Understanding Operating Systems
Fifth Edition

Chapter 4
Processor Management

Topic Hari Ini

- Perbedaan antara penjadwalan job dan penjadwalan proses, serta hubungan keduanya.
- Keuntungan dan kerugian algoritma penjadwalan proses yang sifatnya preemptive dan yang nonpreemptive.
- Tujuan aturan **(policy)** penjadwalan proses pada CPU *single-core*
- 6 macam algoritma penjadwalan proses
- Peran interupsi **(interupt)** internal dan tugas yang dilakukan oleh **interupt handler**

Overview

- **`Single-user systems`** (2 keadaan)
  - Keadaan sibuk: mengeksekusi sebuah job
  - Keaadaan **`Idle`**: tidak melakukan apa-apa
  - Manajemen prosesor sederhana
- **`Program`** (**`job`**)
  - Unit yang tidak aktif
    - File disimpan di sebuah disk
  - Sebuah unit kerja yang di`submit` oleh user
    - Bukan sebuah proses

Overview (continued)

- **Process** (**task**)
  - Entitas aktif
    - Membutuhkan resourse untuk dapat berfungsi
    - Prosesor dan beberapa register
- **Thread**
  - Bagian dari sebuah proses
  - Berjalan mandiri
- **Processor**
  - Central processing unit (CPU)
  - Mengerjakan kalkulasi dan mengeksekusi program.

Overview (continued)

- **Multiprogramming environment**
  - Prosesor dialokasikan dalam periode waktu tertentu
  - Didealokasikan di saat yang tepat/memungkinkan
- **Interrupt**
  - ‘Panggilan bantuan’
  - Mengaktifasi program yang prioritas-nya lebih tinggi
- **Context Switch**
  - Menyimpan informasi atas **job** yang sedang diproses ketika diinterupsi job lain
- **Single processor**
  - Dapat digunakan untuk banyak **jobs (processes)**
  - Membutuhkan aturan dan algoritma penjadwalan

Tentang Teknologi **Multi-Core**

- **Processor (core) (inti)**
  - Terletak di chip
- **Multi-core CPU** (punya lebih dari 1 prosesor)
  - **Dual-core, quad-core**
- Satu chip bisa memiliki banyak **core**
  - Rekayasa **multi-core**
    - Mengatasi masalah kebocoran dan cepat panas
    - Banyak kalkulasi dapat terjadi bersamaan
    - Lebih kompleks dari **single core**: dibahas pada chapter 6

Job Scheduling VS Process Scheduling

- **Processor Manager**
  - Terdiri dari 2 sub-manajer
    - Ada hirarki diantara keduanya
- **Job Scheduler** punya hirarki lebih tinggi
  - Bertanggung jawab atas penjadwalan job
  - Inisiasi job didasarkan pada kriteria tertentu
- **Process Scheduler**: hirarkinya lebih rendah
  - Bertanggung jawab atas penjadwalan proses
  - Menentukan langkah-langkah eksekusi
  - Proses penjadwalan berdasarkan kriteria tertentu

Job Scheduling VS Process Scheduling (continued)

- Fungsi penjadwalan **job**
  - Memilih **job** yang masuk dari antrian
  - Menentukan kriteria inisiasi **job**
    - algoritma penjadwalan proses dan prioritas
- **Tujuan**
  - Urutan **job**
    - Utilisasi sumber daya sistem yang efisien
  - Menyeimbangkan antara interaksi I/O dan komputasi
  - Menjaga agar sebagian besar komponen sistem hampir selalu bekerja

Process Scheduler

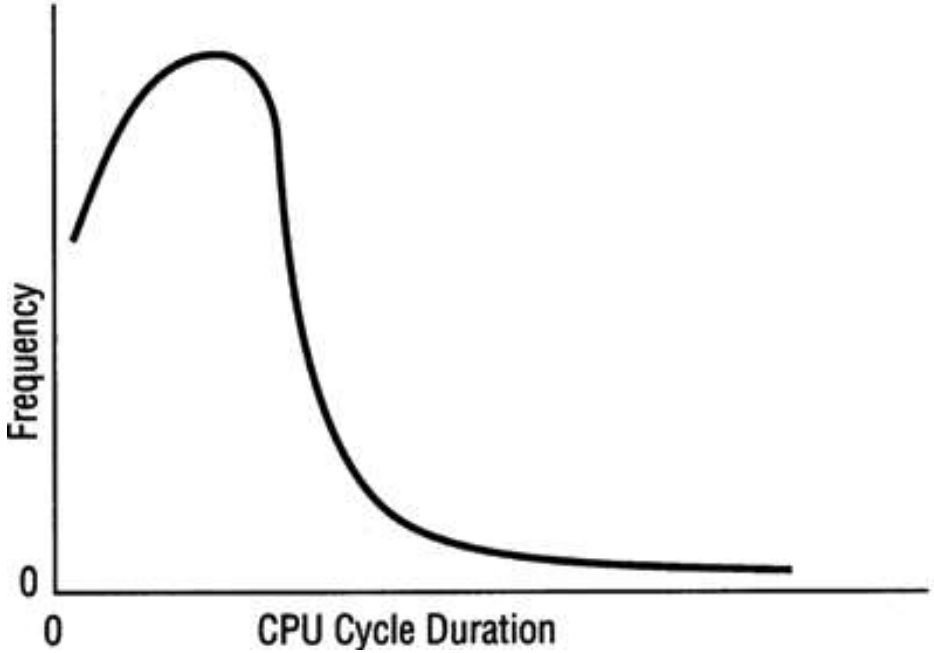
- Fungsi **Process Scheduler**
  - Menentukan **job** mana yang dapat menggunakan CPU
    - Kapan dan seberapa lama
  - Menentukan **process interupt**
  - Menentukan antrian untuk pergantian **job** saat eksekusi
  - Mengetahui ketika sebuah **job** sudah selesai
    - Menentukan terminasi **job**

