

PERTEMUAN 1

PENGANTAR SISTEM OPERASI & ARSITEKTUR KOMPUTER

Mata Kuliah	Sistem Operasi (SI-204) — 3 SKS (2 Teori, 1 Praktikum)
Pertemuan	1 dari 16
Sub-CPMK	Sub-CPMK-1: Mampu menjelaskan peran, fungsi, dan layanan sistem operasi serta posisinya dalam arsitektur SI organisasi (C2).
CPMK Terkait	CPMK-1 (didukung praktikum untuk CPMK-4)
Indikator	• Menjelaskan fungsi OS sebagai pengelola sumber daya • Membedakan kernel mode dan user mode • Mengidentifikasi OS pada tiap tier infrastruktur SI
Bentuk & Waktu	Kuliah interaktif & diskusi (TM 2×50'), Praktikum (1×170'), Penugasan Terstruktur (2×60'), Belajar Mandiri (2×60')
Penilaian	Kuis singkat dan Laporan Praktikum 1 (bobot 2% dari nilai akhir, bagian komponen Partisipasi dan Latihan/Praktikum)
Referensi	[1] Bab 1; [2] Bab 1; [5]

A. TUJUAN PEMBELAJARAN

Setelah mengikuti pertemuan ini, mahasiswa diharapkan mampu:

1. Menjelaskan definisi sistem operasi serta membedakan sistem operasi, kernel, program sistem, dan program aplikasi.
2. Menjelaskan tiga peran utama sistem operasi (penyedia abstraksi, pengalokasi sumber daya, dan program pengawas) beserta contohnya.
3. Menjelaskan komponen dasar organisasi sistem komputer yang dibutuhkan sistem operasi: CPU, memori, hierarki penyimpanan, dan interupsi.
4. Membedakan *user mode* dan *kernel mode* serta menjelaskan mengapa pemisahan ini penting bagi keamanan dan stabilitas sistem.
5. Mengidentifikasi jenis-jenis sistem operasi dan sistem operasi yang digunakan pada tiap tier infrastruktur sistem informasi.
6. (Praktikum) Memasang Ubuntu Server pada mesin virtual dan menjalankan perintah dasar *shell* Linux.

B. APA ITU SISTEM OPERASI?

B.1 Ilustrasi Awal

Analogi: Pengelola Gedung Perkantoran

Bayangkan sebuah gedung perkantoran dengan puluhan penyewa. Setiap penyewa ingin memakai lift, listrik, ruang rapat, dan tempat parkir. Jika setiap penyewa boleh mengatur sendiri mesin lift dan panel listrik, gedung akan kacau dan berbahaya.

Karena itu gedung memiliki **pengelola gedung**. Pengelola (1) menyediakan layanan yang mudah dipakai, sehingga penyewa cukup menekan tombol lift tanpa perlu tahu cara kerja motornya; (2) membagi fasilitas secara adil, misalnya jadwal ruang rapat; dan (3) menjaga keamanan, misalnya penyewa tidak boleh masuk ke ruang panel listrik.

Dalam komputer, **penyewa = program aplikasi**, **fasilitas gedung = perangkat keras**, dan **pengelola gedung = sistem operasi**.

B.2 Definisi Sistem Operasi

Sistem operasi (*Operating System/OS*) adalah perangkat lunak sistem yang menjadi perantara antara pengguna/program aplikasi dan perangkat keras komputer. OS mengelola sumber daya perangkat keras (CPU, memori, penyimpanan, dan perangkat I/O) serta menyediakan layanan agar program dapat berjalan dengan **mudah, efisien, dan aman**.

Perlu diketahui bahwa tidak ada satu definisi OS yang disepakati secara universal. Definisi yang paling sering dipakai dalam buku teks adalah: OS terdiri atas **kernel** (program yang selalu berjalan selama komputer menyala), ditambah **program sistem** dan **middleware** yang menyertainya [1].

B.3 Membedakan Kernel, Program Sistem, Middleware, dan Aplikasi

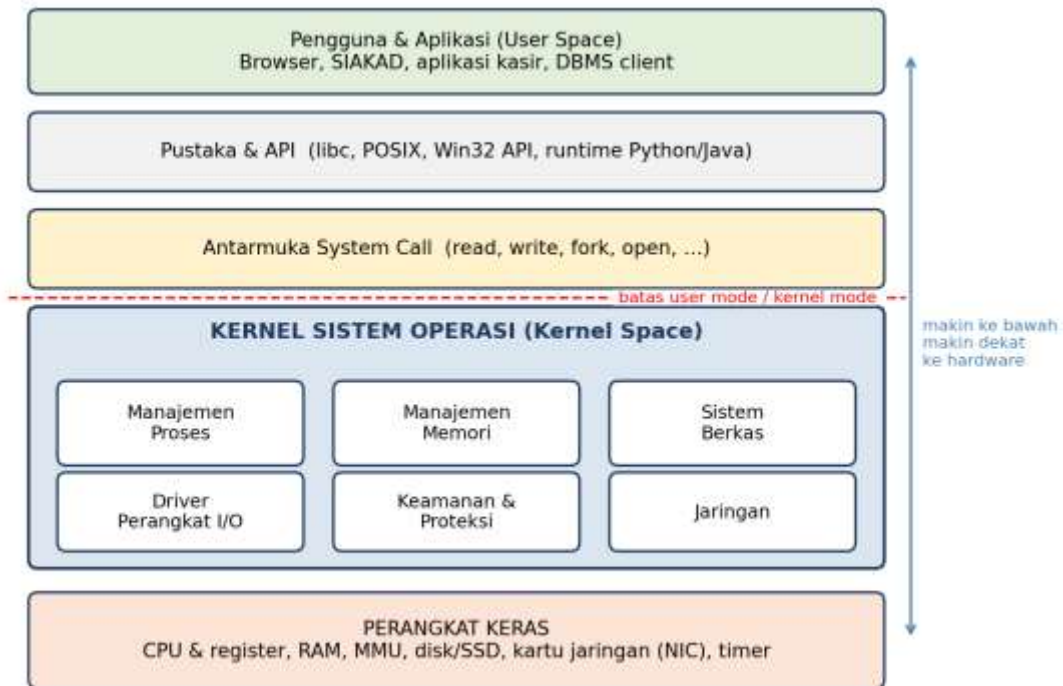
Komponen	Penjelasan	Contoh
Kernel	Inti OS yang selalu berada di memori dan berjalan dalam <i>kernel mode</i> . Kernel mengatur CPU, memori, perangkat, dan keamanan.	Kernel Linux, kernel Windows NT, XNU (macOS/iOS)
Program sistem	Utilitas bawaan OS untuk mengelola sistem. Berjalan di <i>user mode</i> seperti aplikasi biasa, tetapi disediakan oleh OS.	<code>bash</code> , <code>ls</code> , <code>systemctl</code> , Task Manager, File Explorer
Middleware	Kumpulan <i>framework</i> dan layanan tambahan yang memudahkan pengembang membuat aplikasi.	Framework aplikasi Android, layanan grafis dan multimedia
Program aplikasi	Program yang dibuat untuk memenuhi kebutuhan pengguna akhir.	Browser, SIAKAD, aplikasi kasir, Excel, DBMS

Tahukah Anda?

Android dan Ubuntu sama-sama memakai **kernel Linux**, tetapi keduanya adalah sistem operasi yang berbeda karena program sistem, middleware, dan antarmukanya berbeda. Contoh ini menunjukkan bahwa **kernel ≠ sistem operasi secara keseluruhan**.

B.4 Posisi Sistem Operasi dalam Sistem Komputer

Sistem komputer dapat dipandang sebagai tumpukan lapisan (Gambar 1.1). Aplikasi berada di lapisan paling atas dan **tidak menyentuh perangkat keras secara langsung**. Setiap kali aplikasi membutuhkan sesuatu dari perangkat keras, seperti membaca file atau mengirim data lewat jaringan, aplikasi meminta bantuan kernel melalui **system call**.



