Dody Herdiana | Esa Firmansyah
Muhammad Agreindra Helmiawan | Yopi Hidayatul Akbar

SISTEM OPERASI

Teori dan Konsep Dasar

NAFAL
Publishing

SISTEM OPERASI

Teori dan Konsep Dasar

Dody Herdiana | Esa Firmansyah
Muhammad Agreindra Helmiawan | Yopi Hidayatul Akbar

NAFAL
Publishing

SISTEM OPERASI
TEORI DAN KONSEP DASAR

Penulis:

Dody Herdiana
Esa Firmansyah
Muhammad Agreindra Helmiawan
Yopi Hidayatul Akbar

Diterbitkan, dicetak, dan didistribusikan oleh
Nafal Publishing
PT Nafal Global Nusantara
Jl. Utama 1 Metro 34112
Telp: +62823-7716-1512, +62 858-0920-7521
Email: nafalglobalnusantara@gmail.com
Anggota IKAPI No. 017/LPU/2024

Cetakan I, Januari 2025

Perancang sampul: Dicky Gea
Penata letak: Hasanuddin

ISBN: 978-634-7025-52-4

xiv + 152 hlm. ; 15,5x23 cm.

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas terselesaikannya buku ini, "Sistem Operasi: Teori dan Konsep Dasar". Buku ini disusun sebagai referensi komprehensif bagi mahasiswa Teknik Informatika, Ilmu Komputer, dan bidang terkait lainnya, membahas dasar-dasar hingga topik lanjutan dalam sistem operasi.

Materi dalam buku ini mencakup konsep-konsep inti seperti manajemen memori, penjadwalan proses, dan sistem file, serta membahas tren terbaru seperti virtualisasi, cloud computing, dan aplikasi dalam Internet of Things (IoT) dan perangkat mobile. Kami juga menampilkan studi kasus sistem operasi populer seperti Linux, Windows, macOS, Android, dan iOS untuk memberikan gambaran praktis yang mendalam.

Buku ini diharapkan dapat membantu mahasiswa memahami dasar-dasar sistem operasi yang menjadi fondasi penting dalam pengembangan aplikasi dan sistem. Kami juga berharap buku ini dapat menjadi panduan pembelajaran di kelas dan laboratorium, serta berguna bagi penelitian lebih lanjut.

Kami menyadari bahwa buku ini masih memiliki kekurangan, dan dengan senang hati menerima saran dan kritik konstruktif demi perbaikan di masa depan. Terima kasih kepada semua pihak yang telah berkontribusi,

terutama rekan pengajar dan mahasiswa yang memberikan masukan berharga. Semoga buku ini bermanfaat dan membantu dalam memperluas pemahaman pembaca tentang sistem operasi.

Selamat membaca dan semoga sukses dalam belajar!

Penulis

DAFTAR ISI

Bab 3

Bab 4

Bab 5

Bab 8

Bab 9

Bab 10

Bab 11

DAFTAR GAMBAR

Bab 1

PENDAHULUAN SISTEM OPERASI

A. Definisi dan Fungsi Sistem Operasi

Sistem operasi (OS) merupakan komponen fundamental dalam dunia komputasi yang menjadi penghubung antara perangkat keras dan perangkat lunak. Seiring dengan perkembangan teknologi, sistem operasi telah menjadi pusat dari seluruh aktivitas komputasi, baik pada komputer pribadi, server, perangkat mobile, hingga perangkat Internet of Things (IoT). Dalam setiap perangkat yang kita gunakan sehari-hari, sistem operasi berperan penting untuk mengelola sumber daya, menjalankan

aplikasi, dan memberikan antarmuka yang memungkinkan interaksi antara pengguna dan mesin.

Pada masa awal perkembangan komputer, sistem operasi tidak seperti yang kita kenal sekarang. Komputer generasi pertama menggunakan pendekatan batch processing, di mana program dijalankan secara bergantian tanpa interaksi langsung dengan pengguna. Namun, dengan munculnya konsep time-sharing pada tahun 1960-an, sistem operasi berkembang untuk mendukung banyak pengguna secara simultan, memungkinkan lebih banyak fleksibilitas dan efisiensi dalam penggunaan komputer. Konsep ini kemudian diperluas dengan kemampuan multitasking dan multiprocessing, yang menjadi dasar dari sistem operasi modern.

Sistem operasi memiliki beberapa fungsi utama, seperti manajemen proses, manajemen memori, manajemen sistem file, serta manajemen perangkat I/O (Input/Output). Dengan fungsi-fungsi ini, OS bertanggung jawab untuk mengalokasikan sumber daya, menangani permintaan aplikasi, dan menjaga stabilitas serta keamanan sistem. Misalnya, ketika pengguna membuka aplikasi di komputer atau ponsel, sistem operasi yang mengatur bagaimana memori dialokasikan, bagaimana data diakses, dan bagaimana tampilan aplikasi ditampilkan kepada pengguna. Tanpa adanya sistem operasi, perangkat keras komputer tidak akan dapat menjalankan instruksi atau berinteraksi dengan aplikasi.