Process Scheduler (continued)

- Mengeksploitasi sifat/metode umum program komputer
  - Program bergantian diantara 2 siklus
    - Siklus CPU dan I/O
    - Frekuensi dan durasi siklus CPU bervariasi
- Beberapa kecenderungan
  - **I/O-bound job**
    - Banyak siklus CPU singkat dan siklus I/O panjang (mencetak/print dokumen)
  - **CPU-bound job**
    - Banyak siklus CPU panjang dan siklus I/O pendek (kalkulasi matematis)

Process Scheduler (continued)

- **Kurva Distribusi Poisson**
  - Siklus CPU untuk `I/O-bound` dan `CPU-bound jobs`



Gambar 4.1
Distribusi waktu siklus CPU. Distribusi ini menunjukkan bahwa jumlah job yang meminta siklus CPU pendek jauh lebih banyak dibanding yang meminta siklus CPU panjang.

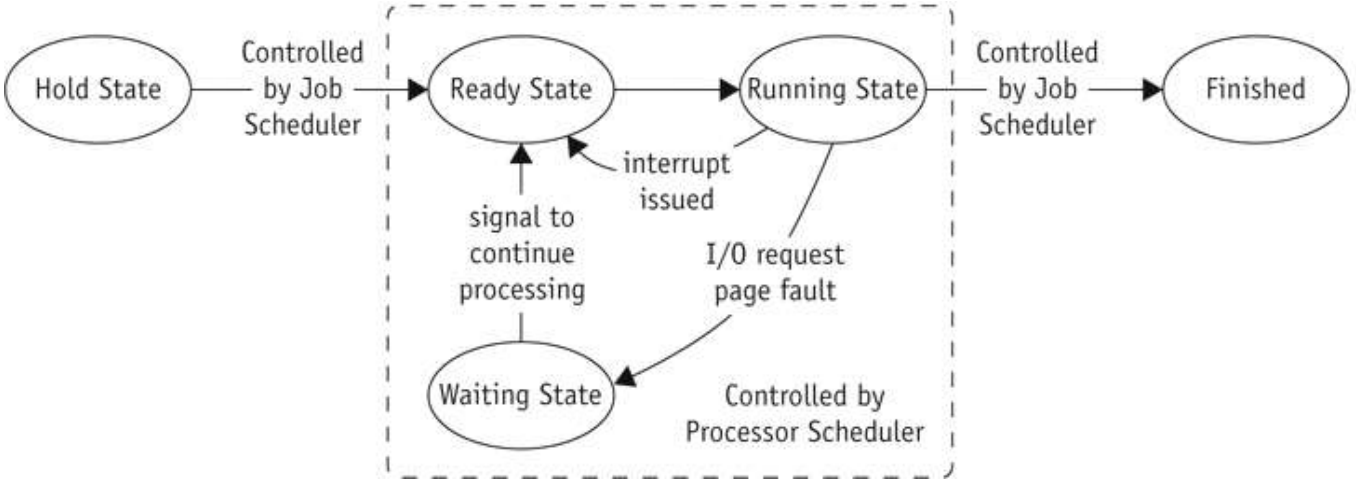
Process Scheduler (continued)

- **Middle-level scheduler**: layer ketiga
- Ditemukan pada lingkungan dengan tingkat interaktivitas tinggi
  - Menangani **overloading**
    - Memindahkan job aktif dari memori
    - Mengurangi tingkat **multiprogramming**
  - Hasilnya job diselesaikan lebih cepat
- **Single-user environment**
  - Tidak ada perbedaan antara penjadwalan **job** dan **process**
  - Dalam satu waktu hanya ada 1 job yang aktif
    - Resourse sistem didedikasikan sepenuhnya selama job dieksekusi

Status **Job** dan **Process**

- 5 keadaan:
  - **HOLD**
  - **READY**
  - **WAITING**
  - **RUNNING**
  - **FINISHED**
- disebut **job status** atau **process status**

Job and Process Status (continued)



(figure 4.2)

A typical job (or process) changes status as it moves through the system from HOLD to FINISHED.

