Penjadwalan Proses

- **Penjadwalan**: pemilihan proses selanjutnya yg akan dieksekusi
- Melakukan *multiplexing* CPU
- Kapan dilakukan penjadwalan?
  - Proses baru dibuat
  - Proses selesai dieksekusi
  - Proses yg sdg dieksekusi diblokir
  - Terjadi I/O interrupt (mis. proses yg diblokir siap utk dieksekusi kembali)
  - Terjadi clock interrupt (mis. sekali 40 mdet)

Tujuan Penjadwalan

- Adil
- Prioritas
- Efisiensi
- Mendukung beban yg berat
- Beradaptasi dgn beragam lingkungan (interaktif, real-time, multimedia)

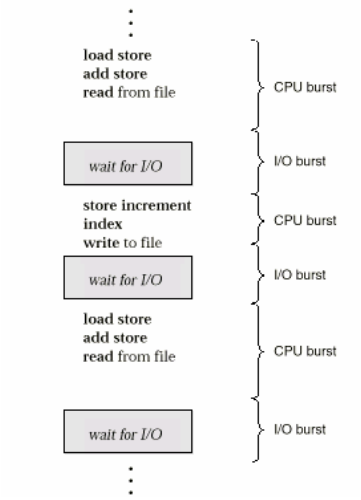
Kriteria Performansi

- **Keadilan**
- **Efisiensi**: optimalisasi penggunaan sumberdaya
- ***Throughput***: # proses yg selesai dalam satuan waktu tertentu
- **Waktu *Turnaround*** (aka: elapse time): waktu yg diperlukan utk menyelesaikan eksekusi sejak proses tsb masuk
- **Waktu Tunggu**: waktu yg diperlukan proses utk menunggu di antrian *ready*
- **Waktu Respon**: jangka waktu sejak proses di-*submit* hingga memperoleh respon pertama
- **Penerapan Kebijakan**: sesuai dgn kebijakan yg telah ditetapkan
- **Proporsionalitas**: memenuhi keinginan user
- **Memenuhi Tenggat**

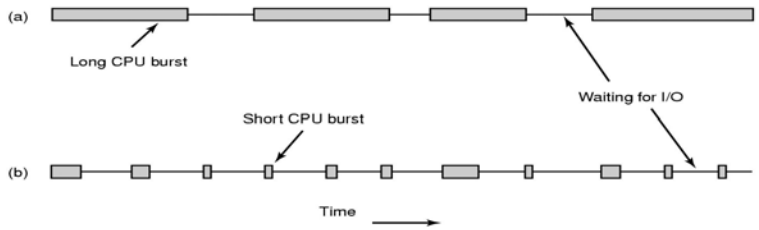
Fokus Penjadwalan pd Berbagai Sistem

- Untuk semua:
  - keadilan, penerapan kebijakan, pemerataan beban
- Sistem Batch
  - maks. troughput, min waktu turnaround, maks penggunaan CPU
- Sistem Interaktif
  - min. waktu respon, proporsionalitas
- Sistem *Real-Time*
  - dpt diprediksi, memenuhi tenggat

CPU dan I/O *Bursts*



Perilaku Program yg Dipertimbangkan dalam Penjadwalan



- CPU *burst* pendek/panjang
- Lingkungan (*batch,* interaktif)
- Prioritas, tingkat kepentingan (*urgency*)
- Frekuensi terjadinya *page fault*, *preemption*
- Waktu eksekusi yg dibutuhkan dan yg telah digunakan

Penjadwalan Preemptive vs Non-preemptive

- Penjadwalan Non-preemptive
  - Proses yg sdg dieksekusi menggunakan CPU hingga proses tsb menyerahkannya secara sukarela
- Penjadwalan Preemptive
  - Proses yg sdg dieksekusi dpt diinterupsi dan dipaksa utk menyerahkan CPU

