OS - Round Robin and threads

1. To implement FCFS | Assn 1 - Tie Breaker

2. Round Robin SA | Assn 2 - RR

3. Threads

→ What are threads?

→ Single vs Multi core

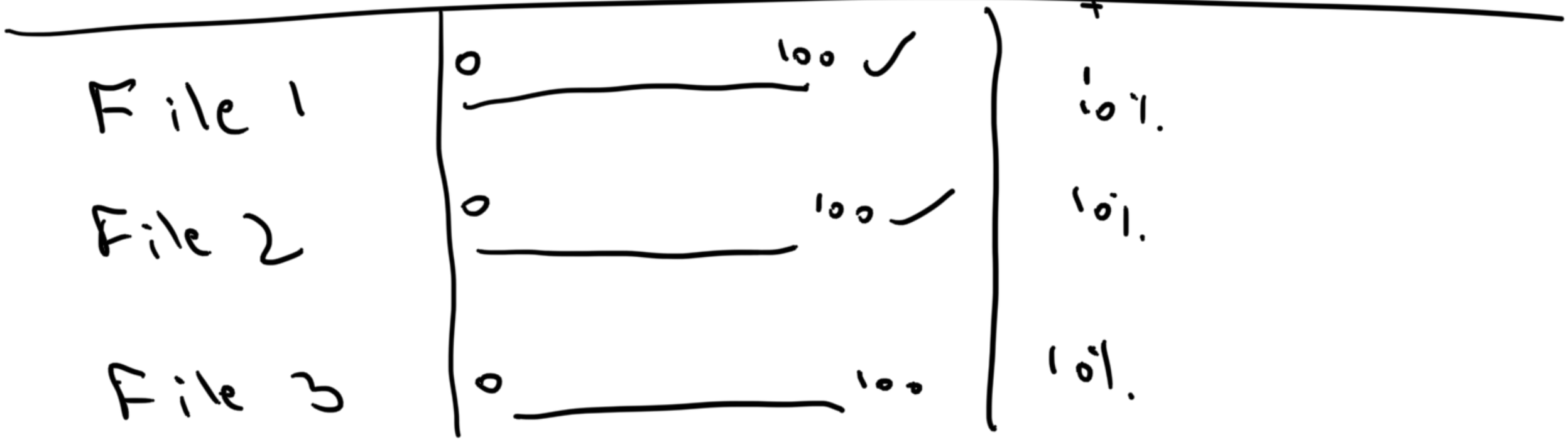
→ Concurrency vs Parallelism

→ Implementation using threads.

Round Robin

SRTF

- Starvation
- Unfair treatment of processes.



Instead of saying that we care about priority

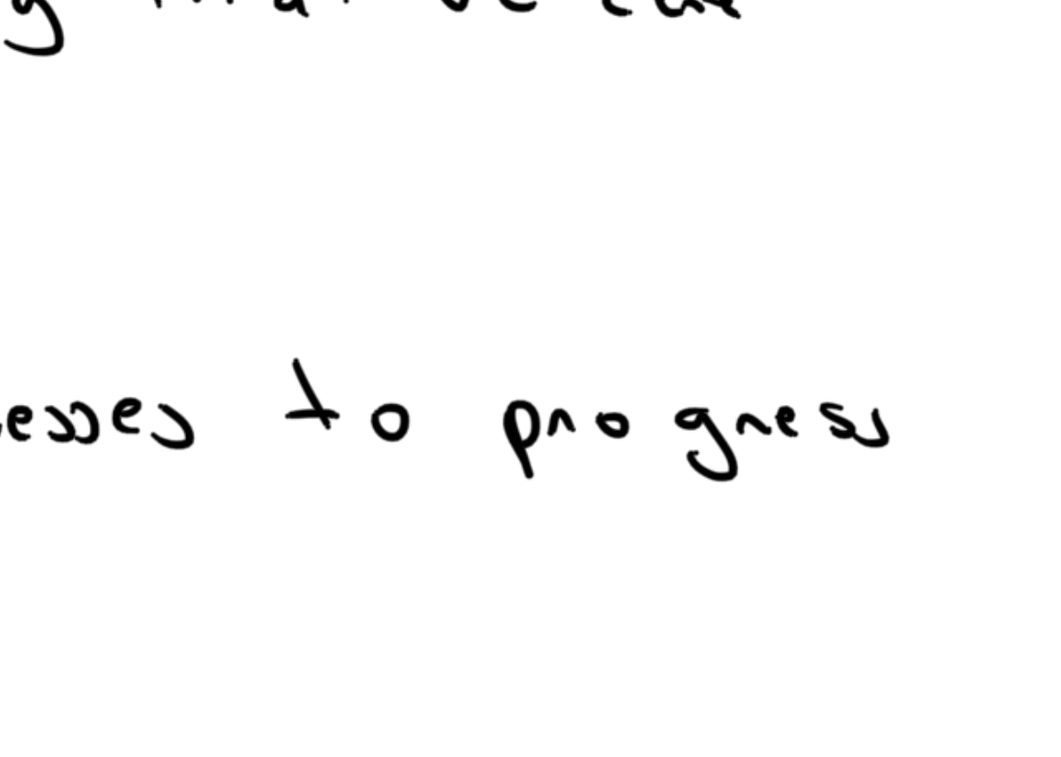
We want all processes to progress together.

