Sistem Operasi 5

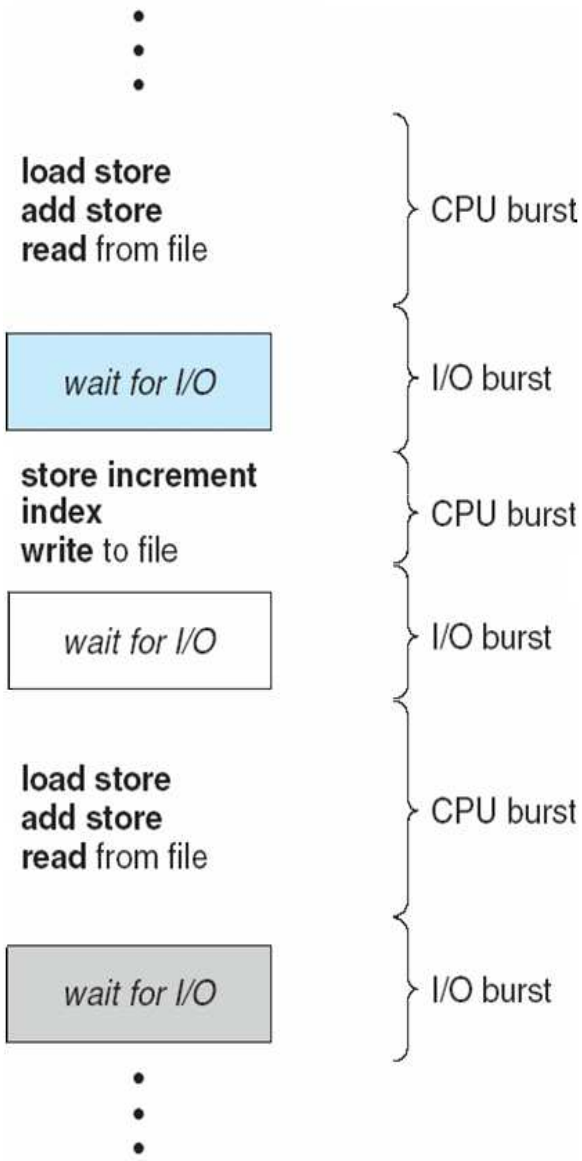
"Process Schedulling"