Gambar 1.1 Posisi sistem operasi di antara aplikasi dan perangkat keras

Contoh dalam SI: ketika dosen menyimpan nilai di SIAKAD, alurnya kurang lebih sebagai berikut: aplikasi web meneruskan data ke DBMS → DBMS memanggil system call `write()` → kernel memeriksa izin dan meneruskan perintah ke *driver* penyimpanan → *driver* memerintahkan SSD menulis data. Pengembang SIAKAD tidak perlu tahu merek SSD atau letak sektor fisiknya. Semua detail itu diurus OS.

B.5 Tiga Peran Utama Sistem Operasi

1) OS sebagai Penyedia Abstraksi (Extended Machine)

Perangkat keras sangat rumit: disk bekerja dengan sektor dan blok, kartu jaringan dengan sinyal dan register kontrol. OS **menyembunyikan kerumitan ini** dan menggantinya dengan konsep yang mudah dipahami, yaitu **abstraksi**. Contohnya:

- **File** menggantikan blok-blok data di disk. Programmer cukup menulis `open("nilai.csv")`.
- **Proses** menggantikan detail register CPU dan alamat memori sebuah program yang berjalan.
- **Socket** menggantikan detail kartu jaringan untuk komunikasi antarkomputer.
- **Antarmuka pengguna** berupa GUI (grafis), CLI (baris perintah), atau layar sentuh.

2) OS sebagai Pengalokasi Sumber Daya (Resource Allocator)

Sumber daya komputer terbatas: waktu CPU, kapasitas RAM, ruang disk, dan *bandwidth* I/O. Ketika banyak program atau pengguna meminta sumber daya secara bersamaan, OS memutuskan **siapa mendapat apa, berapa banyak, dan kapan**. Tujuannya adalah **efisiensi** (sumber daya tidak menganggur) dan **keadilan** (*fairness*, tidak ada program yang terus-menerus dianaktirikan).

Contoh: saat periode KRS online, server akademik menerima ribuan permintaan dalam hitungan menit. OS harus membagi CPU dan memori ke ribuan proses atau *thread* agar semua mahasiswa tetap terlayani.

3) OS sebagai Program Pengawas (Control Program)

OS mengawasi jalannya program agar tidak terjadi kesalahan fatal atau penyalahgunaan. Setiap program diisolasi di ruang memorinya sendiri. Jika satu aplikasi *crash*, OS cukup menghentikan aplikasi tersebut tanpa mengganggu aplikasi lain maupun kernel.

Luruskan Miskonsepsi

OS **tidak dapat mencegah bug di dalam program**, misalnya *memory leak* (memori yang terus dipesan tetapi tidak pernah dilepas). Yang dapat dilakukan OS adalah **membatasi dampaknya**: mengisolasi proses, menghentikan proses yang melanggar aturan, dan mengambil kembali seluruh memori milik proses ketika proses berakhir.

Konteks SI: Dua Sudut Pandang

Sudut pandang pengguna: yang penting adalah OS mudah dipakai dan responsif.

Sudut pandang administrator server SI: yang penting adalah pemanfaatan sumber daya yang optimal, stabilitas (layanan tidak mudah mati), dan keamanan data.

Mata kuliah ini banyak membahas sudut pandang kedua, karena lulusan SI sering berperan dalam perencanaan dan pengelolaan infrastruktur layanan.

C. ORGANISASI SISTEM KOMPUTER: DASAR PERANGKAT KERAS BAGI OS

Sistem operasi sangat bergantung pada fitur perangkat keras. Karena itu, sebelum mempelajari cara kerja OS, kita perlu memahami beberapa komponen dasar komputer.

C.1 Komponen Utama

1. **Central Processing Unit (CPU)**, "otak" yang menjalankan instruksi program. Bagian pentingnya:
 - **Arithmetic Logic Unit (ALU)**: melakukan operasi aritmetika (tambah, kurang) dan logika (AND, OR, perbandingan).
 - **Control Unit (CU)**: mengatur siklus instruksi *fetch* → *decode* → *execute*.
 - **Program Counter (PC)**: register berisi alamat instruksi **berikutnya** yang akan dijalankan.
 - **Instruction Register (IR)**: register berisi instruksi yang **sedang** didekode dan dieksekusi.
 - **Stack Pointer (SP)**: menunjuk ke puncak *stack*, yaitu area memori untuk menyimpan data sementara saat pemanggilan fungsi.
 - **Register status (PSW/FLAGS)**: menyimpan *flag* hasil operasi (Zero, Carry, Overflow) dan **mode bit** yang menandai CPU sedang di *user mode* atau *kernel mode* (lihat Bagian D).
 - **Multicore**: CPU modern berisi beberapa inti (*core*) dalam satu chip. Setiap core dapat menjalankan instruksi secara paralel, sehingga OS harus pandai membagi pekerjaan ke banyak core.
2. **Memori utama (RAM)**: tempat program dan data yang sedang digunakan. RAM bersifat *volatile* (isinya hilang saat listrik padam). CPU hanya dapat mengeksekusi instruksi yang ada di memori utama, sehingga program yang tersimpan di disk harus **dimuat (load)** ke RAM terlebih dahulu.
3. **Pengendali perangkat (device controller) dan device driver**: setiap perangkat (disk, keyboard, kartu jaringan) memiliki *controller* dengan *buffer* kecil. OS berkomunikasi dengan *controller* melalui **driver**, yaitu program khusus yang "menerjemahkan" perintah OS menjadi perintah yang dipahami perangkat.
4. **Bus sistem**: jalur komunikasi antarkomponen, terdiri atas *address bus* (alamat tujuan), *data bus* (data yang dikirim), dan *control bus* (sinyal kendali seperti *read/write* dan permintaan interupsi).

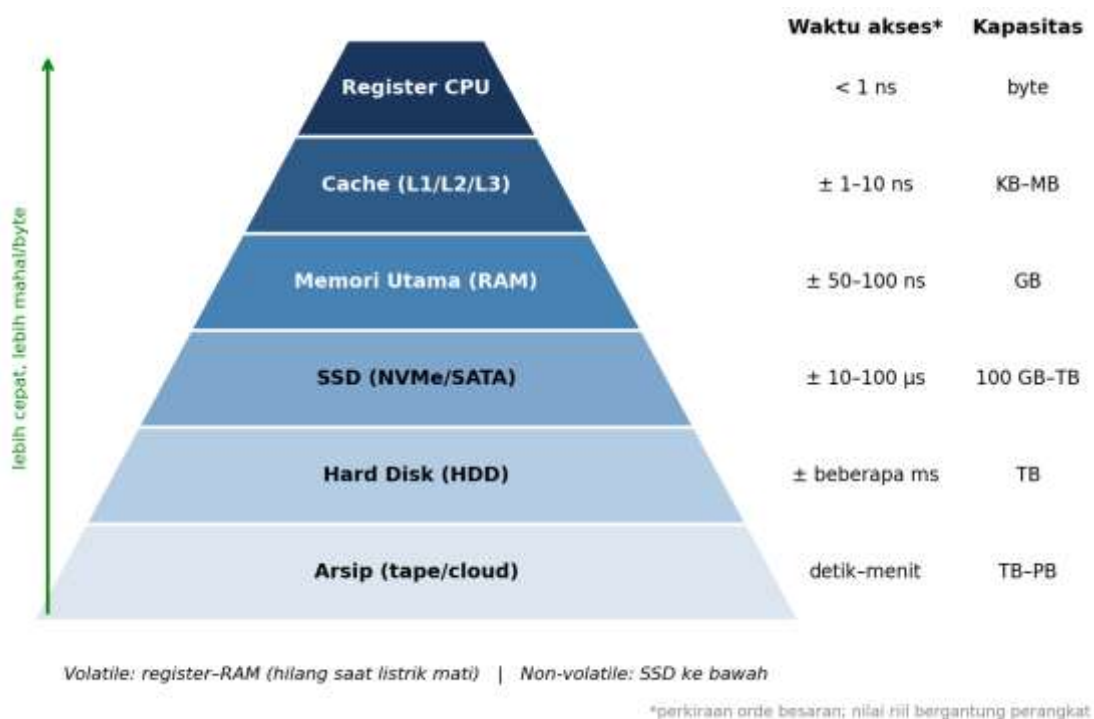
5. **Memory Management Unit (MMU):** perangkat keras yang menerjemahkan alamat logis milik program menjadi alamat fisik di RAM sekaligus memeriksa hak aksesnya. Materi ini dibahas mendalam pada Pertemuan 9.