Job and Process Status (continued)

- User mengirimkan **job (batch/interactive)**
  - **Job** diterima
    - Diberi status **HOLD** dan ditempatkan dalam antrian
  - Keadaan **job** berubah dari **HOLD** ke **READY**
    - Mengindikasikan kalau **job** sedang menunggu CPU
  - Keadaan **job** berubah dari **READY** ke **RUNNING**
    - Ketika dapat menggunakan CPU dan diproses
  - Keadaan **job** berubah dari **RUNNING** ke **WAITING**
    - Karena membutuhkan sumber daya yang tidak tersedia
  - Keadaan **job** berubah ke **FINISHED**
    - **Job** selesai dieksekusi (baik sukses maupun gagal)

Job and Process Status (continued)

- **Job Scheduler (JS)** atau **Process Scheduler (PS)** bertanggung jawab atas transisi keadaan job.
  - **HOLD** ke **READY**
    - **JS** menginisiasi berdasarkan aturan yang sudah ditentukan sebelumhya
  - **READY** ke **RUNNING**
    - **PS** menginisiasi berdasarkan algoritma yang sudah ditentukan sebelumnya
  - **RUNNING** kembali lagi ke **READY**
    - **PS** menginisiasi menurut batas waktu atau kriteria lain yang sudah ditentukan sebelumnya.
  - **RUNNING** ke **WAITING**
    - **PS** menginisiasi dengan instruksi ke **job**

Job and Process Status (continued)

- **WAITING** ke **READY**
  - **PS** menginisiasi dengan sinyal dari manajer I/O
  - Sinyal memberitahukan kalau permintaan I/O dikabulkan
- **RUNNING** ke **FINISHED**
  - **PS** atau **JS** menginisiasi saat job diselesaikan
  - Baik dengan sempurna maupun dengan **error**

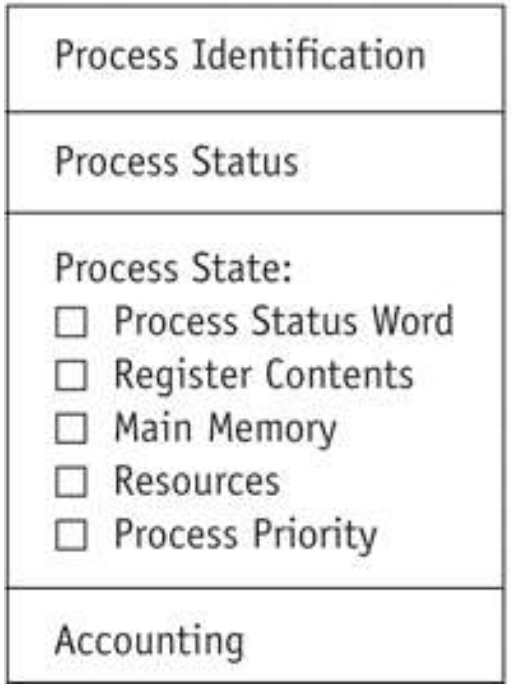
Process Control Blocks

- Struktur data
- Berisi informasi tentang job
  - Apa
  - Akan kemana
  - Seberapa banyak proses diselesaikan
  - Disimpan dimana
  - Berapa lama waktu yang diperlukan untuk menggunakan resourse

Process Control Blocks (continued)

- Komponen **Process Control Block (PCB)**
  - Identifikasi **process**
    - *Unique*
  - Status **process**
    - Keadaan **job** (**HOLD, READY, RUNNING, WAITING**)
  - Keadaan **process**
    - Isi register **process status word**, info memori utama, resource, prioritas **process**
  - Accounting
    - Pengukuran billing dan kinerja
    - Waktu CPU, waktu total, penggunaan memori, operasi I/O, jumlah pembacaan **input records**, dll.

Process Control Blocks (continued)