Antonius Rachmat C, S.Kom, M.Cs

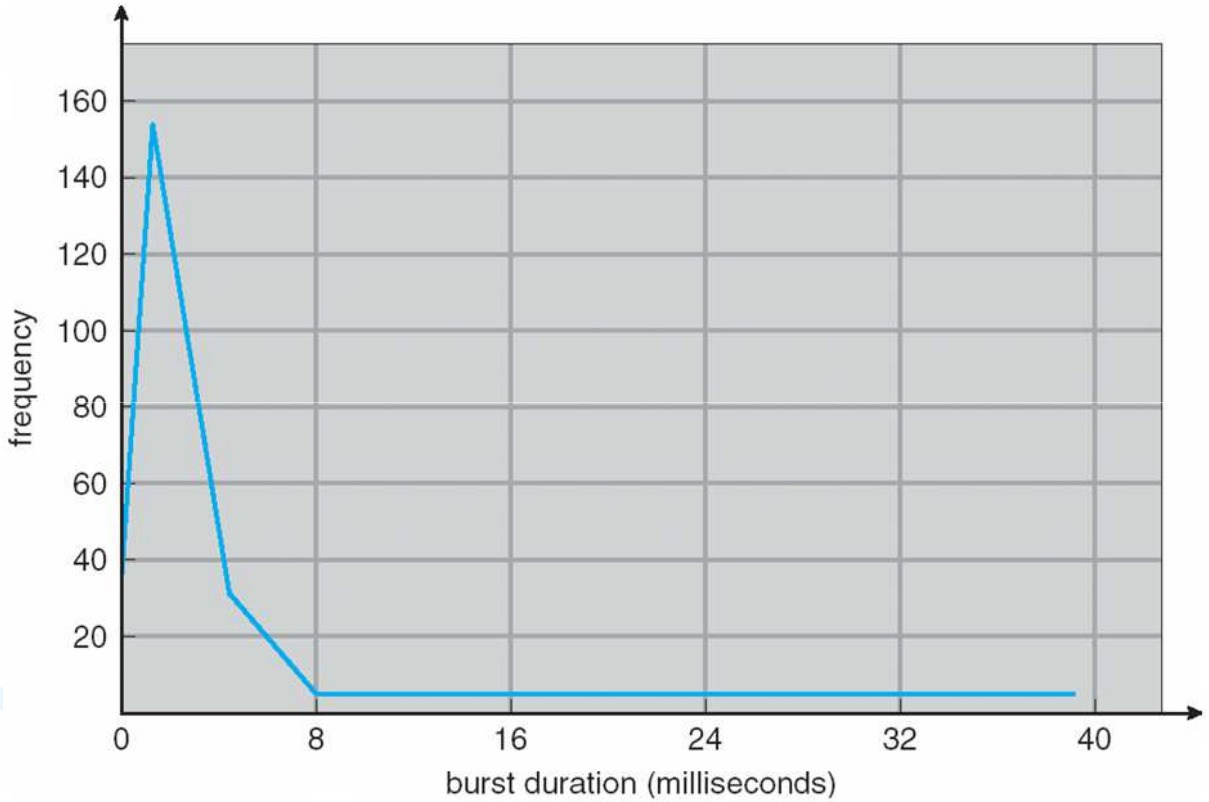
Basic Concept

- Tujuan Utama : agar proses-proses berjalan secara **konkuren** dan untuk **memaksimalkan** kinerja dari CPU.
- Pemanfaatan CPU maksimum diperoleh dengan **multiprograming**
  - Proses dieksekusi secara **bergantian**
- **CPU-I/O Burst Cycle** adalah pelaksanaan proses yg terdiri dari suatu siklus tunggu I/O dan eksekusi CPU

Alternating Sequence of CPU And I/O Bursts



Histogram of CPU-burst Times



CPU Scheduler

- **Short term scheduler** memilih dari sekian proses yang ada di ready queue yang sudah siap dieksekusi, dan mengalokasikan CPU untuk kepadanya
- Penjadwalan CPU akan terjadi perubahan state:
  1. Berubah dari running ke waiting state.
  2. Berubah dari running ke ready state.
  3. Berubah dari waiting ke ready.
  4. Terminates.
- Penjadwalan 1 dan 4 adalah **non preemptive**; sekali CPU telah dialokasikan untuk sebuah proses, maka tidak bisa di ganggu,
  - contoh pada windows 3.x dan DOS
- Selain itu bersifat **preemptive** (> win 95 dst)

Dispatcher

- Merupakan modul pemberi kontrol CPU kepada proses, yg fungsinya :
  - switching context
  - switching to user mode
  - Lompat dari suatu bagian di progam user untuk mengulang progam.
- **Dispatch Latency**: waktu yang dibutuhkan untuk menstop satu proses dan menjalankan proses lainnya
  - Sebisa mungkin secepat-cepatnya.

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible (ideal: 40-90%)
- **Throughput** – # of processes that complete their execution per time unit
  - Rata-rata untuk proses yang sebentar 10 proses per dtk
- **Turnaround time** – amount of time to execute a particular process
  - dibawa ke memori, menunggu di ready queue, eksekusi di CPU, dan I/O
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment)

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Scheduling Algorithm

- First-Come, First-Served
- Shortest-Job-First
- Priority
- Round-Robin
- Multilevel Queue
- Multilevel Feedback Queue

FCFS

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3 The Gantt Chart for the schedule is:

- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

FCFS (2)

Suppose that the processes arrive in the order

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:

- Waiting time for $P_1 = 6; P_2 = 0, P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case
- *Convoy effect* short process behind long process