Algoritma Penjadwalan Prosesor Tunggal

- Sistem *Batch*
  - First Come First Serve (FCFS)
  - Shortest Job First (SJF)
- Sistem Interaktif
  - Round Robin
  - Penjadwalan Prioritas
  - Multi Queue dan Multi Level Feedback
  - Shortest Process Time
  - Guaranteed Scheduling
  - Lottery Scheduling
  - Fair Sharing Scheduling

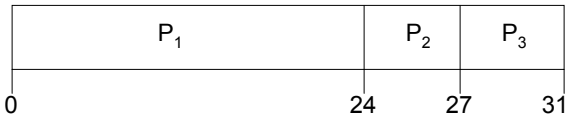
First Come First Serve (FCFS)

- Proses yg meminta CPU duluan yg dialokasikan CPU duluan
- Disebut juga FIFO
- Non-preemptive
- Digunakan pada sistem batch
- Analogi dunia nyata: restoran cepat saji
- Implementasi: antrian FIFO
  - Proses baru memasuki belakang antrian
  - Scheduler memilih dari depan antrian
- Metrik performansi: waktu tunggu rata-rata
- Parameter:
  - Burst time (dlm ms), waktu dan urutan kedatangan

FCFS: Contoh (1)

Proses	Waktu	Urutan	Kedatangan
P1	24	1	0
P2	3	2	0
P3	4	3	0

Penjadwalan:



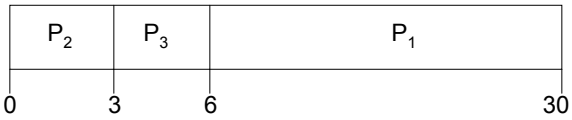
Waktu tunggu: P1=0, P2=24, P3=27

Waktu tunggu rata-rata: (0 + 24 + 27)/3 = 17

FCFS: Contoh (2)

- Jika proses datang dengan urutan P2, P3, dan P1

Penjadwalan:



Waktu tunggu: P1=7, P2=0,P3=3

Waktu tunggu rata-rata: (7 + 0 + 3)/3=3.3

- Jauh lebih baik dari contoh(1)

FCFS: Masalah

- Non-preemptive
- AWT tidak optimal
- Tidak bisa menggunakan sumberdaya secara paralel:
  - Asumsi: 1 proses CPU-bounded dan banyak proses I/O-bounded
  - Hasil: **Convoy effect**, utilisasi CPU dan perangkat I/O sangat rendah

Shortest Job First (SJF)

- Dahulukan job dengan waktu eksekusi tersingkat
- Digunakan pada sistem batch
- Ada 2 tipe:
  - Non-preemptive
  - Preemptive
- Kebutuhan: waktu eksekusi harus diketahui terlebih dahulu
- Optimal jika semua job tersedia pada waktu yg sama
- Memberikan waktu tunggu rata-rata terbaik

Non-preemptive SJF: Contoh(1)

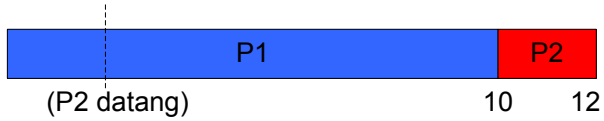
Proses	Waktu	Urutan	Kedatangan
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0

- Waktu tunggu: P1=3, P2=16, P3=9, P4=0
- Waktu tunggu rata-rata= 28/4 = 7
- Total waktu eksekusi = 24

Non-preemptive SJF: Contoh(2)

- Bagaimana jika kedatangan proses tidak serentak?

Proses	Waktu	Urutan	Kedatangan
P1	10	1	0
P2	2	2	2



- Waktu tunggu: P1=0, P2=8
- Waktu tunggu rata-rata: 8/2=4
- SJF tidak selalu optimal!

Preemptive SJF

- Disebut juga Shortest Remaining Time First (SRTF)
  - Jadwalkan dulu job dengan sisa waktu eksekusi yg paling singkat
- Kebutuhan: waktu eksekusi yg telah terpakai (*elapse time*) harus diketahui

Preemptive SJF: Contoh

Proses	Waktu	Urutan	Kedatangan
P1	10	1	0
P2	2	2	2

- Waktu tunggu: P1=0+(4-2)=2, P2=0
- Waktu tungu rata-rata: (2+0)/2=2

Masalah pada SJF

P2

4

- Starvation
  - Pada kondisi tertentu, suatu job mungkin tidak pernah menyelesaikan eksekusinya
  - Contoh:

    Proses A dgn elapse time 1 jam tiba pd waktu 0. Namun, pd waktu yg sama dan setiap 1 menit berikutnya tiba proses singkat dgn elapse time 2 menit.