Round robin

→ Pain treatment

→ t → time slice

→ time quantum



5 5 5

Large wait time

P1 - 0 s

P2 - 5s

P3 - 10s

running_time rt

- burst time b
- time q

P1 - 1s

P2 - 3s

P3 - 4s

q → 2s

P1	P2	P3	P2	P3
0 — 1	1 — 3	3 — 5	5 — 6	

if $T_B < T_Q$

$$T_R = T_B$$

else

$$T_R = T_Q$$

$$\boxed{T_R = \min(T_B, T_Q)}$$

Algorithm :

① If no process is running,
pick the next process from

Ready Q

② Process runs for $\min(T_B, T_Q)$

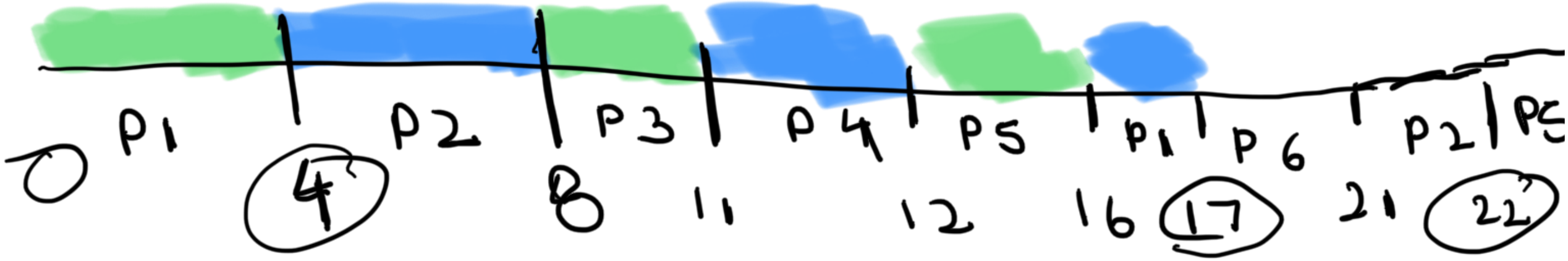
③ Pause running process & pick
the next process from the Q.
→ Add it to the end of the q