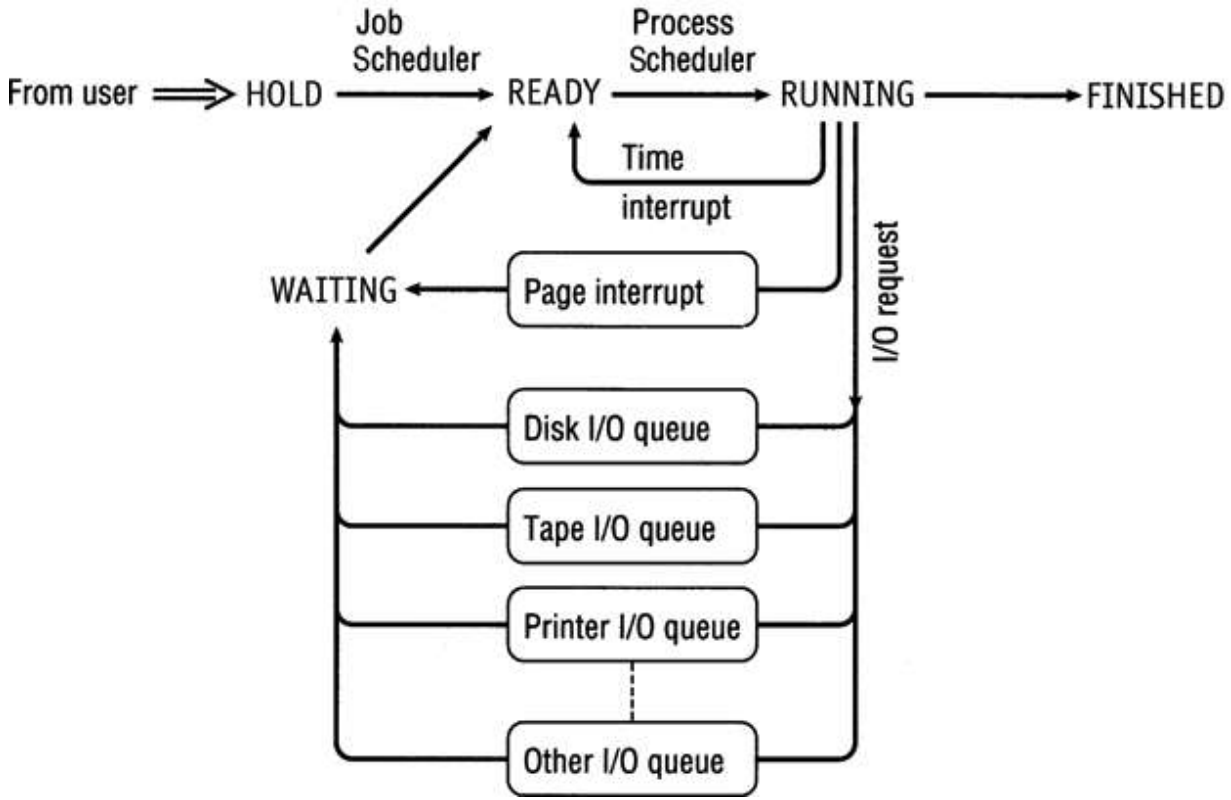
(figure 4.3)

Contents of each job's Process Control Block.

PCBs and Queuing (continued)

- **Job PCB**
  - Dibuat ketika **Job Scheduler** menerima **job**
  - Selalu diupdate selama **job** dieksekusi
  - **Antrian** menggunakan **PCB** untuk melacak **jobs**
  - Berisi seluruh data penting **job management processing**
  - **PCB** dihubungkan (*linked)* untuk membentuk antrian (job tidak di-*link*)
  - Mengatur antrian menggunakan algoritma dan aturan **process scheduling**

PCBs and Queuing (continued)



(figure 4.4)

Queuing paths from HOLD to FINISHED. The Job and Processor schedulers release the resources when the job leaves the RUNNING state.

Process Scheduling Policies

- Di lingkungan multiprogramming
  - Terdapat lebih banyak job dibandingkan sumber daya yang tersedia dalam satu waktu
- Tugas SO sebelum penjadwalan
  - Mengatasi 3 keterbatasan sistem
    - Jumlah sumber daya yang terbatas (disk drives, printers, tape drives)
    - Beberapa sumber daya tidak bisa digunakan bersama-sama (printers)
    - Beberapa sumber daya membutuhkan intervensi operator (tape drives)

Process Scheduling Policies (continued)

- Kriteria **process scheduling policy** yang baik
  - Memaksimalkan **throughput**
    - Menjalankan sebanyak mungkin job dalam satu waktu
  - Meminimalkan waktu respon
    - Lebih interaktif
  - Meminimalkan waktu **turnaround**
    - Job cepat keluar masuk sistem
  - Meminimalkan waktu tunggu
    - Memindahkan job dari antrian **READY** dengan cepat

Process Scheduling Policies (continued)

  - Memaksimalkan efisiensi CPU
    - Menjaga CPU tetap sibuk 100% waktu
  - Memastikan semua job mendapat giliran
    - Memberikan setiap job waktu yang sama untuk memakai CPU dan I/O
- Kriteria untuk **scheduling policy** ditetapkan oleh desainer sistem

Process Scheduling Policies (continued)

- Masalah
  - Job menggunakan CPU lama sekali sebelum permintaan I/O dibuat
    - Akibatnya antrian **READY** membengkak sementara antrian I/O kosong
    - Membuat sistem menjadi tidak seimbang
- Perbaikan
  - **Interrupt**
    - Digunakan oleh **process scheduler** berdasarkan batas waktu expire
    - Aktifitas job yang ada dihentikan sementara **(suspended)**
    - Job dijadwalkan ulang ke antrian **READY**

Process Scheduling Policies (continued)

- Tipe **scheduling policies**
  - **Preemptive**
    - Digunakan di sistem **time-sharing**
    - Menginterupsi pemrosesan **job**
    - Mentransfer CPU ke **job** lain
  - **Nonpreemptive**
    - Berfungsi tanpa interupsi internal
  - **Infinite loops** tetap diinterupsi di keduanya

Process Scheduling Algorithms

- Berdasarkan aturan tertentu
  - Mengalokasikan CPU dan memindahkan job di dalam sistem
- 6 jenis algoritma
  - **First-come, first-served (FCFS)**
  - **Shortest job next (SJN)**
  - **Priority scheduling**
  - **Shortest remaining time (SRT)**
  - **Round robin**
  - **Multiple-level queues**
- Sistem yang ada saat ini menekankan pada interaktivitas dan waktu respon (menggunakan **preemptive policies**)

