Operating Systems - I

Agenda

- Intro to OS
- Uni vs multi programming
- Process
- CPU scheduling
- Scheduling algorithms
  - First come first serve (FCFS)

- Shortest Remaining time First (SRTF)

OS → Linux

Distros ↓

→ Ubuntu

→ Kali

→ Windows - 11

→ Mac Os

What is an OS?

Interface to hardware

Developers

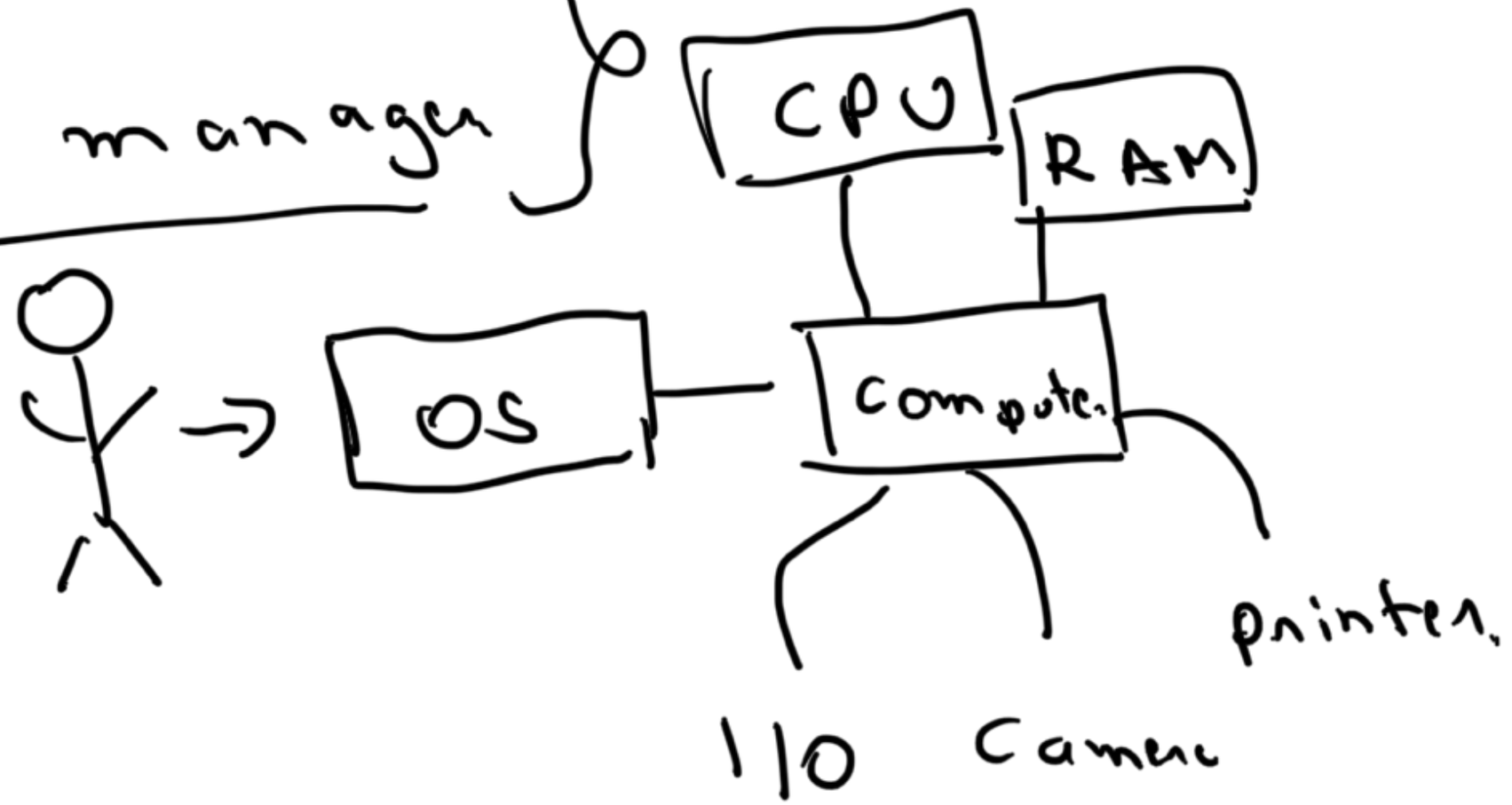
sys.out

import os

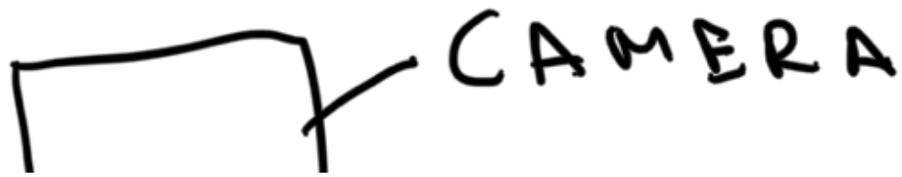
os.cwd()

