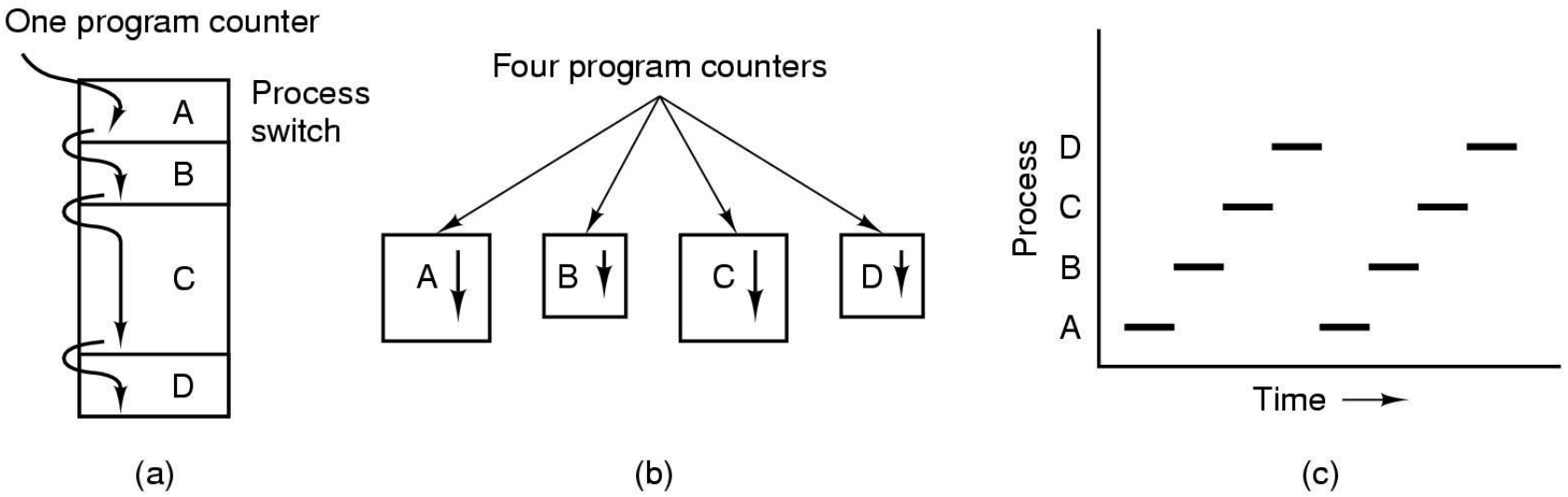

# SISTEM OPERASI

Departemen Ilmu Komputer/ Informatika
Universitas Diponegoro
Semester Gasal 2018/ 2019

# Proses

1. Deskripsi Proses
2. Pemodelan Proses
3. Implementasi Proses
4. Pengalihan Proses dan Konteks

# Proses Pada Multiprogramming




a) Multiprograming untuk empat program
b) Model konseptual untuk 4 proses yang independen dan sequensial
c) Hanya satu program yang aktif

# Deskripsi Proses

- Proses ~ Task, program yang sedang di eksekusi
- Analogi : resep vs memasak
- Unit kerja terkecil, secara individu memiliki sumber daya
- Sumber daya yang dimaksud adalah: prosesor, penyimpanan, perangkat I/O, perangkat komunikasi, dan data
  - SO mengelola semua proses di sistem, dan
  - SO mengalokasikan sumber daya ke proses2 sesuai kebijakan sistem

# Multiprograming, Multiprocessing, Distributed processing

- Multiprograming/ Multitasking
  - Banyak proses, satu prosesor
  - Pemakai memandang banyak proses dijalankan besamaan pada satu saat. Masing2 proses mendapat memori dan kendali sendiri.
  - SO mengalihkan layanan prosesor antara proses2 tersebut, Interleave → **pseudoparalelism**
  - Sifat proses2 yang dijalankan pd sistem multiprog:
    - Saling tidak bergantung (independent)
    - Satu program pada satu saat (one program at any instant)

# ...Multiprocessing, distributed (2)

- **Multiprocessing**
  - Banyak proses, banyak prosesor
  - Komputer dengan banyak prosesor (berbagi memori yang sama)
  - **true hardware paralelism**
- **Distributed processing**
  - Banyak proses, banyak sistem komputer yang tersebar (terdistribusi) pada suatu jaringan.