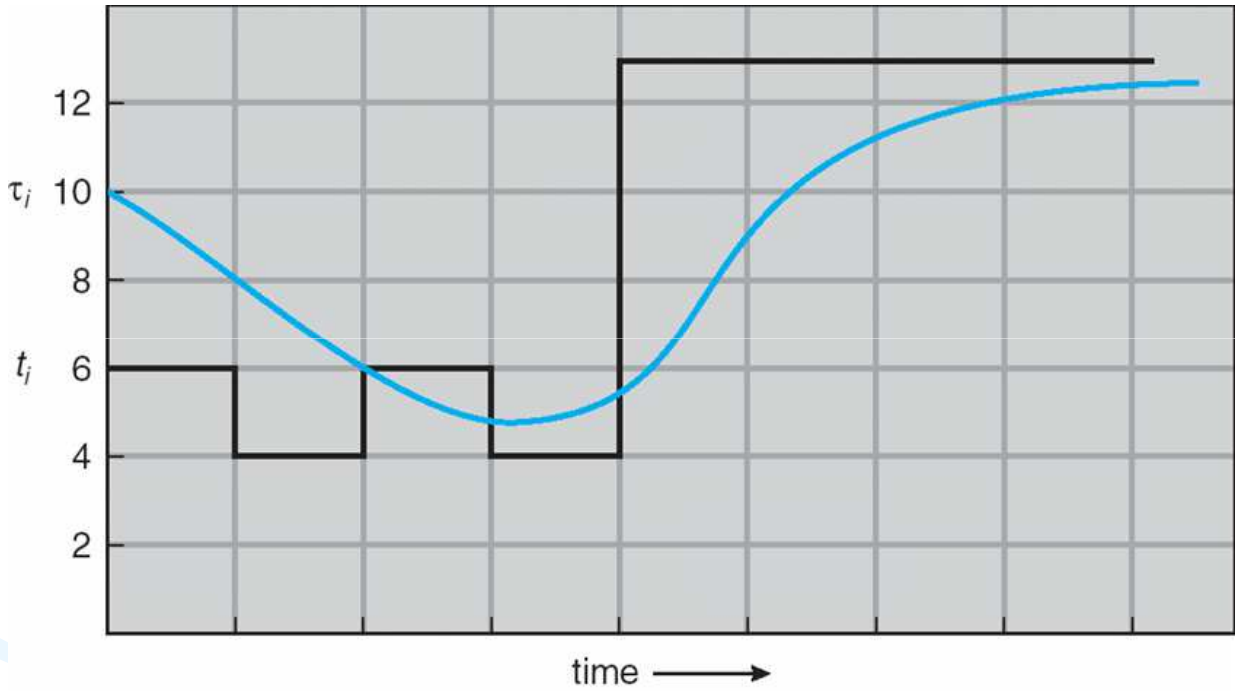
Bersifat non preemptive

Demo FCFS

Shortest Job First

- Mendahulukan proses dengan burst time terkecil (shortest next CPU burst)
- Average waiting time terkecil (optimal)
- Sulit memprediksi panjang CPU burst
- Two schemes:
  - **Non preemptive** – once CPU given to the process it cannot be preempted until completes its CPU burst
  - **preemptive** – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the **Shortest-Remaining-Time-First (SRTF)**

Prediction of the Length of the Next CPU Burst

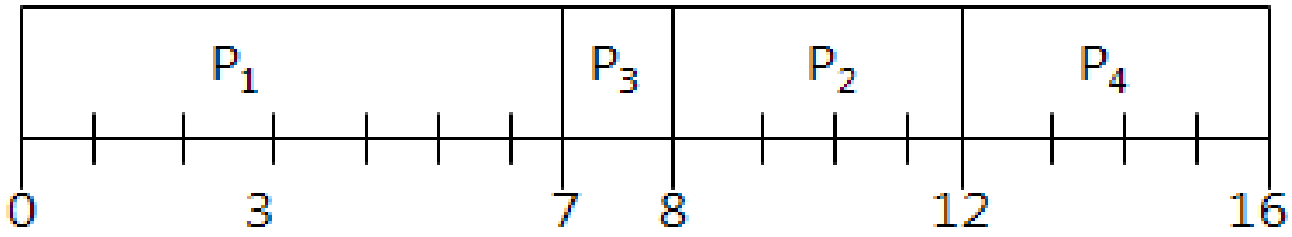


CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

SJF non preemptive

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



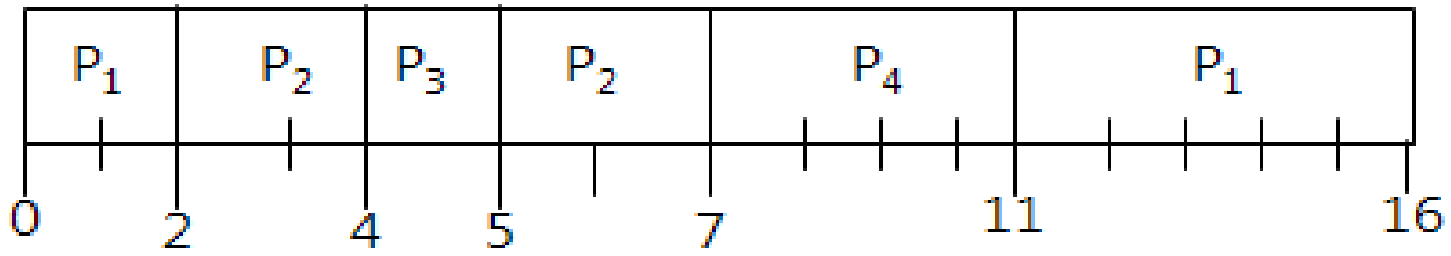
- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Demo SJF Non Preemptive

SJF preemptive

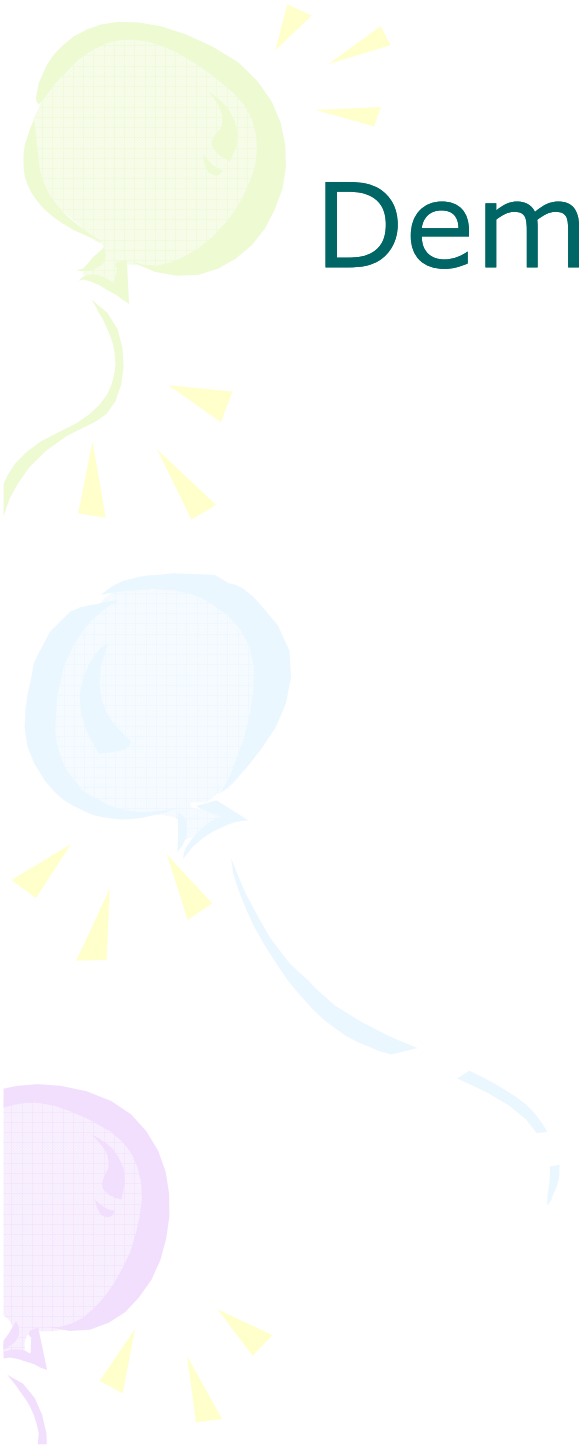
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

