CSE 323 / L-12 / 21.03.2024 /

Review of Midterm Syllabus

L-13 / 23.03.2024 /

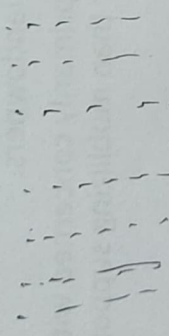
Midterm-1 Exam

L-14 / 28.03.2024 /

Chapter - 3

Processes

Program



PC = Program counter.
Basically store the address of next instruction.

* OS executes variety of programs.

  (i) user program or task ⇒ time-shared system

  (ii) process or jobs ⇒ Batch system

* Process:

  - a program in execution

  - process execution must progress in sequential fashion.

* Multiple parts of process:

  - text section: contains the program code

  - program counter: store the address of next instruction

  - register: saved the data used for the process.

  - stack: contains temporary data, function parameters, return addresses, local variable.

  - Data section: contains global variable.

  - Heap: contains memory dynamically allocated during run time.

* Program is passive, stored on disk.

  Process is active or running program.

  When a program start executing, its became the process.

States of process:

- new:
- running
- waiting
- ready
- terminated

Slide - 3.6

PCB ⇒ Process Control Block

- also known as task control block
- store information related to the process:

⇒

- process state
- Program counter
- CPU Register
- CPU scheduling information
- memory management information
- Accounting information
  - CPU used
  - clock time elapsed
  - time limit
- I/O status information

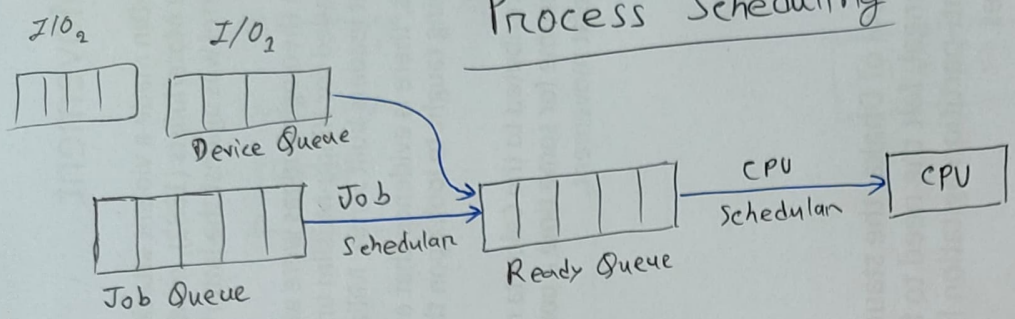
* if we want to increase the multitasking, then CPU idle time will increase.

Slide - 3.8

Program switch frequency $\propto$ CPU idle time

L-15 / 30.03.2024

Process Scheduling



* Three types of Queue:
  - (i) Job Queue: contains all process in the system.
  - (ii) Ready Queue: contains process that are ready and waiting to execute.
  - (iii) Device Queue: process that needs I/O operations.

Slide - 3.10-11

* Two types of schedular:
  - (i) Short term schedular - CPU schedular. execute to frequently
  - (ii) Long term schedular - Job schedular -

* When Ready Queue is nearly empty. the job schedular executed and fill it again.
  That's why CPU schedular is short time & job schedular is long time.

- Execution frequency:

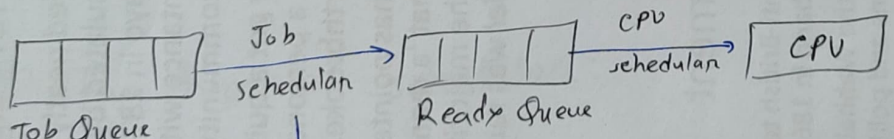
  CPU schedular > Job schedular

- Degree of multiprogramming depends on long term schedular.
- I/O bond process and CPU bond process will run randomly in mix to reduce the load and idle time. Like, 2 CPU bond and 1 I/O bond will execute normally.

  - Job schedular do that.

- Which schedular are responsible for degree of ~~multiprogramming~~ multiprogramming and why? **

L-16/18.04.2024/



→ responsible for multiprogramming

→ need to make good process mix combined of I/O bound and CPU bond process.

* Medium term schedular :

- if user want to focus on one program, then CPU needs to run the process of that program too frequently, medium term schedular remove the extra process from the ready Queue and keep the process of that particular program. Then, re-swap it in the ready Queue.