# Agenda

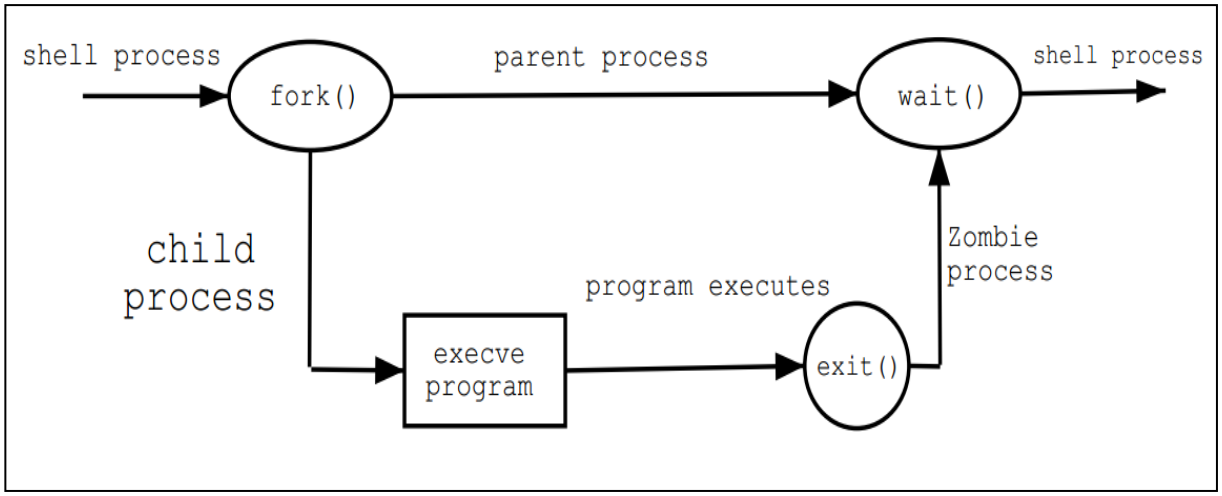
1. Deskripsi Proses
2. **Pemodelan Proses**
3. Implementasi Proses
4. Pengalihan Proses dan Konteks

# Pembuatan Proses

- Proses dibuat ketika:
  - Inisialisasi sistem
  - Proses menciptakan proses lain (proses anak)
  - User membuat proses baru → Pada lingkungan interactive
  - Inisiasi suatu batch job → Pada lingkungan batch: Mainframe

# ...Pembuatan Proses

- Linux
  - Fork()
  - Execve()



# Terminasi proses

- Proses akan terminasi ketika:
  - Normal Exit (voluntary)
    - exit()
  - Error Exit (voluntary)
    - file  not exist
  - Fatal Error (Involuntary)
    - Illegal instruction, dividing by zero
  - Killed by another process (Involuntary)