Demo SJF Preemptive



Priority Schedulling

- Tiap proses diberikan skala **prioritas**
- Skala bisa ditentukan secara internal/eksternal
- Preemptive or not.
  - Non preemptive = FCFS berprioritas
  - Preemptive = SJF berprioritas
- Problem: **Starvation** - low priority processes may never execute
- Solution: **Aging** – as time progresses increase the priority of the process

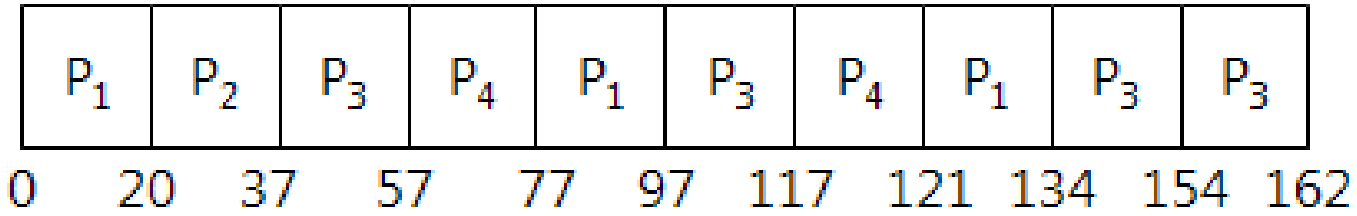
Round Robin

- Digilir selama **time quantum** (q)
  - Sekitar 10-100ms
- Performa tergantung besar **time quantum**.
- Adil, semua proses mendapat jatah CPU 1/n, dan tidak akan menunggu lebih dari (n-1)/q
- Jika time quantum sangat besar (infinite), akan seperti FCFS.
- Kalau terlalu kecil, context switch (perpindahan) terlalu banyak

RR with time quantum= 20

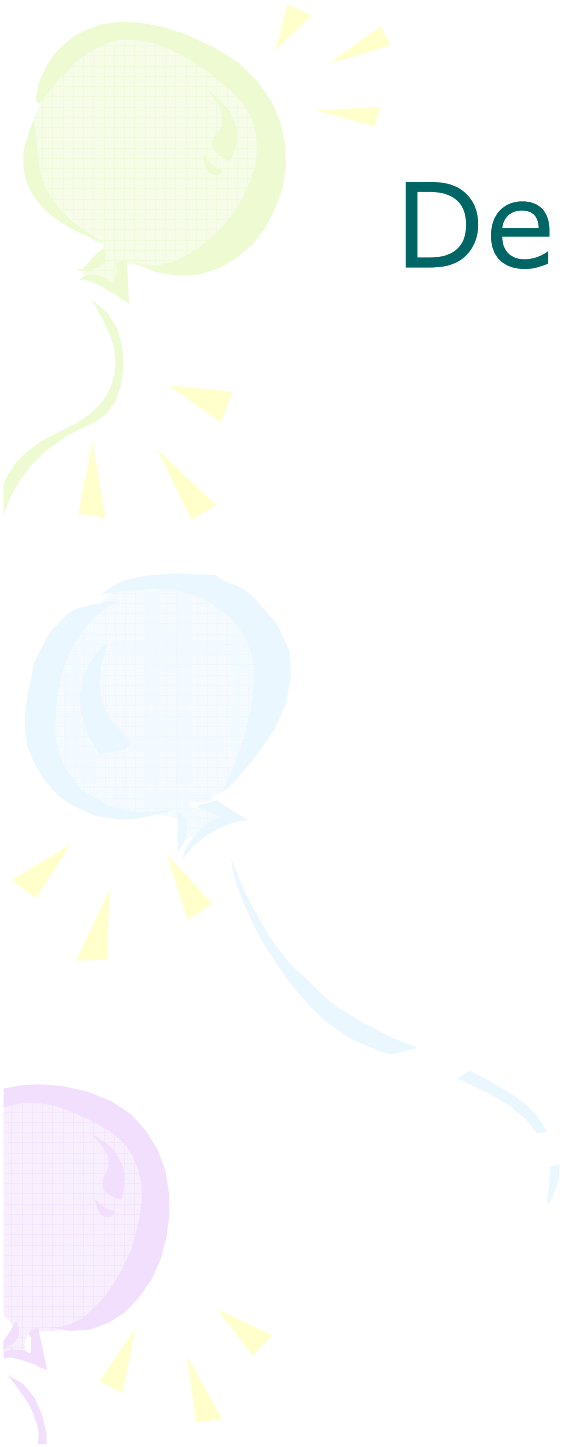
Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better *response*