Siklus Instruksi (Fetch–Decode–Execute)

1. **Fetch:** CPU mengambil instruksi dari memori pada alamat yang ditunjuk PC, lalu menyimpannya di IR. Nilai PC dinaikkan agar menunjuk instruksi berikutnya.
2. **Decode:** Control Unit menerjemahkan instruksi, misalnya "tambahkan dua bilangan".
3. **Execute:** ALU atau komponen lain menjalankan instruksi, lalu hasilnya disimpan ke register atau memori.
4. Sebelum mengulang siklus, CPU memeriksa apakah ada **interupsi** yang menunggu. Jika ada, CPU menanganinya lebih dulu (Bagian C.3).

C.2 Hierarki Penyimpanan dan Caching

Tidak ada media penyimpanan yang sekaligus sangat cepat, sangat besar, murah, dan permanen. Karena itu komputer memakai **hierarki penyimpanan** (Gambar 1.2): semakin ke atas semakin cepat, tetapi kapasitasnya kecil dan harga per byte-nya mahal.



Gambar 1.2 Hierarki penyimpanan (angka adalah perkiraan orde besaran)

Caching adalah teknik menyimpan salinan data yang sering dipakai di media yang lebih cepat. Contohnya, OS Linux menyimpan isi file yang baru dibaca di RAM (*page cache*), sehingga pembacaan berikutnya jauh lebih cepat karena tidak perlu ke disk lagi.

Konteks SI

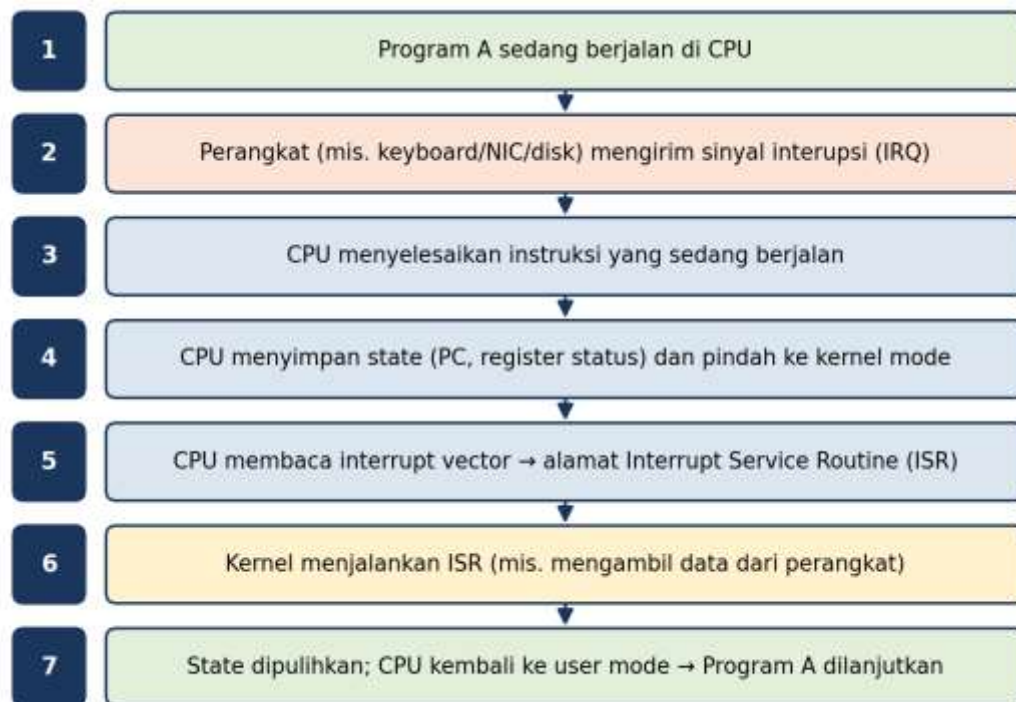
Hierarki ini menjelaskan mengapa **query basis data yang sama sering lebih cepat saat dijalankan kedua kali** (datanya sudah ada di cache RAM), dan mengapa server basis data dianjurkan memiliki **RAM besar dan SSD**. Keputusan pengadaan server yang tepat membutuhkan pemahaman ini.

C.3 Interupsi (Interrupt)

Interupsi adalah sinyal yang memberi tahu CPU bahwa ada peristiwa yang perlu segera ditangani, sehingga CPU menunda sementara pekerjaan yang sedang berjalan. OS modern disebut **interrupt-driven**, artinya sebagian besar aktivitas kernel dipicu oleh interupsi.

Analogi: Bel Pintu vs Mengintip Pintu

Tanpa interupsi, CPU harus terus-menerus memeriksa apakah perangkat sudah siap (*polling*), seperti orang yang bolak-balik mengintip pintu setiap menit untuk melihat apakah kurir sudah datang. Dengan interupsi, CPU cukup bekerja seperti biasa dan baru berhenti ketika "bel pintu" berbunyi. Hasilnya, waktu CPU jauh lebih efisien.



Gambar 1.3 Urutan penanganan interupsi perangkat keras

Pada langkah 5, CPU memakai **interrupt vector**, yaitu tabel berisi alamat rutin penanganan (*Interrupt Service Routine/ISR*) untuk setiap jenis interupsi, sehingga CPU langsung tahu kode mana yang harus dijalankan tanpa perlu mencari.

Jenis-jenis Interupsi

Jenis	Sifat	Penjelasan	Contoh
Interupsi perangkat keras (<i>hardware interrupt</i>)	Asinkron (bisa terjadi kapan saja)	Dipicu perangkat di luar CPU untuk memberi tahu bahwa ada peristiwa.	Tombol keyboard ditekan, paket jaringan tiba, disk selesai membaca, <i>timer</i> berbunyi
Trap (<i>software interrupt</i>)	Sinkron, disengaja	Dipicu oleh instruksi khusus dalam program yang sengaja meminta layanan kernel.	System call <code>read()</code> , <code>write()</code> , <code>fork()</code>
Exception: Fault	Sinkron, tidak disengaja	Kesalahan saat menjalankan instruksi. Jika OS dapat memperbaikinya, instruksi diulang. Jika tidak dapat diperbaiki, proses dihentikan.	<i>Page fault</i> (dapat diperbaiki); akses alamat ilegal atau pembagian dengan nol (proses dihentikan)

Jenis	Sifat	Penjelasan	Contoh
Exception: Abort	Sinkron, fatal	Kesalahan berat yang tidak dapat dipulihkan.	Kesalahan perangkat keras serius (<i>machine check</i>)

Catatan: istilah *trap*, *exception*, dan *software interrupt* dipakai sedikit berbeda antar buku dan antar arsitektur CPU. Yang penting dipahami adalah perbedaan **sumber pemicu** (dari luar CPU atau dari instruksi program) dan **kesengajaannya**.

C.4 Operasi I/O dan Direct Memory Access (DMA)

Ketika program membaca data dari disk, *driver* memberi perintah ke *controller*. *Controller* memindahkan data ke *buffer*-nya, lalu mengirim interupsi ke CPU bahwa pekerjaan selesai. Untuk data berukuran besar, mengirim interupsi per byte akan sangat membebani CPU. Solusinya adalah **DMA**: *controller* memindahkan satu blok data **langsung ke RAM** tanpa melibatkan CPU, lalu hanya mengirim **satu interupsi per blok**. Selama itu, CPU bebas mengerjakan hal lain.

D. DUAL-MODE OPERATION DAN PROTEKSI PERANGKAT KERAS

D.1 Mengapa Perlu Dua Mode?

Jika setiap aplikasi bebas mengakses perangkat keras dan seluruh memori, satu *bug* kecil atau satu program jahat (*malware*) dapat merusak data aplikasi lain, bahkan menghentikan seluruh sistem. Karena itu CPU menyediakan setidaknya **dua mode kerja** yang dibedakan oleh **mode bit** pada register status.

