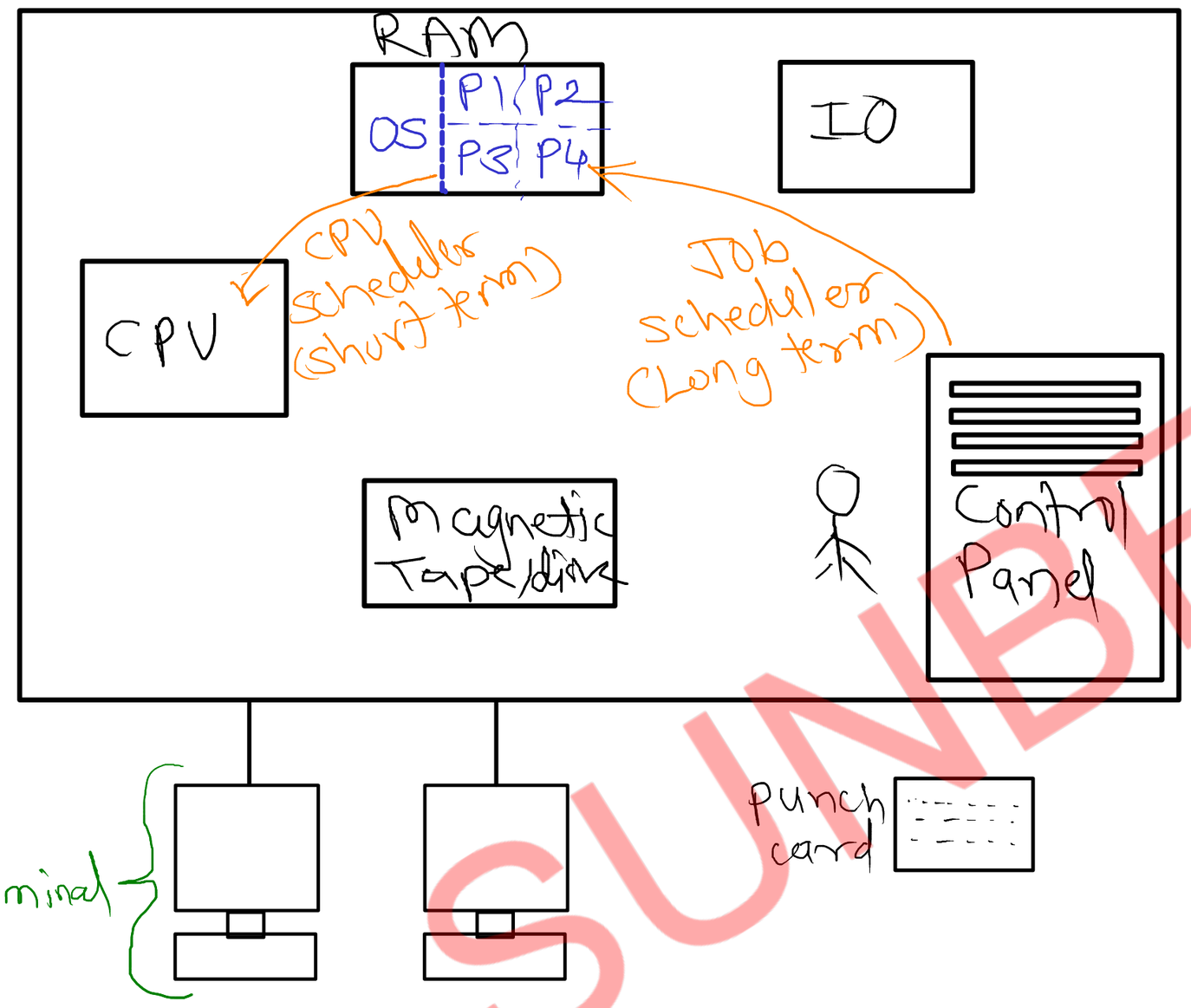
Types of Operating System

- Depending on type of targetted hardware OSs are classified as
    1. Desktop OS
    2. Server OS
    3. Handheld OS
    4. Embedded OS
    5. Real Time OS
    6. Distributed OS
    7. Mainframe OS

Types of Operating System



1) Resident Monitor
2) Batch system
3) Multiprogramming System
   - multiple programs are loaded into RAM

Degree of Multiprogramming
↳ no. of processes loaded into RAM.

CPU burst/time
↳ time spent on CPU

IO burst/time
↳ time spent to perform IO

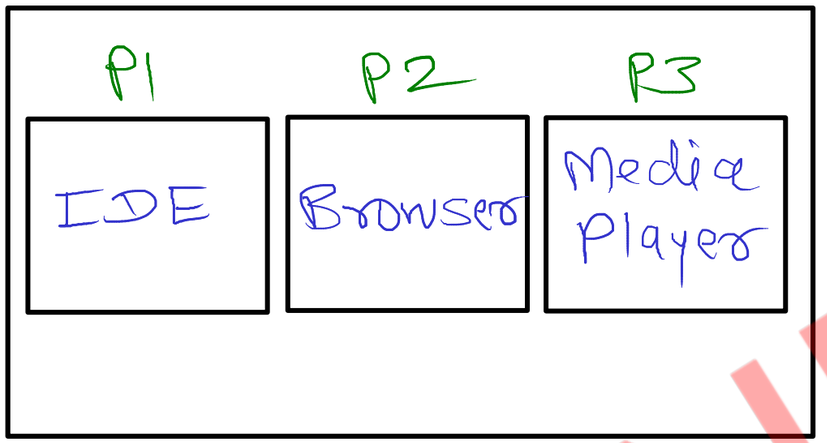
CPU burst > IO burst — CPU bound process

