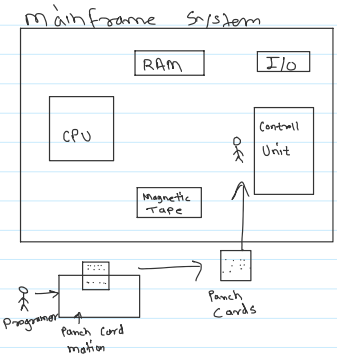
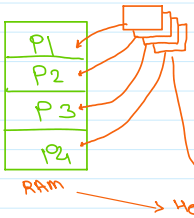


① Resident Monitor
- → Hal & libs for I/O
- → Reside in main memory
- → monitor execution of Programs (job)
- → Handle error

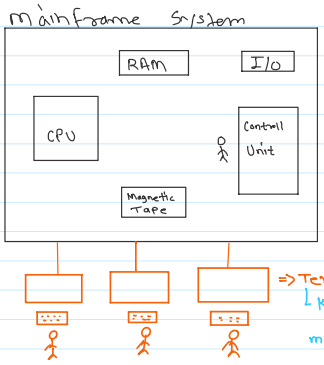
② Batch System
- → Submit a batch of similar programs (job)
- → Os pick a Program & execuit it

③ Multi - Programming System

execution time of Prog = CPU Burst Time + I/O Burst Time



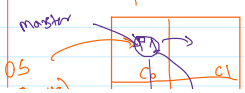
CPU Bound = CPU burst > I/O burst
IO Bound = CPU burst < IO burst
Loading multiple prog on main to increase CPU utilization
Should be good comb^n of CPU bound and I/o Bound
→ Here Ram memory manage by → Conticous allocation

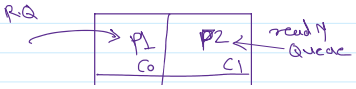


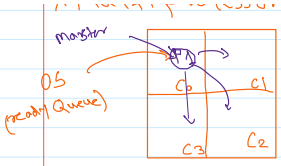
④ Time sharing System / Multitasking Sys
- → CPU time shared with among multiple jobs present in Ready Queue.
- → Response time < 1 sec should be.
- → multiple jobs executed concurrently.
  - ↳ Process Based Multitasking: Independent processes are running concurrently. eg. FTP server.
  - ↳ Thread Based Multitasking: Multiple Threads in the same process running concurrently
    e.g. media player

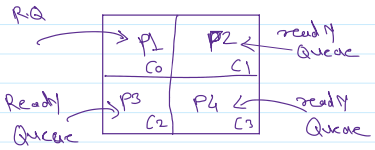
⑤ Multiuser System
- → multiple users execute multiple taks at same time (concurrently)
- → multi user command. linux
  - ① tty
  - ② who
  - ③ whoami
  - ④ who am i
  - ⑤ w

* Multi processor:







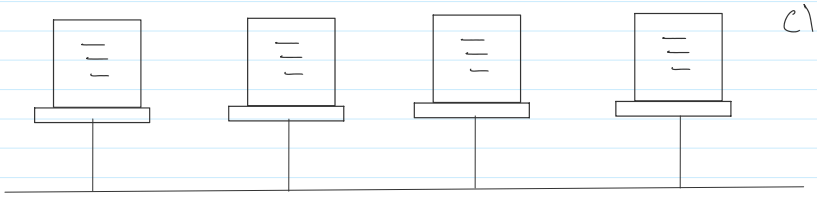


Asymmetric multip pro. (AMP) — Symmetric M processor (SMP)
↳ used some part in super computer

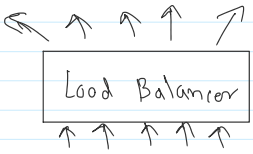
Ready Queue → all processes ready for CPU execution are stored in ready queue. & scheduler pick up next process from it.

terminal → lscpu
terminal → uname.

Distributed System



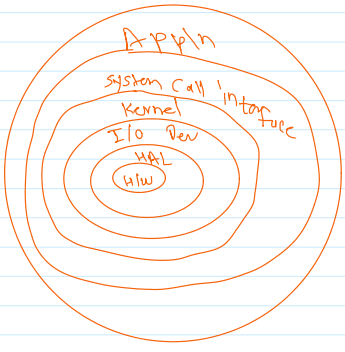
Cluster



It get

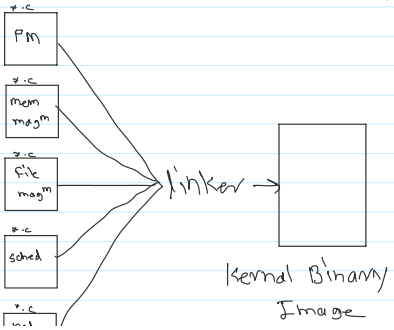
1. higher Computing
2. higher Memory
3. high Storage
4. higher Resource
5. Fault tolerant
6. reliable
7. higher availability / salability

* OS / Kernel Structure

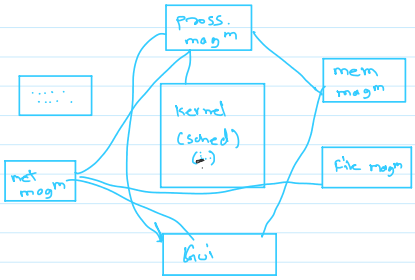


ex. unix, Linux. etc.

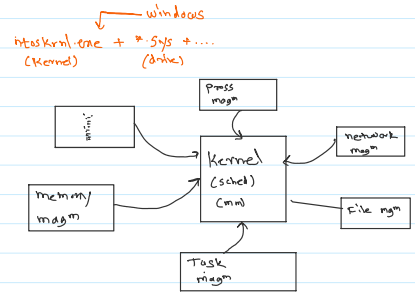
* Monolithic kernel



* micro kernel



* Modular kernel



↳ each fun - sep modules
↳ kernel is very small

Kernal Binary Image

- It is Faster
* In case of error / bug whole kernel crash
* if any modification in any component whole kernel rebuilt

→ each fun - server (sep- Process)
→ kernel is very small - robust
→ if error, only that Component fail
→ if modified, only the Component needs rebuild
* Servers Comm^n using IPC (msg passing) & hence slower

↳ each fun - sep modules
↳ kernel is very small
↳ modules are loaded & executed in kernel process itself. faster
↳ If updated, only module needs to rebuild.

* Hybrid Kernel

OS kernel is made of multiple kernels

* Mac OS X = BSD (UCB) + Mach (km)

* Linux = Static Components + Dynamic Components

1. scheduler
2. process mang^m
3. memory mang^m
4. I/o subsystem
5. System calls.

(vmlinuz /Boot)

1. device driver
2. File system drivers.

linux kernel object (*.ko)
/lib/modules