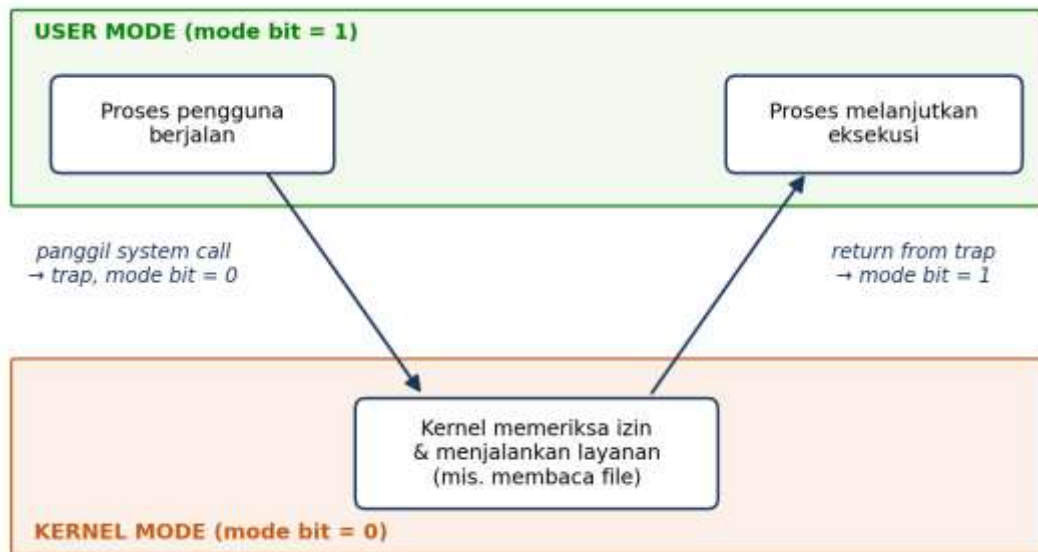
D.2 User Mode vs Kernel Mode

Aspek	User Mode (mode bit = 1)	Kernel Mode (mode bit = 0)
Siapa yang berjalan	Program aplikasi dan program sistem	Kernel sistem operasi (termasuk <i>driver</i>)
Akses memori	Hanya ruang memori milik prosesnya sendiri	Seluruh memori, termasuk ruang kernel
Instruksi	Hanya instruksi biasa (<i>non-privileged</i>)	Semua instruksi, termasuk <i>privileged instruction</i>
Akses perangkat keras	Tidak boleh langsung; harus lewat system call	Boleh mengendalikan perangkat secara langsung
Dampak jika terjadi kesalahan	Biasanya hanya proses tersebut yang dihentikan	Dapat menjatuhkan seluruh sistem (<i>kernel panic</i> di Linux, <i>Blue Screen</i> di Windows)

Catatan arsitektur: konvensi 0 = kernel dan 1 = user berasal dari Silberschatz [1]. Pada prosesor x86, mode diwujudkan sebagai *protection ring* (ring 0 untuk kernel, ring 3 untuk aplikasi). Pada ARM, mode diwujudkan sebagai *exception level* (EL0 untuk aplikasi, EL1 untuk kernel). CPU modern juga memiliki mode tambahan untuk *hypervisor* yang akan dibahas pada Pertemuan 14.

D.3 Perpindahan Mode

Saat komputer dinyalakan (*boot*), CPU berada di kernel mode. Setelah OS siap, OS menjalankan aplikasi di user mode. Setiap kali terjadi **interupsi, trap, atau exception**, CPU otomatis berpindah ke kernel mode. Setelah kernel selesai menanganinya, CPU kembali ke user mode sebelum melanjutkan aplikasi (Gambar 1.4).



Gambar 1.4 Perpindahan user mode ↔ kernel mode saat system call

Contoh: aplikasi memanggil `read()` untuk membaca file → instruksi *trap* dijalankan → mode bit menjadi 0 → kernel memeriksa apakah proses berhak membaca file tersebut → kernel memerintahkan *driver* membaca disk → data disalin ke memori aplikasi → mode bit kembali menjadi 1 → aplikasi melanjutkan eksekusi. Mekanisme system call dibahas lebih rinci pada Pertemuan 2.

D.4 Privileged Instruction

Privileged instruction adalah instruksi yang hanya boleh dijalankan di kernel mode. Jika aplikasi di user mode mencoba menjalankannya, CPU **menolak** dan membangkitkan *trap* ke OS. OS kemudian biasanya menghentikan proses tersebut. Contohnya:

- Mengubah mode bit.
- Mengatur *timer*.
- Menonaktifkan atau mengaktifkan interupsi (pada x86: **CLI/STI**).
- Mengakses port perangkat I/O secara langsung (pada x86: **IN/OUT**).
- Mengubah pengaturan proteksi memori (register *base/limit*, tabel halaman).
- Menghentikan CPU (pada x86: **HLT**).

D.5 Timer: Mencegah Monopoli CPU

Bagaimana jika sebuah aplikasi terjebak dalam *infinite loop* dan tidak pernah mengembalikan CPU kepada OS? Untuk mencegah hal ini, perangkat keras menyediakan **timer**:

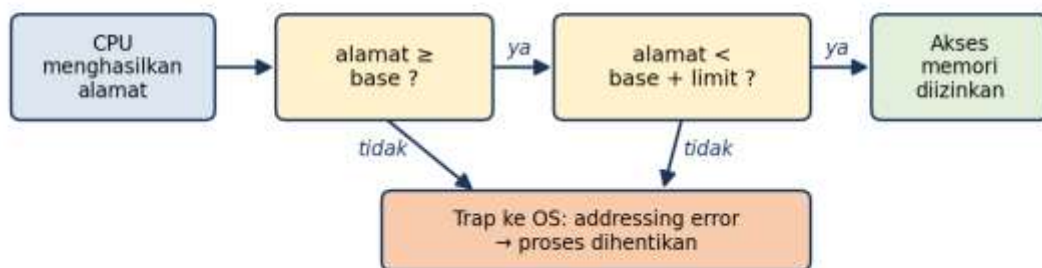
1. Sebelum menyerahkan CPU ke aplikasi, OS mengatur *timer* agar berbunyi setelah jangka waktu tertentu, misalnya beberapa milidetik.
2. Setiap detak *clock*, nilai *counter timer* **dikurangi** satu.
3. Ketika *counter* mencapai 0, *timer* membangkitkan **interupsi**, sehingga CPU dipaksa kembali ke kernel.
4. Kernel dapat memutuskan untuk melanjutkan aplikasi tersebut atau menggantinya dengan proses lain. Mekanisme ini menjadi dasar penjadwalan *preemptive* yang dibahas pada Pertemuan 4.

Karena mengatur *timer* termasuk *privileged instruction*, aplikasi tidak dapat mematikan *timer* untuk menguasai CPU. Ibarat pertandingan olahraga, *timer* adalah **wasit dengan stopwatch** yang memastikan setiap pemain mendapat giliran.

D.6 Proteksi Memori Dasar: Register Base dan Limit

Setiap proses hanya boleh mengakses ruang memorinya sendiri. Cara paling sederhana untuk menegakkan aturan ini adalah dengan dua register: **base** (alamat awal yang sah) dan **limit** (ukuran ruang yang sah). Setiap alamat yang diakses proses di user mode diperiksa oleh perangkat keras (Gambar 1.5).

Contoh: $\text{base} = 300040$, $\text{limit} = 120900 \rightarrow$ alamat sah: 300040 s.d. 420939



Gambar 1.5 Pemeriksaan alamat menggunakan register base dan limit

OS modern memakai mekanisme yang lebih canggih, yaitu *paging* melalui MMU (Pertemuan 9), tetapi prinsipnya sama: **setiap akses memori diperiksa perangkat keras, dan pelanggaran memicu trap ke OS**. Di Linux, proses yang melanggar biasanya berhenti dengan pesan *Segmentation fault*.

E. FUNGSI MANAJEMEN SISTEM OPERASI

OS modern menjalankan beberapa fungsi manajemen secara terpadu. Tabel berikut merangkum tanggung jawab tiap fungsi, contoh perintah untuk mengamatinya di Linux, dan relevansinya bagi pengelolaan SI.