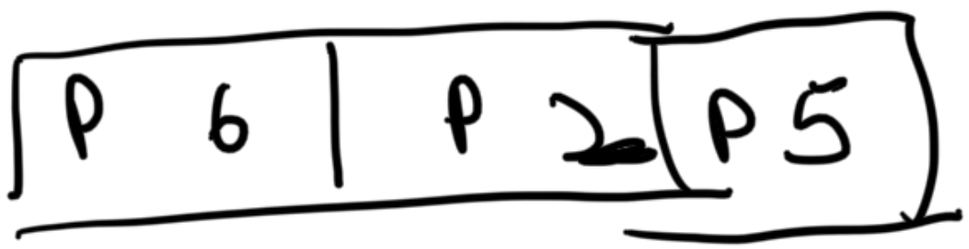
④ Repeat until no process is left

Scalen Topics

	Arrival	Burst
P1	0	0
P2	1	2
P3	2	0
P4	3	0
P5	4	1
P6	6	4

$t_q = 4$





Advantages

* Starvation Free
* Fair allocation of CPU
* Easy to implement

— T_B not needed

Assignment - II

→ Implement Round robin

Disadvantages

① A lot of context switching

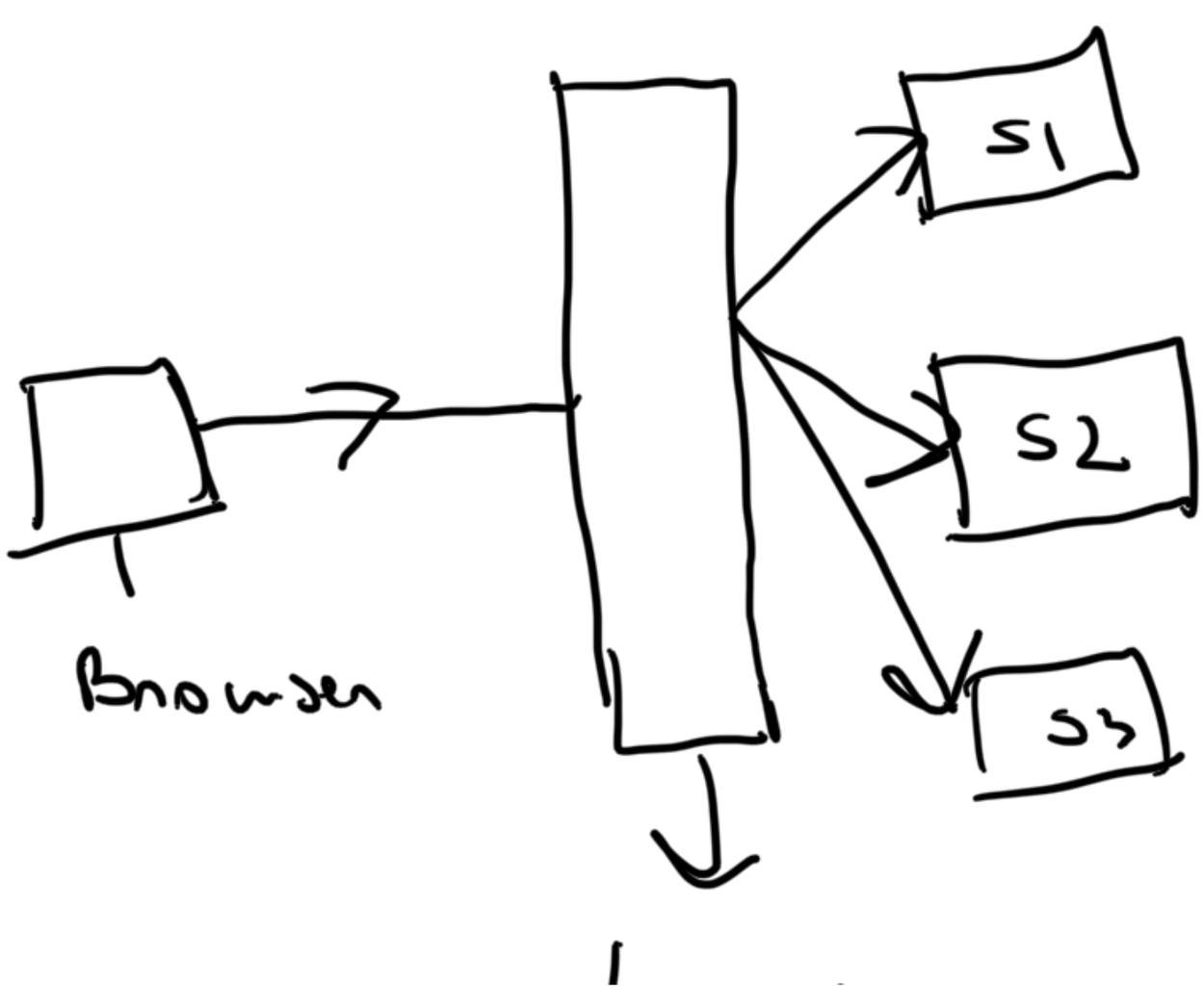
② Low throughput

③ Priority

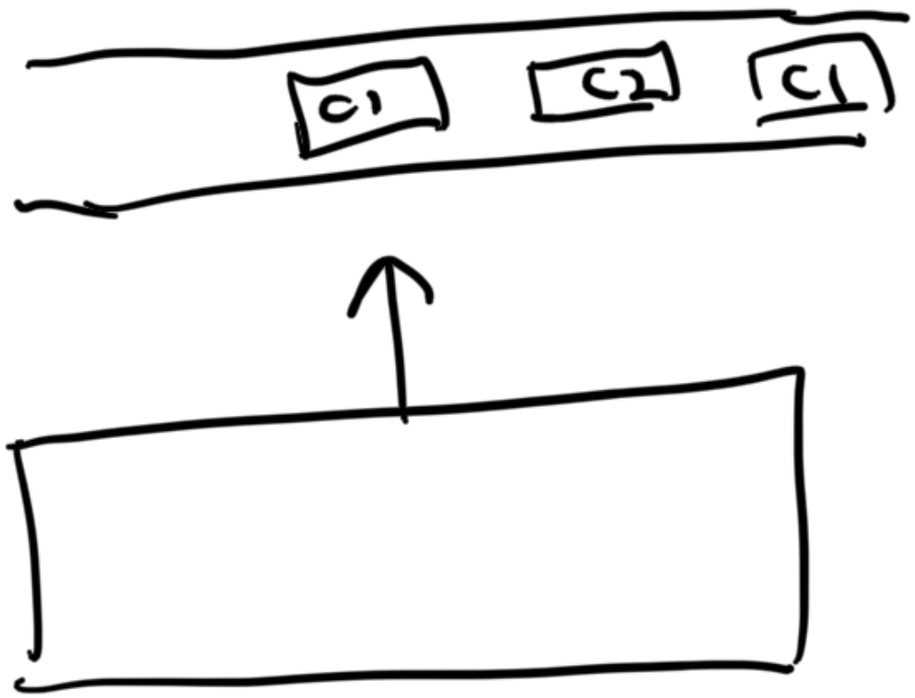
with higher T_q

RR → FCFS

Load balancers



Load
Balanci.