IO burst > CPU burst — IO bound process

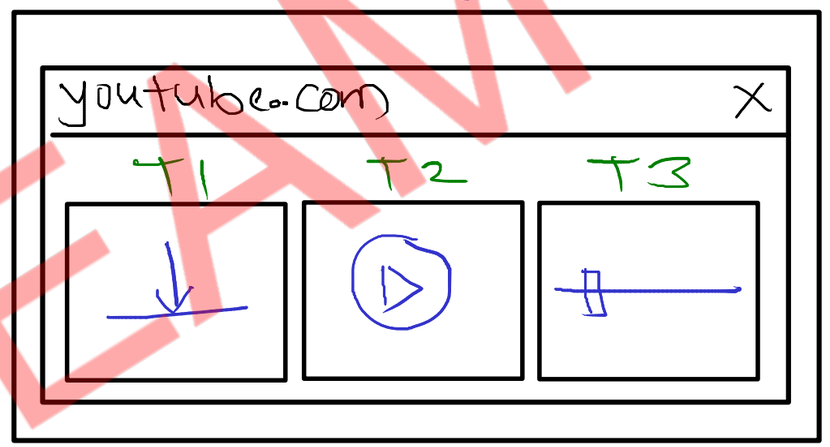
- mixture of CPU bound & IO bound processes is loaded into RAM

4) Time sharing System / Multitasking System
- CPU time is shared in all the processes of RAM
Response time < 1 sec
- there are two types of Multitasking

i) Process based Multitasking ii) Thread based Multitasking (Multithreading)



- multitasking within system



- multitasking which process

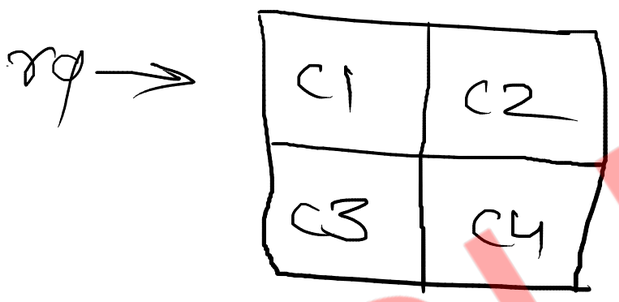
5) Multiuser system
- multiple terminals (Monitor + keyboard) are connected to single system
- multiple users can operate single system through these terminals

commands: whoami, who, w, tty

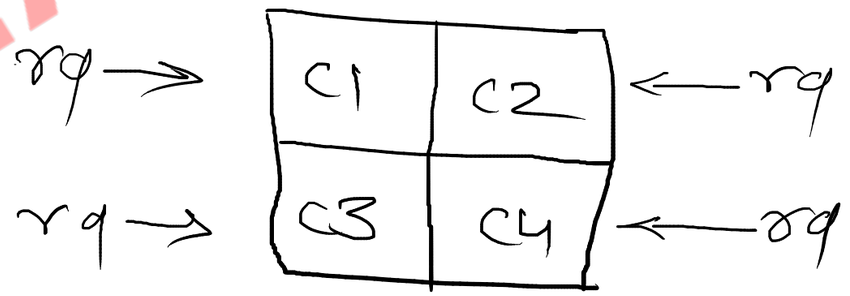
6) Multiprocessing system / Parallel System

- multiple CPUs are fitted on single chip, such processor chip is known as "multiprocessor" / "multicore"
- OS can schedule multiple processes at a time for every core, in short multiple processes can be processed parallelly
- there are two types of multiprocessing