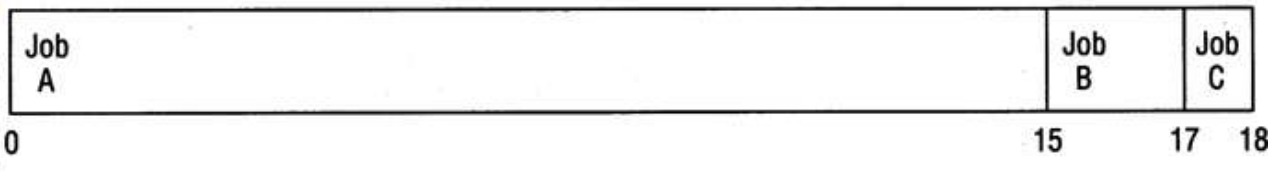
First-Come, First-Served

- **Nonpreemptive**
- Job ditangani berdasarkan waktu kedatangan
  - Job yang datang awal, ditangani lebih awal
- Implementasi algoritma sederhana
  - Menggunakan antrian **first-in, first-out (FIFO)**
- Bagus untuk sistem **batch**
- Tidak bisa digunakan dalam sistem interaktif
  - Waktu **turnaround** tidak dapat diprediksi
- Kekurangan
  - Rata-rata waktu **turnaround** bervariasi; jarang diminimalisir

First-Come, First-Served (continued)

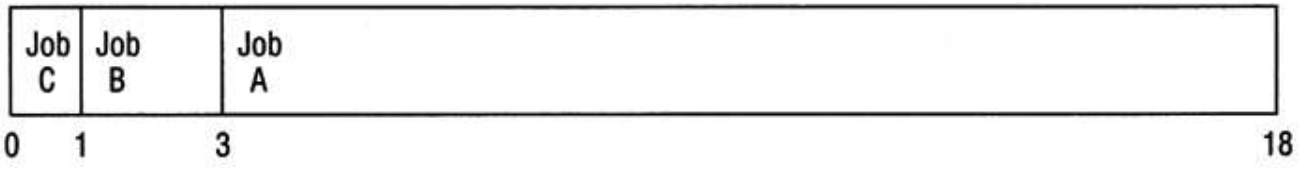
(figure 4.5)

Timeline for job sequence A, B, C using the FCFS algorithm



(figure 4.6)

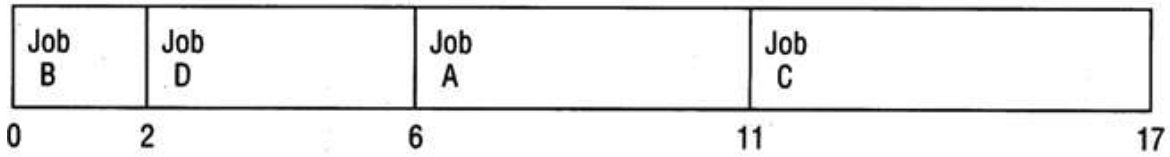
Timeline for job sequence C, B, A using the FCFS algorithm.



Shortest Job Next

- **Nonpreemptive**
- Dikenal juga dengan sebutan **shortest job first (SJF)**
- Job ditangani berdasarkan panjang waktu siklus CPU
- Implementasinya mudah dalam sistem **batch**
  - Waktu yang diperlukan untuk menggunakan CPU sudah terlebih dahulu diketahui
- Tidak cocok untuk sistem interaktif
- Algoritma optimal
  - Semua job tersedia dalam waktu bersama-sama
  - Perkiraan CPU tersedia dan akurat

Shortest Job Next (continued)



(figure 4.7)

Timeline for job sequence B, D, A, C using the SJN algorithm.

Priority Scheduling

- **Nonpreemptive**
- Job yang penting memiliki prioritas lebih tinggi
  - Program dengan prioritas lebih tinggi di proses terlebih dahulu
  - Tidak ada interupsi sampa siklus CPU selesai
- Antrian **READY** bisa jadi memiliku 2 job dengan prioritas yang sama
  - Gunakan **FCFS** untuk job dengan prioritas sama
- Administrator sistem dan **Processor Manager** menggunakan metode berbeda-beda untuk menentukan prioritas

Priority Scheduling (continued)

- Metode penetapan prioritas oleh **Processor Manager**
  - Kebutuhan memori
    - Jobs yang membutuhkan banyak memori
    - Diberi prioritas rendah (dan sebaliknya)
  - Jumlah dan jenis perangkat periferal
    - Jobs yang membutuhkan banyak perangkat periferal
    - Diberi prioritas rendah (dan sebaliknya)

Priority Scheduling (continued)

- Total waktu CPU
  - Jobs yang memiliku siklus waktu CPU yang lama
  - Diberi prioritas rendah (dan sebaliknya)
- Jumlah waktu yang sudah dihabiskan di dalam sistem
  - Waktu total sejak job mulai diproses
  - Naikkan prioritas kalau job sudah diproses terlalu lama