# Model Proses
## Tiga Keadaan(1)

- **Running**

  Prosesor sedang mengeksekusi instruksi suatu proses

- **Ready**

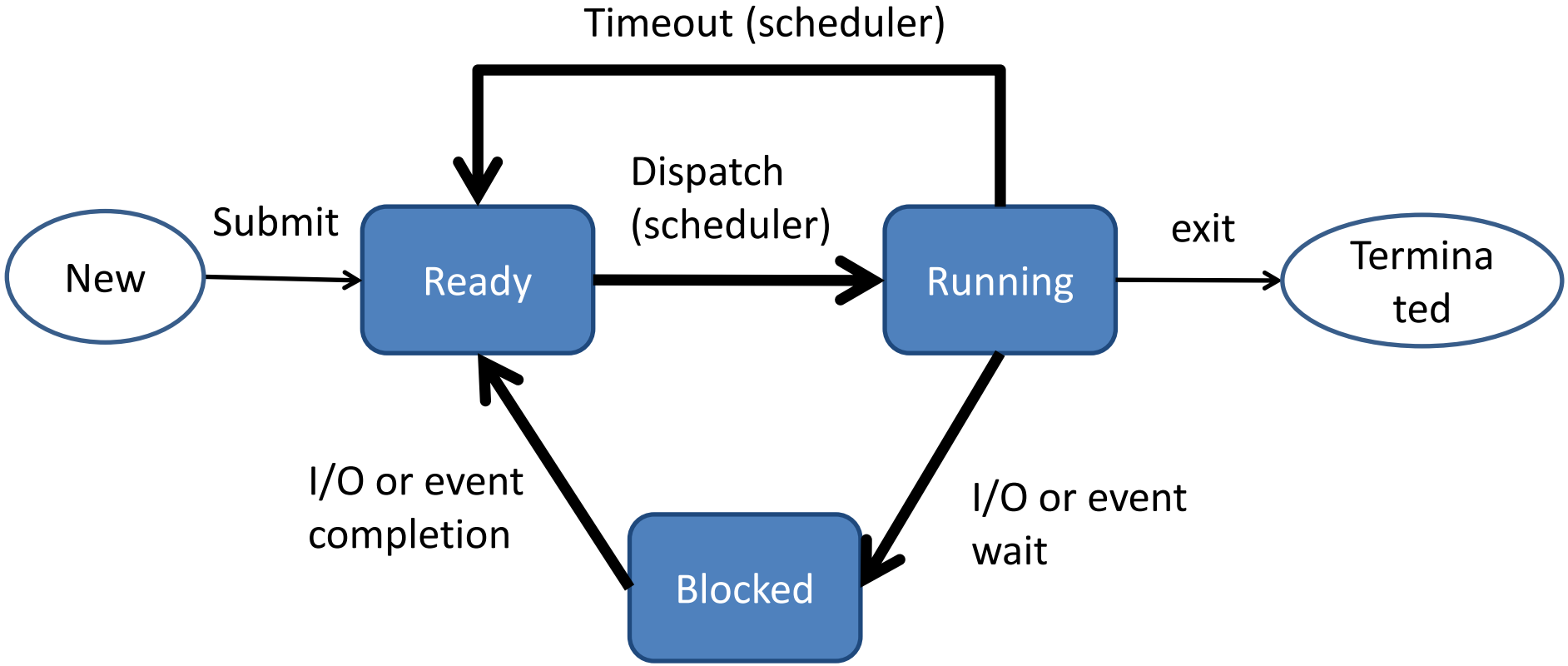
  Proses dalam keadaan siap dieksekusi, tetapi prosesor belum mengeksekusinya

- **Blocked**

  Proses menunggu kejadian tertentu selesai, mis: Menunggu selesainya operasi perangkat I/O

# Model Proses

## Tiga Keadaan(2)




Contoh:

```
$ Cat chapter1 chapter2 chapter3 | grep process
```

# Penjadwalan Proses

## First Come First Serve VS Shortest Job First

- Kita ambil contoh Kasus pada:
  - Sistem batch, Non Pre-emptive
  - Beberapa proses siap dilayani CPU, ketika *running* tidak bisa disela.
- SO bertanggungjawab mengelola antrian "terbaik".
- Suatu batch Job A, B, C, D memiliki waktu eksekusi 8, 7, 6, 5. Hitung Rata2 Waktu tunggu (*Waiting time*)