Fungsi	Tanggung Jawab OS	Contoh di Linux	Relevansi bagi SI
Manajemen proses	- Membuat dan menghapus proses - Menjadwalkan proses ke CPU - Menunda dan melanjutkan proses - Menyediakan sinkronisasi dan komunikasi antarproses	<i>ps</i> , <i>top</i> , <i>systemctl</i>	Layanan web dan basis data harus tetap responsif saat beban puncak (Pert. 3–6)
Manajemen memori	- Mencatat bagian memori yang terpakai dan pemakainya - Mengalokasikan dan membebaskan memori - Mengelola memori virtual	<i>free -h</i> , <i>vmstat</i>	Kapasitas RAM server menentukan jumlah pengguna serentak (Pert. 9–10)
Manajemen berkas & penyimpanan	- Menyediakan file dan direktori - Memetakan file ke media fisik - Mengatur hak akses - Mengelola ruang kosong dan penjadwalan disk	<i>ls -l</i> , <i>df -h</i> , <i>mount</i>	Integritas dan ketersediaan data transaksi; backup (Pert. 11–12)

Fungsi	Tanggung Jawab OS	Contoh di Linux	Relevansi bagi SI
Manajemen I/O	- Menyediakan antarmuka <i>driver</i> yang seragam - Buffering, caching, spooling	<code>lsblk, dmesg</code>	Kinerja aplikasi sangat dipengaruhi I/O disk dan jaringan (Pert. 12)
Keamanan & proteksi	- Autentikasi pengguna - Pengaturan hak akses - Isolasi antarproses	<code>id, sudo, chmod</code>	Melindungi data pribadi sesuai UU No. 27 Tahun 2022 tentang Pelindungan Data Pribadi (Pert. 13)

Tiga Teknik Manajemen I/O dalam Bahasa Sederhana

1. **Buffering:** menampung data sementara ketika pengirim dan penerima berbeda kecepatan. Contohnya video *streaming* yang di-*buffer* agar tidak tersendat.
2. **Caching:** menyimpan salinan data yang sering dipakai di media cepat agar akses berikutnya lebih cepat (lihat C.2).
3. **Spooling:** mengantrekan pekerjaan untuk perangkat yang hanya dapat melayani satu permintaan pada satu waktu. Contohnya antrean cetak printer: sepuluh orang dapat menekan "Print" bersamaan, tetapi printer mencetaknya satu per satu.

F. EVOLUSI DAN JENIS-JENIS SISTEM OPERASI

F.1 Evolusi Singkat

Era	Perkembangan Utama
1940–1950-an	Belum ada OS. Operator memasang program secara manual menggunakan kartu plong atau panel kabel.
1950–1960-an	Sistem batch: pekerjaan (<i>job</i>) dikumpulkan dan dijalankan berurutan secara otomatis tanpa interaksi pengguna.
1960–1970-an	Multiprogramming (beberapa program di memori sekaligus) dan time-sharing (banyak pengguna berbagi CPU secara bergantian dengan cepat). UNIX mulai dikembangkan di Bell Labs pada 1969.
1980–1990-an	Era komputer pribadi dan GUI: MS-DOS (1981), Macintosh (1984), Windows, dan kernel Linux (1991).
2000–2010-an	Virtualisasi server dan komputasi awan; OS mobile (iPhone 2007, Android 2008); kontainer populer sejak Docker (2013).
Saat ini	OS dirancang untuk cloud, perangkat IoT, keamanan yang lebih kuat, dan beban kerja AI.

F.2 Jenis-jenis Sistem Operasi

Jenis	Ciri Utama	Contoh	Contoh Penggunaan dalam SI
Batch	Pekerjaan diproses berurutan tanpa interaksi pengguna	Pengelola <i>job</i> pada mainframe (mis. z/OS)	Proses penggajian bulanan, tutup buku malam hari
Time-sharing / multi-user	Banyak pengguna berbagi CPU secara bergantian dengan sangat cepat	UNIX, Linux	Server yang diakses banyak administrator melalui SSH

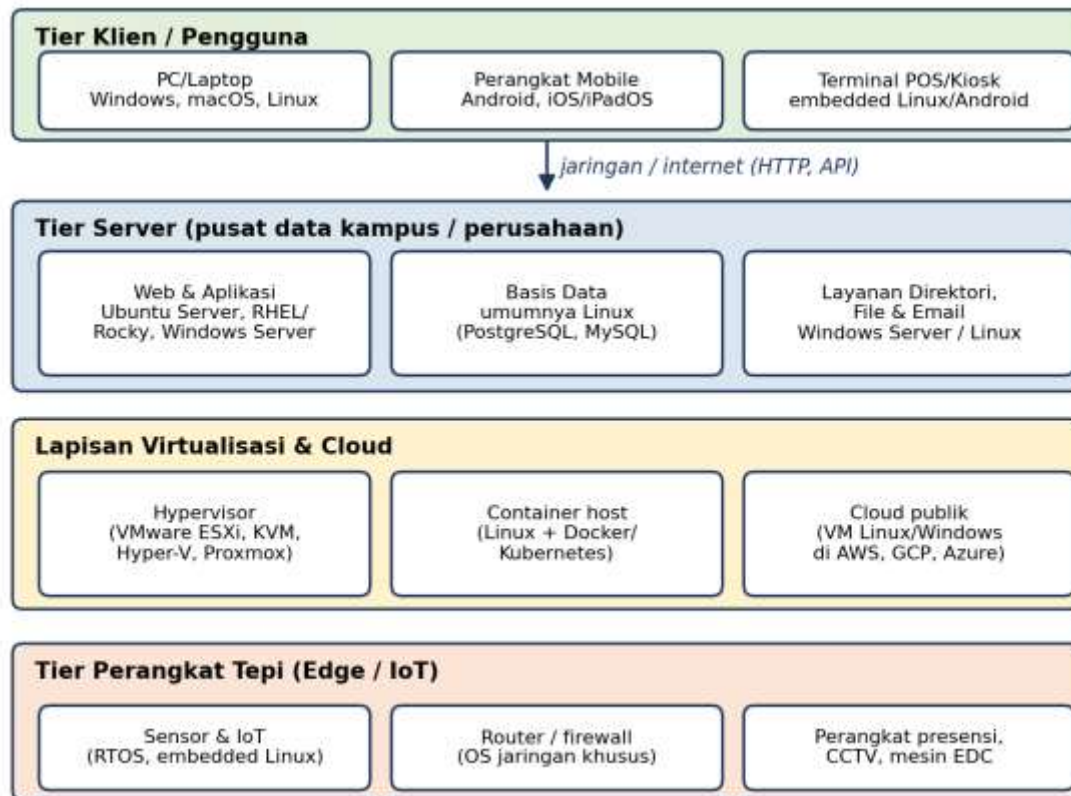
Jenis	Ciri Utama	Contoh	Contoh Penggunaan dalam SI
Desktop	Fokus pada kemudahan dan pengalaman satu pengguna	Windows 11, macOS, Ubuntu Desktop	Komputer staf administrasi dan laboratorium
Server	Stabil, aman, melayani banyak klien, sering tanpa GUI	Ubuntu Server, RHEL, Rocky Linux, Windows Server	Server SIAKAD, web, basis data, email
Mobile	Layar sentuh, hemat daya, aplikasi dalam <i>sandbox</i>	Android, iOS	Aplikasi mobile banking, aplikasi akademik
Real-time (RTOS)	Tugas wajib selesai dalam batas waktu yang ketat	FreeRTOS, Zephyr, VxWorks	Sistem kendali industri, perangkat medis
Embedded	Tertanam pada perangkat khusus dengan sumber daya terbatas	Embedded Linux, Android pada mesin EDC	Mesin EDC, kiosk, perangkat presensi
Terdistribusi / cloud	Sumber daya banyak mesin dikelola sebagai satu kesatuan	Platform cloud; Kubernetes (orkestrator, bukan OS murni)	Layanan SI berbasis cloud yang dapat diskalakan

F.3 Open Source vs Proprietary

Linux bersifat *open source* (lisensi GPL): kode sumbernya terbuka dan dapat dipakai tanpa biaya lisensi. Dukungan teknis tersedia dari komunitas atau secara berbayar dari perusahaan seperti Canonical (Ubuntu) dan Red Hat (RHEL). **Windows Server** bersifat *proprietary*: memerlukan lisensi berbayar, tetapi terintegrasi erat dengan ekosistem Microsoft (Active Directory, Microsoft 365). Dalam praktik, keputusan memilih OS tidak cukup didasarkan pada harga lisensi. Yang perlu dihitung adalah **Total Cost of Ownership (TCO)**: lisensi, dukungan, pelatihan SDM, dan risiko *downtime*.