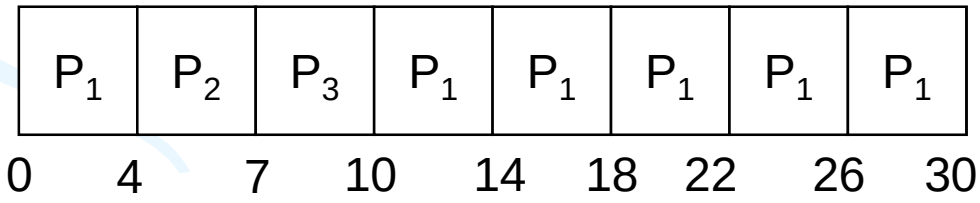
Demo Round Robin



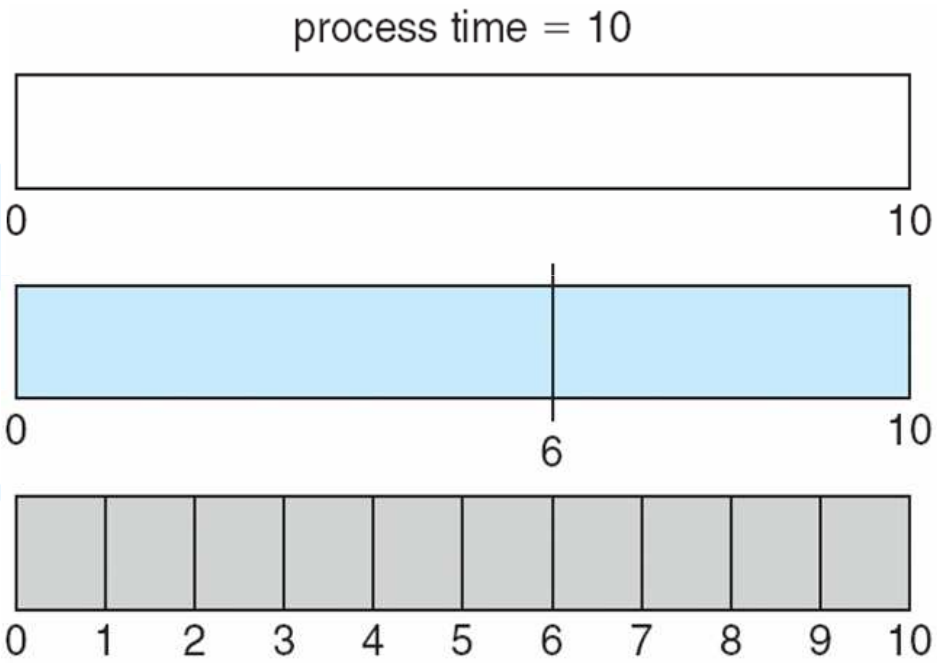
RR with Time Quantum = 4

Process	Burst Time
P_1	24
P_2	3
P_3	3

The Gantt chart is:



