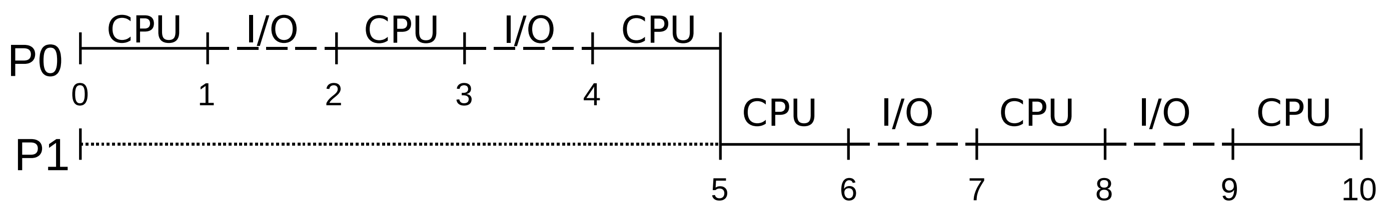
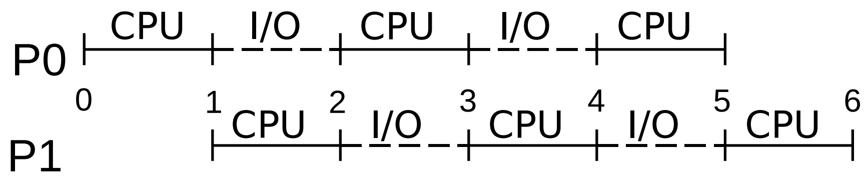
Multiprogramming

- Without multiprogramming



time = 10 tu
throughput = 0.2 proc/tu
average execution time = 7.5 tu
CPU usage = 50%

