1) FCFS

2) Shortest Job First (Preemptible / Non-preemptible)
   ⇓
   Shortest Job Remaining First

   Starvation of CPU-bound jobs

3) Round-Robin — Performance depends a lot on predecided time quanta.
   ↓
   Large → Close to FCFS
   Small → Repeated process switching

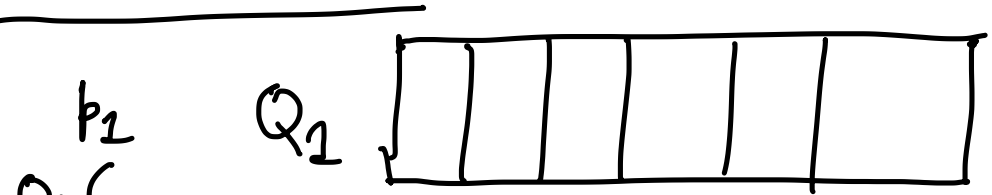
Types of Processes

1) I/O Bound or interactive ⟶ Desktops or mobiles
   ⇓
   Playing music, bash shell, file manager

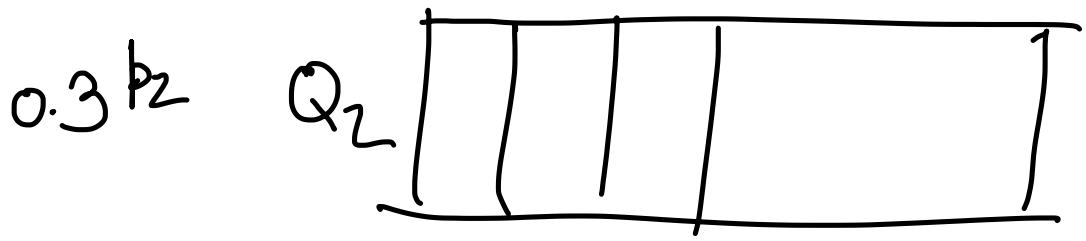
2) CPU Bound or Compute-intensive
   ⇓
   Server systems
   ⇓
   servers for web server / computation

---

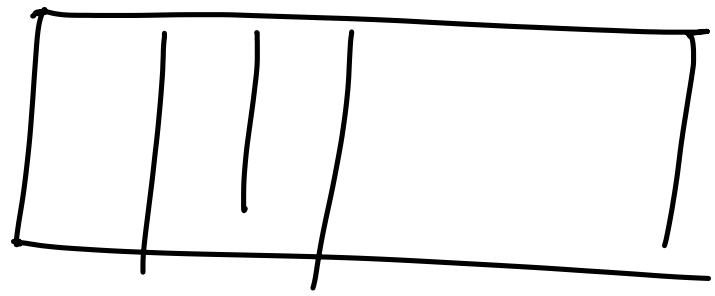
4) Multilevel Queue :—



⟶ Round robin with some time quanta q_1



q_2



$p_1 > p_2 > p_3$

$q_3 = \infty$ (FCFS) ✓

$q_1 < q_2 < q_3$

Interactive processes or processes with high priority go to higher queues.

One possible way: → If there is some process in both Q_i and Q_j, $(i < j)$, then execute always with high prob. from Q_i

✗ 1) Somebody has to predict where to place the process.