Time Quantum and Context Switch Time

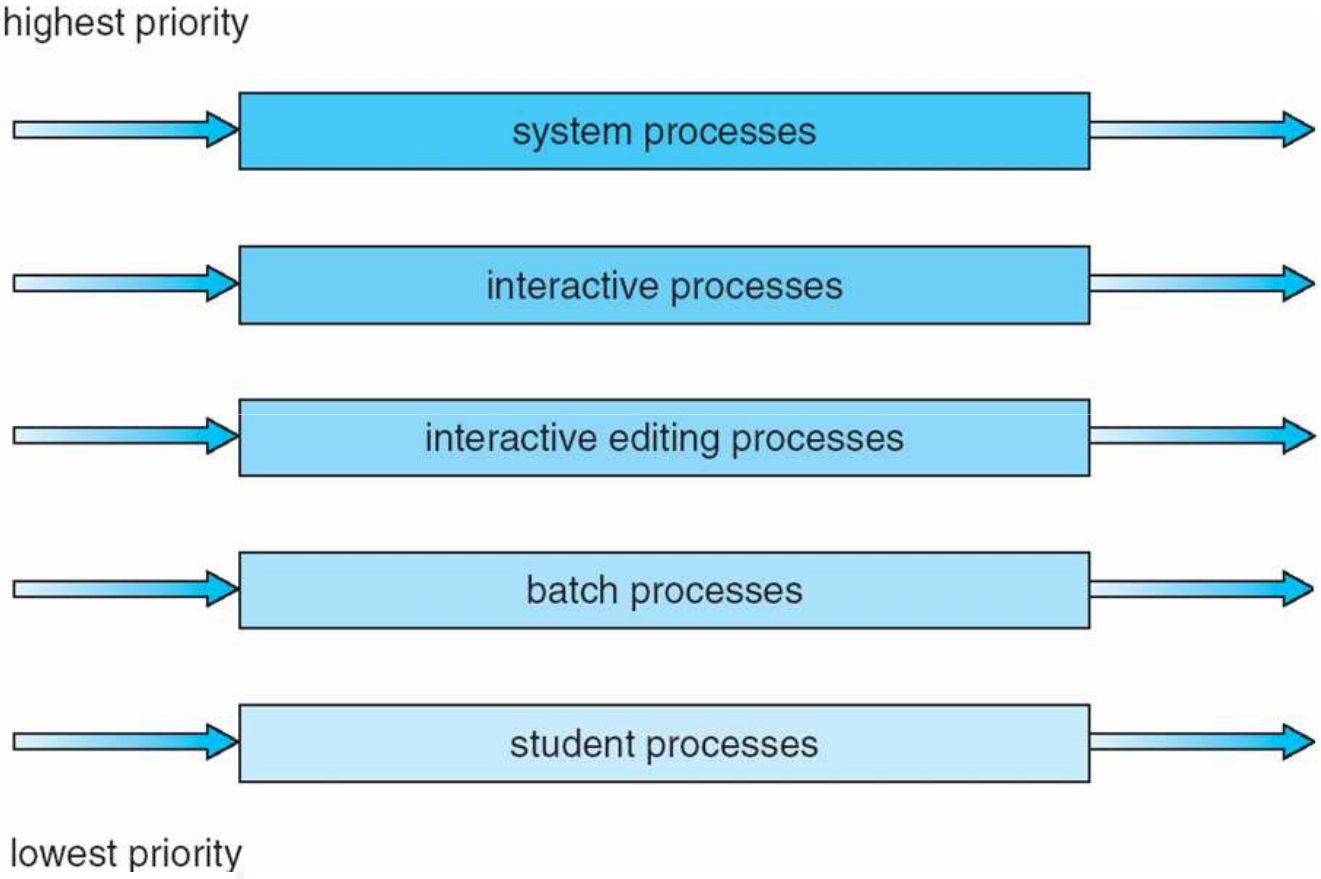


quantum	context switches
12	0
6	1
1	9

Multi level queue

- Terdiri dari beberapa antrian (queue):
  - foreground (interactive)
  - background (batch)
- Tiap bagian antrian mempunyai **skala prioritas**
- Antrian tidak akan mendapat jatah CPU, selama masih ada antrian dg prioritas lebih tinggi yg belum mendapat jatah
- **Time slice** – each queue gets a certain amount of CPU time
  - 80% to foreground in RR, and 20% to background in FCFS
- Each queue has its own scheduling algorithm
  - foreground – RR
  - background – FCFS