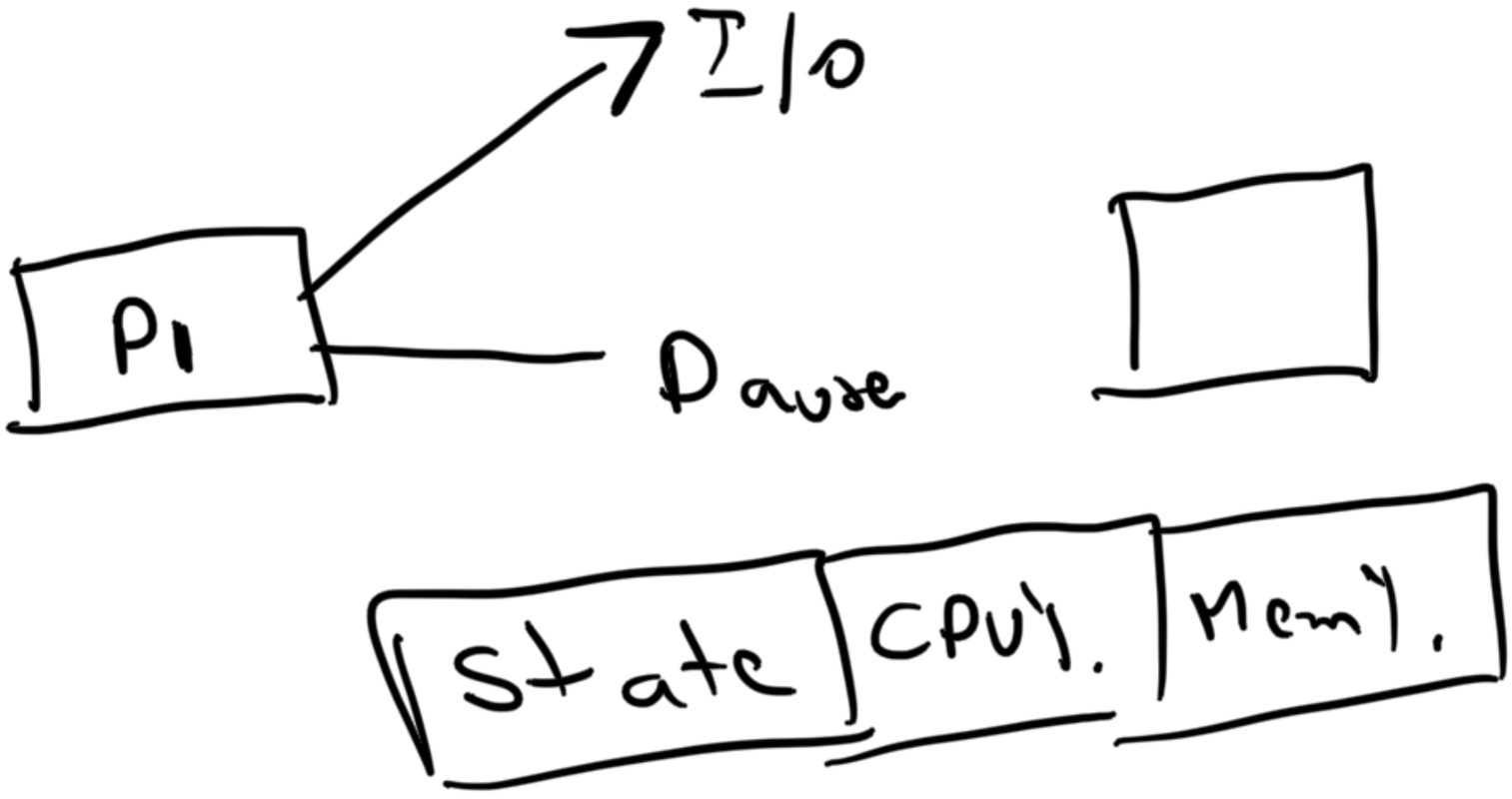


BREAK

5:50 : 5:55

10:25

Zombie process



① Estimation

- past trends

(2) Padding - loss

Threads

→ Light weight process

Process

Do unload

Process

Viewing

Downloader

Program

Disk

Mem

CODE

Data

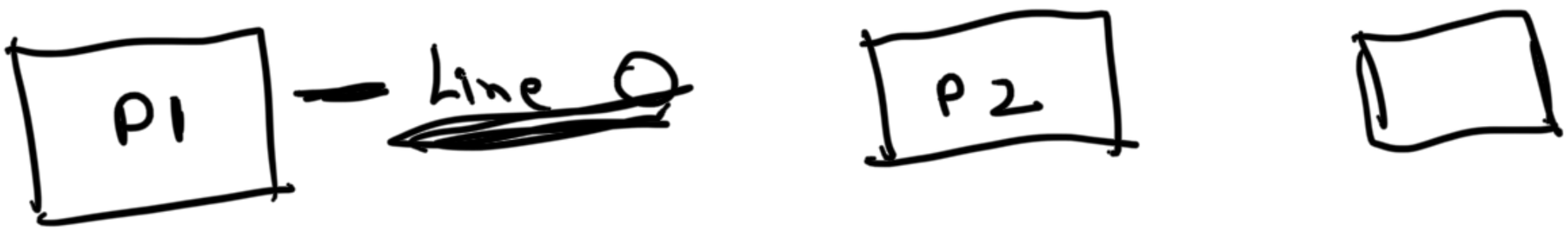
Resources

CODE

Data

Multiple processes

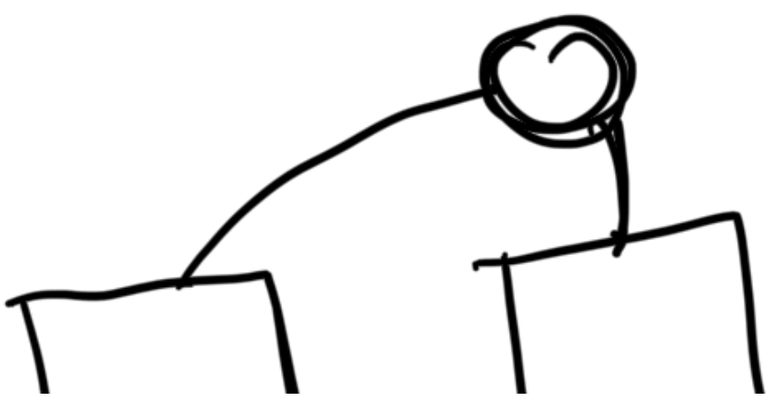
→ Less performant



```
#include <iostream>
using namespace std;
void function1();
int main()
{
    function1();
}
void function1()
{
    function2();
}
```