2) Starvation of CPU-intensive processes is still possible

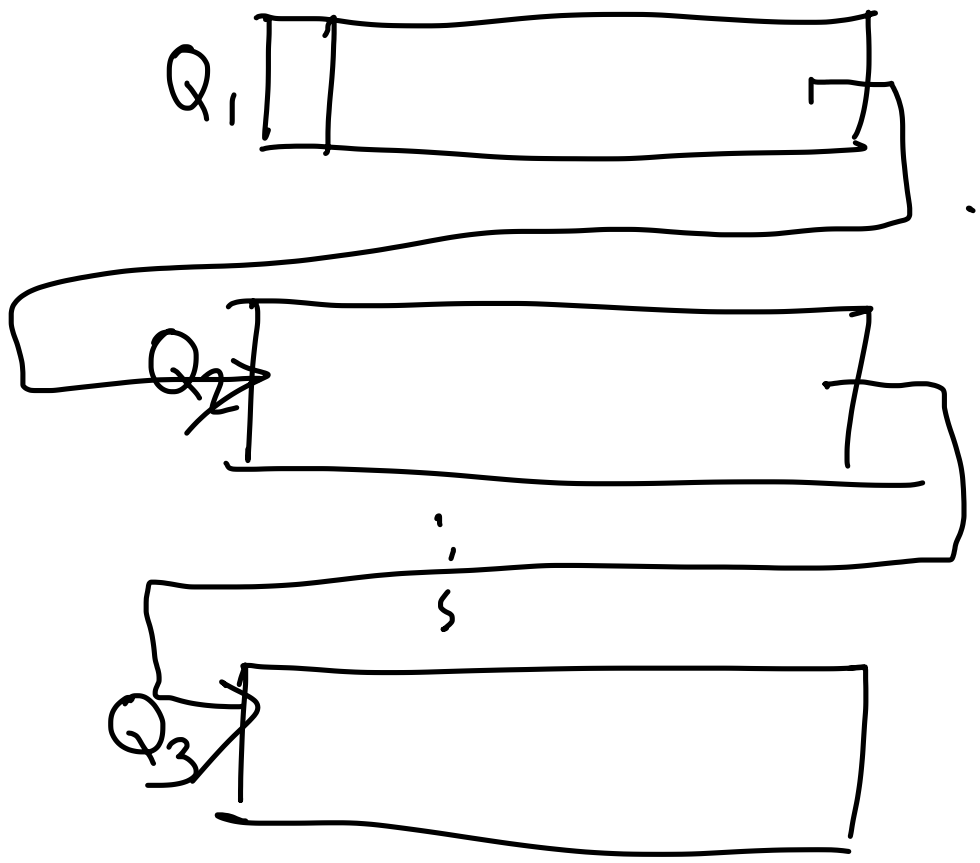
⇓

Method 2 problem eliminated

0.4 random (0, 1)

3) ~~✓~~ ✗ Somebody needs to decide which queue it should be placed in.

5) Multilevel Feedback Queue



All processes will start in Q_1. If it does not finish in one time quantum, it will be pushed down to Q_2 for scheduling again.

Linux 2.4

Priorities assigned to processes with values ranging from 0 to 139

Priorities from 0 to 99 was for real-time processes.

⇓

if some type of performance guarantee is required

⇓

process has to be executed within a deadline

- Soft real-time
  - ⇓
  - You can occasionally miss the deadline, but the more you miss, the
- Hard real-time
  - ⇓
  - You cannot miss the deadline.
  - ⇓