1) Symmetric Multiprocessing



2) Asymmetric Multiprocessing



- Windows Vista } first time multiprocessing is supported.
- Linux 2.5+

OS's Data Structures

1. Job queue / Process list

- All processes of memory (RAM) are kept in this queue

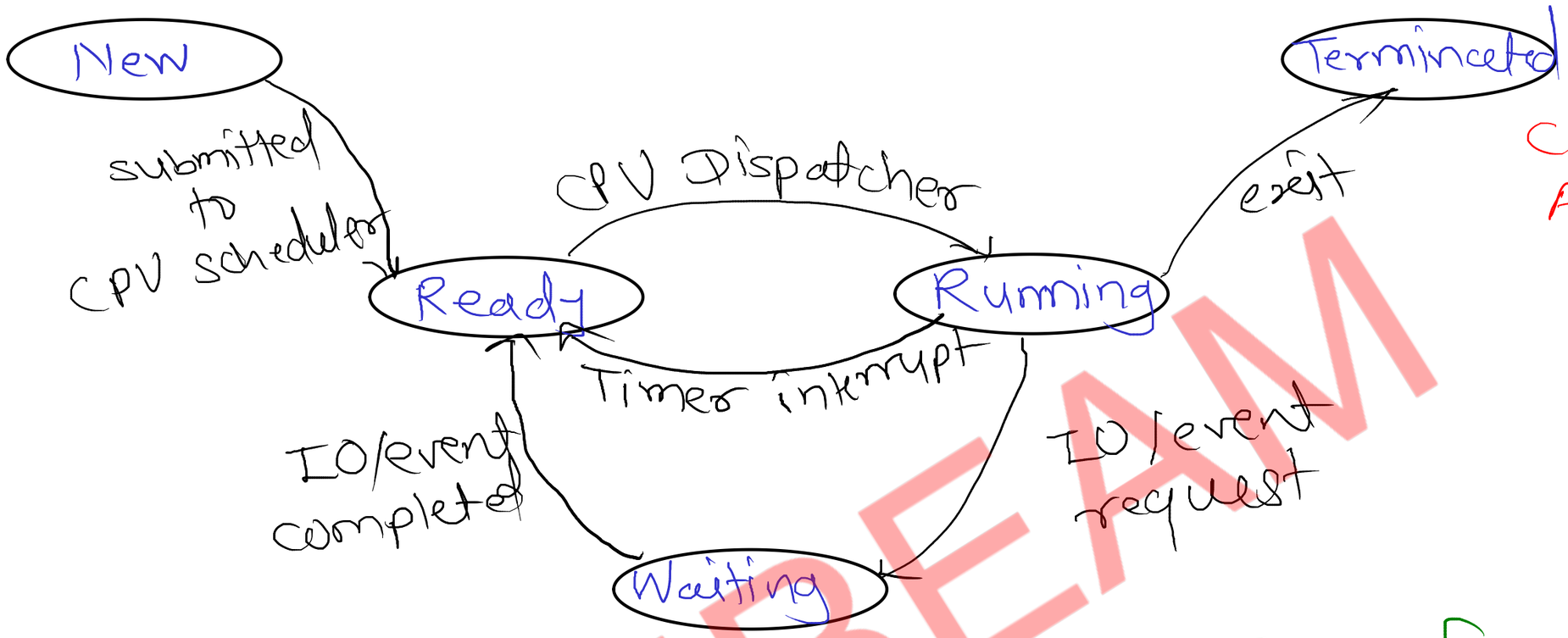
2. Ready queue

- processes which are ready to execute on CPU
- CPU scheduler always select process from ready queue

3. Waiting queues

- processes which are waiting for completion of I/O/event
- waiting queues are multiple (per device)

Process Life cycle



CPU scheduling Algorithms :-

1) FCFS
2) SJF
3) Priority
4) RR
5) fairshare

1) Running → Terminated
2) Running → Waiting
3) Running → Ready
4) Waiting → Ready

} volunteerily
} forcefully

Types of scheduling

1) Preemptive
CPU access is given to another process forcefully

2) Non Preemptive
CPU access is given to another process volunteerily

CPU Scheduling Criterias

1. CPU Utilization (Max)

- How much CPU is busy | idle

$$\frac{\text{Process time}}{\text{total time}} = \frac{70}{100} = 0.7$$

Desktop OS – 70% Server OS – 90%

2. Through put (Max)

- amount of work done in unit time
- How many processes are completed in unit time

3. Waiting time (Min)

- total time spent by process into ready queue for waiting to get CPU access.

4. Response time (Min)

- time from arrival of process into ready queue upto first time getting scheduled on CPU.