    Hasilnya: A tidak pernah mendapat jatah eksekusi

Algoritma Penjadwalan Interaktif

- Biasanya preemptive
  - Waktu eksekusi dibagi dalam kuantum (interval waktu)
  - Keputusan penjadwalan dibuat pd awal tiap kuantum
- Kriteria performansi
  - Waktu respon minimum
  - Proporsional terbaik
- Algoritma
  - Berbasis prioritas
  - Round-robin
  - Multi Queue & Multi-level Feedback
  - Shortest process time
  - Guaranteed Scheduling
  - Lottery Scheduling
  - Fair Sharing Scheduling

Penjadwalan Prioritas

- Tiap proses diberi prioritas
- Penjadwalan FCFS within each priority level.
- Proses dgn prioritas lebih tinggi dijadwalkan duluan
  - Preemptive
  - Non-preemptive
- Masalah:
  - Mungkin tidak menghasilkan waktu tunggu rata-rata yg baik
  - Dpt menyebabkan infinite blocking atau starvation pd proses dgn prioritas rendah

Penjadwalan Prioritas: Penentuan Prioritas

- Ada 2 pendekatan:
  - Statis (untuk sistem dgn perilaku aplikasi yg teratur dan telah diketahui)
  - Dinamis (sebaliknya)
- Prioritas dpt ditentukan berdasarkan:
  - Biaya terhadap user
  - Tingkat kepentingan user
  - Umur proses (aging)
  - % waktu CPU yg telah digunakan pd x jam terakhir

Penjadwalan Prioritas: Contoh

Proses	Waktu	Prioritas	Kedatangan
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0

- Waktu tunggu: P1=18, P2=0, P3=11, P4=8
- Waktu tunggu rata-rata: (18+0+11+8)/4=9.25 (lebih jelek dari Non-preemptive SJF)

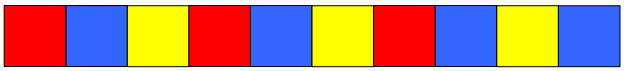
Round Robin

- Tiap proses memperoleh alokasi waktu CPU dlm kuantum waktu, biasanya 10-100 ms
- Setelah kuantum waktu lewat, proses di-preempted dan dimasukkan ke belakang antrian ready
- Jika ada n proses pd antrian ready dan kuantum waktu=q, maka:
  - Pada gilirannya tiap proses memperoleh 1/n waktu CPU selama q
  - Tidak ada proses yg menunuggu lebih dari (n-1)q unit waktu
- Performansi:
  - q besar → FIFO
  - q kecil → overhead utk context switch sangat besar