Seiring berjalannya waktu, sistem operasi tidak hanya berkembang pada komputer pribadi atau mainframe, tetapi juga pada perangkat mobile, server cloud, dan perangkat embedded. Sistem operasi mobile seperti Android dan iOS dirancang untuk mengoptimalkan pengalaman pengguna pada ponsel pintar, dengan fitur keamanan dan manajemen energi yang canggih. Di sisi lain, sistem operasi server seperti Linux dan Windows Server mengutamakan stabilitas, keandalan, dan kemampuan untuk menangani beban kerja yang besar. Sementara itu, perkembangan di bidang virtualisasi dan cloud computing telah memungkinkan satu perangkat keras untuk menjalankan banyak sistem operasi secara

bersamaan, memberikan fleksibilitas dan efisiensi yang belum pernah terjadi sebelumnya.

Selain itu, tantangan dan peluang baru muncul dengan hadirnya Internet of Things (IoT), di mana jutaan perangkat terhubung ke internet dan saling berinteraksi. Sistem operasi untuk perangkat IoT, seperti RIOT OS atau Contiki, dirancang agar efisien dalam penggunaan daya dan mendukung komunikasi jaringan yang cepat dan aman. Hal ini membuka peluang baru bagi pengembangan teknologi di berbagai sektor, mulai dari rumah pintar hingga industri manufaktur.

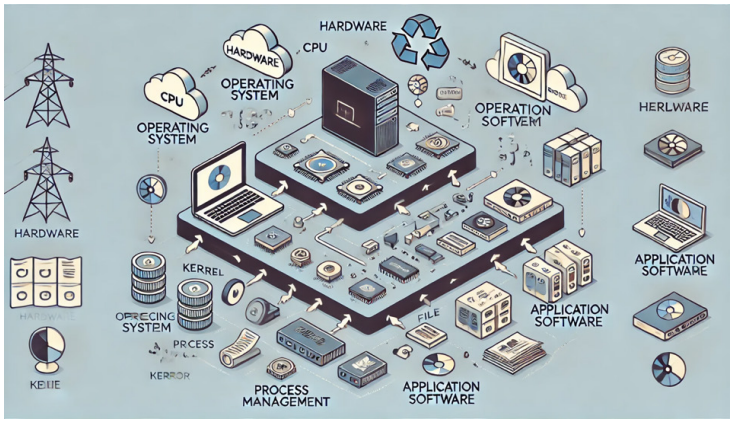
Secara keseluruhan, sistem operasi telah berevolusi menjadi lebih kompleks dan canggih, mengakomodasi kebutuhan yang semakin beragam dari pengguna dan aplikasi. Pada bab-bab berikutnya dalam buku ini, kita akan mempelajari lebih lanjut berbagai komponen sistem operasi, termasuk bagaimana OS mengelola proses, memori, sistem file, dan perangkat I/O. Kita juga akan membahas tren terbaru seperti virtualisasi, cloud computing, serta sistem operasi untuk perangkat embedded dan real-time.

Secara garis besar OS adalah perangkat lunak inti yang berfungsi sebagai penghubung antara pengguna dan perangkat keras komputer. OS mengelola sumber daya perangkat keras dan menyediakan layanan inti untuk perangkat lunak aplikasi, seperti alokasi memori, penjadwalan proses, serta manajemen file dan keamanan.

Fungsi utama Sistem Operasi:

1. **Manajemen Proses**: Mengelola eksekusi proses, mulai dari pembuatan hingga penghentian. Mengelola penjadwalan CPU agar proses mendapatkan waktu yang tepat dan menghindari konflik.
2. **Manajemen Memori**: Memungkinkan alokasi memori secara efisien, mengoptimalkan ruang penyimpanan, dan menyediakan memori virtual untuk menangani program yang lebih besar dari kapasitas RAM fisik.

3. **Manajemen Sistem File**: Mengatur penyimpanan, pengambilan, dan manipulasi data di disk. Memastikan file yang disimpan tetap aman dan dapat diakses dengan mudah.
4. **Manajemen Perangkat Input/Output (I/O)**: Mengontrol perangkat I/O dan memastikan data ditransfer antara perangkat dan memori utama tanpa konflik.
5. **Keamanan dan Proteksi**: Menjamin bahwa hanya pengguna yang memiliki izin yang dapat mengakses sumber daya tertentu dalam sistem, termasuk data sensitif dan program penting.



Gambar 1. Hubungan antara perangkat keras, sistem operasi, dan perangkat lunak aplikasi

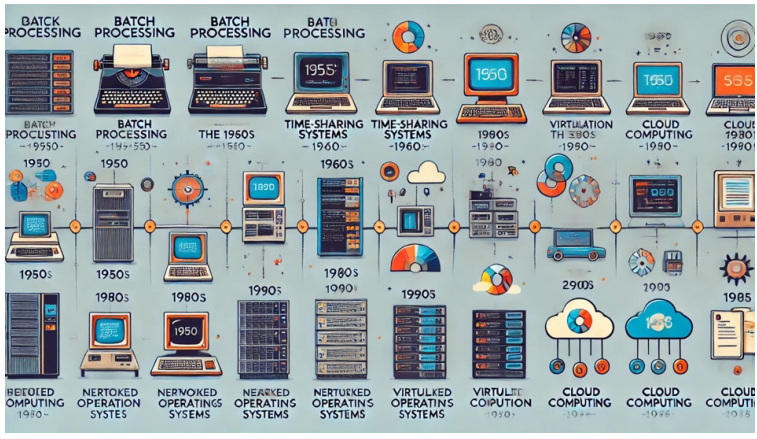
B. Sejarah dan Evolusi Sistem Operasi

Perkembangan OS dapat dibagi menjadi beberapa era utama, yang masing-masing membawa perubahan besar dalam cara sistem operasi berfungsi dan digunakan.