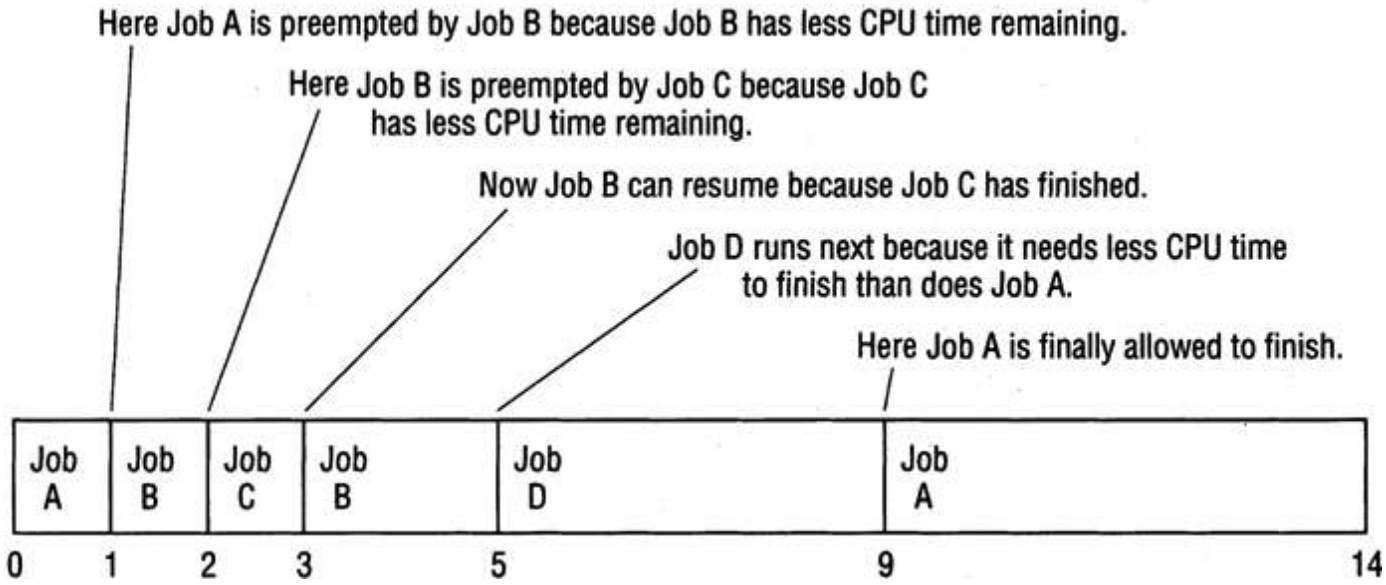
Shortest Remaining Time

- Versi **preemptive SJN**
- Prosesor dialokasikan ke job yang sudah hampir selesai dieksekusi
  - **Preemptive** jika ada job baru yang lebih mendekati selesai
- Sering digunakan dalam sistem **batch**
  - Job-job pendek diberi prioritas
- Tidak bisa diimplementasikan untuk sistem interaktif
- Lebih banyak overhead dibanding **SJN**
  - Sistem mengamati waktu CPU untuk job di antrian **READY**
  - Menjalankan **context switching**

Shortest Remaining Time (continued)

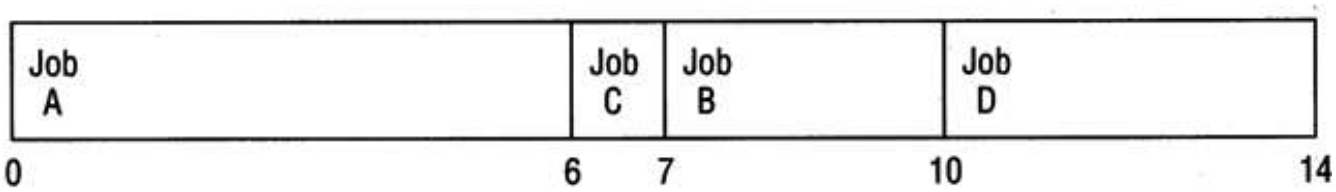
(figure 4.8)

Timeline for job sequence A, B, C, D using the preemptive SRT algorithm. Each job is interrupted after one CPU cycle if another job is waiting with less CPU time remaining.



(figure 4.9)

Timeline for the same job sequence A, B, C, D using the nonpreemptive SJN algorithm.