Round Robin: Contoh

Proses	Durasi	Urutan	Kedatangan
P1	3	1	0
P2	4	2	0
P3	3	3	0

Asumsi: kuantum waktu=1 unit; P1, P2, & P3 tidak pernah diblokir

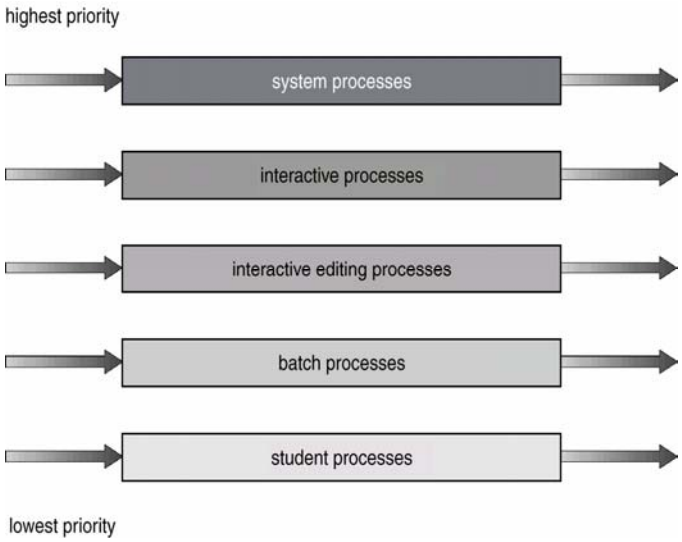


- Waktu tunggu:
  P1=0+2+2=4,
  P2=1+2+2+1=6
  P3=2+2+2=6
- Waktu tunggu rata-rata: (4+6+6)/3=5.33

Multilevel-Queue (1)

- Antrian ready queue dibagi menjadi beberapa antrian terpisah :
  - foreground (interaktif)
  - background (batch)
- Tiap antrian memiliki algoritma penjadwalan tersendiri:
  - foreground – RR
  - background – FCFS
- Penjadwalan harus dilakukan antar antrian
  - Fixed prioritas: melayani semua proses foreground, kemudian background → kemungkinan starvation
  - Slot waktu: tiap antrian memperoleh alokasi waktu CPU tertentu yg digunakan untuk menjadwalkan proses2 pd antrian tsb (mis. 80% utk foreground, 20% utk background)

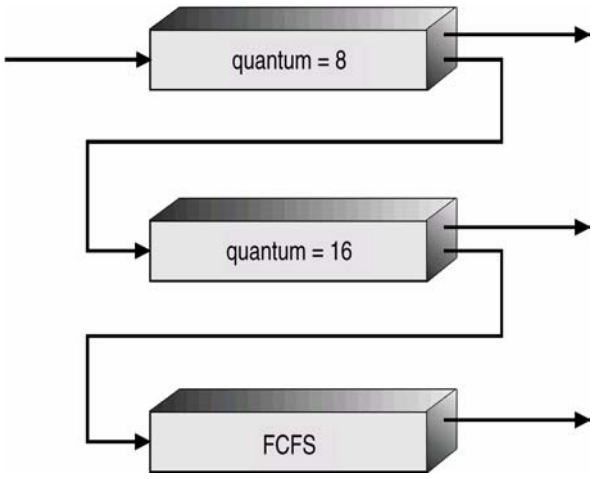
Multilevel-Queue (2)



Multilevel Feedback (1)

- Proses dapat berpindah antar berbagai antrian; mis. Implementasi aging
- Scheduler multilevel-feedback-queue didefinisikan oleh parameter berikut:
  - Banyaknya antrian
  - Algoritma penjadwalan pd tiap antrian
  - Metode utk menentukan kapan proses dinaikkan atau diturunkan prioritasnya
  - Metode utk menentukan antrian mana yg akan dimasuki proses ketika proses tsb membutuhkan layanan

Multilevel Feedback: Contoh (1)



Multilevel Feedback: Contoh (2)

- 3 antrian:
  - Q_0 – kuantum waktu 8 ms
  - Q_1 – kuantum waktu 16 ms
  - Q_2 – FCFS
- Penjadwalan
  - Proses baru masuk antrian Q_0. Ketika mendapat giliran CPU, proses ini memperoleh jatah waktu 8ms. Jika tidak selesai dalam 8 ms, proses dipindahkan ke antrian Q_1
  - Pada Q_1 proses dilayani kembali oleh FCFS dan memperoleh tambahan 16 ms. Jika masih belum selesai, proses ini di-preempted dan dipindahkan ke antrian Q_2

Shortest Process Time

- Berdasarkan SJF, yg diubah utk sistem interaktif
  - Mempertimbangkan waktu respon rata-rata utk tiap input/command user
  - Estimasi dibuat berdasarkan respon sebelumnya
    - Waktu rata-rata sebelumnnya: T0, T1, T2, T3
    - Waktu rata-rata berikutnya diperkirakan:
      T0 /8 + T1 /8 + T2 /4 + T3 /2
  - Apa masalahnya?