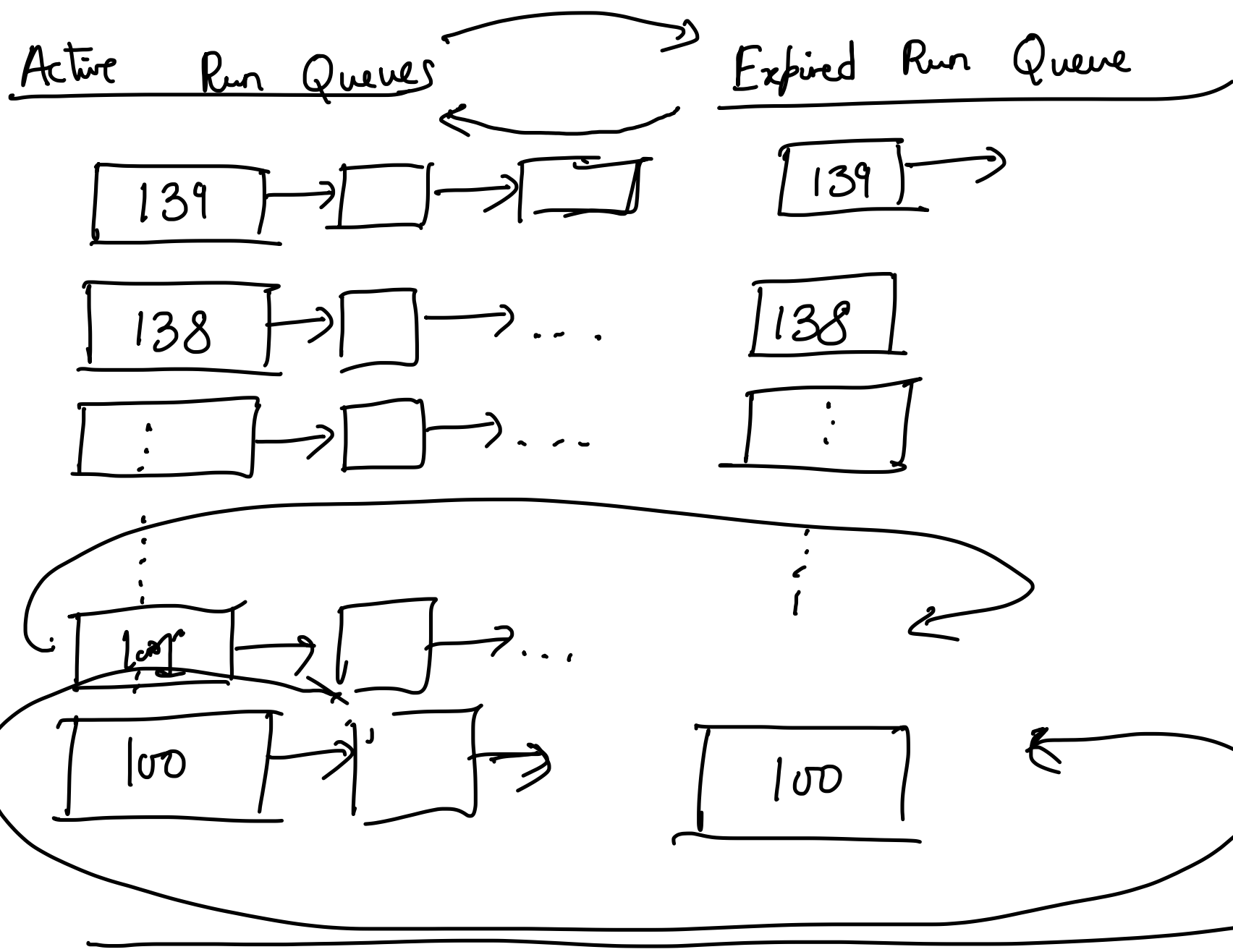
worse is the performance of the scheduler.

Cannot be done on desktop/mobile system

⇓

By specialized hardware

Normal processes → priority values from 100 to 139.



Static priority → whatever priority a process starts with

100 to 139

⇓

120 + n, where n is the "nice number"

n = 19 to -20

n is greater, that process has lower priority

By default, n is 0.

Root users can make their nice numbers negative.

Non-root users cannot.

All users can set positive nice numbers for their processes.

Dynamic priority : $\longrightarrow$ max (100, MIN (static priority − bonus + 5, 139))

$\Downarrow$

Distinguishing between I/O and CPU oriented processes.

CPU-bound process should get less priority; so if your waiting time is higher, it should get higher bonus.

$$\text{bonus} = \left\lfloor \frac{\text{Waiting time in ms}}{100} \right\rfloor \quad \text{if waiting time} < 1s$$

$$= 10, \quad \text{if } \gg \gg > 1s$$

if priority < 120; Time slice $= (140 - \text{prio}) \times 20$ ms

≥ 120; $\gg \gg = (140 - \text{prio}) \times 5$ ms