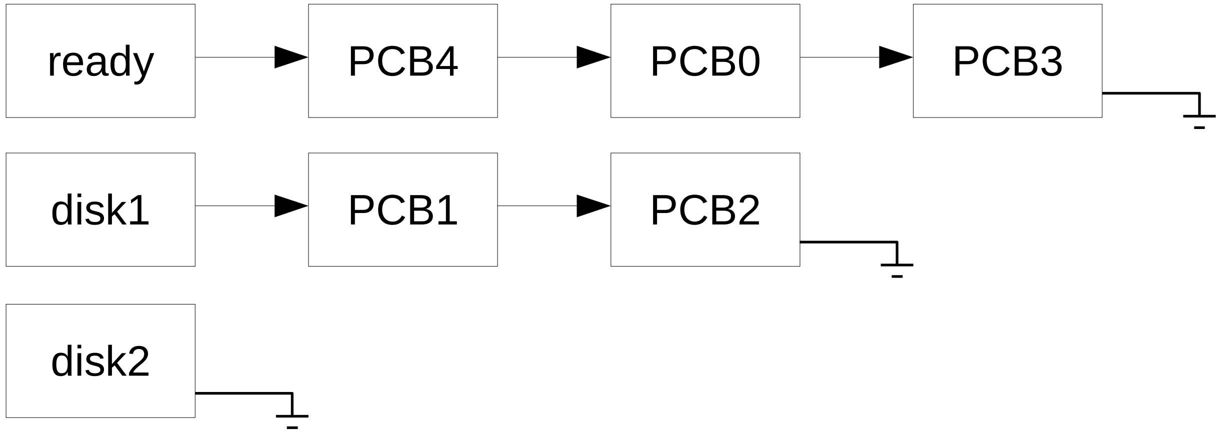
- With multiprogramming



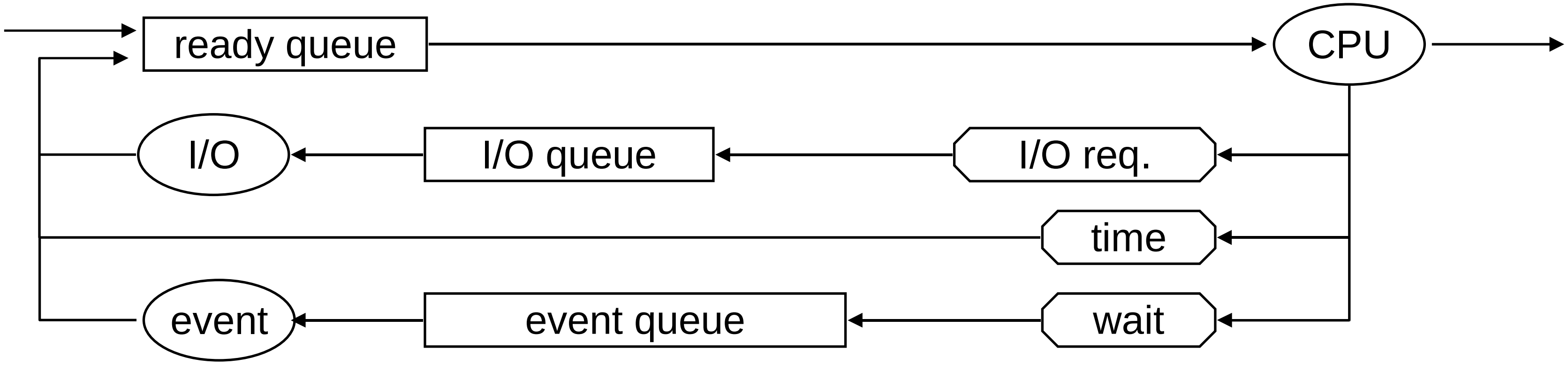
time = 6 tu
throughput = 0.33 proc/tu
average execution time = 5.5 tu
CPU usage = 100%

Process Scheduling Structures

- Ready and I/O queues



- System queue diagram

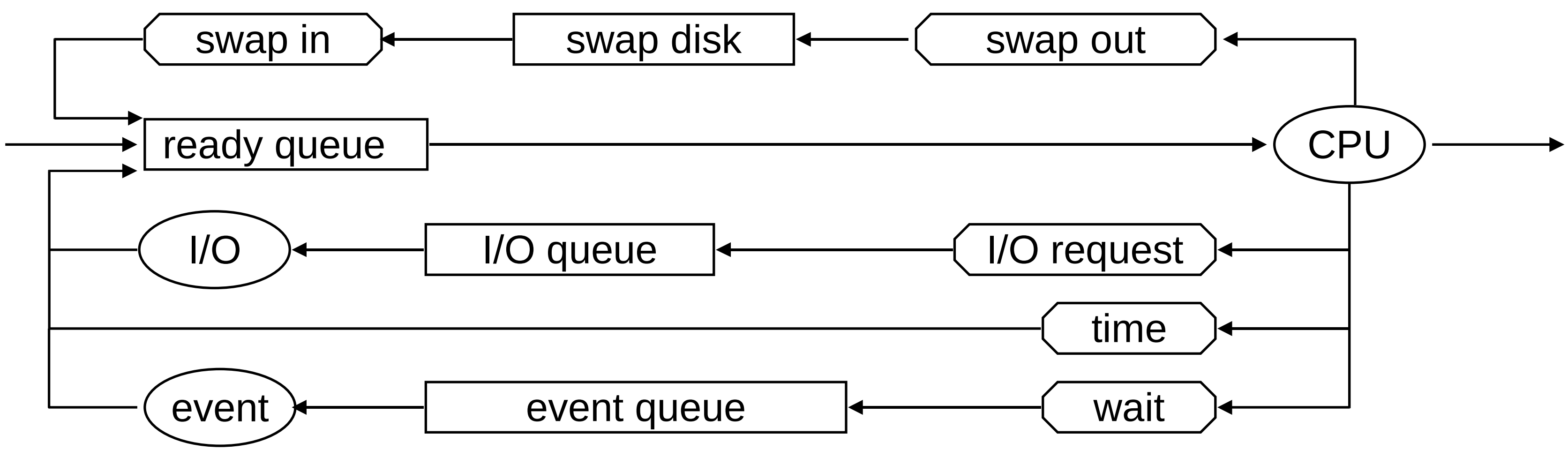


Process Schedulers

- Short term (CPU)
  - Selects processes from the ready queue
  - Runs very often and therefore must be very efficient
- Long term (jobs)
  - Selects processes that will be allowed in the system
  - Tries to balance I/O-bound e CPU-bound processes