FCFS

| A | B | C | D |
|---|---|---|---|
| 8 | 7 | 6 | 5 |

SJF

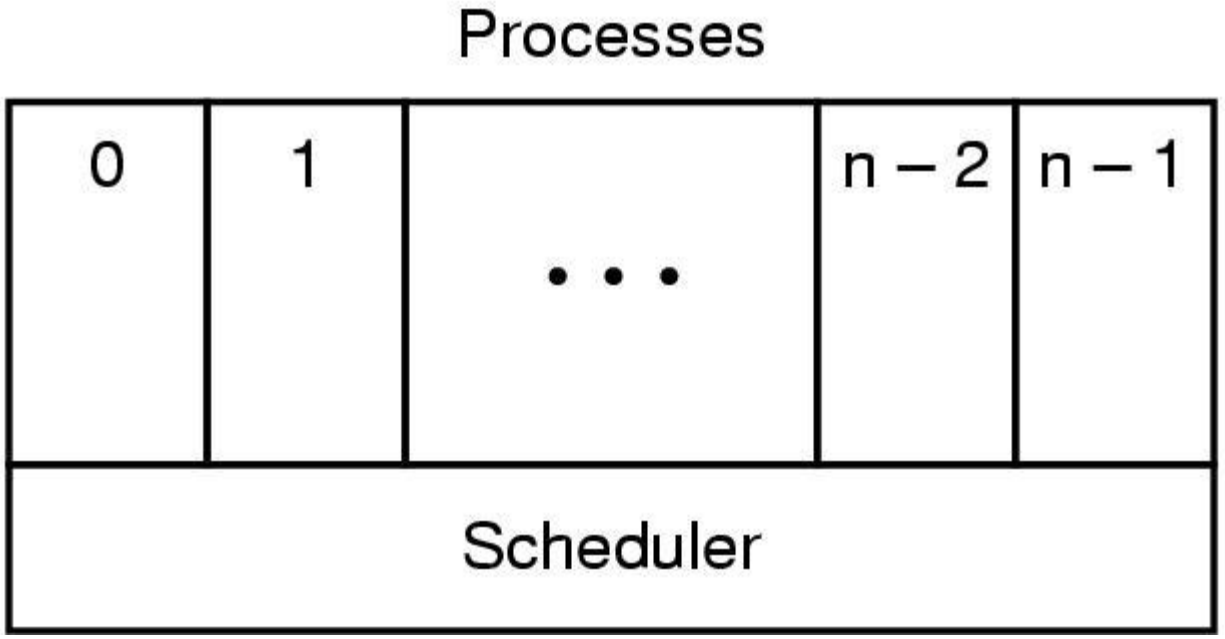
| D | C | B | A |
|---|---|---|---|
| 5 | 6 | 7 | 8 |

| Proses | Waiting time | Waiting time |
|---|---|---|
| | FCFS | SJF |
| A | 0 | 18 |
| B | 8 | 11 |
| C | 15 | 5 |
| D | 21 | 0 |
| **Rata-rata** | **11** | **8,5** |

# Agenda

1. Deskripsi Proses
2. Pemodelan Proses
3. Implementasi Proses
4. Pengalihan Proses dan Konteks

# Implementasi Proses (1)




The lowest layer of a process-structured operating system handles **interrupts** and **scheduling**. Above that layer are sequential processes.

# Implementasi Proses (2)

- Untuk mengimplementasikan model Proses, SO me-maintain suatu tabel yang disebut sebagai **Tabel Proses** (disebut juga *Process Control Blocks*)
- Tabel Proses tersebut berisi informasi penting yang harus disimpan ketika proses beralih dari keadaan ***running*** ke ***ready*** atau ***blocked*** agar nantinya dapat di restart kembali seolah-olah tidak pernah berhenti

# Implementasi Proses (3)

| Process management | Memory management | File management |
|---|---|---|
| Registers | Pointer to text segment info | Root directory |
| Program counter | Pointer to data segment info | Working directory |
| Program status word | Pointer to stack segment info | File descriptors |
| Stack pointer | | User ID |
| Process state | | Group ID |
| Priority | | |
| Scheduling parameters | | |
| Process ID | | |
| Parent process | | |
| Process group | | |
| Signals | | |
| Time when process started | | |
| CPU time used | | |
| Children's CPU time | | |
| Time of next alarm | | |

Some of the fields of a typical **process table** entry.

# Implementasi Proses (4)

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

Skeleton of what the lowest level of the operating system does when an interrupt occurs.

