

# Proses

Praktikum Sistem Operasi

Ilmu Komputer IPB

2017

# Intro

# Tim Praktikum

- ▶ Auriza Rahmad Akbar
- ▶ M Mukhibillah Asshidiqy
- ▶ Kurnia Saputra
- ▶ Lu William Hanugra
- ▶ Selfi Qisthina

# Peraturan

- ▶ Pakaian sopan, tidak ketat
  - ▶ pelanggaran lebih dari 3 kali: sanksi sedang (nilai 0)
- ▶ Kehadiran minimal 80%
- ▶ Toleransi terlambat 15 menit
- ▶ Tidak membawa makanan ke lab



Gambar 1: Tata tertib mahasiswa IPB

# LMS

- ▶ <https://lms.ipb.ac.id/course/view.php?id=154>
  - ▶ *key*: so2017
- ▶ Buku acuan:
  - ▶ Silberschatz *et al.* 2013. *Operating System Concepts*. Ed ke-9.
- ▶ Proporsi nilai praktikum:
  - ▶ UTSP: 30%
  - ▶ UASP: 30%
  - ▶ Tugas: 40%

# Proses

# Apa itu proses?



# Apa itu proses?

Program yang sedang berjalan.

*A program in execution.*<sup>1</sup>

---

<sup>1</sup>Silberschatz et al. (2013), *Operating System Concepts*, hlm 105.

# Bagaimana cara menjalankan program?

Misalkan kita ingin menjalankan program Firefox. Ada berapa cara?

# Bagaimana cara menjalankan program?

Misalkan kita ingin menjalankan program Firefox. Ada berapa cara?

Dua cara:

1. **CLI**: buka *shell*, lalu ketikkan perintah `firefox`.
2. **GUI**: klik ikon Firefox pada menu aplikasi<sup>2</sup>.

---

<sup>2</sup>jika ikon diklik, program akan tetap dijalankan melalui *shell*; coba cek isi *file* `/usr/share/applications/firefox.desktop`.

# Shell

# Apa itu *shell*?

# Apa itu *shell*?

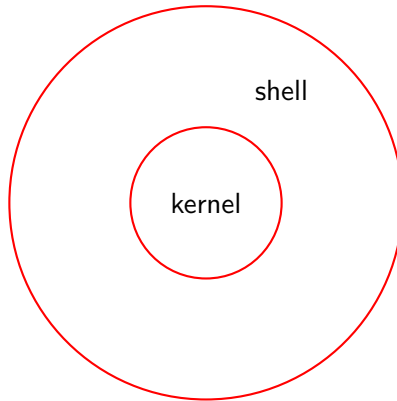
- ▶ *Shell* adalah antarmuka antara pengguna dengan *kernel*.

# Kernel vs Shell

- ▶ *kernel* = ...
- ▶ *shell* = ...

## Kernel vs Shell

- ▶ *kernel* = inti
- ▶ *shell* = kulit





# Kernel vs Shell

- ▶ *Kernel* adalah inti dari sistem operasi.
- ▶ *Shell* adalah antarmuka antara pengguna dengan *kernel*.
- ▶ *Shell* bertugas untuk menjalankan aplikasi pengguna.
  - ▶ *user* → *shell* → *kernel*.

# Contoh *kernel*

- ▶ UNIX
  - ▶ BSD
  - ▶ AIX
  - ▶ HP-UX
  - ▶ Solaris
  - ▶ Linux
- ▶ Windows NT

## Contoh *shell*

- ▶ Bourne shell (`sh`)
- ▶ Bourne-again shell (`bash`)
- ▶ Korn shell (`ksh`)
- ▶ Z shell (`zsh`)
- ▶ Windows PowerShell

## Bagaimana *shell* bisa membuat proses?

**Tugas:** baca Silberschatz *et al.* (2013), hlm 116–118 sebagai tugas sekaligus materi praktikum pekan depan.

system()

# Fungsi system()

```
int system(char *command);
```

- ▶ Menjalankan `command` dengan menjalankan *shell* terlebih dahulu<sup>3</sup>:
  - ▶ `sh -c "command"`

---

<sup>3</sup>lihat 'man system'

# Contoh

- ▶ Menjalankan perintah “ps --forest”.

```
// system.c
int main()
{
    puts("Running command");

    system("ps --forest");

    puts("Done");
    return 0;
}
```

# Hierarki proses

```
..
  \_ bash
      \_ ./system
          \_ sh
              \_ ps
```



- ▶ Bisa menjalankan rangkaian beberapa perintah sekaligus.
- ▶ Contoh:

```
int main()
{
    system("hostname | rev");
    return 0;
}
```

# Latihan

- ▶ Buat program untuk menjalankan perintah 'ps -A'!
- ▶ Buat program untuk mencetak kalender bulan Desember!

exec()

# Fungsi exec()

```
int execvp(char *file, char *argv[]);  
int execlp(char *file, char *arg, ...);
```

- ▶ Menggantikan proses yang ada dengan proses baru<sup>4</sup>

---

<sup>4</sup>lihat 'man exec'

## Contoh execlp()

- ▶ Parameter perintah ditempatkan pada *list* argumen.
- ▶ Menjalankan perintah "ps --forest":

```
// exec.c
int main()
{
    puts("Running command");

    execlp("ps", "ps", "--forest", NULL);

    puts("Done");
    return 0;
}
```

## Contoh execvp()

- ▶ Parameter perintah disimpan pada variabel *string array*.
- ▶ Menjalankan perintah “ps --forest”:

```
// exec.c
int main()
{
    puts("Running command");

    char *args[] = {"ps", "--forest", NULL};
    execvp(args[0], args);

    puts("Done");
    return 0;
}
```

# Hierarki proses

```
..  
  \_ bash  
      \_ ./exec
```

Setelah pemanggilan fungsi exec, proses lama akan ditimpa.

```
..  
  \_ bash  
      \_ ps --forest
```

# Latihan

- ▶ Buat program untuk menjalankan perintah 'ps -A'!
- ▶ Buat program untuk mencetak kalender bulan Juni!



## system() vs exec()

- ▶ Fungsi `system()` lebih mudah digunakan
  - ▶ namun tidak efisien dalam penggunaan memori dan waktu
  - ▶ karena harus membuat dua proses baru untuk tiap perintah
- ▶ Fungsi `exec()` lebih efisien
  - ▶ langsung menimpa proses yang sudah ada
  - ▶ dipakai oleh *shell* untuk membuat proses baru