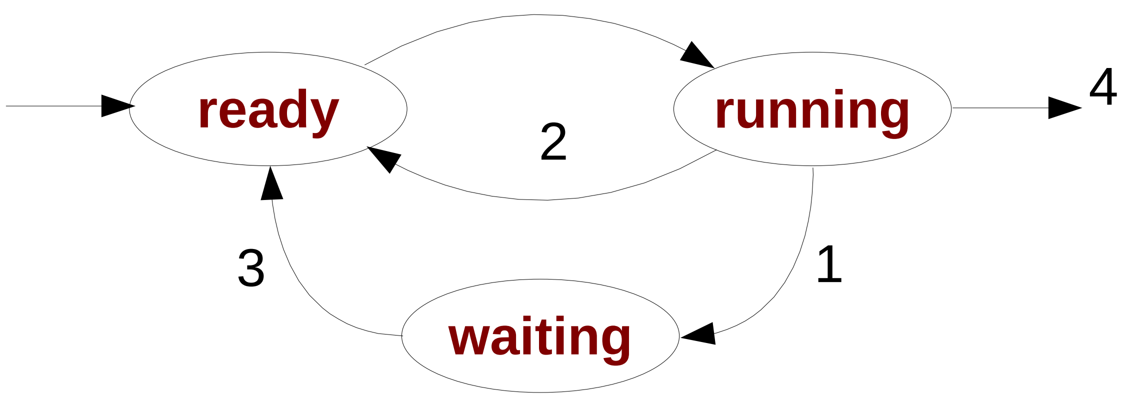
Process Schedulers

- Medium-term (*swapper*)
  - Temporarily suspends processes
    - To keep the balance between I/O e CPU usage
    - Due to memory depletion



Preemptive and Non-preemptive Process Scheduling

- A process must be chosen to occupy the CPU whenever a process
  1 - Changes state from *running* to *waiting*
  2 - Changes state from *running* to *ready*
  3 - Changes state from *waiting* to *ready*
  4 - Finishes
  - Preemptive scheduling: 1, 2, 3 and 4
  - Non-preemptive scheduling: 1 and 4



Process Scheduling Criteria

- Maximize CPU utilization
- Maximize system throughput (jobs/time)
- Minimize turnaround time (total time)
- Minimize waiting time (time waiting to run)
- Minimize and stabilize (user) response time

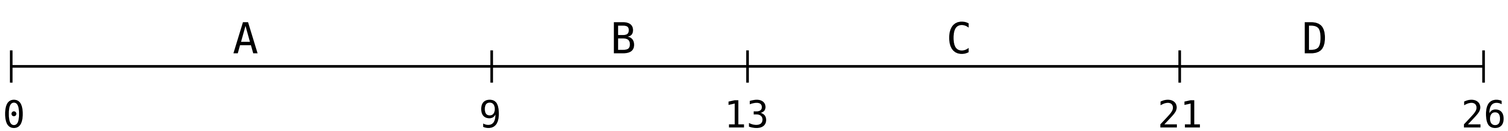
Process Scheduling Policy

- First Come First Served
- Shortest Job First
- Static Priority
- Dynamic Priority
- Round-Robin
- Multilevel Queue
- And thousands of derivations thereof

First Come First Served (FCFS)

- Policy
  - Ready queue under FIFO policy
  - New processes are inserted at the end
  - Non-preemptive
- Performance
  - Extremely poor when a CPU-bound process blocks an I/O-bound process
- Example