# Agenda

1. Deskripsi Proses
2. Pemodelan Proses
3. Implementasi Proses
4. **Pengalihan Proses dan Konteks**

# Pengalihan Konteks

- Apa yang dimaksud Konteks?

  Informasi yg mungkin diubah oleh eksekusi *interrupt handler*, dan yg akan diperlukan utk melanjutkan program yg sedang diinterupsi.

  Misal: isi PC, register proc, stack, dll

# Pengalihan proses VS pengalihan konteks?

- Ketika interupsi terjadi, SO menyimpan konteks program yang sedang dieksekusi & mengeset PC ke alamat awal program interupt handling.
- Kemudian setelah melayani interupsi, jika ternyata proses yang tadi diinterupsi dilanjutkan lagi, ini artinya baru saja terjadi (hanya) **pengalihan konteks**.
- Namun bila setelah melayani interupsi, ternyata ada proses baru yang harus dieksekusi maka perlu adanya kerja tambahan, yaitu SO harus menyimpan beberapa informasi lagi dari PCB sebagai referensi untuk melanjutkan eksekusi proses ini (yang baru saja di interupsi), kondisi ini kita sebut terjadi **pengalihan proses**.

# Pengalihan Proses

- **Pengalihan proses**

  Terjadi jika proses yg running beralih ke keadaan lain (*ready, blocked, dsb*), kemudian SO harus membuat perubahan2 berarti terhadap lingkungannya

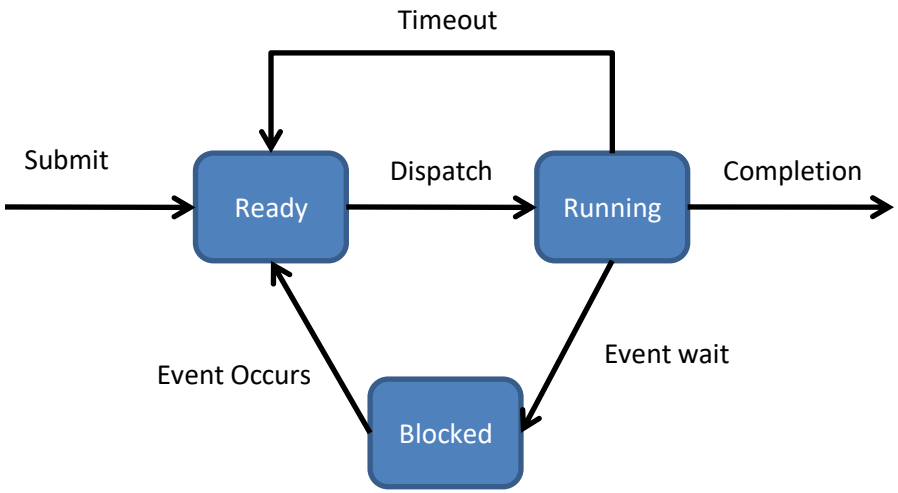
- Pengalihan proses melibatkan **pengalihan konteks** + **perubahan keadaan** → suatu usaha yg lbh besar dibanding hanya pengalihan mode/ konteks

# Penyebab pengalihan Proses

1. Interupsi
2. Trap
3. Supervisor Call

# Penyebab pengalihan proses
# 1. Interupsi

- Disebabkan kejadian **eksternal** dan tidak bergantung proses yg saat itu sdg dalam keadaan *running*
- Tipe interupsi :
  - ***Clock intterupt:*** proses telah mencapai batas waktu maksimum yang diijinkan
  - ***I/O interrupt:*** Event I/O selesai dikerjakan
  - ***Page/ memory fault:*** alamat memory berada di virtual memory (hard disk-I/O) sehingga perlu diambil ke memori utama