G. SISTEM OPERASI DALAM ARSITEKTUR SISTEM INFORMASI

Sebuah sistem informasi organisasi berjalan di atas banyak perangkat dengan OS yang berbeda-beda. Gambar 1.6 menunjukkan pemetaan umum OS pada tiap tier infrastruktur SI.



Gambar 1.6 Sistem operasi pada tiap tier infrastruktur sistem informasi

1. **Tier klien:** perangkat yang dipakai langsung oleh pengguna. OS-nya berfokus pada kemudahan penggunaan.
2. **Tier server:** menjalankan aplikasi, basis data, dan layanan inti. Linux dominan untuk server web dan basis data, sedangkan Windows Server banyak dipakai untuk layanan direktori (Active Directory) dan aplikasi berbasis .NET lama.
3. **Lapisan virtualisasi dan cloud:** satu server fisik dapat menjalankan banyak VM atau kontainer. Setiap VM memiliki OS-nya sendiri, sedangkan kontainer berbagi kernel dengan host (Pertemuan 14).
4. **Tier perangkat tepi (edge/IoT):** perangkat khusus dengan OS ringan atau *real-time*.

Kriteria Pemilihan OS untuk Layanan SI

Kriteria	Pertanyaan Kunci
Kompatibilitas aplikasi	Apakah aplikasi, DBMS, dan <i>framework</i> yang dipakai didukung resmi oleh OS tersebut?
Keamanan & masa dukungan	Berapa lama pembaruan keamanan tersedia? Contoh: Ubuntu 26.04 LTS didukung hingga April 2031. OS yang sudah <i>end-of-life</i> tidak lagi menerima <i>patch</i> keamanan.
Biaya (TCO)	Berapa total biaya lisensi, dukungan, perangkat keras, dan pelatihan selama umur sistem?
Ketersediaan SDM	Apakah tim TI menguasai OS tersebut? Adakah tenaga yang mudah direkrut?
Kinerja & skalabilitas	Apakah OS sanggup menangani beban puncak, misalnya saat KRS atau <i>flash sale</i> ?

Kriteria	Pertanyaan Kunci
Ekosistem & dukungan	Seberapa besar komunitas, dokumentasi, dan dukungan vendornya?

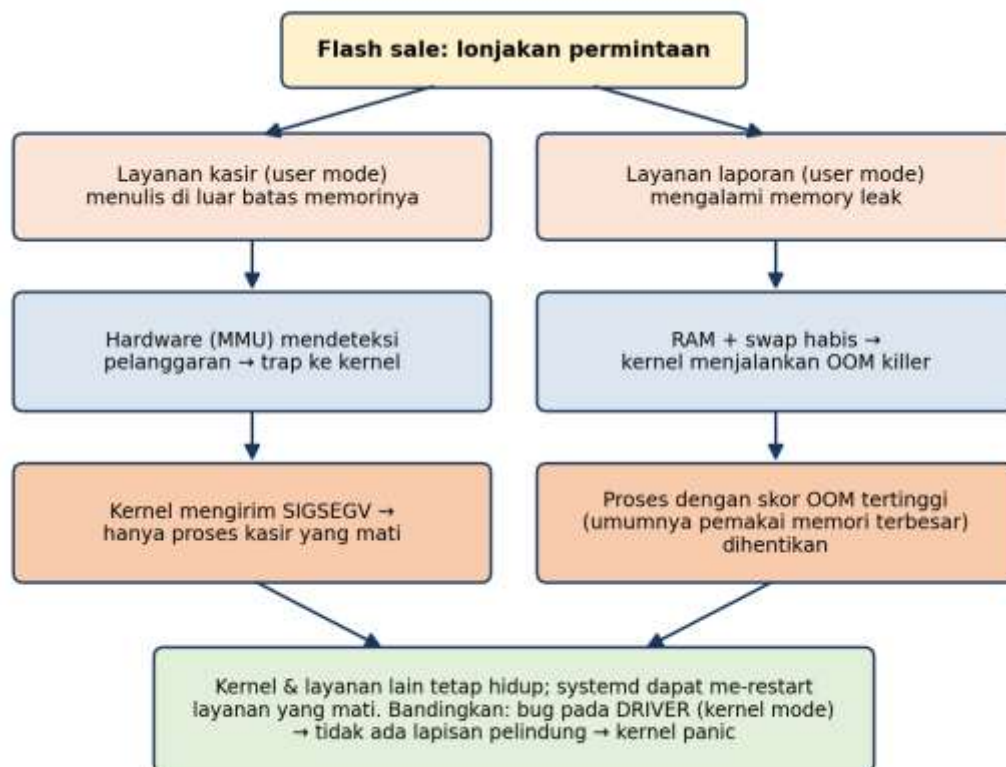
H. STUDI KASUS: "MENGAPA SERVER TIDAK IKUT MATI?"

H.1 Deskripsi Kasus

Toko daring fiktif **BelanjaKita** menjalankan beberapa layanan dalam satu server Linux: layanan web, layanan kasir, layanan laporan penjualan, dan basis data PostgreSQL. Saat program *flash sale* pukul 12.00 terjadi dua insiden:

- Layanan kasir tiba-tiba berhenti. Log mencatat pesan *Segmentation fault*. Investigasi menunjukkan ada *bug* yang membuat program menulis data di luar batas memori yang dialokasikan untuknya.
- Sekitar 20 menit kemudian, layanan laporan yang mengalami *memory leak* membuat RAM dan *swap* penuh. Log kernel mencatat pesan *Out of memory: Killed process*
- Meskipun terjadi dua insiden, kernel, layanan web, dan basis data tetap berjalan. Tim TI me-*restart* layanan kasir dalam dua menit.

Manajer bertanya kepada tim TI: "*Mengapa server tidak mati total? Dan bisakah aplikasi kasir menulis log langsung ke pengendali disk agar lebih cepat?*"



Gambar 1.7 Alur kejadian pada studi kasus BelanjaKita

H.2 Pertanyaan Diskusi

- Mekanisme OS dan perangkat keras apa yang membuat hanya layanan kasir yang berhenti?
- Apakah OS seharusnya dapat mencegah *memory leak* pada layanan laporan? Jelaskan.

3. Mengapa aplikasi kasir tidak boleh menulis langsung ke pengendali disk?
4. Dalam kondisi apa kejadian serupa dapat berujung pada *kernel panic*?

H.3 Pembahasan

1. **Isolasi memori dan dual-mode.** Layanan kasir berjalan di user mode. Ketika mencoba menulis ke alamat di luar ruang memorinya, MMU mendeteksi pelanggaran dan membangkitkan *trap* (exception) ke kernel. Kernel mengirim sinyal **SIGSEGV** ke proses kasir sehingga **hanya proses itu** yang berhenti. Proses lain memiliki ruang memori terpisah sehingga tidak terdampak.
2. **OS tidak mencegah bug, tetapi membatasi dampaknya.** Dari sudut pandang OS, *memory leak* hanyalah proses yang terus meminta memori secara sah. Ketika RAM dan *swap* habis, kernel Linux menjalankan **OOM killer** yang memilih proses dengan skor OOM tertinggi (umumnya pemakai memori terbesar) untuk dihentikan agar sistem tidak macet total. **Risikonya**, OOM killer dapat saja menghentikan proses penting seperti basis data. Karena itu, pemakaian memori tiap layanan perlu dibatasi (cgroups, Pertemuan 14) dan dipantau.
3. **Akses perangkat harus lewat kernel.** Instruksi akses I/O langsung adalah *privileged instruction*, sehingga CPU akan menolaknya di user mode. Cara yang benar adalah memanggil system call **write()**. Kernel lalu memeriksa hak akses, memakai *buffer* dan *cache*, serta mengatur urutan penulisan agar data tidak rusak ketika banyak proses menulis bersamaan. Menulis langsung ke *controller* justru berisiko merusak sistem berkas.
4. **Kernel panic terjadi jika kesalahan ada di kernel mode.** Jika *bug* serupa terdapat pada *driver* atau modul kernel, tidak ada lapisan pengawas di atas kernel yang dapat menghentikannya. Akibatnya seluruh sistem berhenti (*kernel panic*). Itulah sebabnya *driver* dan modul kernel harus berasal dari sumber tepercaya dan diuji dengan ketat.