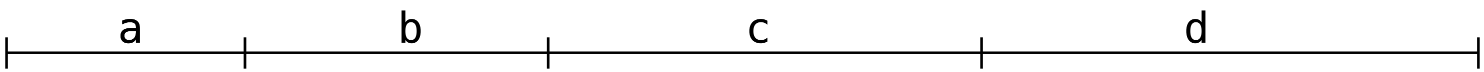
Process	A	B	C	D
CPU time	9	4	8	5
Arrival time	0	0	0	0




```
TA = (9 + 13 + 21 + 26) / 4 = 17.25 tu
WT = (0 + 9 + 13 + 21) / 4 = 10.75 tu
```

Shortest Job First (SJF)

- Policy
  - Process that will need the shortest CPU time is scheduled first
  - Preemptive or non-preemptive
- Performance
  - Optimal algorithm in terms of TA and WT



$$TA = (a + (a+b) + (a+b+c) + (a+b+c+d)) / 4 = (4a + 3b + 2c + d) / 4 \text{ tu}$$

$$WT = (0 + a + (a+b) + (a+b+c)) / 4 = (3a + 2b + c) / 4 \text{ tu}$$

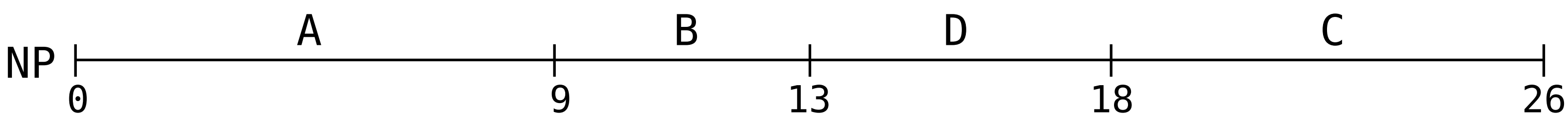
- Useful for processes for which the maximum execution time is known

Operating Systems

Shortest Job First (SJF)

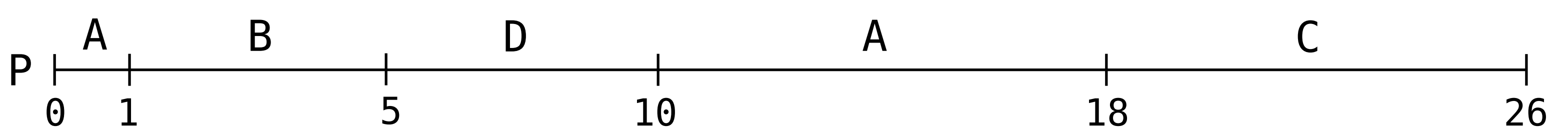
- Example

Process	A	B	C	D
CPU time	9	4	8	5
Arrival time	0	1	2	3



TA = (9 + 12 + 24 + 15) / 4 = 15 tu

WT = (0 + 8 + 16 + 10) / 4 = 8.5 tu



TA = (18 + 4 + 24 + 7) / 4 = 13,25 tu

WT = (9 + 0 + 16 + 2) / 4 = 6.75 tu

SJF Approximation

- Policy
  - Future estimation based on recent past
  - Process that has been having the shortest CPU cycles is scheduled first
- Formula

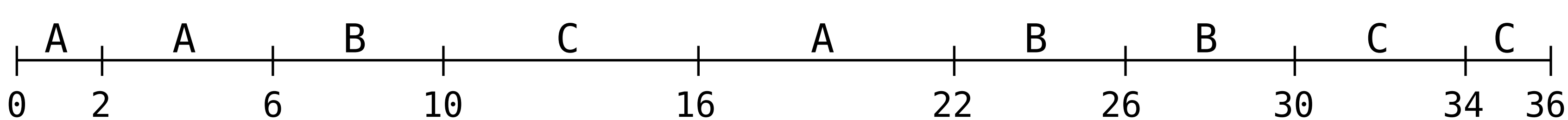
$$\pi_{n+1} = \alpha\, t_n + (1 - \alpha)\, \pi_n$$