① Low level API

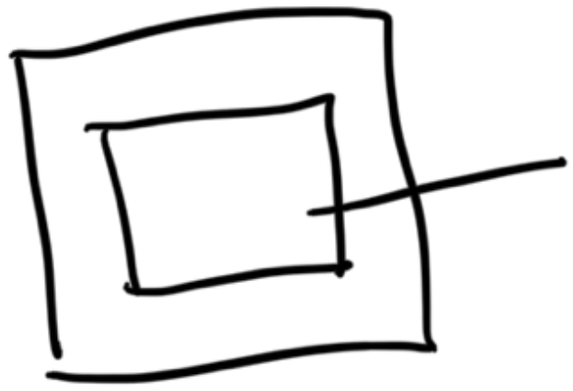
② Resource manager



Zoom



Meet



1 core

1 process

Zoom

Uniprogramming

↳ 1 process can run at a time

* MS-DOS
* ATM
* Washing machine
* Smart devices | IOT

Multiprogramming

↳ run multiple programs at the same time

- PCs
- mobile
- servers

| Multitasking
| Multiprocessing

Types of multiprogramming

P1 P2

* Based on # users

- single user OS ✓
- multi user OS
  - ↳ servers

→ Windows / Linux ↓

→ Macos

AWS

AVS

192.168.1.1

192.168.1.2

Windows

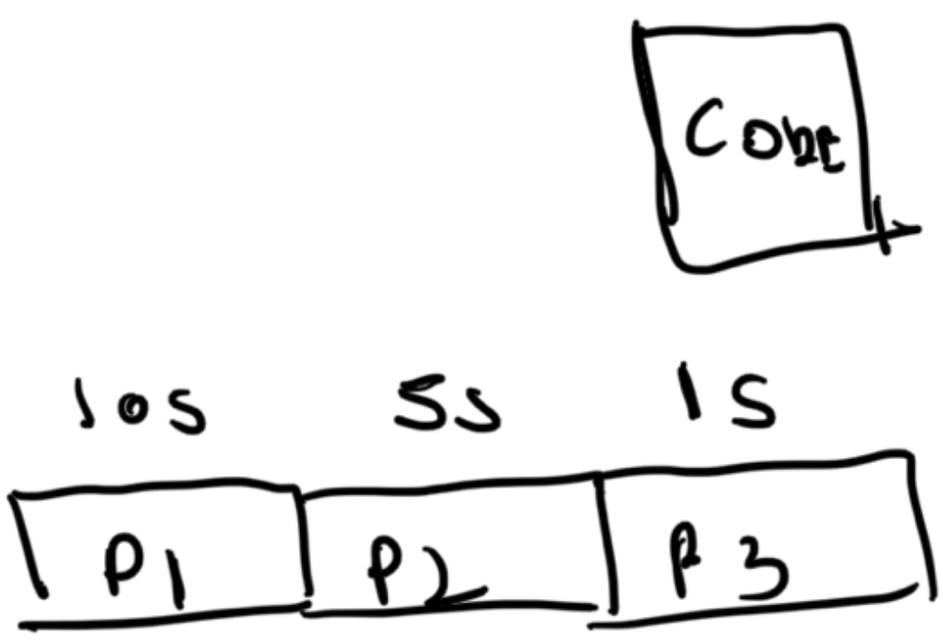
IIS

Tiny OS

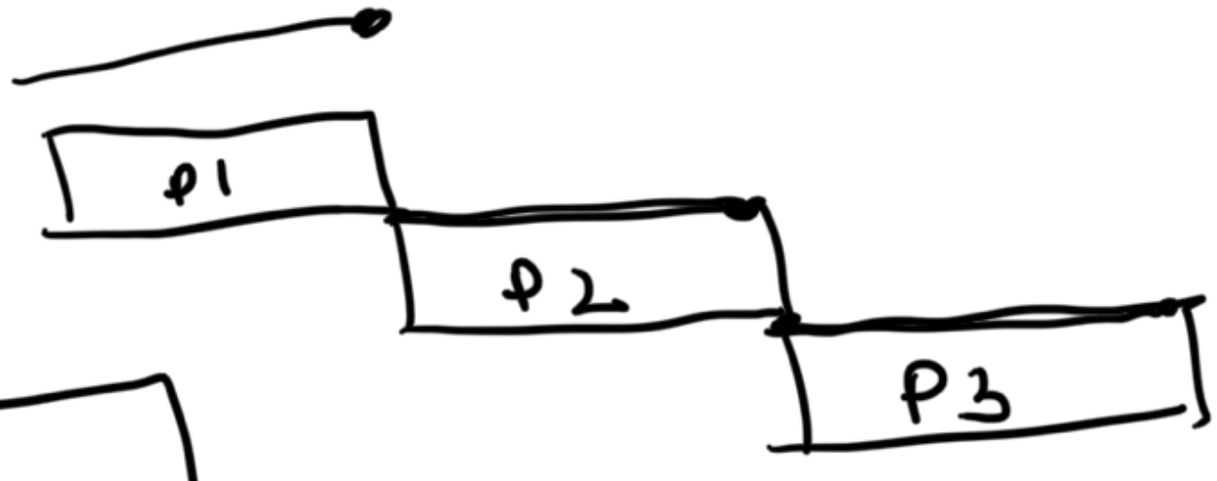
1. Understand what goes on behind the scene