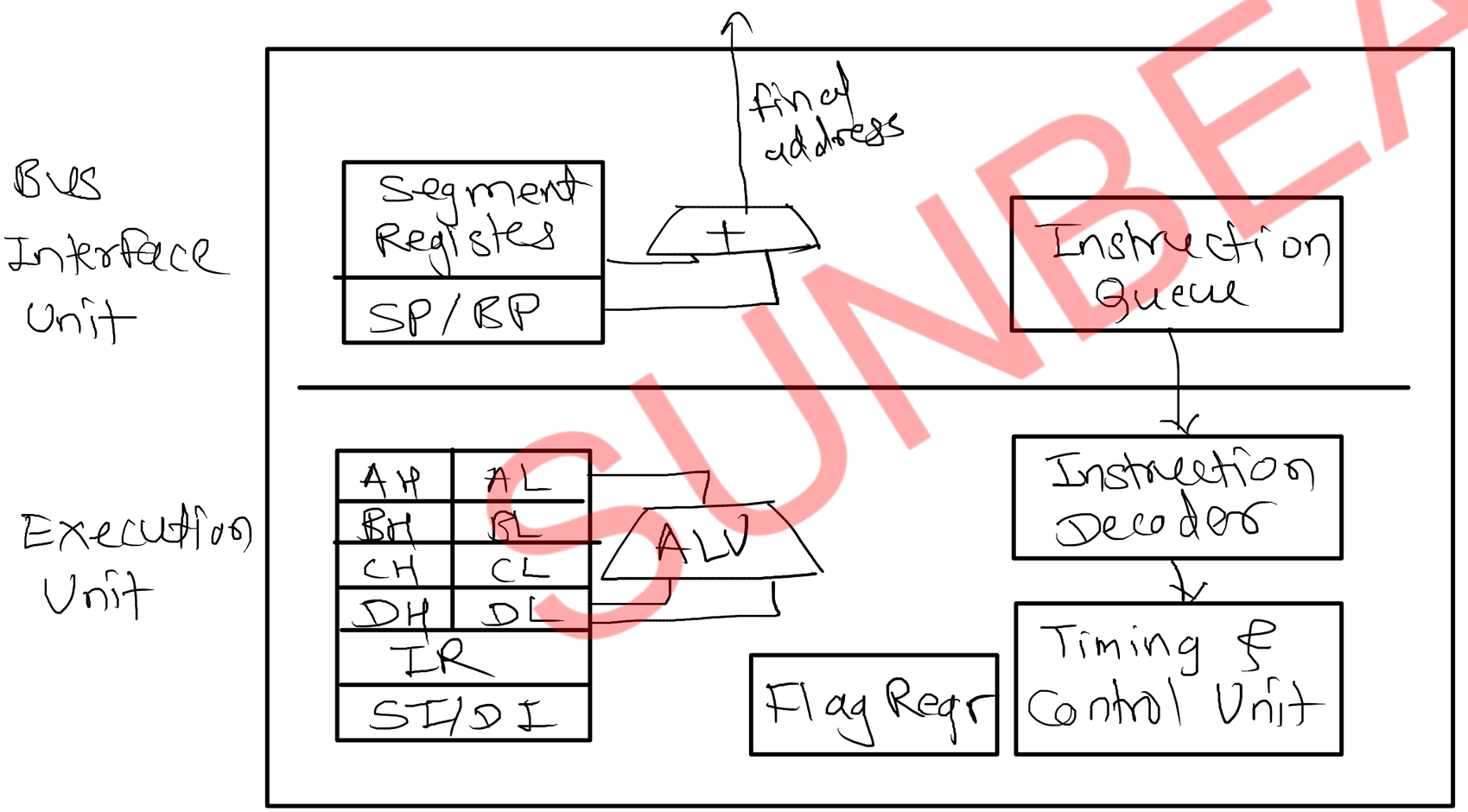
5. Turn Aroud Time(TAT) (Min)

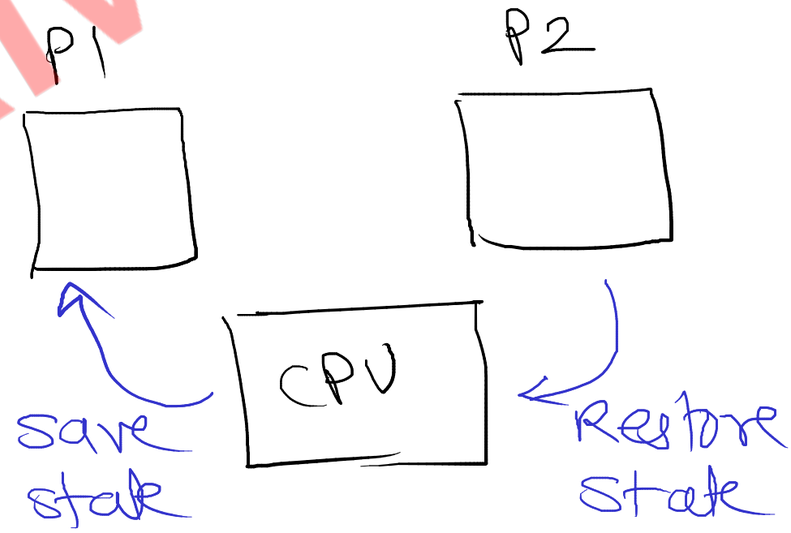
- total time spent by process into memory (RAM)

$$TAT = \underset{\text{time}}{\text{CPU waiting}} + \text{CPU burst} + \underset{\text{time}}{\text{IO waiting}} + \text{IO burst}$$

Context Switching

- process of unloading one process from CPU & loading another process on CPU
- changing the process of CPU
- Execution Context - values of CPU register





- CPU dispatcher is responsible for context switching

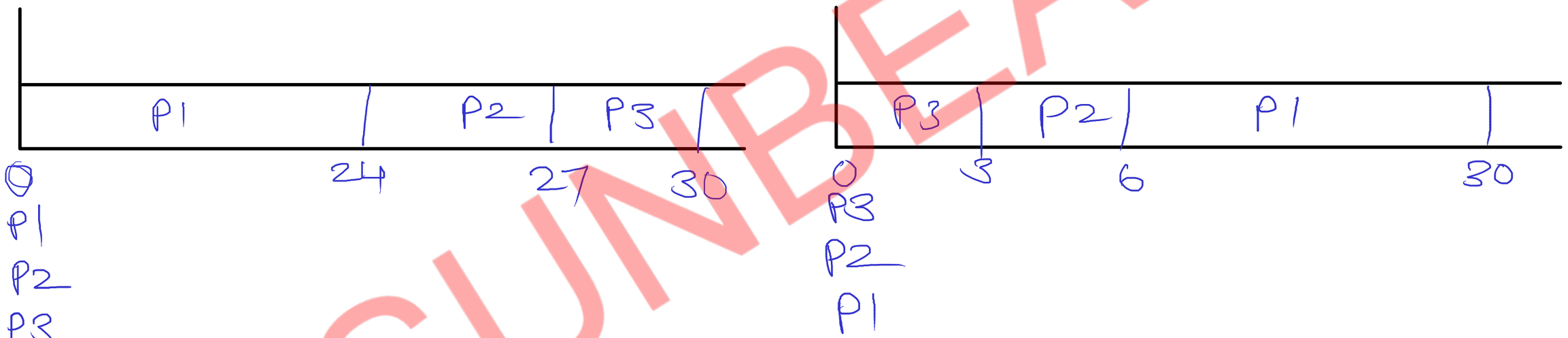
FCFS (First Come First Serve)