Slide - 3.13

* Content Switch:

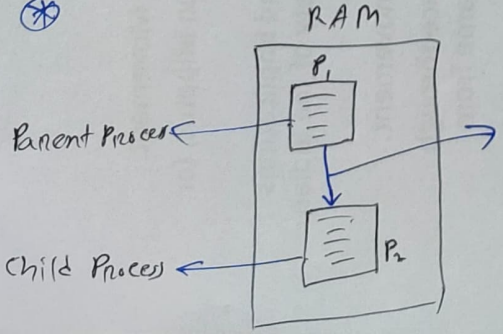
- load and save the process in PCB.
- execute between process switch.

⇒ the more complex OS and PCB ⇒ the longer the content switch.

⇒ multiple ~~the~~ sets of registers per CPU causes multiple content loaded at once

* For multiprogramming, we need to ~~shwi~~ switch too frequently. But it will increase the CPU ~~to~~ idle time.

*



* exec ⇒ replace existing code.

- PID ⇒ Process identifier
  - in general a serial number or index or id of a process.
  - Unique ID.
- Resource sharing option between parent and child process:
  - public : share all resources
  - protected : children share subset of parent's resources
  - private : no resources shared.
- Execution option:
  - Parent and children execute concurrently
  - Parent wait until children terminate.
- Windows allow child process without parent.
  Linux and unix abort the child if no parent.

Slide - 3.12

Slide - 3.12

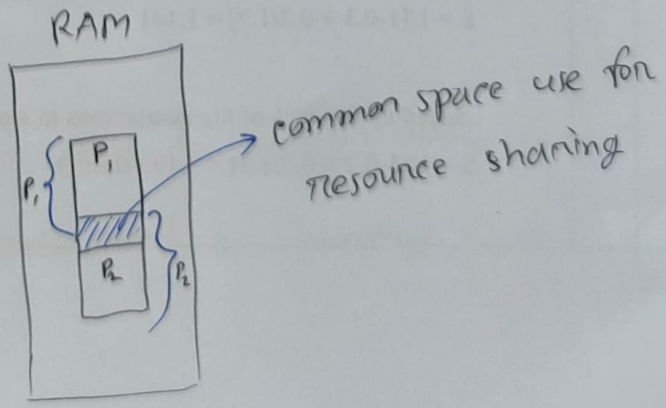
- exit():
  - returns status data from child to parent thru wait()
  - process resources are deallocated by operating system.

* abort():
  - used to terminate the child by parent, if children
    - exceeded allocated resource
    - no longer required
    - ~~if parent~~
  - if parent ~~doesn't exi~~ exiting, then system use it to abort the child.

* IPC ⇒ Interprocess Communication
  - shared memory
  - message passing

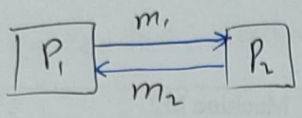
* Reasons for cooperating process:
  - information sharing
  - computation speedup
  - modularity
  - convenience

* Shared memory:

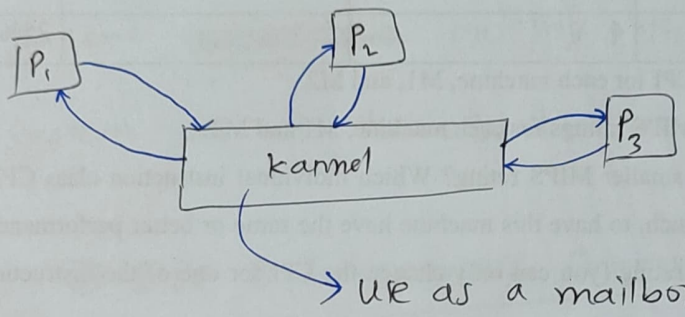


Message Passing:

- Direct message passing



- indirect message passing



For each communication, there are particular mail-box.

after the end of the process, mail-box get closed.

⇒ There are multiple mailbox in the kernel.

IPC facility provides two operations for message passing:

- send (P, message)
- receive (Q, message)

⇒ message size can be fixed or variable.

Shared memory:

- not popular and not so fast
- can be error due to garbage read for unsync. operation.
- OS not interfare here.

* message passing:
  - most popular and fast
  - OS will wait until data is available, then it will resend the mail.

* implementation of communication link: used for message passing.