2. How my application interacts with the OS!

On type of scheduling



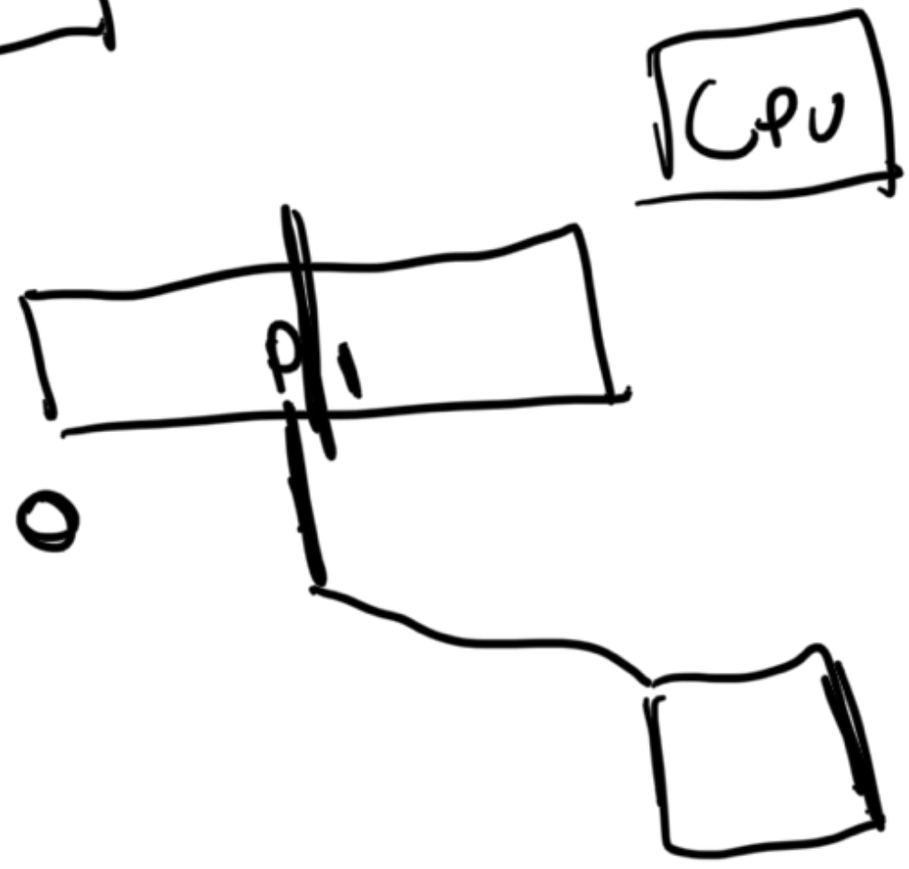
Sequential



- Non - Preemptive

scheduling

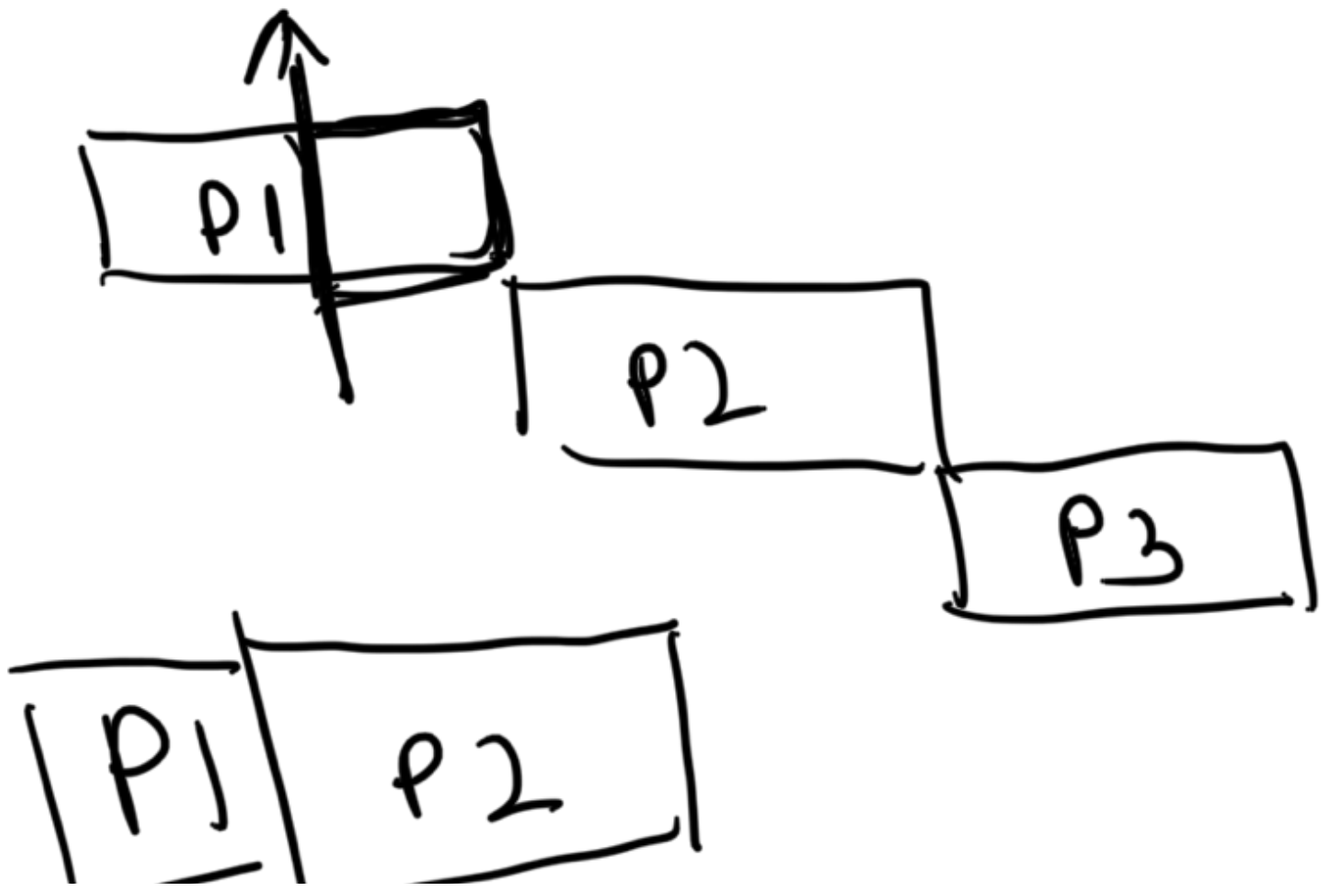
Preemptive



P1 → x+y=10 } CPU

→ Solve x

→ Send x over the network



