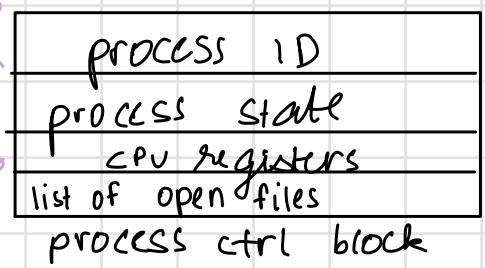
PROCESS : active entity

- every process associated with process control block (it's a data structure associated with processes)
- every process → identified → process ID (unique)

States of processes

1. new : just created

buffer diagram in ppt



when process created, memory to be allocated to it (from)

2. ready : process waits in ready queue has all resources it needs to be executed

3. waiting

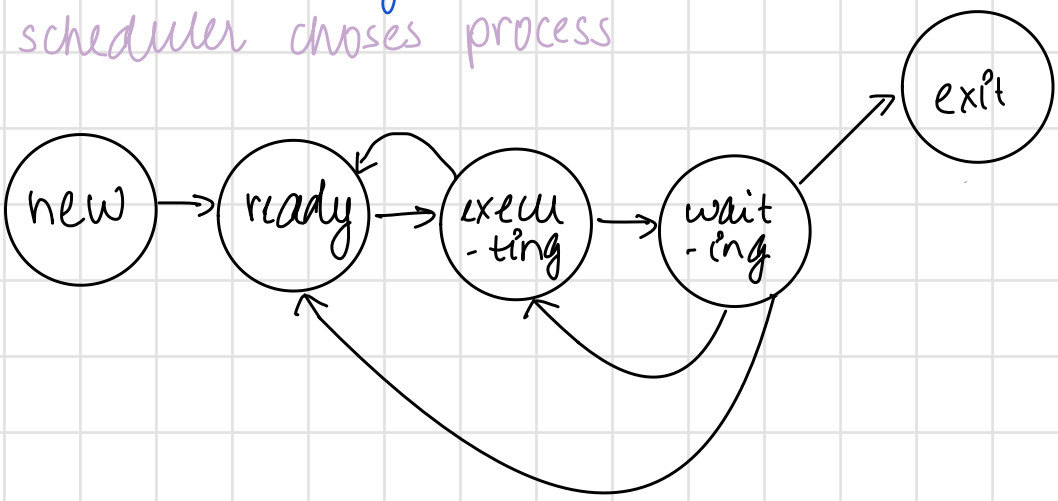
4. executing

5. exiting

note: process ctrl block is imp when new is created or context is switched

imp Q) What is process ? what is process control block ? Draw relevant diagram

scheduler choses process



process control block diagram

When process moves from executing to waiting

→ when process waits for event to occur

→ when process waits for I/O device

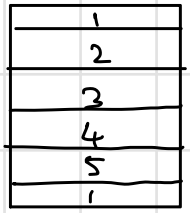
→ when process waits for timer to get expired
↳ ex: washing machine
↳ if process goes for sleep system call

- wait to ready or wait to execute : scheduler decides
- waiting : process doesn't have everything needed
  eg: I'm in a line to buy food but I don't have money so I "wait" for someone to bring it.

- executing to ready state in time-sharing system

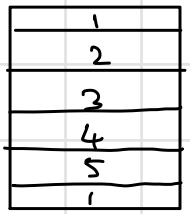
FCFS

new → ready → execute



round robin / time share

new → ready → execute
(time given)
↳ put back to ready queue



priority based

① take 1st year CY & 65 students
② and 2nd year CY & 70 students
within ① & ② they've same priority but ② has more priority than ①
it can be fixed priority levels or dynamic

- lower priority task gets preempted & moves to ready queue

Schedulers :-
- short term (CPU scheduler)
- long term (job scheduler)
- medium term (based on CPU efficiency)

(child process thing refer ppt)

Fork : creation of child process

wait : parent needs to wait for child processes to execute

orphan : parent exits before child

zombie : parent doesn't invoke a wait system call (concurrent execution with child)

process termination :
- abort
- exit : peacefully terminates

↳ if it needs more memory than allocated

Q) operation on processes
- ↳ creation
- ↳ child process (exec() call, types)
- ↳ wait & zombie (with child execute or after all children executed)
- ↳ address space & resource
- ↳ termination, abort (cascaded termination)

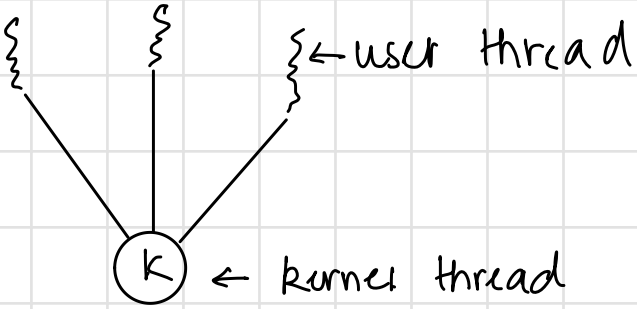
UNIT - 2

MULTI THREADED PROGRAMMING :

thread : lightweight process
kernels also follow multi-threaded structure

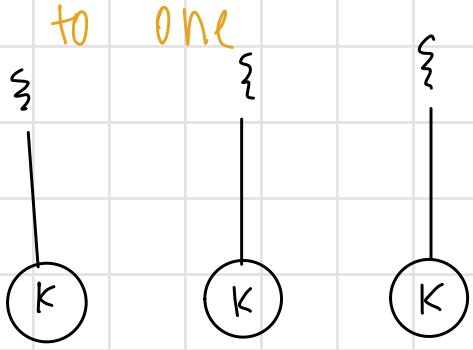
kernel thread - (K)
user thread - {

many to one



eg : Solaris green thread (need to remember eg)
GNU portable thread

one to one



eg : windows
→ more concurrency
→ no of threads per user process limited due to overhead

each thread has its own thread control block