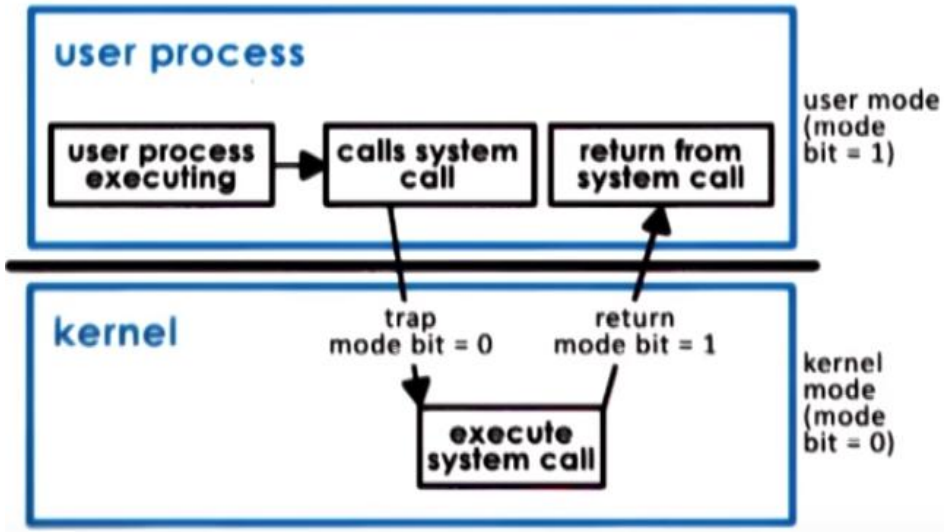

Identifikasikan pengaruh tipe-tipe interupsi tsb pada Diagram 3 keadaan proses !!!

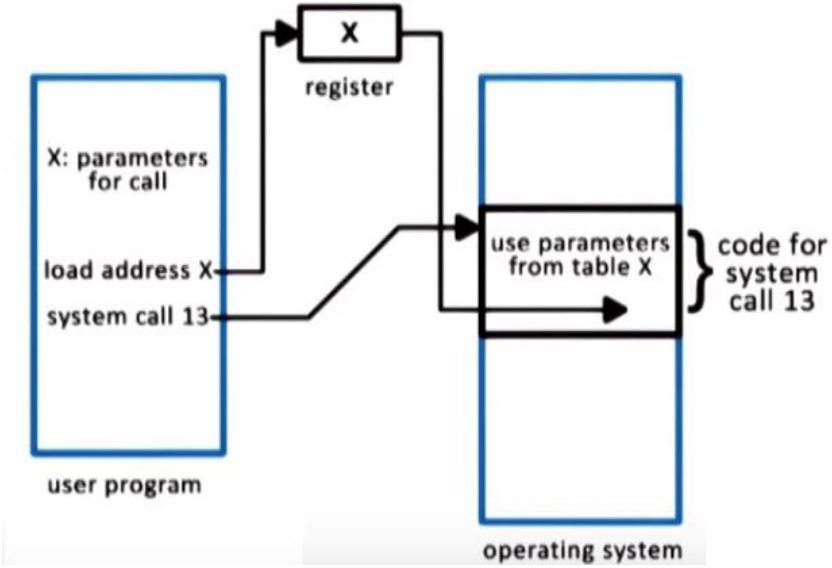
# Penyebab pengalihan proses:
# 2. Trap

- Interupsi yang terjadi karena kesalahan atau kondisi perkecualian (*exception conditions)* yg dihasilkan proses yg running, misalnya usaha ilegal mengakses file.
- SO menentukan :
  - Bila kesalahan fatal → Singkirkan proses yg sdg running → lakukan alih proses
  - Bila tidak fatal → prosedur pemulihan atau peringatan ke pemakai