1. **Batch Processing (1950-an)**: Pada awalnya, komputer hanya menjalankan satu pekerjaan (job) dalam satu waktu. Program dikumpulkan dalam batch dan dijalankan satu per satu tanpa interaksi langsung dengan pengguna.
2. **Time-Sharing Systems (1960-an)**: Sistem time-sharing memperkenalkan konsep multi-programming, di mana banyak

pengguna dapat berinteraksi dengan sistem secara langsung dan seolah-olah mereka memiliki komputer sendiri.

3. **Multiprocessing dan Multitasking (1970-an hingga 1980-an)**: Dengan diperkenalkannya prosesor multi-core, OS berkembang untuk mendukung multitasking dan multiprocessing, memungkinkan lebih dari satu proses dijalankan secara paralel.
4. **Graphical User Interface (GUI) (1980-an hingga 1990-an)**: GUI yang pertama kali diperkenalkan oleh Xerox dan kemudian diadopsi oleh Apple dan Microsoft, memungkinkan pengguna untuk berinteraksi dengan komputer secara lebih intuitif melalui ikon dan menu.
5. **Mobile OS dan Cloud Computing (2000-an hingga sekarang)**: Dengan kemajuan perangkat mobile, OS seperti Android dan iOS berkembang pesat. OS juga mengalami evolusi dalam mendukung cloud computing, dengan layanan seperti Google Cloud Platform dan Amazon Web Services.

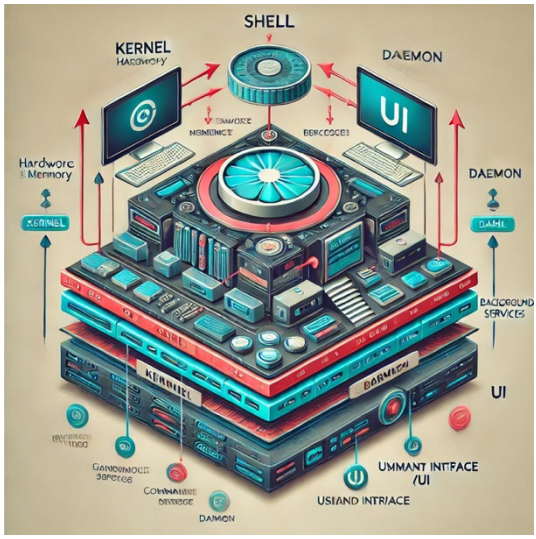


Gambar 2. Sejarah dan Evolusi Sistem Operasi

C. Komponen Utama Sistem Operasi

OS terdiri dari beberapa komponen inti yang bekerja sama untuk mengelola perangkat keras dan menyediakan layanan bagi aplikasi. Berikut penjelasan dari setiap komponen inti sistem operasi:

1. **Kernel**: Komponen inti yang berjalan di mode tertinggi CPU dan berperan dalam manajemen memori, manajemen proses, dan komunikasi antara perangkat keras dan perangkat lunak.
2. **Shell**: Berfungsi sebagai antarmuka antara pengguna dan kernel. Shell dapat berupa **command-line interface (CLI)** seperti bash atau **graphical user interface (GUI)** seperti yang ditemukan di Windows dan macOS.
3. **File System**: Mengatur penyimpanan, pengambilan, dan manipulasi data di media penyimpanan. Contoh sistem file termasuk NTFS pada Windows dan ext4 pada Linux.
4. **Daemons dan Services**: Program latar belakang yang berjalan tanpa interaksi langsung dengan pengguna. Contoh daemon adalah httpd untuk web server di Linux, yang menangani permintaan web dari klien.
5. **User Interface (UI)**: Antarmuka yang memungkinkan pengguna berinteraksi dengan sistem. UI dapat berupa GUI atau CLI.



Gambar 3. Struktur OS, menunjukkan interaksi antara kernel, shell, daemon, dan UI.