  Physical:
  - shared memory
  - Hardware bus
  - Network

  Logical:
  - Direct or indirect
  - synchronous or asynchronous
  - automatic or explicit buffering

* Properties of communication Link in direct communication:
  - Links are established automatically
  - one link ⇒ one pair of communicating proces
  - one pair ⇒ exactly one link
  - usually bi-directional.

* Properties of communication link in indirect communication:
  - Link established only if process share a common mail box.
  - one link ⇒ many process
  - each pair ⇒ many links
  - uni- or bi directional.

* Blocking considered as synchronous

- Blocking send: sender is blocked until message is recieved
- Blocking receive: receiver is blocked until a message is available.

* Non-Blocking considered as asynchronous:

- Non-blocking send: sender sends the message and continue
- Non-blocking receive: the receiver receives a valid message or null message.

* Buffering:

- Zero capacity - no message are queued on a link.
  - sender must wait for receiver
- Bounded capacity - finite length of n message
  sender must wait if link is full
- Unbounded capacity - infinite length
  - sender never waits.

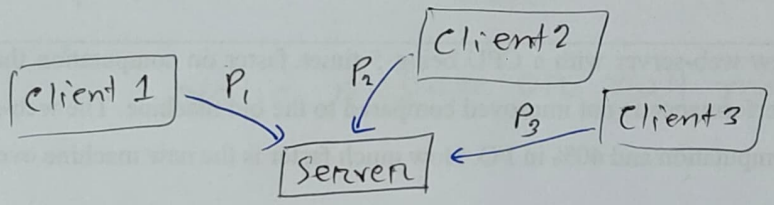
L-17/25.04.2028/

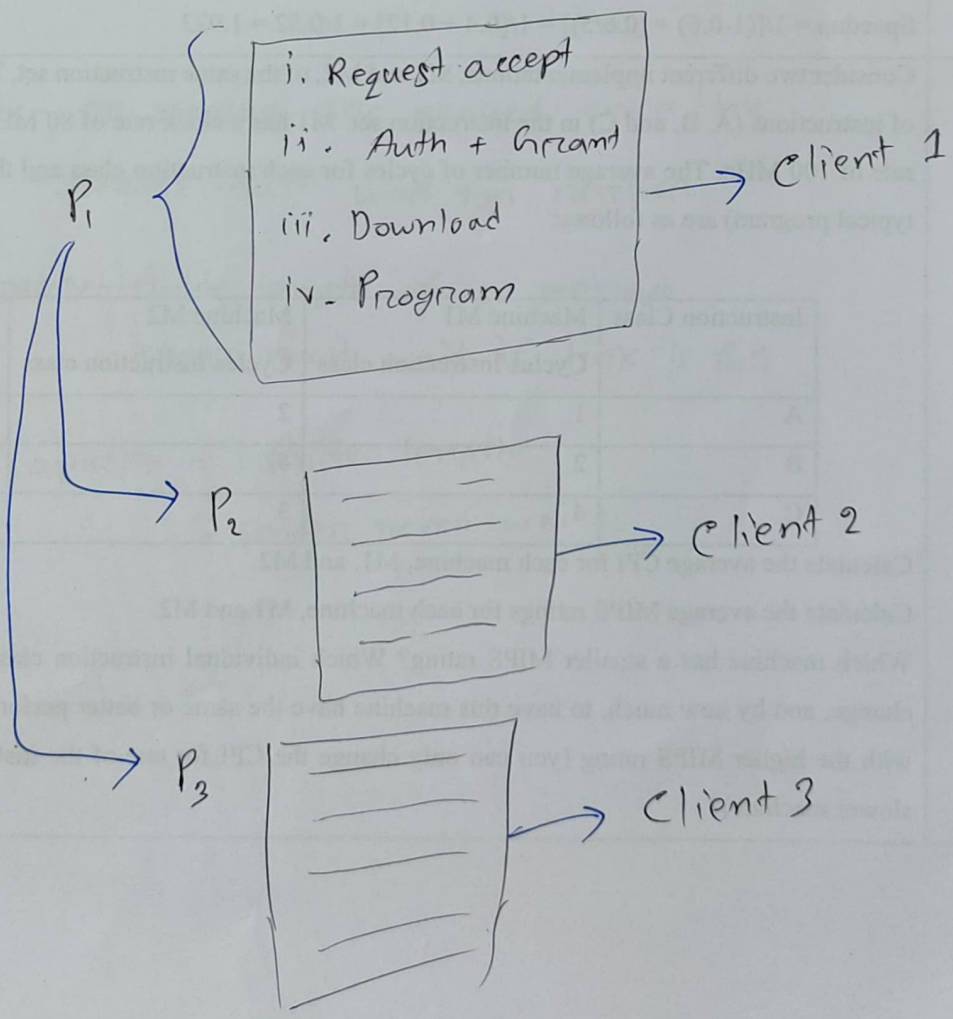
Midterm-2
ch-3,4
Final
ch-5,6,7,8

Chapter-4

Threads

* every process must have it own memory space, even the child process have too.