Preempt → Pause a process

→ Start another one

Zoom → Disk

↓

open it → Process

Program → Process

Class → Objects

Salem → Student, Instructor } class

Process {

PID

runtime

Variables

name

state

memory

resources

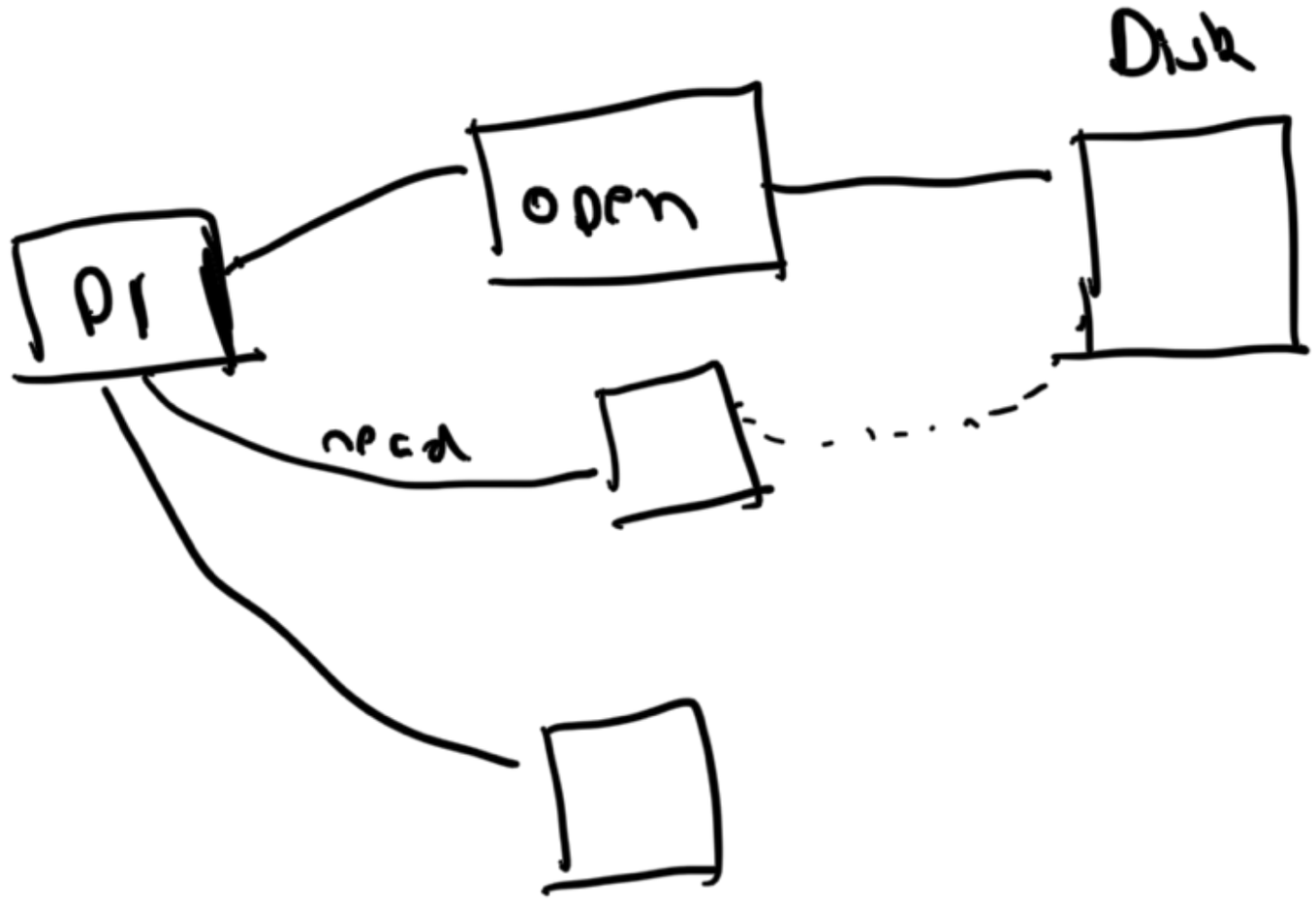
}

→ Process Control [PCB]

Block

① I/O bound process

→ wait for IO operation to complete



② CPU bound

→ finds the 1000th digit in Pi π

→ Calculate prime no. till 1m

③ I/O + CPU

---

① OS — Low level APIs

→ Resource manager

② Uni prog. — one program at a time

Multiprog. — multiple "