D. Tipe Sistem Operasi

Sistem operasi berkembang untuk memenuhi kebutuhan dari berbagai jenis perangkat dan lingkungan. Berikut adalah tipe-tipe OS utama beserta contoh dan fungsinya:

1. **Desktop OS**: Sistem operasi yang dirancang untuk pengguna komputer pribadi, memberikan UI yang interaktif. Contoh: **Windows**, **macOS**, dan **Linux**.
2. **Server OS**: OS yang dioptimalkan untuk server, dengan fokus pada stabilitas, manajemen jaringan, dan kemampuan menangani banyak koneksi. Contoh: **Windows Server**, **Ubuntu Server**, dan **Red Hat Enterprise Linux (RHEL)**.
3. **Mobile OS**: OS yang dioptimalkan untuk perangkat mobile, dengan fokus pada efisiensi daya dan interface sentuh. Contoh: **Android**dan **iOS**.
4. **Embedded OS**: Sistem operasi ringan yang dirancang untuk perangkat dengan sumber daya terbatas seperti router, ATM, dan perangkat IoT. Contoh: **FreeRTOS** dan **VxWorks**.
5. **Real-Time Operating System (RTOS)**: OS yang dirancang untuk aplikasi real-time yang membutuhkan respon deterministik dalam waktu yang singkat. RTOS sering digunakan dalam sistem industri dan perangkat medis.
6. **Distributed OS**: OS yang mengatur beberapa komputer yang saling terhubung untuk bertindak sebagai satu sistem tunggal. Sistem ini banyak digunakan pada sistem berbasis cloud atau pusat data.

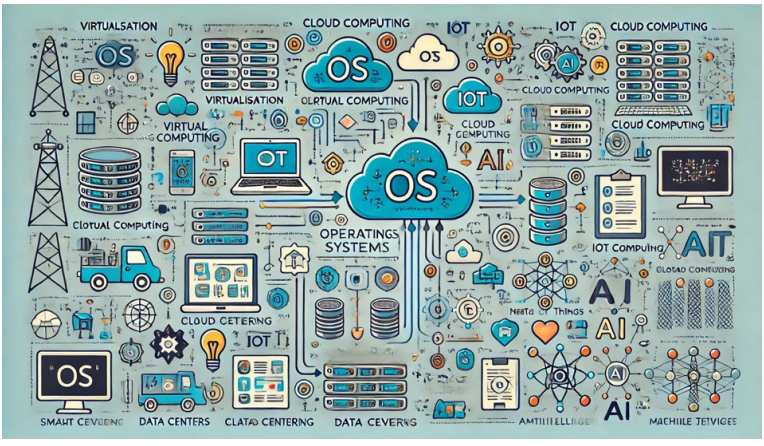
E. Lingkup dan Peran Sistem Operasi dalam Komputasi Modern

Sistem operasi memainkan peran yang sangat penting dalam komputasi modern. Berikut ini adalah beberapa peran utama OS dalam teknologi terkini:

1. **Virtualisasi dan Cloud Computing**: OS mendukung virtualisasi yang memungkinkan beberapa OS berjalan pada satu mesin fisik melalui

hypervisor (misalnya VMware atau KVM), yang menjadi dasar dari infrastruktur cloud computing.

2. **Containerization**: OS juga mendukung container, seperti Docker, yang memungkinkan pengembangan dan penyebaran aplikasi dalam lingkungan yang konsisten dan terisolasi.
3. **Internet of Things (IoT)**: OS untuk perangkat IoT dioptimalkan untuk konsumsi daya rendah dan pengelolaan sumber daya yang efisien. Contoh OS IoT termasuk RIOT OS dan Contiki OS.
4. **Keamanan Cyber**: OS modern memiliki lapisan keamanan yang lebih canggih, seperti fitur sandboxing, kontrol akses berbasis peran (RBAC), enkripsi, dan dukungan untuk autentikasi multifaktor.
5. **Artificial Intelligence (AI)**: OS modern mendukung kemampuan AI, khususnya dalam manajemen sumber daya pada GPU dan TPU yang digunakan untuk komputasi intensif AI.



Gambar 4. Diagram yang menunjukkan aplikasi OS dalam komputasi modern, seperti virtualisasi, cloud, IoT, dan AI

Bab 2

STRUKTUR SISTEM OPERASI

A. Arsitektur Sistem Operasi

Arsitektur sistem operasi merupakan fondasi utama yang menentukan bagaimana sistem operasi mengelola sumber daya dan menjalankan berbagai tugas di perangkat keras. Seperti halnya sebuah bangunan yang dirancang dengan struktur tertentu untuk menopang keseluruhan sistem, arsitektur sistem operasi juga dirancang untuk mengatur alur kerja, komunikasi antar komponen, serta pengelolaan perangkat keras dan

aplikasi. Arsitektur sistem operasi menggambarkan cara kerja internal OS dalam memproses permintaan dari aplikasi, mengelola memori, mengatur akses perangkat, dan mengontrol komunikasi antar proses.

Pada awal perkembangan sistem operasi, arsitektur yang digunakan sangat sederhana. Banyak sistem operasi generasi pertama menggunakan monolithic architecture, di mana semua fungsi sistem operasi diletakkan dalam satu blok besar di dalam kernel. Monolithic kernel mencakup manajemen memori, penjadwalan proses, sistem file, serta manajemen perangkat I/O dalam satu unit yang saling terhubung. Keuntungan dari arsitektur ini adalah kinerja yang tinggi karena semua fungsi dijalankan dalam satu ruang alamat yang sama, namun kelemahannya adalah kompleksitas dan sulitnya memelihara serta memperbarui sistem tanpa mempengaruhi bagian lain.