Guaranteed Scheduling (QoS)

- Memberikan jaminan performansi bagi user
- Contoh:
  - Dgn n proses running, scheduler memastikan bahwa tiap proses memperoleh 1/n siklus CPU
- Penjadwalan:
  - Hitung rasio pemakaian waktu CPU sebenarnya dgn waktu CPU yg dijatahkan
  - Pilih proses dgn rasio terendah
  - Dapat menyebabkan starvation?

Lottery Scheduling

- Lebih umum digunakan
- Berdasarkan probabilitas:
  - Tiap proses diberikan tiket undian
  - Pd saat penjadwalan, salah satu tiket dipilih secara acak, dan proses yg memiliki tiket akan dialokasikan CPU
  - Proses dgn prioritas lebih tinggi memiliki lebih banyak tiket
- Keuntungan:
  - sederhana
  - Sangat responsif
  - Dapat mendukung kerjasama antar proses
  - Mudah utk mendukung kebutuhan prioritas dan proporsionalitas

Fair-share Scheduling

- Apakah Round-robin cukup adil?
  - Ya (dari sudut pandang proses)
  - Mungkin tidak (dari sudut pandang user)
- fair share scheduling berbasis user
  - Tiap user memperoleh jatah secara adil
- Contoh:
  - Ani memiliki 4 proses: A1, A2, A3, A4
  - Budi memiliki 1 proses: B1
  - A1, A2, A3, dan A4 berempat berhak atas 50% waktu CPU
  - B1 sendiri berhak atas 50% waktu CPU

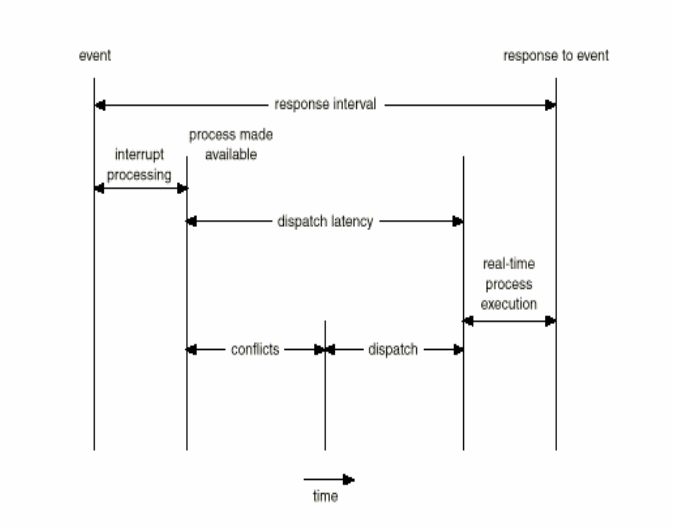
Multiprosesor Scheduling

- Lebih kompleks jika terdapat beberapa CPU
- Pilihan penjadwalan:
  - **Self-Scheduled (symmetric)**: tiap CPU menjalankan proses dari antrian ready sesuai dgn skema penjadwalannya
  - **Master-Slave**
    - 1 CPU menjadwalkan CPU lainnya
  - **Asymmetric**
    - 1 CPU menjalankan kernel dan yg lainnya menjalankan aplikasi user
    - 1 CPU menangani jaringan dan yg lainnya menangani aplikasi

Real-time Scheduling

- Sistem *Hard real-time*– perlu jaminan menyelesaikan task kritis dlm waktu tertentu
- Komputasi *Soft real-time*– proses kritis memperoleh prioritas lebih tinggi dari proses lainnya

Dispatch Latency (1)



Dispatch Latency (2)

- Tujuan: menjaga dispatch latency
- Masalah: system call
  - OS sederhana dpt memaksa proses utk menunggu selesainya system call atau berlangsungnya I/O block
- Solusi: perlu system calls yg dpt di-preempted
  - Sisipkan titik preemption (dpt ditempatkan pd lokasi dimana struktur data kernel tdk dimodifikasi)
  - Membuat kernel dpt di-preempted (semua struktur kernel harus diproteksi melalui berbagai mekanisme sinkronisasi)