Slide - 4.2

* Benefits of thread:
  - Responsiveness - continued execution,
  - Resource sharing
  - Economy - switching thread is faster
  - Scalability - ~~Thread~~ advantage of multiprocessor architecture

* Difference between process and thread:

Slide - 4.4

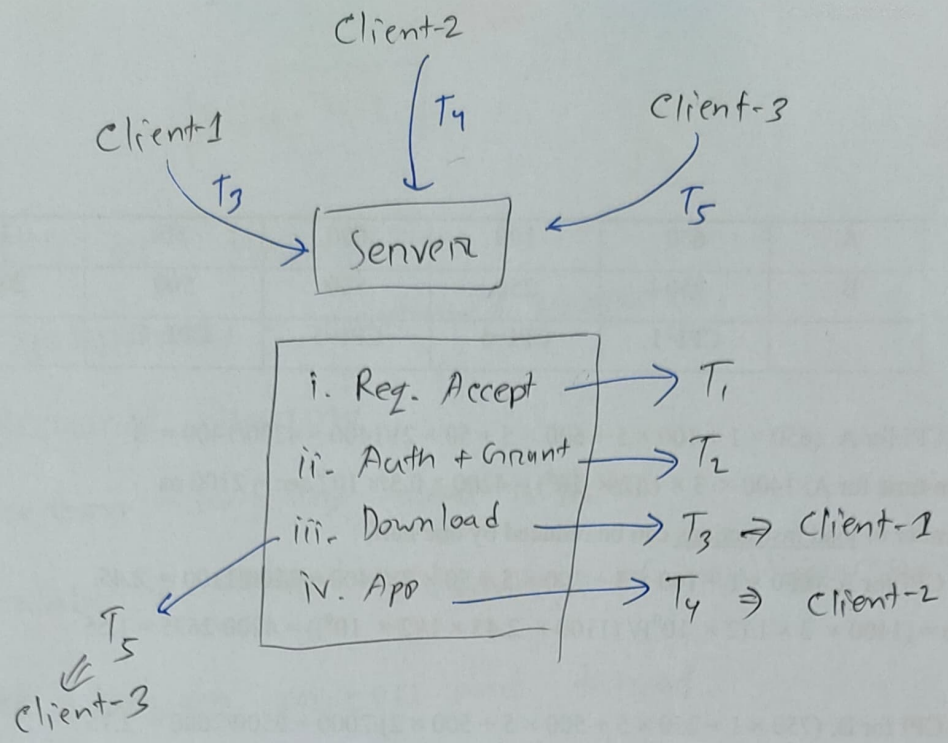
* Challenges for multicore or multiprocessor:
  - Dividing activities
  - Balance
  - Data splitting
  - Data dependency
  - Testing and debugging

* Parallelism : perform more than one task simultaneously.

* Concurrency : supports more than one task making progress.

Slide - 4.6

* Thread doesn't copy code, share the same code. but run time data are stored in different register sets.



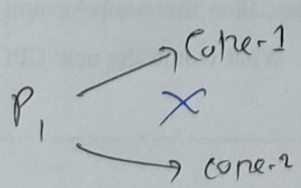
In process system, we need to create a process in kernel mode. So, kernel became busy for that.

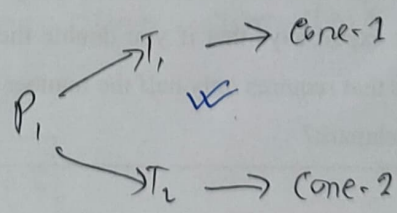
But for thread, we can create either in user mode or kernel mode.

If we create more user mode thread, the kernel will be free, for other task.

We can't run same process in multiple core in parallel.

But in thread, we can run in parallel.





Parallelism:

(i) Data Parallelism:

- Divide in subsets of same data and assign to multiple core to apply same operation

(ii) Task Parallelism: each thread perform unique operation of a task.

Thread libraries:

- POSIX Pthreads
- Windows threads
- Java threads

mid term-2
09.05.2024