Seiring perkembangan teknologi, arsitektur sistem operasi mengalami perubahan untuk meningkatkan modularitas, fleksibilitas, dan keamanan. Salah satu pendekatan yang muncul adalah layered architecture, di mana OS dibagi menjadi beberapa lapisan yang masing-masing memiliki fungsi spesifik. Lapisan terendah berhubungan langsung dengan perangkat keras, sementara lapisan tertinggi menyediakan antarmuka bagi aplikasi pengguna. Keuntungan dari pendekatan ini adalah struktur yang lebih terorganisir dan memungkinkan perbaikan pada satu lapisan tanpa harus mengubah seluruh sistem. Namun, layered architecture sering kali memiliki overhead tambahan karena adanya komunikasi antar lapisan.

Selain layered architecture, muncul pula konsep microkernel, yang memisahkan fungsi-fungsi inti sistem operasi (seperti manajemen memori, komunikasi antar proses, dan penjadwalan) ke dalam unit kecil yang terpisah dan berjalan di mode user. Microkernel berfokus hanya pada fungsi dasar dan memindahkan sebagian besar layanan sistem operasi (seperti sistem file dan driver perangkat) ke user space. Arsitektur ini menawarkan keandalan dan keamanan yang lebih baik karena setiap komponen berjalan terpisah, mengurangi risiko kegagalan sistem akibat bug di salah satu modul. Namun, kelemahannya adalah kinerja yang

mungkin lebih rendah karena adanya komunikasi antar komponen yang lebih sering.

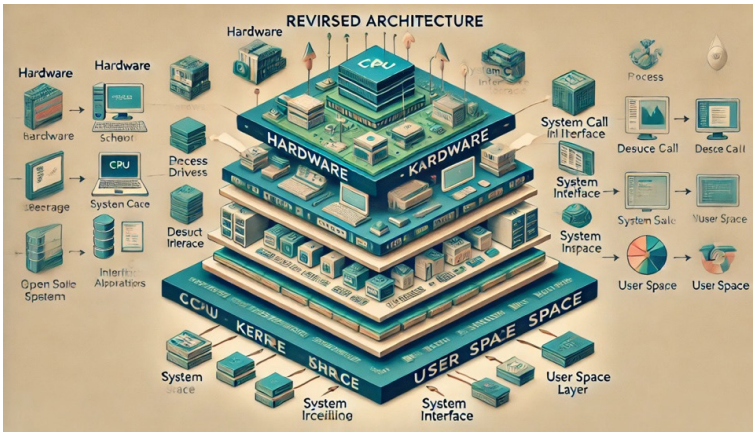
Kemudian muncul modular architecture, yang diadopsi oleh sistem operasi modern seperti Linux. Modular architecture memungkinkan OS untuk menambah atau menghapus modul kernel secara dinamis tanpa harus me-reboot sistem. Setiap modul dapat ditambahkan sesuai kebutuhan, seperti driver perangkat atau protokol jaringan, sehingga memberikan fleksibilitas yang tinggi bagi pengguna dan pengembang. Hal ini memungkinkan sistem untuk beradaptasi dengan berbagai konfigurasi perangkat keras dan kebutuhan aplikasi tanpa mengubah inti kernel.

Pendekatan-pendekatan arsitektur ini menunjukkan bagaimana sistem operasi telah berevolusi untuk memenuhi tuntutan kinerja, stabilitas, dan keamanan yang semakin tinggi. Pilihan arsitektur yang tepat bergantung pada konteks penggunaan sistem operasi tersebut, apakah untuk server yang membutuhkan stabilitas tinggi, perangkat mobile yang membutuhkan efisiensi daya, atau perangkat IoT yang memiliki keterbatasan sumber daya.

Arsitektur sistem operasi menentukan bagaimana komponen OS diatur dan berinteraksi satu sama lain. Berikut adalah penjelasan mendalam tentang beberapa arsitektur yang paling umum:

1. **Monolithic Architecture**
    a. Pada arsitektur monolitik, seluruh komponen OS (seperti manajemen memori, penjadwalan, sistem file, dan komunikasi I/O) terintegrasi dalam satu blok besar yang berjalan dalam satu ruang alamat (kernel space).
    b. Keuntungan: Kinerja tinggi karena semua fungsi OS berjalan dalam ruang kernel dan tidak ada overhead yang besar dalam komunikasi antar modul.
    c. Kerugian: Kode yang sangat besar dan kompleks sehingga rentan terhadap bug; sulit untuk memelihara dan mengimplementasikan pembaruan tanpa memengaruhi sistem secara keseluruhan.