(Non Preemptive)

Process	Arrival	CPU Burst	WT	RT	TAT
P1	0	24	0	0	24
P2	0	3	24	24	27
P3	0	3	27	27	30

Process	Arrival	CPU Burst	WT	RT	TAT
P3	0	3	0	0	3
P2	0	3	3	3	6
P1	0	24	6	6	30

Gantt's chart



P1
P2
P3

P3
P2
P1

Convy's effect:

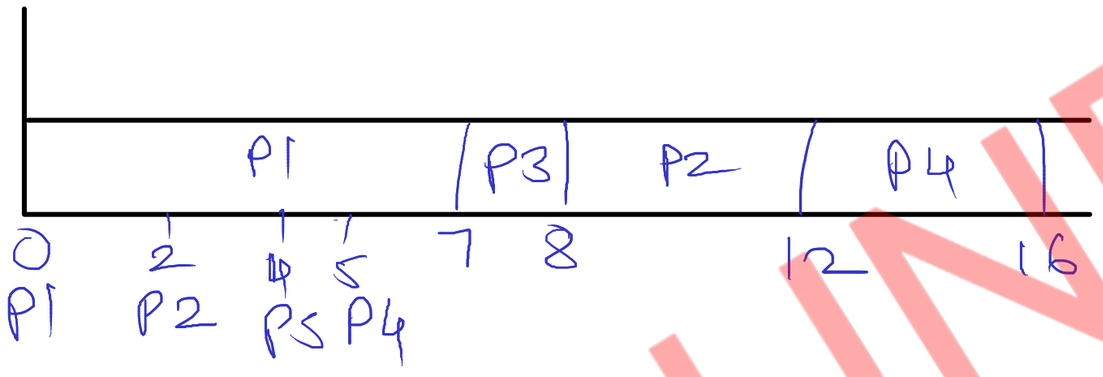
due to arrival of longer process early, all remaining processes has to wait for longer time

SJF (Shortest Job first)

(Non Preemptive)

Process	Arrival	CPU Burst
P1	0	7
P2	2	4
P3	4	1
P4	5	4

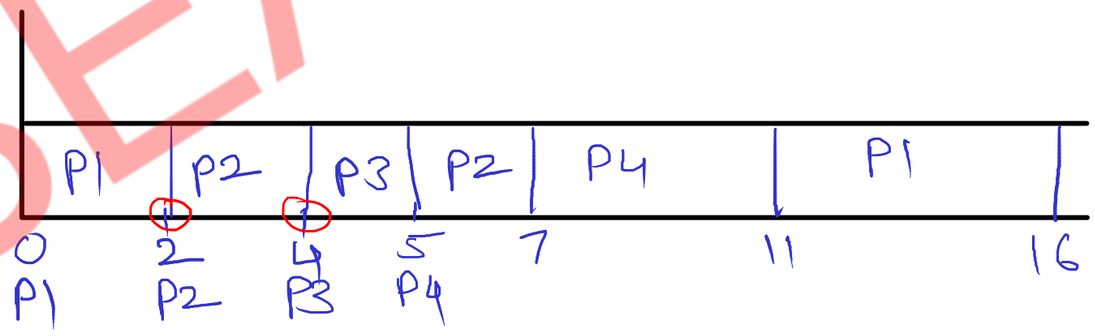
WT	RT	TAT
0	0	7
6	6	10
3	3	4
7	7	11



(Preemptive)
(Shortest Remaining Time First)

Process	Arrival	CPU Burst
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Remain time	WT	RT	TAT
5	9	0	16
2	1	0	5
	0	0	1
	2	2	6
		0.5	