③ Single user — PCs

Multi user — Servers

④ Non-preemptive — cannot pause a process

preemptive — can pause & replay a process

⑤ Processes — PCB

— I/O Bound

— CPU Bound

6:01 : 6:05

: 10:35

PCB

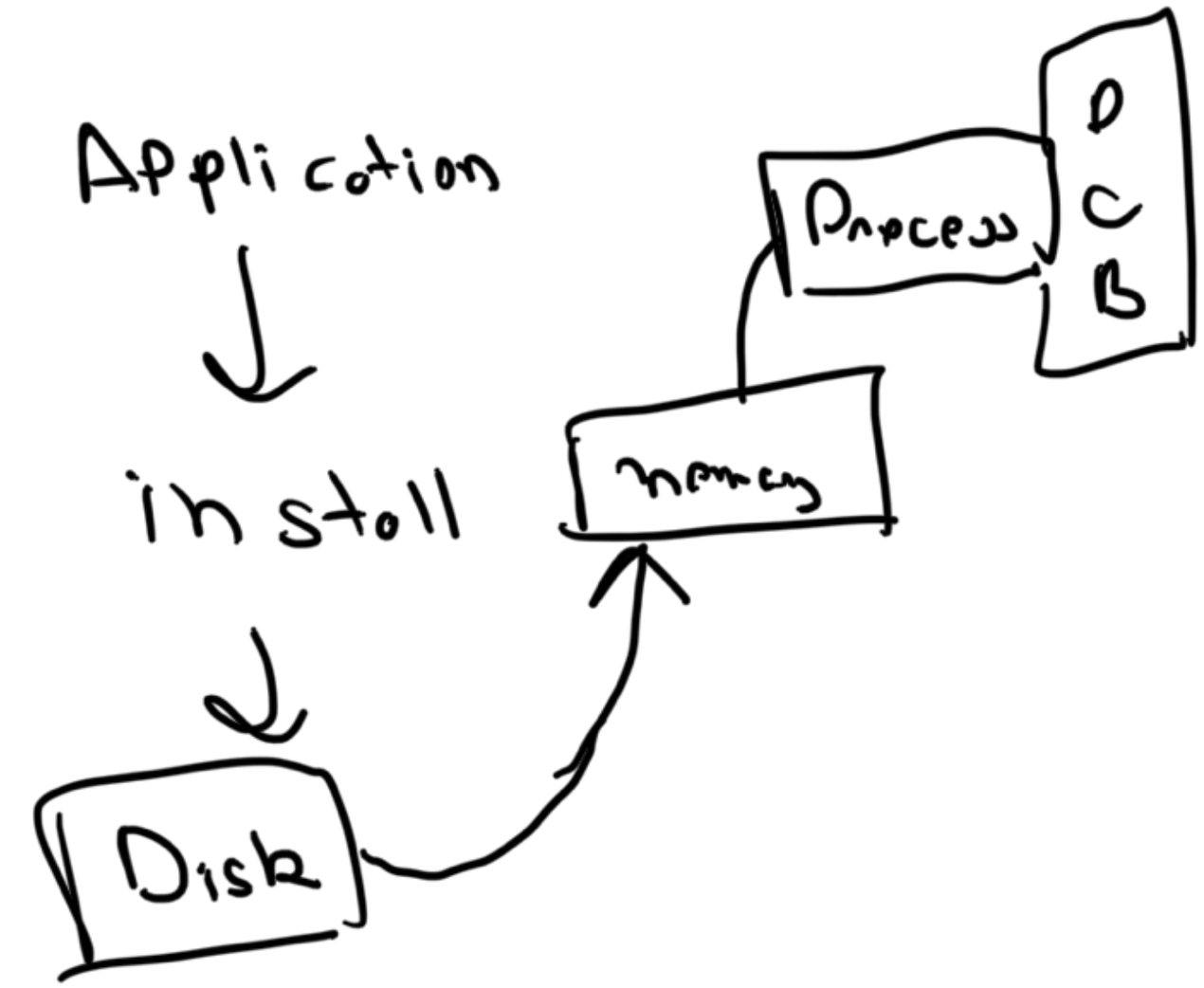
PCB

Process {

- id
- CPU
- memory
- variables
- status

}

Code → Application → install