Process 1 | I/O

1. Creating new process takes time
2. Context switching is slow



Threads

CODE
DATA
Resources
Shared
PC
Stack
PC
Stack
PC
Stack
Multi threaded

download() line 190

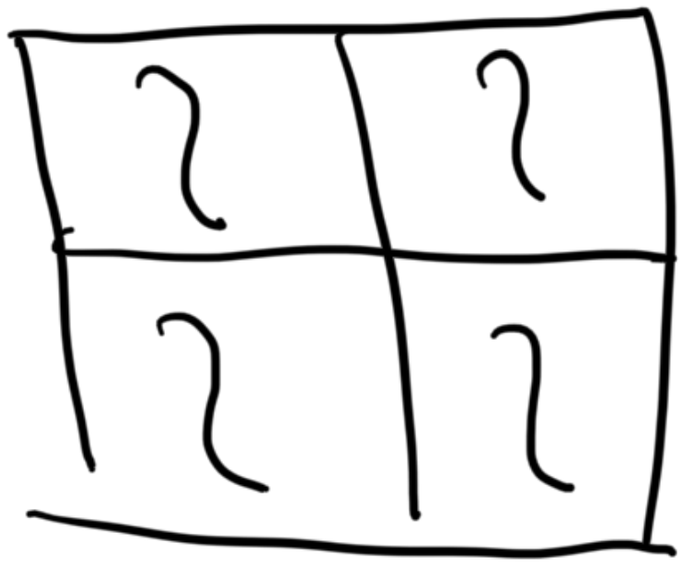
paint() line 200

* Thread is the final unit of execution of a CPU

16 core processor

→ 16

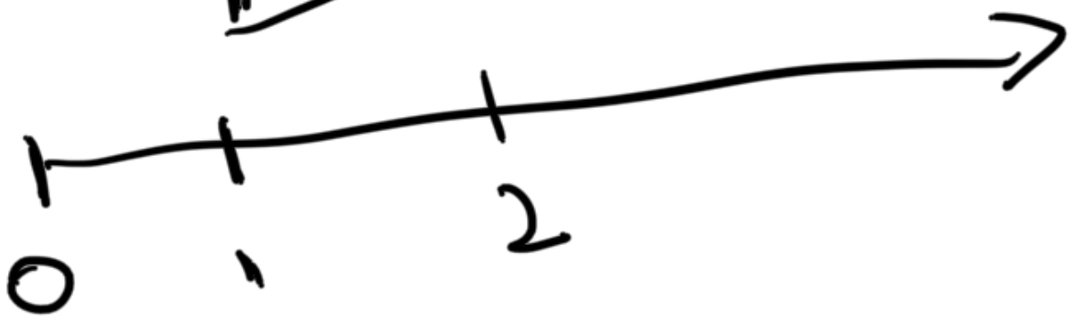