2. **Layered Architecture**
    a. OS dibagi menjadi beberapa lapisan (layer), di mana setiap lapisan hanya berinteraksi dengan lapisan di atas atau di bawahnya. Lapisan paling bawah berinteraksi langsung dengan perangkat keras, sedangkan lapisan atas menyediakan API untuk pengguna.
    b. Keuntungan: Struktur yang lebih teratur dan modular; setiap lapisan dapat diubah tanpa memengaruhi lapisan lainnya.
    c. Kerugian: Overhead dari komunikasi antar lapisan, yang dapat menurunkan kinerja dibandingkan dengan arsitektur monolitik.
3. **Microkernel**
    a. Microkernel memisahkan layanan dasar seperti komunikasi antar proses, manajemen memori, dan manajemen I/O dalam bagian yang lebih kecil yang berjalan di user space, sedangkan hanya fungsi minimal yang berjalan di kernel space.
    b. Keuntungan: OS yang lebih aman dan stabil karena hanya fungsi inti yang berjalan di kernel, dan perubahan dalam komponen tertentu tidak akan memengaruhi keseluruhan OS.
    c. Kerugian: Kompleksitas komunikasi antar komponen yang berjalan di user space menambah overhead, mengakibatkan potensi penurunan kinerja.
4. **Modular Architecture**
    a. OS modular memungkinkan pengembang untuk memuat dan melepas modul secara dinamis. Linux adalah contoh OS modular di mana banyak bagian dapat ditambahkan tanpa harus me-reboot sistem.
    b. Keuntungan: Fleksibilitas tinggi; modul dapat ditambahkan atau dihapus sesuai kebutuhan tanpa mengganggu operasi keseluruhan OS.
    c. Kerugian: Lebih kompleks untuk dikelola karena modul harus kompatibel satu sama lain.



Gambar 5. Arsitektur Sistem Operasi

B. Proses dan Thread

1. Definisi Proses

Proses adalah program dalam eksekusi. Saat program dimuat ke dalam memori dan mulai berjalan, ia menjadi sebuah proses. Setiap proses memiliki alamat memori, variabel, dan instruksi yang spesifik.

2. Siklus Hidup Proses

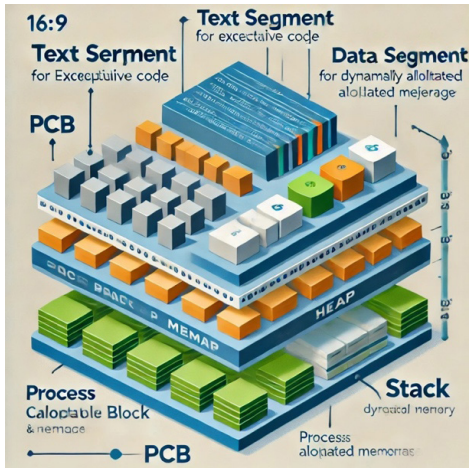
a. **New**: Proses baru dibuat.
b. **Ready**: Proses siap untuk dieksekusi oleh CPU.
c. **Running**: Proses sedang dieksekusi.
d. **Waiting**: Proses menunggu suatu kejadian eksternal seperti input dari perangkat.
e. **Terminated**: Proses telah selesai dijalankan.

Gambar 6. Diagram state dari siklus hidup proses

3. Struktur Proses dalam Memori

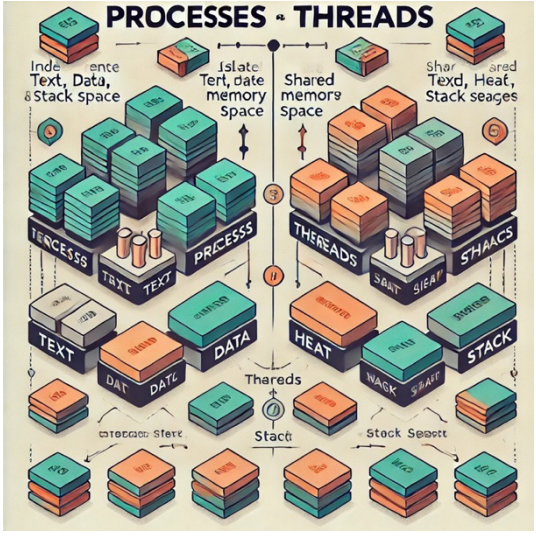
Setiap proses dalam memori memiliki beberapa bagian, yaitu:

a. **Text**: Berisi kode program yang dieksekusi.
b. **Data**: Berisi variabel global yang diinisialisasi.
c. **Heap**: Area untuk alokasi dinamis.
d. **Stack**: Berisi data lokal dari tiap fungsi, termasuk parameter dan alamat return.



Gambar 7. Diagram struktur proses dalam memori

4. **Definisi dan Manfaat Thread**
   a. **Thread** adalah unit terkecil dari eksekusi dalam sebuah proses. Dalam satu proses, dapat terdapat beberapa thread yang berjalan secara paralel, memungkinkan peningkatan kinerja dan penghematan sumber daya.
   b. **Keuntungan Penggunaan Thread**:
      1) Meningkatkan kinerja dengan menjalankan beberapa operasi secara bersamaan.
      2) Menghemat sumber daya karena thread berbagi memori dalam satu proses.
      3) Mengurangi waktu switching dibandingkan dengan switching antar proses.