CPU | Process Scheduling

(1) Priority

(2) Time taken
- Burst time

(3) First response time

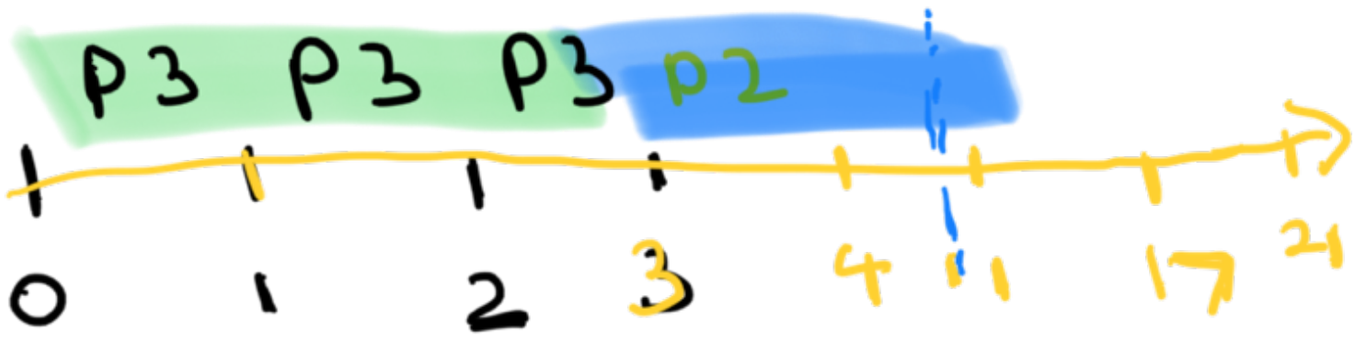
waiting time

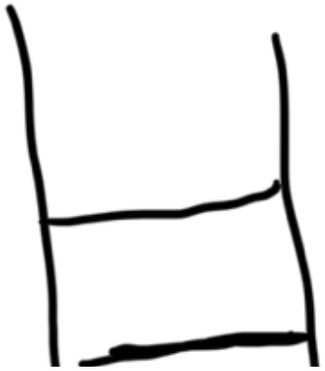
① First come First Serve

PID	Arrival	Burst time
P1	2	6
P2	1	8
P3	1	3
P4	4	4

Tie breaker
① Burst time
② PID

①





- FIFO
FCFS

Ready

Q.