Uni core multi core → 16 = 16

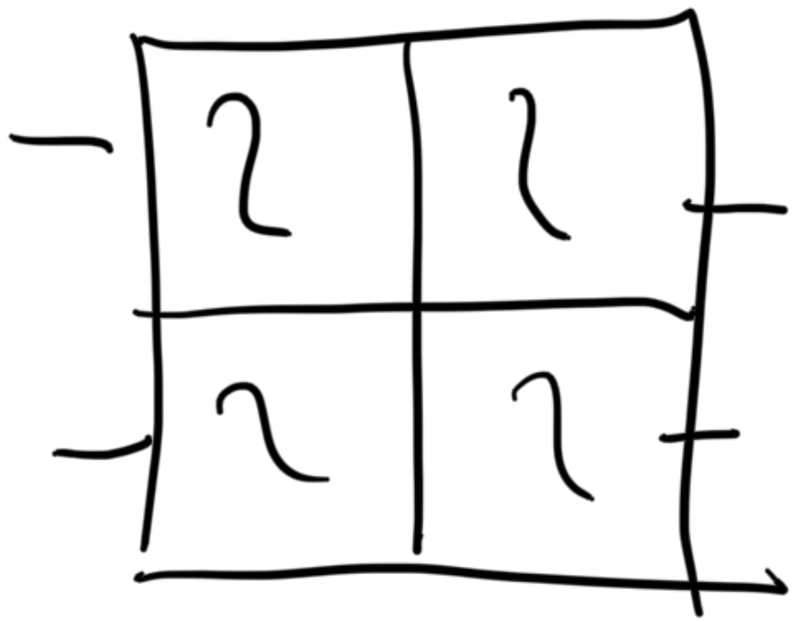


2 - 2

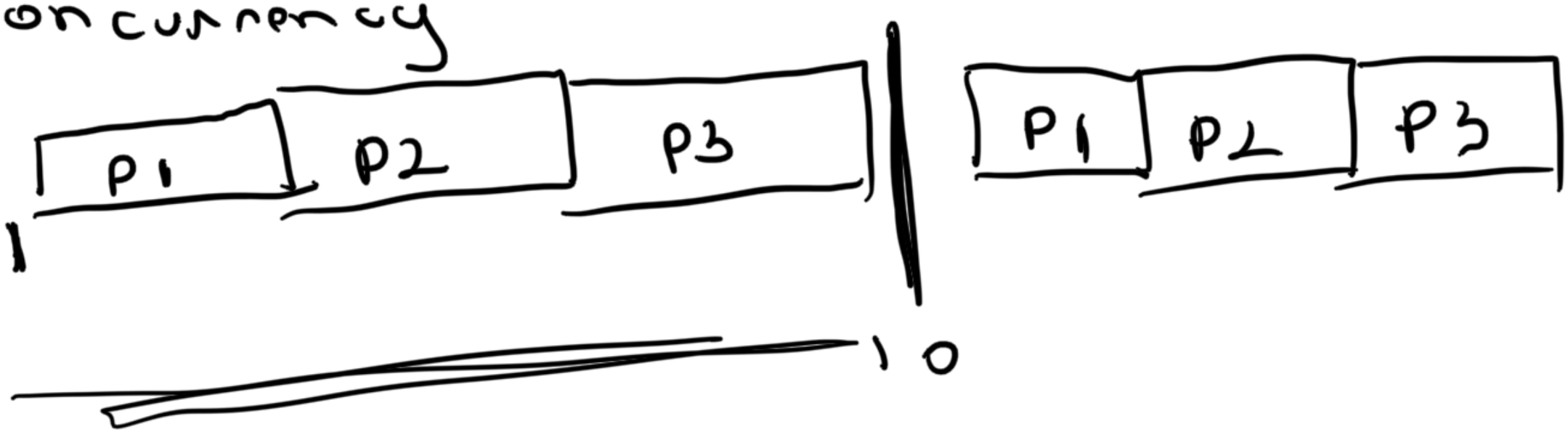
- 100 -100

Parallelism





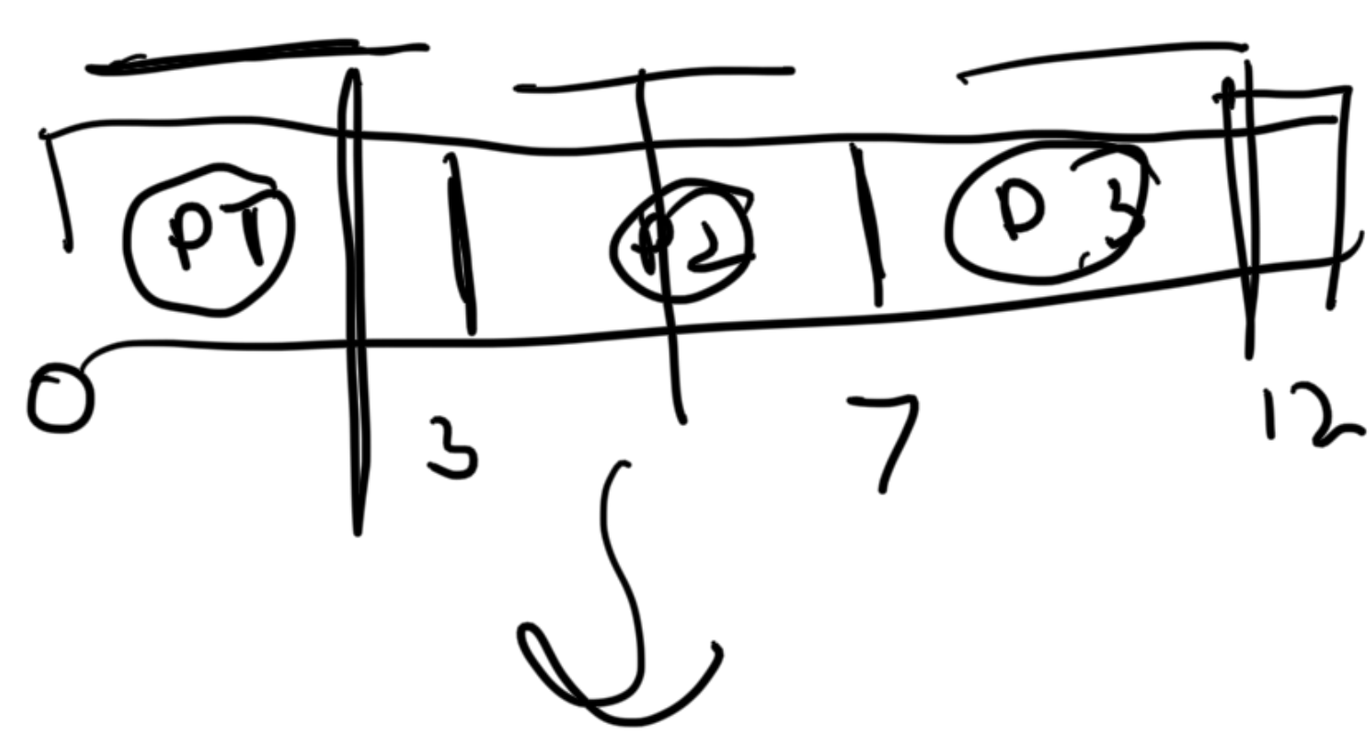
Concurrency

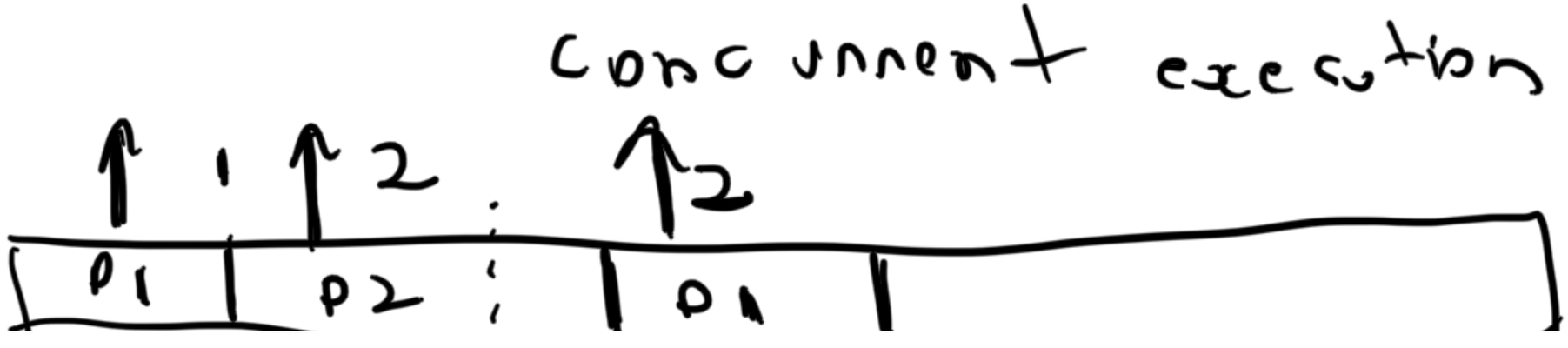


At time t, I can have progress

on different state of execution of multiple processes.

① P1 – 3
P2 – 4
P3 – 5





P1 → 30'

P2 → 50'

At same time but not at same instant

Threads

→ unit of CPU execution

→ Light weight

→ shared memory

→ multiple tasks in same process

Web Browser

1. Create a new thread

2. Chrome → creates a new process for each tab
   - → performance
   - → memory consumption

3. Mozilla Firefox — 4 process.

1
2
3
4
CPU