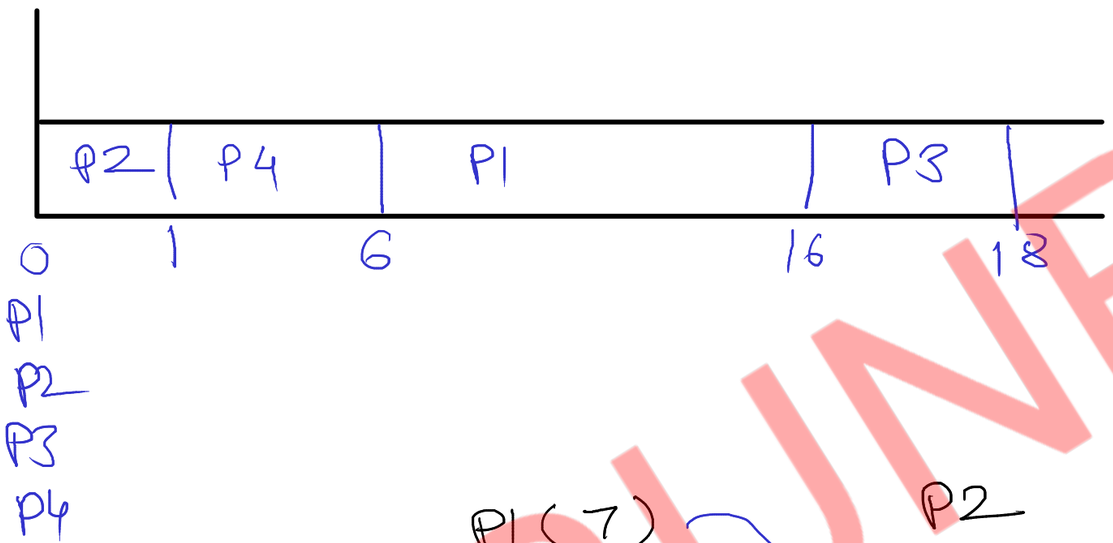


Starvation :- due to longer CPU burst process doesn't get scheduled for longer duration

Priority

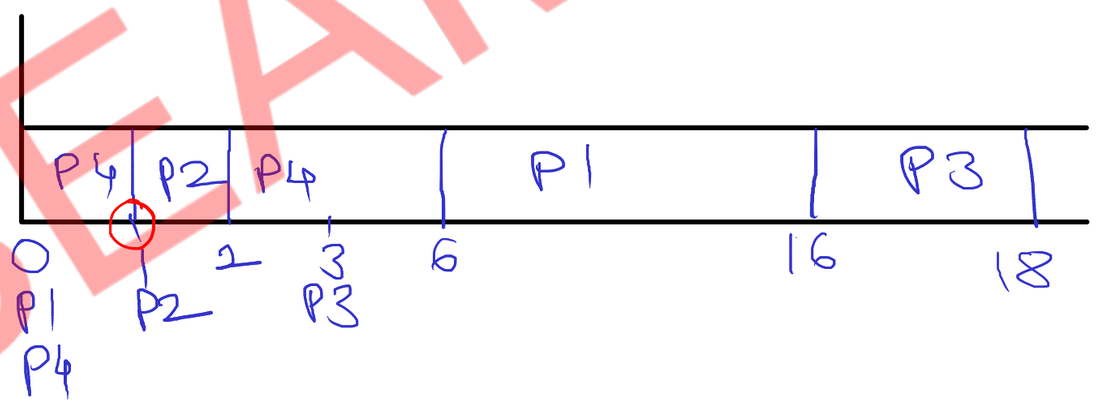
(Non Preemptive)

Process	Arrival	CPU Burst	Priority	WT	RT	TAT
P1	0	10	3	6	6	16
P2	0	1	1 (H)	0	0	1
P3	0	2	4 (L)	16	16	18
P4	0	5	2	1	1	6



(Preemptive)

Process	Arrival	CPU Burst	Priority	WT	RT	TAT
P1	0	10	3	6	6	16
P2	1	1	1 (H)	0	0	1
P3	3	2	4 (L)	13	13	15
P4	0	5	2	1	0	6



P1 (7)
P2 (4)
P3 (5)
P4 (6)
P5 (5)
P6 (5)

P2
P3
P5
P4
P6
P4
P1

Starvation:
due to less priority process don't get enough time to execute

Aging:
increase the priority of starved process gradually

RR (Round Robin)

Time Quantum = 20

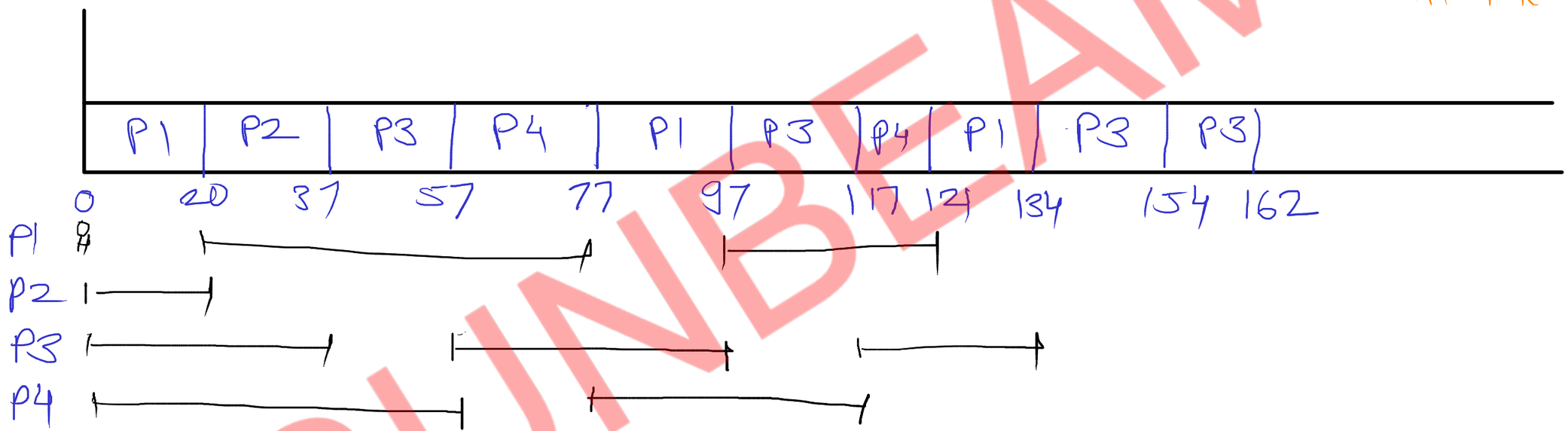
Process	CPU Burst
P1	53
P2	17
P3	68
P4	24