Round Robin

- **Preemptive**
- Banyak sekali digunakan dalam sistem interaktif
- Berdasarkan irisan waktu yang sudah ditentukan sebelumnya
  - Setiap job diberikan **time quantum**
- Ukuran time quantum
  - Penting bagi kinerja sistem
  - Bervariasi mulai dari 100 ms hingga 1-2 seconds
- CPU dibagi merata diantara proses – proses yang aktif
  - Tidak didominasi satu job saja

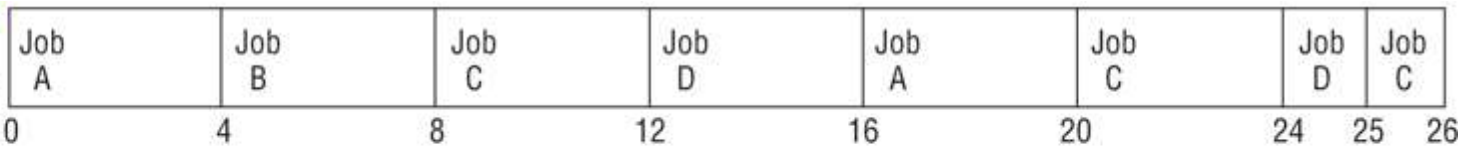
Round Robin (continued)

- Job ditempatkan dalam antrian **READY** (skema **FCFS**)
- **Process Scheduler** memilih job pertama
  - Set timer ke time quantum
  - Alokasikan CPU
- Timer expires
- Jika siklus CPU job > time quantum
  - Job dihentikan sementara dan ditempatkan di akhir antrian **READY**
  - Informasi disimpan di **PCB**

Round Robin (continued)

- Jika siklus CPU job < time quantum
  - Job diselesaikan: sumber daya yang telah dialokasikan dirilis dan job dikembalikan ke user
  - Di interupsi oleh permintaan I/O: informasi disimpan di **PCB** dan di-link ke antrian I/O
- Setelah permintaan I/O dipenuhi
  - Job dikembalikan ke akhir antriandan menunggu CPU

Round Robin (continued)

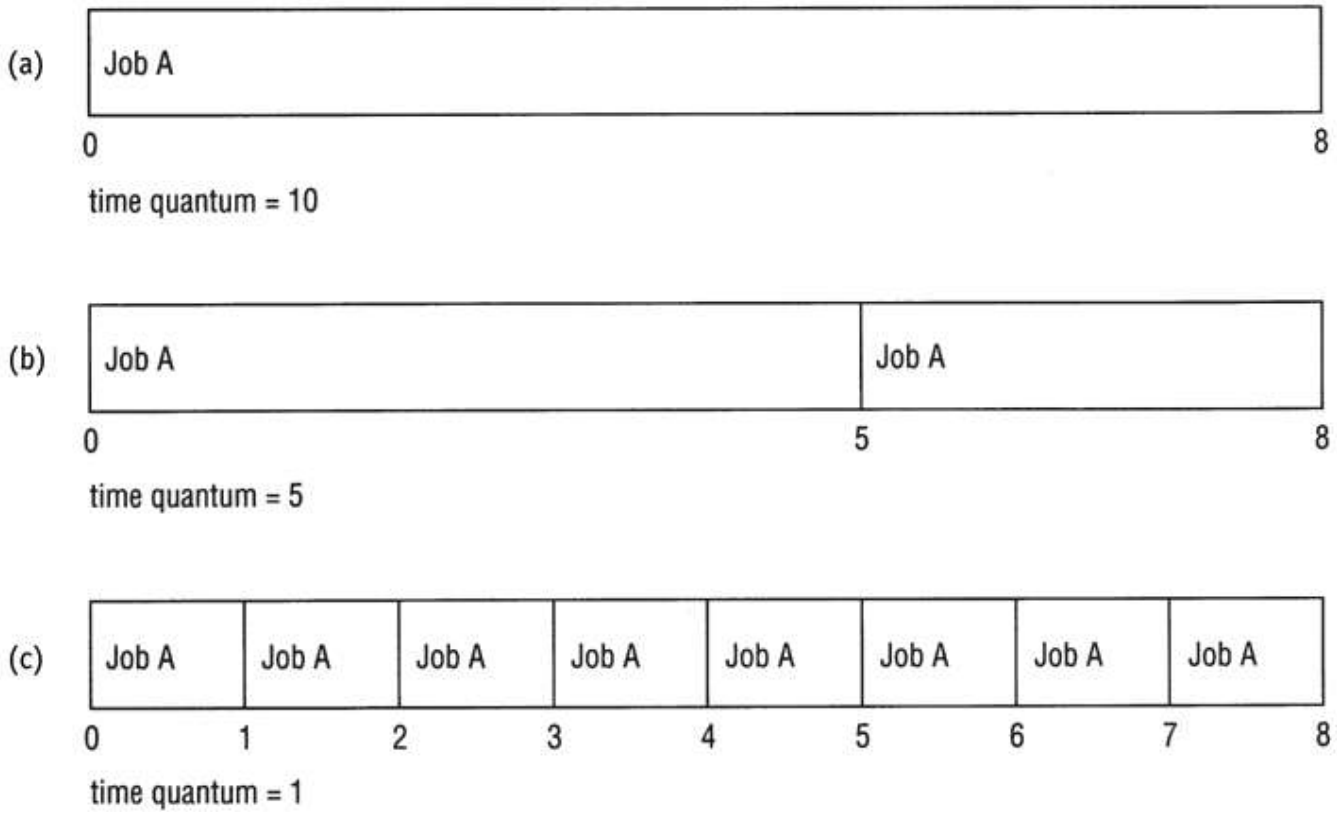


(figure 4.10) *Timeline for job sequence A, B, C, D using the preemptive round robin algorithm with time slices of 4 ms.*

Round Robin (continued)

- Efficiency
  - Depends on time quantum size
    - In relation to average CPU cycle
- Quantum too large (larger than most CPU cycles)
  - Algorithm reduces to FCFS scheme
- Quantum too small
  - Context switching occurs
    - Job execution slows down
    - Overhead dramatically increased

Round Robin (continued)



(figure 4.11)

Context switches for Job A with three different time quantums. In (a) the job finishes before the time quantum expires. In (b) and (c), the time quantum expires first, interrupting the job.