Multilevel Queue Scheduling



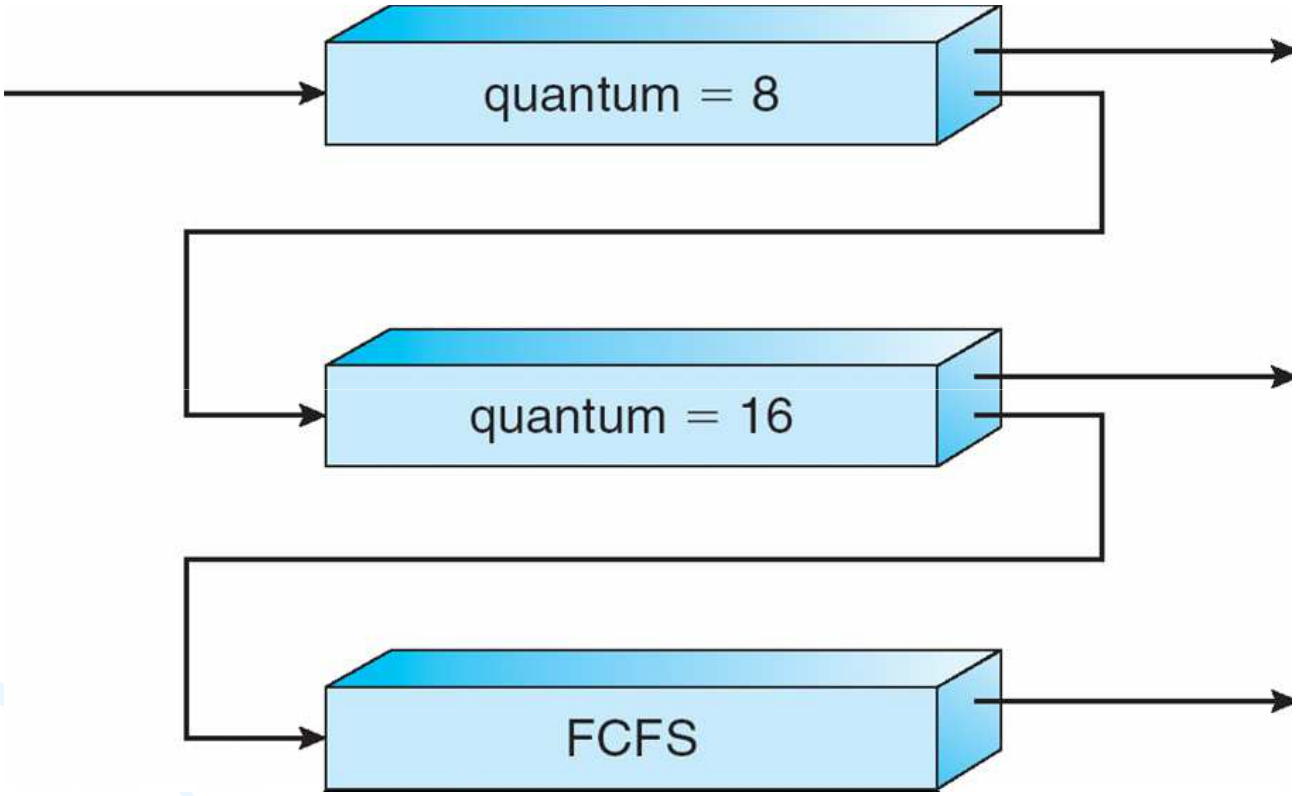
Multi level feedback queue

- Proses bisa pindah antar antrian
  - Dengan metode **Aging**
  - Prioritas tertinggi untuk proses yang CPU burstnya terkecil
- Umumnya antrian high priority menggunakan RR, yang rendah FCFS
- Lebih mendukung interaktivitas
- Parameternya
  - Jumlah antrian
  - Algoritma tiap antrian
  - Antrian mana yg akan dimasukki oleh proses
  - Kapan menaikkan proses ke prioritas yg lebih tinggi dan menurunkan proses ke prioritas yg lebih rendah

Example of Multilevel Feedback Queue

- Ex: in three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – FCFS time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q_1.
  - At Q_1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.

Multilevel Feedback Queues



Thread Scheduling

- Distinction between **user**-level and **kernel**-level threads
- Many-to-one, one-to-one, many-to-many models, thread library schedules user-level threads to run on LWP (Light Weight Process)
  - Known as **process-contention scope (PCS)** since scheduling competition is within the process
- Kernel thread scheduled onto available CPU is **system-contention scope (SCS)** – competition among all threads in system

Real Time Schedulling

- Asumsi:
  - Ada batasan waktu: ada **deadline**
  - Semua kejadian dapat **diprediksi**
  - Jika ada banyak proses terjadi bersamaan, maka semua deadline harus **terpenuhi**.

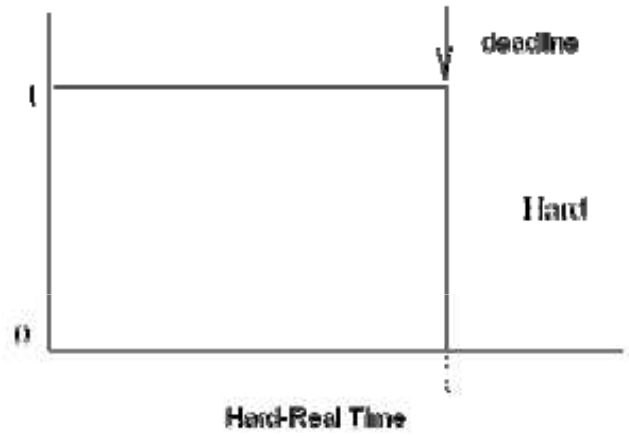
Real time schedulling

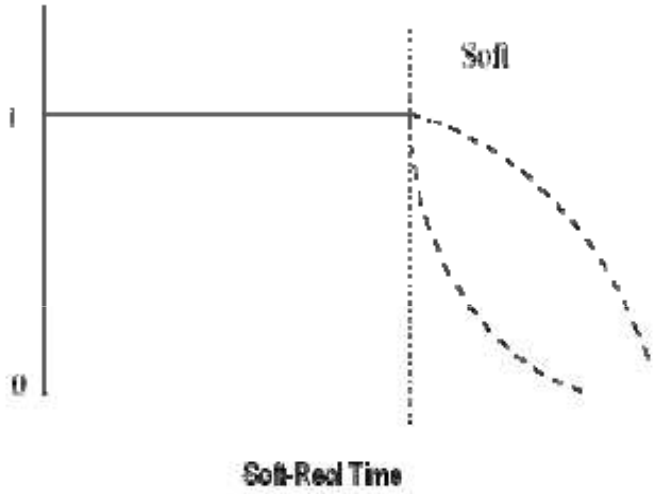
- Hard-Real Time
  - Menjamin proses dapat diselesaikan dengan tepat waktu.
    - Biasanya <= 100 mikro detik
  - Pada saat proses dikirim, terdapat statement yang menyatakan **jumlah waktu** yang diperlukan untuk menyelesaikan proses tersebut.
  - Setelah deadline, proses langsung **berhenti**.
  - **Scheduler** memainkan peranan yang penting.
  - Jika permintaan alokasi waktu terlalu besar (atau tidak dapat diprediksi), maka scheduler akan menolaknya.
  - Contoh: pengkontrol pesawat terbang

Real time schedulling (2)

- Soft-Real Time
  - Memiliki keterbatasan **yang lebih rendah** dari hard-time system. (lebih longgar)
  - Setelah deadline, proses langsung berhenti bertahap.
  - Critical task diberikan prioritas yang **lebih tinggi** dari yang lainnya.
  - Memerlukan desain scheduler yang lebih cermat, karena harus men-set prioritas.
  - Dapat menyebabkan pembagian resource yang kurang adil, delay yang lama, sampai terjadinya starvation.
  - Contoh: alat penjual / pelayanan otomatis.