Remain time
53, 13X
X
48, 28, 8
4X

WT	RT
0+57+24	0
20+	20
37+40+17	37
57+40	57

TQ = 100
↳ behave like FCFS

TQ = 4
↳ CPU overhead will increase



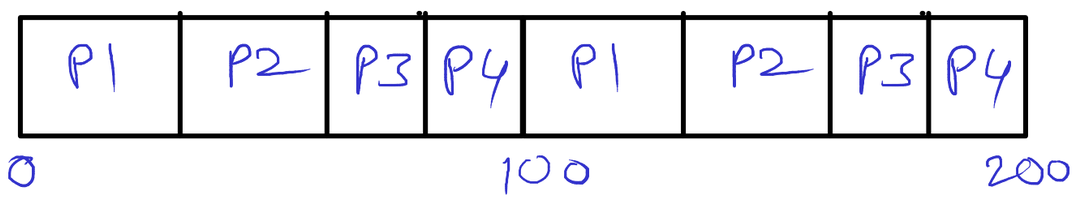
Fair Share

- CPU time is divided into time slices (epoch)
- some share of each epoch is given to the processes which are in ready queue.
- share is given to the process on the basis of their priority
- priority of every process is decided by its nice value
- nice values range ---> -20 to +19 (40 values)
  * -20 - highest priority
  * +19 - lowest priority