π_{n+1} = next cycle estimate
α = past importance factor
t_n = cycle *n* effective time

- Example (α = 1/2)

TA = (22 + 30 + 36) / 3 = 29.3 tu
WT= (10 + 18 + 24) / 3 = 17.3 tu

Process	π_0	t_0	π_1	t_1	π_2	t_2	π_3
A	1	2	1	4	2	6	4
B	1	4	2	4	3	4	3
C	1	6	3	4	3	2	2



Priority

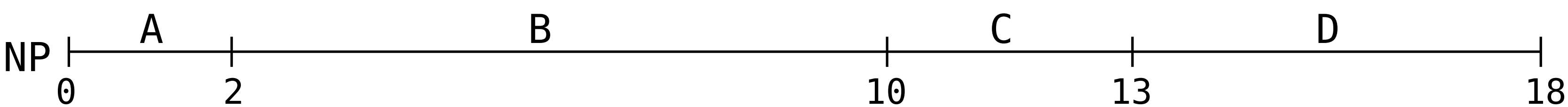
- Policy
  - Process with highest priority is scheduled first
  - Priorities can be assigned to processes either statically or dynamically
  - Preemptive or non-preemptive
- Processes might wait indefinitely
  - Low-priority processes only run when high-priority processes are *waiting*
- Typical of real-time systems

Operating Systems

Static Priority

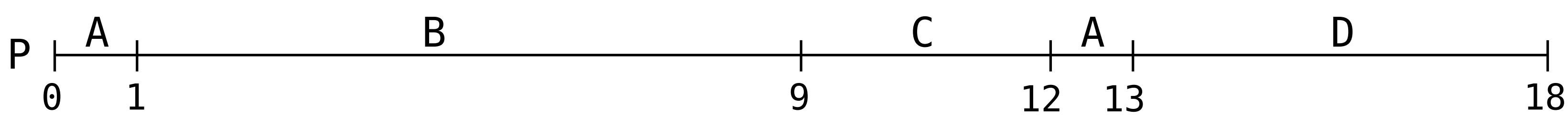
- Example

Process	A	B	C	D
CPU time	2	8	3	5
Priority	3	1	2	3
Arrival time	0	1	2	3



TA = (2 + 9 +11 + 15) / 4 = 9.25 tu

WT = (0 + 1 + 8 + 10) / 4 = 4.75 tu



TA = (13 + 8 + 10 + 15) / 4 = 11.5 tu

WT = (11 + 0 + 7 + 10) / 4 = 7 tu

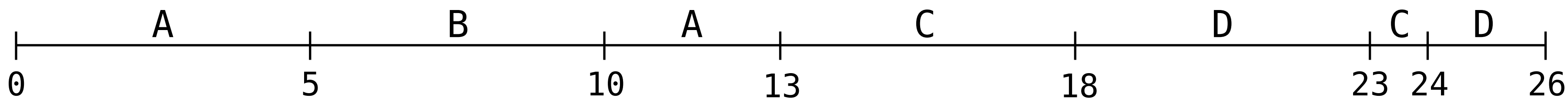
Round-Robin

- Policy
  - Processes are rescheduled periodically based on a time *quantum*
  - FIFO circular queue
  - Preemptive
- Formula
  - For a given set of processes with n elements and a time quantum of q:
    - Each process gets $1/n$ of CPU time in cycles that are no longer than q time units
    - Maximum waiting time = $(n - 1)\, q$
- Typical of time-sharing systems

Round-Robin

- Example (q = 5 tu)

Process	A	B	C	D
CPU time	8	5	6	7
Arrival time	0	4	9	14



TA = (13 + 6 + 15 + 12) / 4 = 11.5 tu
Wt = (5 + 1 + 9 + 5) / 4 = 5 tu

Multilevel Queue

- Policy
  - Processes are grouped
    - E.g. system, interactive, batch
  - Each group has its own queue under a specific policy
  - Processes might be allowed to change groups