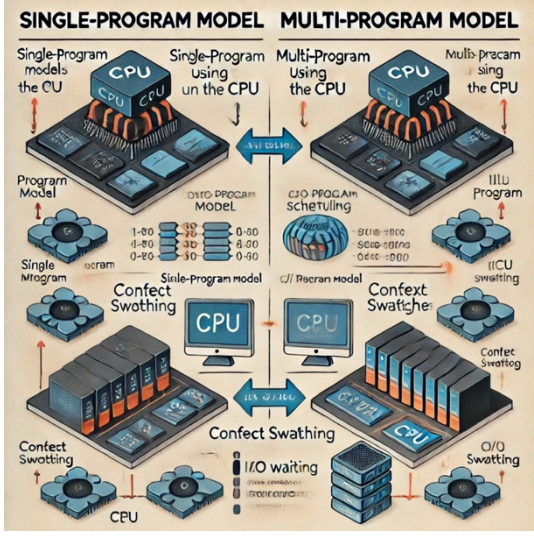


Gambar 8. Proses Thread

C. Model Eksekusi Program

Pada OS, model eksekusi program menentukan bagaimana proses atau thread dijalankan pada CPU. Ada dua model utama:

1. **Single Program Model**
   a. Dalam model ini, CPU hanya menjalankan satu proses dalam satu waktu. Setiap proses harus selesai sebelum proses lain dapat berjalan.
   b. Cocok untuk sistem operasi sederhana dan real-time yang membutuhkan eksekusi deterministik.
   c. Kekurangan: Tidak efisien untuk aplikasi multitasking karena CPU tidak digunakan secara optimal.
2. **Multi-Program Model**
   a. Pada model multi-program, beberapa proses dapat berjalan secara bersamaan melalui teknik penjadwalan CPU. OS beralih dari satu proses ke proses lainnya, memungkinkan ilusi bahwa beberapa proses berjalan sekaligus.
   b. Keuntungan: Meningkatkan pemanfaatan CPU dan mendukung multitasking.
   c. Kekurangan: Memerlukan manajemen memori yang lebih kompleks untuk mencegah konflik.



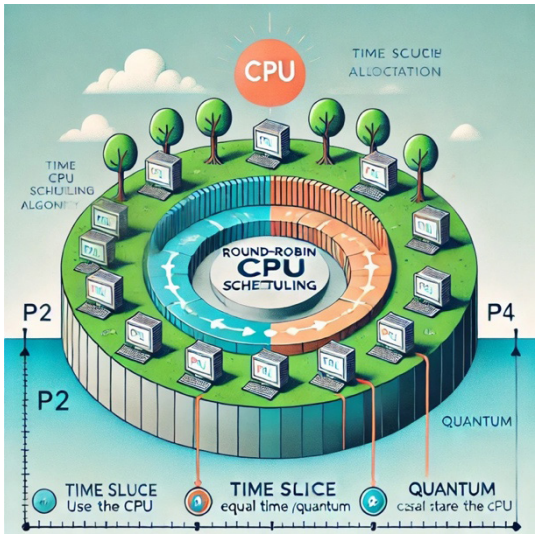
Gambar 9. Ilustrasi model single program vs multi-program dengan contoh penjadwalan CPU.

D. Penjadwalan dan Eksekusi dalam Sistem Operasi

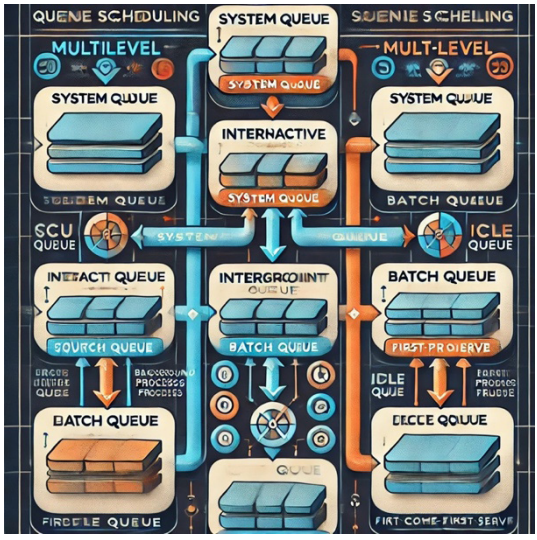
Penjadwalan adalah salah satu komponen utama OS yang bertanggung jawab untuk menentukan urutan eksekusi proses. Berikut adalah penjelasan lebih rinci tentang algoritma penjadwalan yang umum:

1. **First-Come, First-Served (FCFS):**
   a. Proses pertama yang datang akan dijalankan terlebih dahulu. Algoritma ini sederhana tetapi kurang optimal karena tidak memperhitungkan waktu eksekusi proses.
   b. Kekurangan: Masalah **convoy effect**, di mana proses yang lambat dapat menghambat proses lain yang lebih cepat.
2. **Shortest Job First (SJF):**
   a. Proses dengan waktu eksekusi terpendek diprioritaskan untuk dijalankan lebih dulu.
   b. Keuntungan: Meminimalkan waktu rata-rata untuk menyelesaikan proses.
   c. Kekurangan: Sulit untuk diterapkan dalam kasus nyata karena waktu eksekusi setiap proses seringkali tidak diketahui.
3. **Round-Robin (RR):**
   a. Setiap proses mendapatkan jatah waktu CPU dalam suatu siklus atau **quantum**. Jika proses tidak selesai dalam jatah waktu tersebut, ia akan dipindahkan ke antrian belakang.
   b. Keuntungan: Mendukung multitasking dan memberikan setiap proses waktu yang adil.
   c. Kekurangan: Pemilihan **quantum** yang tidak tepat dapat menyebabkan **context switching** berlebihan atau waktu tunggu yang lama.
4. **Multilevel Queue:**
   a. Proses dikelompokkan ke dalam beberapa antrian berdasarkan prioritas atau jenisnya (misalnya, interaktif atau batch). Setiap antrian memiliki kebijakan penjadwalan tersendiri.
   b. Keuntungan: Cocok untuk sistem dengan banyak jenis proses yang berbeda.