Round Robin (continued)

- Ukuran time quantum terbaik
  - Tergantung sistem
    - **Interactive**: faktor kunci pada waktu respon
    - **Batch:** faktor kunci pada waktu **turnaround**
  - Petunjuk umum
    - Cukup lama untuk menyelesaikan 80% siklus CPU
    - Paling tidak 100 kali lebih lama dari waktu yang dibutuhkan untuk **context switch**

Multiple-Level Queues

- Bekerja berdampingan dengan skema lain
- Baik untuk sistem dimana job dikelompokkan atas karakteristik yang sama
  - Berdasarkan prioritas
    - Antrian berbeda untuk setiap level prioritas
  - **CPU-bound jobs** di satu antrian dan **I/O-bound jobs** di antrian lain
  - **Hybrid environment**
    - Job **batch** di antrian latar belakang
    - Job **Interactive** di antrian depan
- **Scheduling policy** berdasarkan skema yang telah ditentukan
- 4 metode utama

Case 1: Tidak ada pergerakan diantara antrian

- Sederhana
- Menguntungkan job dengan prioritas tinggi
  - Prosesor dialokasikan menggunakan **FCFS**
- Prosesor dialokasikan ke job dengan priotitas rendah
  - Hanya jika antrian job dengan prioritas tinggi kosong
- Lingkungan yang bagus:
  - Terdapat sedikit job dengan prioritas tinggi
  - Menghabiskan lebih banyak waktu untuk job dengan prioritas rendah

Case 2: Ada pergerakan diantara antrian

- Prosesor menyesuaikan prioritas setiap job
- Job ber-prioritas tinggi
  - Prioritas awal disukai
    - Diperlakukan seperti job-job yang lain setelahnya
- **`Quantum interrupt`**
  - Job dihentikan sementara
    - Dipindahkan ke antrian dengan prioritas lebih rendah
    - Nantinya prioritas bisa dinaikkan lagi
- Baik untuk lingkungan:
  - Jobs ditangani berdasarkan karakteristik siklus (CPU atau I/O)
  - Sistem interaktif

Case 3: Variable Time Quantum Per Queue

- Varian Case 2: pergerakan diantara antrian
- Setiap antrian diberikan ukuran quantum time
  - Ukurannya 2 kali lebih panjang dari antrian sebelumnya
- **Turnaround** tinggi untuk **CPU-bound jobs**
- **CPU-bound jobs** dieksekusi lebih lama dan diberikan waktu lebih lama
  - Mempercepat waktu selesai

Case 4: Penuaan

- Memastikan job antrian level lebih rendah pada akhirnya selesai dieksekusi
- Sistem melacak waktu tunggu job
- Kalau terlalu “tua”
  - Sistem memindahkan job ke antrian yang lebih tinggi
  - Seterusnya sampai job “tua” berada di antrian tertinggi
  - Memungkinkan perpindahan job tua ke antrian tertinggi
- Keuntungan
  - Menjaga jangan sampai ada sebuah job yang terus-menerus tertunda-tunda sampai tidak pernah selesai
    - Masalah utama: akan didiskusikan di chapter 5

Tentang Interupsi (**interupt**)

- Tipe interupsi
  - **Page interrupt (memory manager)**
    - Mengakomodasi permintaan **job**
  - Interupsi karena time quantum kadaluarsa
  - Interupsi I/O
    - Disebabkan karena perintah **READ** or **WRITE** I/O
  - Interupsi internal
    - **Synchronous**
    - Disebabkan oleh operasi aritmatika atau instruksi job

Tentang Interupsi (**`interupt`**) (continued)

- Interupsi operasi aritmatika ilegal
  - Pembagian oleh bilangan 0; operasi floating-point yang buruk
- Interupsi intruksi ilegal job
  - Percobaan akses ke penyimpanan yang diproteksi
- **`Interrupt handler`**
  - Program pengendalian
    - Menangani urutan kejadian saat interupsi

Tentang Interupsi (**interupt**) (continued)

- SO mendeteksi erorr yang tidak bisa di **recover**
  - Urutan **interrupt handler**
    - Tipe interupsi dideskripsikan dan disimpan
    - Keadaan proses yang diinterupsi disimpan
    - Interupsi diproses
    - **Processor** dikembalikan ke operasi normal

MINGGU DEPAN MID TERM!

BAHAN CHAPTER 1 - 4

dan

MATERI PRESENTASI