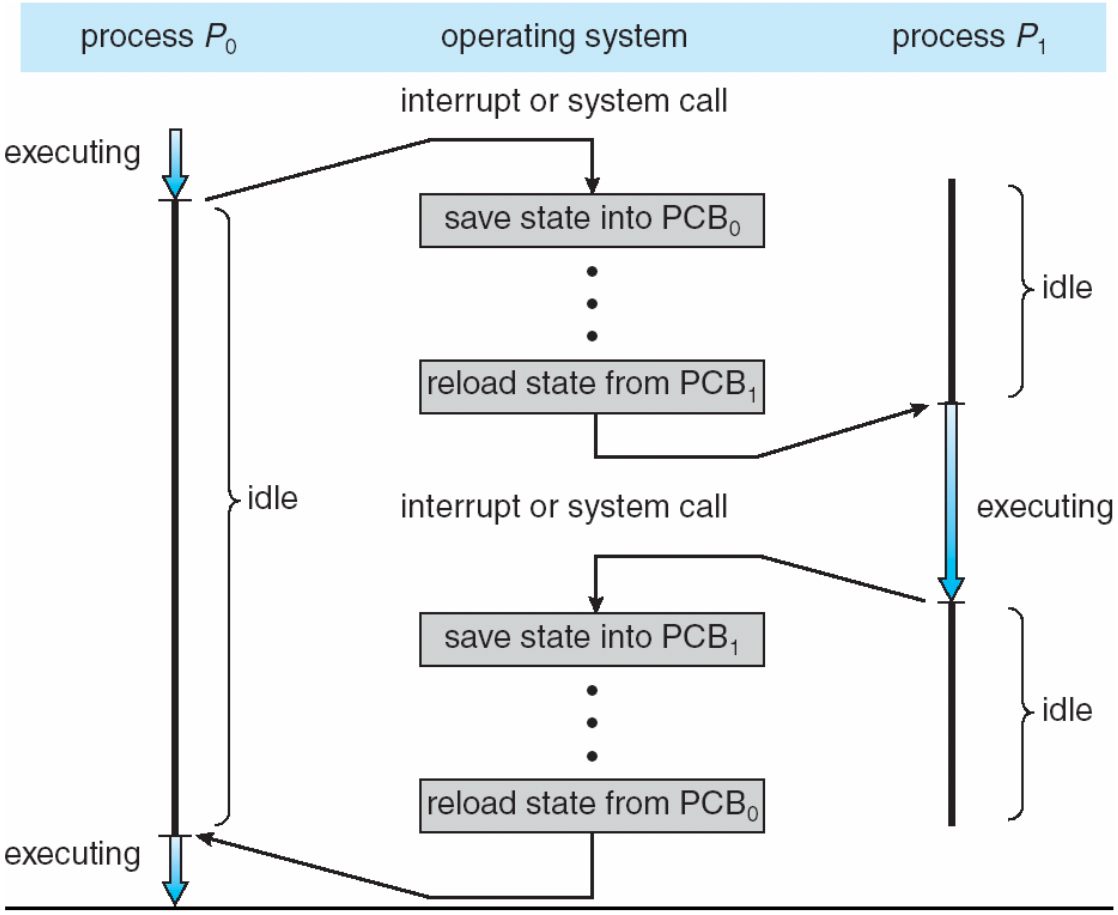
# Penyebab pengalihan proses
## 3. Supervisor call

- Ketika terjadi panggilan untuk meminta atau mengaktifkan bagian sistem operasi (kernel) → *system call*
  - Penggunaan system call mengakibatkan proses user *blocked* krn diaktifkannya proses kernel.