Process	Nice Value
P1	10
P2	10
P3	10
P4	10

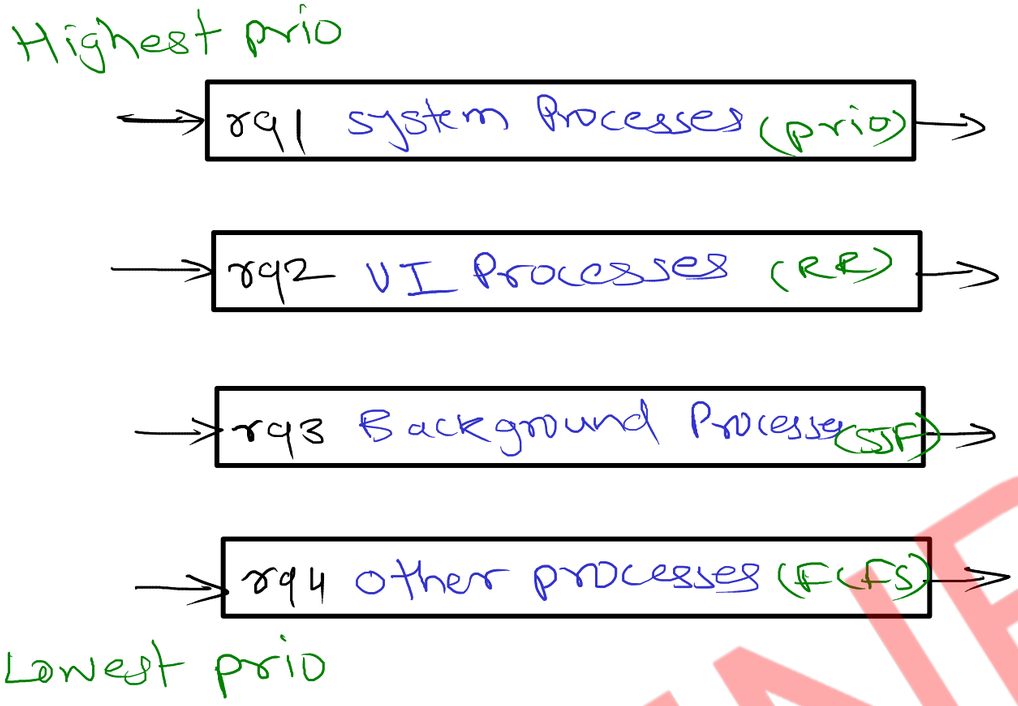
Epoch - 100

Process	Nice Value
P1	5
P2	5
P3	10
P4	10

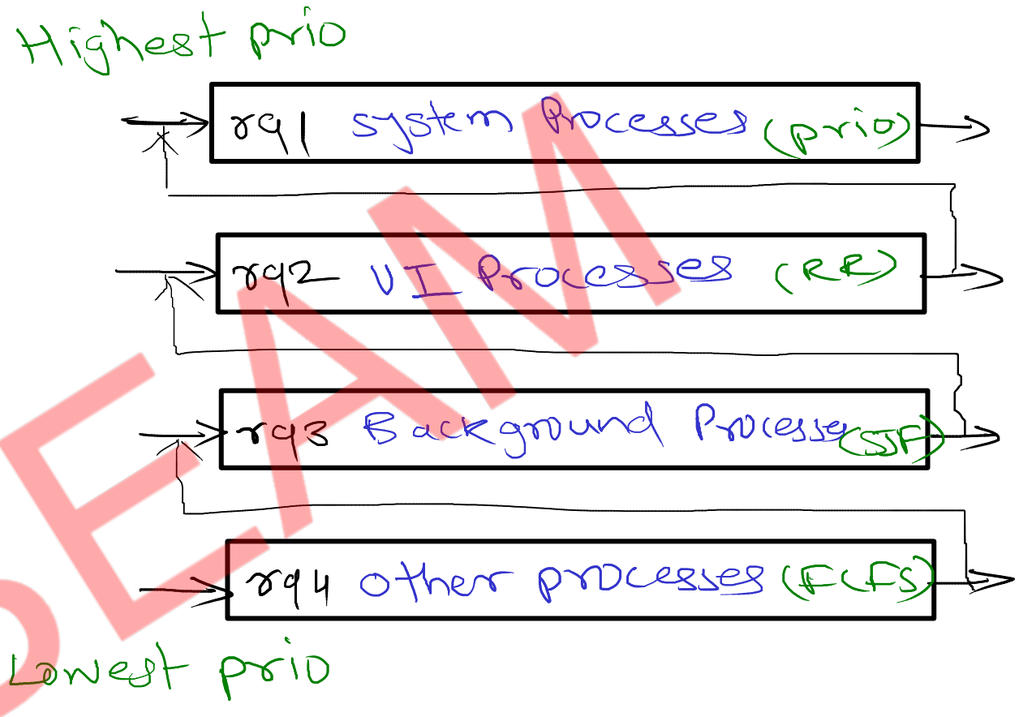


Completely Fair Scheduler (CFS)

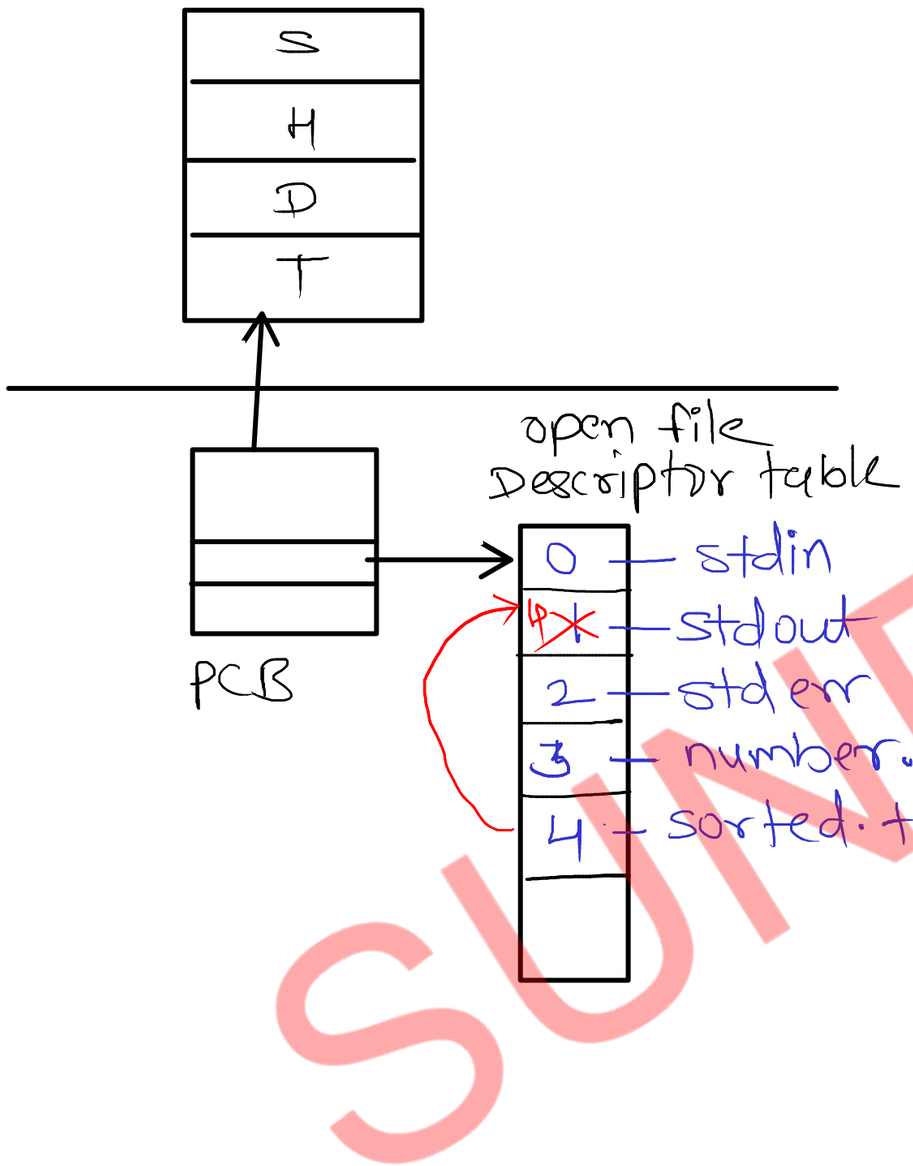
Multi Level Ready Queue



Multi Level Feedback Ready Queue



Redirection



Pipe

sort numbers.txt | uniq