P3 → P2 → P1 → P4

Non - preemptive

Assignment

→ Implement FCFS

Process

{ pid → PCB

List<Process>

arrival
burst

Output

Sorted | FCFS based
schedule

Output {
index
Process
}

→ Any language you want
→ Python

Python

1. Create a DS for Process (PCB)
2. Input — List<Process>
3. Output — FCFS schedule

Disadvantages

Advantages

① simple

② No starvation

① Not efficient; low t/p
tabes more
time

② Higher wait time

③ High CPU idle
time

Shortest remaining time first (SRTF)

PID	AT	BT	Remaining time

FCFS

- process completion on t=0

SRTF

- process completion
- new process arrival

Algorithm

① If queue is empty, CPU idle

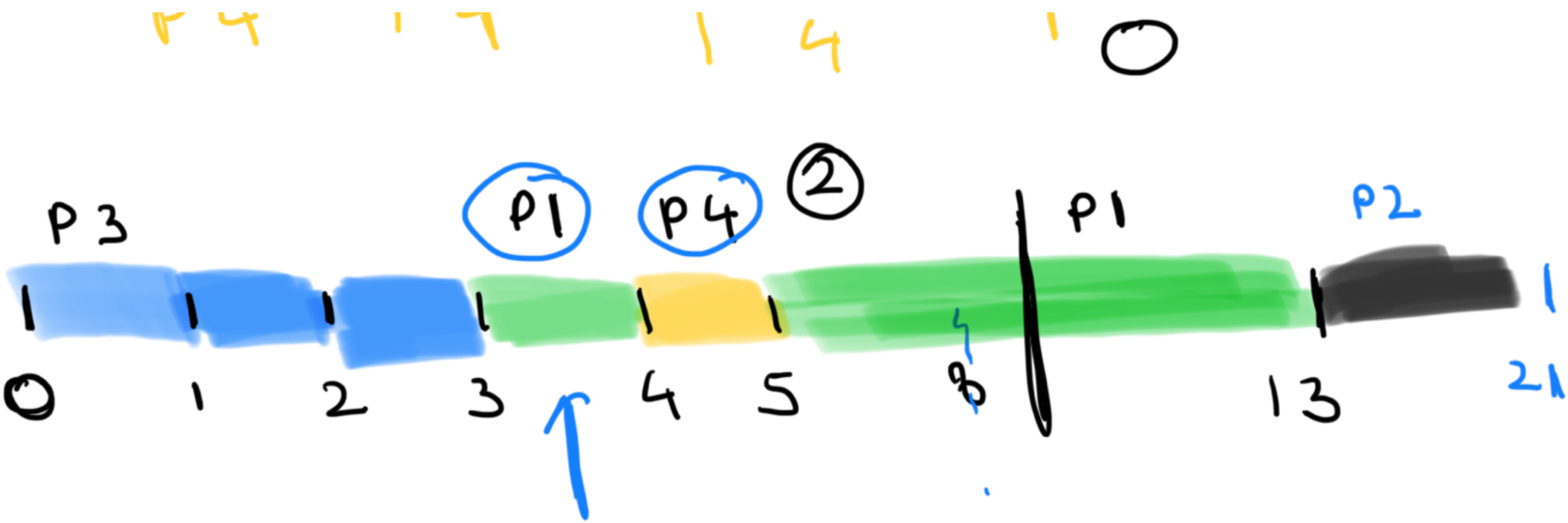
② A new process comes in

→ if remaining(new) < current

schedule(new)

→ add it to ready.

PID	AT	BT	Remaining
P1	2	6	
P2	1	8	
P3	0	3	



Advantages

1. Preemptive

Disadvantages

1. Starvation

(2) Less wait time

(3) Higher throughput

(2) Higher context switching

(3) Burst time is not always available.

Context Switching

① FCFS

② SRTF

SJF (Shortest job first)