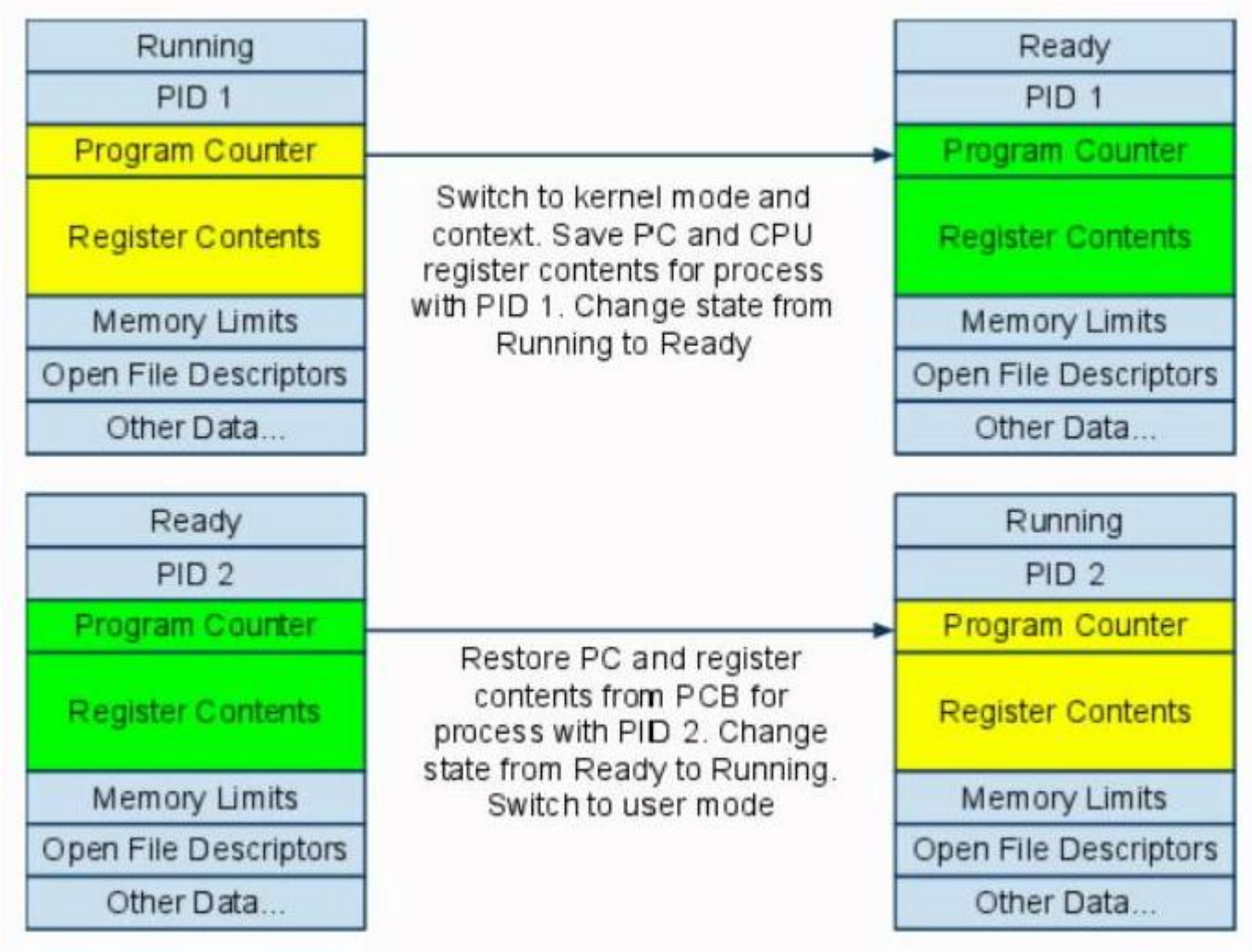


# Ilustrasi Pengalihan Proses (1)



# Ilustrasi Pengalihan Proses (2)



# ❑ Review

## Bagian 1. Deskripsi Proses

1. Jelaskan perbedaan antara proses dengan program.
2. Bagaimana multiprogramming dapat diimplementasikan menggunakan prosesor tunggal?

## Bagian 2. Pemodelan Proses

1. Apa saja penyebab terciptanya proses? Dan apa saja penyebab proses mengalami terminasi?
2. Jelaskan fungsi *fork()* dan *execve()* pada sistem Unix.
3. Gambarkan diagram 3 keadaan proses secara tepat.
4. Suatu urutan batch Job A, B, C, D memiliki waktu eksekusi 15, 8, 10, 3. Hitung: rata-rate *waiting time* untuk penjadwalan dengan FCFS dan SJF.

# ❑ Review

## Bagian 3. Pengalihan Proses dan Pengalihan Konteks

1. Jelaskan perbedaan *interrupt* dengan *trap*?
2. Jelaskan perbedaan pengalihan proses dengan pengalihan konteks.

## Bagian 4. Implementasi Proses

1. Jelaskan fungsi tabel proses (*Process Control Block*).
2. Mengapa **Interupsi** dan **Penjadwalan** terletak pada *layer* terbawah Sistem Operasi?