Pelajaran bagi Pengelola SI

- Pisahkan layanan dan batasi sumber daya per layanan.
- Gunakan *service manager* yang dapat me-*restart* layanan secara otomatis (mis. **Restart=on-failure** pada systemd).
- Pantau pemakaian memori dan log kernel secara rutin, bukan hanya ketika terjadi masalah.
- Hanya pasang *driver/modul* kernel dari sumber resmi.

I. RANGKUMAN

- Sistem operasi adalah perantara antara aplikasi dan perangkat keras. Intinya disebut **kernel**.
- Tiga peran utama OS: **penyedia abstraksi**, **pengalokasi sumber daya**, dan **program pengawas**.
- OS bergantung pada fitur perangkat keras: CPU dan registernya, hierarki penyimpanan, interupsi, DMA, MMU, dan *timer*.
- **Interupsi** membuat CPU efisien karena tidak perlu terus memeriksa perangkat. **Trap** dipakai program untuk meminta layanan kernel (system call).
- **Dual-mode** (user/kernel) dan **privileged instruction** memastikan aplikasi tidak dapat merusak sistem. **Timer** mencegah monopoli CPU. Proteksi memori mencegah proses mengganggu ruang memori proses lain.
- Fungsi manajemen OS meliputi proses, memori, berkas/penyimpanan, I/O, serta keamanan dan proteksi.
- Organisasi memakai beragam OS di tiap tier SI. Pemilihan OS harus mempertimbangkan kompatibilitas, masa dukungan, TCO, SDM, dan kinerja.

J. GLOSARIUM

Istilah	Arti Singkat
Kernel	Inti sistem operasi yang berjalan di kernel mode dan mengelola sumber daya
System call	Cara program meminta layanan kepada kernel, misalnya membaca file
Interupsi	Sinyal ke CPU bahwa ada peristiwa yang perlu segera ditangani
ISR	<i>Interrupt Service Routine</i> , rutin kernel yang menangani interupsi tertentu
Trap	Interupsi yang dipicu secara sengaja oleh program, misalnya untuk system call
Exception	Interupsi akibat kesalahan saat eksekusi instruksi, misalnya <i>page fault</i>
Mode bit	Bit pada register status yang menandai user mode (1) atau kernel mode (0)
Privileged instruction	Instruksi yang hanya boleh dijalankan dalam kernel mode
Timer	Perangkat keras yang membangkitkan interupsi berkala agar OS mendapatkan kembali kendali CPU
DMA	<i>Direct Memory Access</i> , pemindahan blok data antara perangkat dan RAM tanpa melibatkan CPU
Caching	Menyimpan salinan data yang sering dipakai di media yang lebih cepat
Spooling	Mengantrekan pekerjaan untuk perangkat yang hanya dapat melayani satu permintaan pada satu waktu
Kernel panic	Kondisi ketika kernel Linux mendeteksi kesalahan fatal dan menghentikan seluruh sistem
OOM killer	Mekanisme kernel Linux yang menghentikan proses ketika memori habis
LTS	<i>Long Term Support</i> , versi OS dengan masa dukungan pembaruan yang panjang

K. KUIS SINGKAT

Pilih satu jawaban yang paling tepat.

1. Bagian OS yang selalu berada di memori dan berjalan dalam kernel mode disebut ...
a. shell b. kernel c. middleware d. program aplikasi
2. Peran OS dalam membagi waktu CPU dan kapasitas RAM secara efisien dan adil disebut ...
a. extended machine b. resource allocator c. control program d. system program
3. Register yang menyimpan alamat instruksi berikutnya adalah ...
a. Instruction Register b. Stack Pointer c. Program Counter d. Register status
4. Urutan media penyimpanan dari yang paling cepat adalah ...
a. RAM – cache – SSD b. register – cache – RAM – SSD c. cache – register – HDD – RAM
d. SSD – RAM – cache
5. Kartu jaringan menerima paket data lalu memberi tahu CPU. Peristiwa ini merupakan contoh ...
a. trap b. interupsi perangkat keras c. abort d. system call
6. Program memanggil `read()` untuk membaca file. Mekanisme perpindahan ke kernel yang terjadi adalah ...
a. trap (system call) b. interupsi timer c. DMA d. polling

7. Menurut konvensi Silberschatz, mode bit bernilai 0 menandakan ...
a. user mode b. kernel mode c. mode hibernasi d. mode aman
8. Komponen yang mencegah sebuah program menguasai CPU tanpa batas adalah ...
a. MMU b. cache c. timer d. DMA
9. Jenis OS yang paling tepat untuk mesin kendali industri dengan batas waktu ketat adalah ...
a. OS batch b. RTOS c. OS desktop d. OS time-sharing
10. Aplikasi di user mode mencoba mengakses alamat di luar ruang memorinya. Yang paling mungkin terjadi adalah ...
a. kernel panic b. hanya proses tersebut dihentikan c. semua proses dihentikan d. data tertulis ke memori kernel

L. LATIHAN MANDIRI

1. Jelaskan dengan kata-kata Anda sendiri apa yang terjadi pada CPU dan OS ketika aplikasi memanggil `read()` untuk membaca file dari disk. Kaitkan penjelasan Anda dengan *trap* dan perubahan mode bit.
2. Jelaskan perbedaan interupsi perangkat keras, *trap*, dan *exception*. Berikan satu contoh peristiwa nyata untuk masing-masing.
3. Sebuah komputer tidak memiliki *timer* perangkat keras. Apakah OS pada komputer tersebut masih dapat menerapkan *preemptive multitasking*, yaitu mengambil CPU secara paksa dari program yang sedang berjalan? Jelaskan alasannya secara logis.
4. Sebutkan lima fungsi manajemen OS. Untuk masing-masing, berikan satu contoh perintah atau fitur di Linux yang berkaitan dengan fungsi tersebut.
5. Mengapa *kernel panic* hampir selalu disebabkan oleh kode yang berjalan di kernel mode, bukan oleh aplikasi biasa?

M. PENUGASAN TERSTRUKTUR (PT-1): PEMETAAN OS PADA LAYANAN SI

Bentuk: individu atau berpasangan. **Batas pengumpulan:** sebelum Pertemuan 2, melalui LMS (PDF).

Instruksi:

1. Pilih satu organisasi, misalnya kampus Anda, UMKM, instansi pemerintah, atau tempat kerja.
2. Identifikasi minimal **enam** perangkat atau layanan yang tersebar pada minimal **tiga tier** (klien, server, virtualisasi/cloud, atau edge). Untuk setiap perangkat/layanan, catat nama layanan, jenis perangkat, OS beserta versinya (jika diketahui), dan sumber informasinya.
3. Gunakan sumber informasi yang sah: wawancara dengan staf TI, dokumentasi resmi, atau pengamatan langsung pada perangkat yang Anda gunakan. **Dilarang** melakukan pemindaian jaringan atau sistem milik organisasi tanpa izin tertulis.
4. Gambarkan diagram sederhana pemetaan tersebut (boleh mengadaptasi Gambar 1.6).
5. Tulis analisis maksimal satu halaman: mengapa OS tersebut dipilih pada tiap tier, apakah ada OS yang sudah atau hampir *end-of-life*, satu risiko utama yang Anda temukan, dan rekomendasi Anda.

Aspek Penilaian	Bobot
Kelengkapan identifikasi (≥ 6 layanan, ≥ 3 tier)	30%
Ketepatan identifikasi OS dan tier	20%

Aspek Penilaian	Bobot
Kualitas analisis dan rekomendasi	35%
Kerapian penyajian (tabel, diagram, bahasa)	15%

N. PANDUAN PRAKTIKUM 1: INSTALASI UBUNTU SERVER DAN PERINTAH DASAR SHELL

N.1 Tujuan

1. Membuat mesin virtual (VM) dan memasang Ubuntu Server 26.04 LTS.
2. Menggunakan perintah dasar *shell* Linux untuk navigasi dan pengelolaan file.
3. Mengamati konsep teori pertemuan ini (kernel, CPU, cache, memori, interupsi, proteksi) pada sistem nyata.