Hard vs Soft Graphics





Setelah deadline, proses langsung berhenti.

Setelah deadline, proses berhenti bertahap.

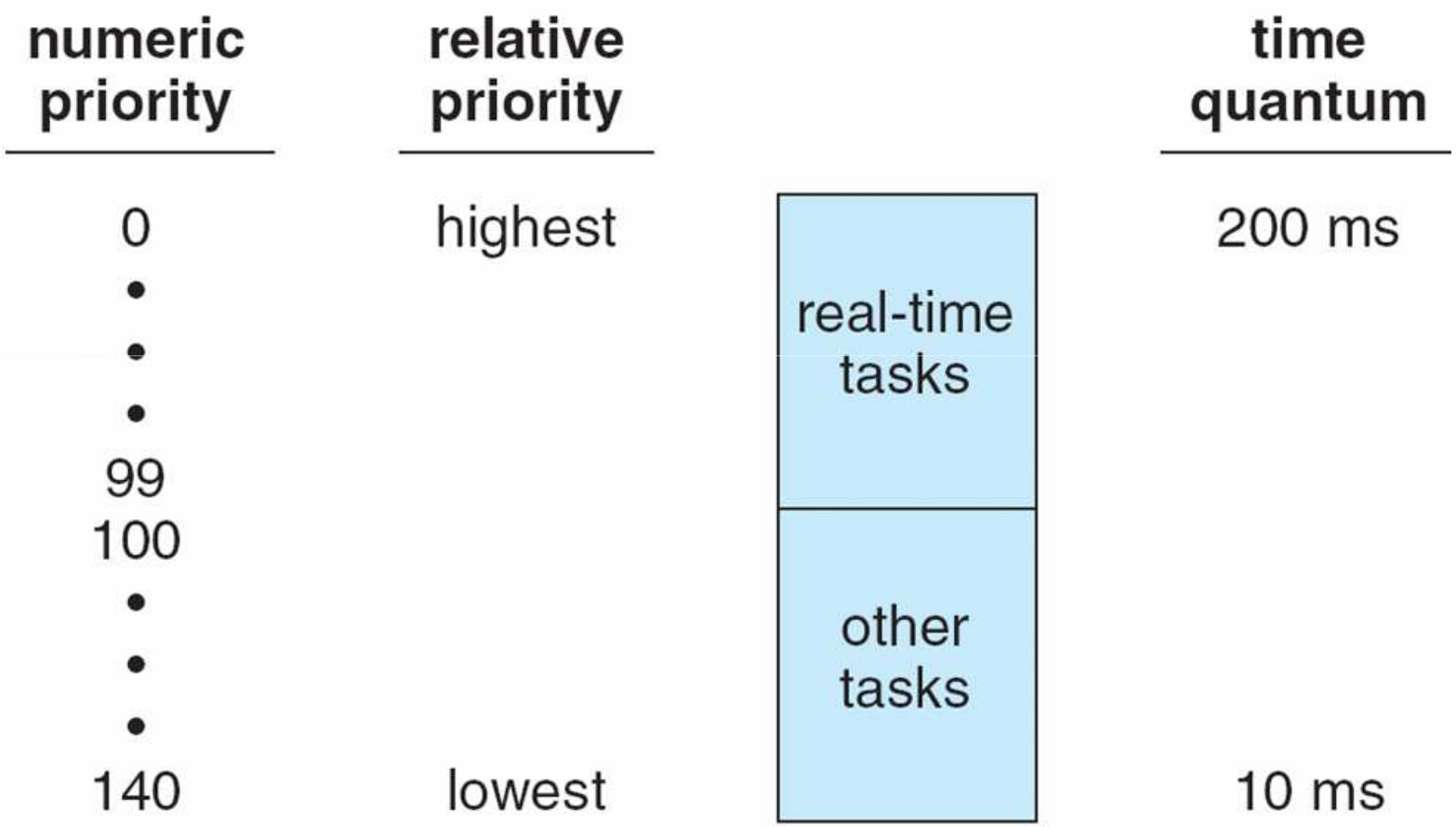
Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
- **Homogeneous processors** within a multiprocessor
- **Asymmetric multiprocessing** – only one processor accesses the system data structures (sharing)
- **Symmetric multiprocessing (SMP)** – each processor is self-scheduling, all processes in common ready queue, or each has its own private queue of ready processes

Windows XP Priorities

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Linux Priorities: the Relationship Between Priorities and Time-slice length



NEXT

- Presentasi Tugas 1 dan Quiz