Sistem Operasi (INF-201)

Pertemuan 1: Pengantar SO, Dual-Mode Operation, & System Calls

Tim Dosen Sistem Operasi

Program Studi Informatika – Universitas Bengkulu

Semester Ganjil 2026/2027

Capaian Pembelajaran Pertemuan (Sub-CPMK 1)

- Mahasiswa mampu menjelaskan definisi, peran, dan tujuan Sistem Operasi.
- Mahasiswa mampu membedakan mode operasi hardware (*User Mode* vs *Kernel Mode*).
- Mahasiswa memahami mekanisme **Limited Direct Execution (LDE)** & Hardware Traps (OSTEP Ch. 6).
- Mahasiswa mampu menganalisis silsilah UNIX (AT&T $\rightarrow$ BSD vs GNU/Linux) dan lisensi (GPL vs BSD).
- Mahasiswa mampu melacak alur eksekusi *System Calls* menggunakan `strace`.

Apa itu Sistem Operasi?

Definisi Sistem Operasi (OSTEP Ch. 2)

Program yang bertindak sebagai perantara antara pengguna komputer (*user*) dan perangkat keras komputer (*hardware*), menyediakan abstraksi *Virtualization*, *Concurrency*, dan *Persistence*.

Tujuan Utama OS:

1. Menjalankan program pengguna.
2. Kemudahan penggunaan (*User-convenience*).
3. Efisiensi & Proteksi *hardware*.

Dua Peran Utama:

- **Resource Allocator**: Mengelola CPU, RAM, I/O secara adil & efisien.
- **Control Program**: Mencegah error & penggunaan tak sah.

Proteksi Perangkat Keras: Dual-Mode Operation

- Untuk mencegah program pengguna merusak OS, hardware menyediakan minimal 2 mode eksekusi:

User Mode (Mode Bit = 1)	**Kernel Mode (Mode Bit = 0)**
Eksekusi kode aplikasi biasa	Eksekusi kode kernel/OS
Akses memori terbatas (isolated)	Akses penuh ke hardware & memori
Instruksi biasa saja	Instruksi terproteksi (*Privileged*)

Mekanisme Alih Mode (OSTEP Ch. 6)

Perpindahan dari User Mode ke Kernel Mode terjadi secara aman via **Hardware Interrupt**, **Trap/Exception**, atau **System Call**.

Mekanisme LDE (Limited Direct Execution) – OSTEP Ch. 6

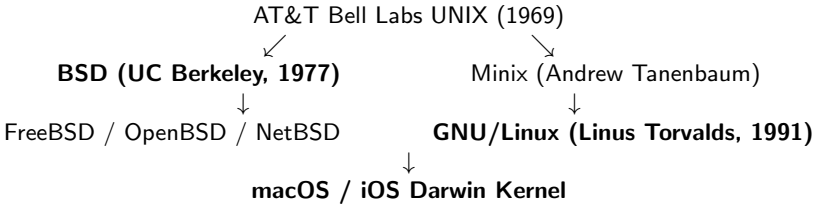
Konsep Limited Direct Execution

OS menjalankan program langsung di CPU (*Direct Execution*) untuk performa tinggi, namun membatasinya (*Limited*) agar OS tetap memegang kendali kontrol CPU.

Dua Fase Utama LDE:

1. **Saat Booting (Kernel Mode)**: Kernel menginisialisasi **Trap Table** di RAM dan mendaftarkan alamat *Trap Handler* ke CPU.
2. **Saat Running (User Mode)**: Ketika aplikasi memanggil system call, CPU menyimpan register ke *Kernel Stack*, mengganti Mode Bit ke 0, dan meloncat ke *Trap Handler*.

Silsilah UNIX: Perkembangan BSD vs Linux



Lisensi GNU GPL (Linux): Copyleft; setiap modifikasi kode kernel wajib dibuka kembali ke publik.

Lisensi BSD (FreeBSD): Permisif; kode bebas dipakai di produk komersial tertutup (misal: macOS, PlayStation OS).

System Calls: Antarmuka Aplikasi ke Kernel

System Call

Layanan resmi yang disediakan oleh kernel OS untuk program aplikasi melalui antarmuka standar (POSIX API).

Alur Eksekusi System Call (`read()` / `write()`):

1. Aplikasi memanggil fungsi pustaka C (misal `printf()`).
2. Pustaka C menyiapkan argumen & mengeksekusi instruksi trap (`syscall` / `int 0x80`).
3. CPU beralih ke **Kernel Mode (Mode Bit 0)**.
4. Kernel mengeksekusi handler sesuai Syscall Table Number.
5. Kernel mengembalikan hasil ke User Space & CPU kembali ke **User Mode**.

Praktikum & Melacak System Calls (strace)

Perintah CLI Linux (*TLCL Ch. 10*) untuk melacak *System Calls* yang dipanggil oleh aplikasi:

strace (System Trace)

```
$ strace -c ls
```

Membuat rangkuman statistik jumlah dan durasi *system calls* saat menjalankan `ls`.

Contoh output strace:

```
openat(AT_FDCWD, "file.txt", O_RDONLY) = 3
read(3, "Hello World\n", 512)          = 12
write(1, "Hello World\n", 12)          = 12
close(3)                               = 0
```

Tugas Membaca Mingguan (Reading Assignment)

Persiapan Pertemuan 2: Arsitektur Kernel & Booting

Sebelum mengikuti kuliah minggu depan, mahasiswa wajib membaca:

- **[OSTEP]** Chapter 4: *Abstraction: The Process* (hal. 27–38)
- **[xv6 RISC-V]** Chapter 2: *Operating System Organization* (hal. 17–28)
- **[TLCL]** Chapter 24: *Writing Your First Script* (hal. 340–352)

Pertanyaan Pemicu Diskusi Mandiri

Mengapa Microkernel (seperti Mach/seL4) lebih aman namun cenderung lebih lambat dibandingkan Monolithic Kernel (seperti Linux)?