N.2 Kebutuhan

Komputer host	RAM minimal 8 GB, ruang disk kosong ≥ 30 GB, fitur virtualisasi (Intel VT-x/AMD-V) aktif di BIOS/UEFI
Perangkat lunak	Oracle VirtualBox 7.x (atau VMware Workstation). Pengguna Mac Apple Silicon: gunakan UTM/VMware Fusion dengan ISO Ubuntu Server versi ARM64
Berkas ISO	Ubuntu Server 26.04 LTS (amd64) dari situs resmi ubuntu.com
Spesifikasi VM	2 vCPU, RAM 2 GB (kebutuhan minimal resmi 1,5 GB), disk virtual 25 GB (<i>dynamically allocated</i>), adapter jaringan NAT

N.3 Langkah 1 — Membuat Mesin Virtual

1. Buka VirtualBox, lalu klik **New**.
2. Isi *Name* dengan **srv-os-NIM** (ganti NIM dengan NIM Anda). Pilih berkas ISO Ubuntu Server. Centang **Skip Unattended Installation** agar Anda dapat mengikuti proses instalasi secara manual.
3. Atur RAM 2048 MB dan 2 CPU, lalu buat disk virtual 25 GB.
4. Klik **Finish**, pilih VM tersebut, lalu klik **Start**.

N.4 Langkah 2 — Instalasi Ubuntu Server

Tampilan installer dapat sedikit berbeda antarversi. Ikuti inti langkahnya.

1. Pilih bahasa **English** dan tata letak keyboard yang sesuai.
2. Pilih tipe instalasi **Ubuntu Server** (bukan *minimized*).
3. Pada bagian jaringan, biarkan DHCP berjalan otomatis dan catat alamat IP yang diperoleh.
4. Lewati pengaturan *proxy*, lalu gunakan *mirror* bawaan.
5. Pada bagian penyimpanan, pilih **Use an entire disk** (pengaturan LVM bawaan boleh dipertahankan), lalu konfirmasi.
6. Isi profil: nama Anda, *server name* **srv-NIM**, *username* berupa nama depan + NIM, dan *password* yang kuat. **Catat password Anda.**
7. Pada tawaran Ubuntu Pro, pilih *Skip for now*.

8. Centang **Install OpenSSH server** (dipakai pada pertemuan berikutnya).
9. Lewati pilihan *featured server snaps*. Tunggu hingga instalasi selesai, lalu pilih **Reboot Now**. Jika diminta, keluarkan ISO dari *drive* virtual.

N.5 Langkah 3 — Login dan Memperbarui Sistem

Login menggunakan *username* dan *password* Anda, lalu jalankan:

```
sudo apt update      # memperbarui daftar paket
sudo apt upgrade -y  # memasang pembaruan
ip a                 # melihat alamat IP VM
```

N.6 Langkah 4 — Perintah Dasar Shell

Perintah	Fungsi	Contoh
<code>pwd</code>	Menampilkan direktori aktif	<code>pwd</code>
<code>ls</code>	Menampilkan isi direktori	<code>ls -la /etc</code>
<code>cd</code>	Berpindah direktori	<code>cd /var/log, cd ~</code>
<code>mkdir</code>	Membuat direktori	<code>mkdir -p praktikum/p1</code>
<code>touch</code>	Membuat file kosong	<code>touch catatan.txt</code>
<code>nano</code>	Menyunting file teks	<code>nano catatan.txt</code>
<code>cat, less</code>	Menampilkan isi file	<code>cat /etc/os-release</code>
<code>cp, mv</code>	Menyalin dan memindahkan/mengganti nama	<code>cp catatan.txt salin.txt</code>
<code>rm</code>	Menghapus file (hati-hati, tanpa <i>recycle bin</i>)	<code>rm salin.txt</code>
<code>man</code>	Membuka manual perintah	<code>man ls</code> (tekan q untuk keluar)
<code>whoami, id</code>	Menampilkan identitas pengguna aktif	<code>id</code>
<code>sudo</code>	Menjalankan perintah dengan hak administrator	<code>sudo apt update</code>
<code>sudo poweroff</code>	Mematikan server dengan aman	<code>sudo poweroff</code>

N.7 Langkah 5 — Mengamati Konsep Teori pada Sistem Nyata

```
uname -r              # versi KERNEL Linux
cat /etc/os-release    # nama dan versi DISTRIBUSI (OS)
lscpu                  # jumlah core dan ukuran cache L1/L2/L3
free -h                # kapasitas RAM dan swap
cat /proc/interrupts   # jumlah interupsi per jenis per CPU
sleep 10; cat /proc/interrupts # bandingkan setelah 10 detik
ps -e | wc -l          # jumlah proses yang sedang berjalan
ps -p 1 -o pid,comm     # proses pertama (PID 1) yang dijalankan kernel
sudo dmesg | less       # pesan kernel saat boot (deteksi perangkat & driver)
cat /etc/shadow         # coba tanpa sudo, amati hasilnya
sudo head -3 /etc/shadow # coba dengan sudo
```

Catatan

Penolakan akses ke `/etc/shadow` (berisi *hash* password) adalah contoh **proteksi hak akses berkas** yang ditegakkan kernel. Ini berbeda dengan *privileged instruction* pada tingkat CPU, tetapi tujuannya sama, yaitu mencegah pengguna biasa menyentuh sumber daya yang bukan haknya. Perintah `sudo` tidak mengubah mode bit CPU. `sudo` menjalankan proses dengan identitas *root*, sedangkan proses tersebut tetap berjalan di user mode.

N.8 Tugas dan Format Laporan Praktikum 1

Setiap tangkapan layar harus memperlihatkan *prompt* yang memuat *username*/NIM Anda. Jawablah pertanyaan berikut dalam laporan:

1. Apa versi kernel dan nama distribusi OS pada VM Anda? Jelaskan perbedaan keduanya dengan mengacu pada Bagian B.3.
2. Berapa jumlah core dan ukuran cache L1, L2, dan L3 menurut `lscpu`? Kaitkan hasilnya dengan hierarki penyimpanan.
3. Dari dua kali pembacaan `/proc/interrupts`, jenis interupsi apa yang bertambah paling banyak dalam 10 detik? Mengapa demikian?
4. Berapa jumlah proses yang berjalan padahal Anda belum membuka aplikasi apa pun? Proses apa yang memiliki PID 1?
5. Mengapa `/etc/shadow` tidak dapat dibaca tanpa `sudo`?

Susunan laporan (PDF): sampul; tujuan; langkah kerja beserta tangkapan layar; jawaban pertanyaan; kesimpulan. Laporan dikumpulkan melalui LMS sebelum Pertemuan 2.

N.9 Pemecahan Masalah Umum

Masalah	Solusi
Pesan VT-x/AMD-V is not available	Aktifkan fitur virtualisasi di BIOS/UEFI. Di Windows, nonaktifkan Hyper-V jika terjadi konflik dengan VirtualBox.
VM tidak mendapatkan alamat IP	Pastikan adapter jaringan diatur ke NAT dan kabel virtual tersambung (<i>Cable Connected</i>).
Password tidak diterima	Periksa tata letak keyboard dan tombol Caps Lock. Password tidak tampil saat diketik, dan itu normal.
Kursor "terkunci" di dalam VM	Tekan <i>Host key</i> (bawaan: Ctrl kanan) untuk melepas kursor.
Instalasi sangat lambat	Tutup aplikasi berat di host dan pastikan RAM VM tidak melebihi separuh RAM host.

O. DAFTAR PUSTAKA

- [1] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating system concepts* (10th ed.). Wiley.
- [2] Tanenbaum, A. S., & Bos, H. (2023). *Modern operating systems* (5th ed.). Pearson.
- [3] Stallings, W. (2018). *Operating systems: Internals and design principles* (9th ed.). Pearson.
- [4] Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2018). *Operating systems: Three easy pieces*. Arpaci-Dusseau Books. <https://pages.cs.wisc.edu/~remzi/OSTEP/>
- [5] Nemeth, E., Snyder, G., Hein, T. R., Whaley, B., & Mackin, D. (2017). *UNIX and Linux system administration handbook* (5th ed.). Addison-Wesley.

[6] Canonical Ltd. (2026). *Ubuntu 26.04 LTS release notes*.
<https://documentation.ubuntu.com/release-notes/26.04/>