c. Kekurangan: Lebih kompleks dalam implementasi dan pengelolaan.



Gambar 10. Penjadwalan Round Robin

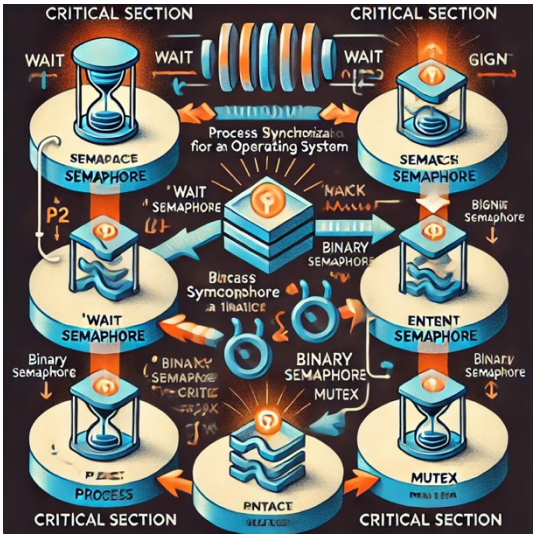


Gambar 11. Multilevel Queue

E. Sinkronisasi dan Komunikasi Antar Proses

Dalam sistem multitasking, beberapa proses atau thread mungkin perlu berkomunikasi atau bekerja sama, yang memerlukan sinkronisasi dan komunikasi antar proses.

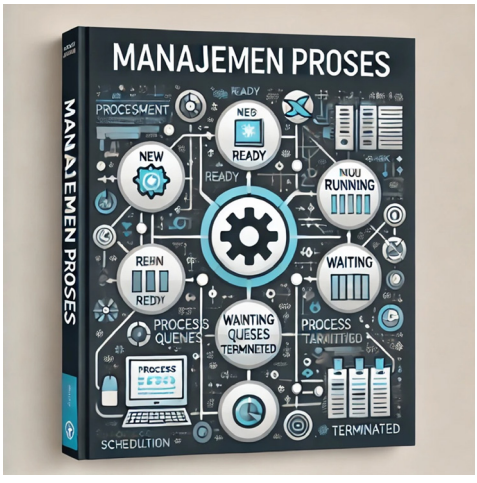
1. **Sinkronisasi Proses**
    a. Sinkronisasi diperlukan saat dua atau lebih proses mengakses data bersama. Mekanisme sinkronisasi mencegah konflik dalam **critical section** (bagian dari kode yang mengakses data bersama).
    b. Teknik sinkronisasi:
        1) **Locks**: Memungkinkan hanya satu proses yang mengakses critical section dalam satu waktu.
        2) **Semaphores**: Digunakan untuk mengatur sinkronisasi antar proses. Ada dua tipe semaphore:
            a) **Counting Semaphore**: Untuk mengelola akses pada sumber daya terbatas.
            b) **Binary Semaphore**: Digunakan untuk akses eksklusif pada critical section.
        3) **Monitors**: Memastikan akses yang aman pada variabel dalam lingkungan yang terkontrol.
2. **Inter Process Communication (IPC)**
    a. **Message Passing**: Proses dapat saling berkomunikasi dengan mengirim dan menerima pesan, misalnya menggunakan **pipes** atau **socket**.
    b. **Shared Memory**: Proses berbagi area memori tertentu, memungkinkan akses cepat namun membutuhkan mekanisme sinkronisasi tambahan untuk menghindari konflik.



Gambar 12. Diagram semaphore dengan critical section, menunjukkan cara sinkronisasi akses data oleh beberapa proses.

Bab 3

MANAJEMEN PROSES



A. Manajemen Proses

Di dalam dunia komputasi, proses dapat diibaratkan sebagai "jantung" dari sistem operasi. Proses adalah program yang sedang dieksekusi, yang mencakup instruksi, data, dan status eksekusi. Setiap kali kita menjalankan aplikasi, seperti editor teks, browser web, atau pemutar video, kita sebenarnya memulai satu atau lebih proses di dalam sistem operasi. Sistem operasi bertanggung jawab penuh dalam mengelola proses-

proses ini, memastikan bahwa mereka berjalan dengan benar dan efisien. Tanpa adanya manajemen proses, aplikasi akan saling berebut sumber daya, menyebabkan kinerja sistem yang buruk atau bahkan crash. Oleh karena itu, manajemen proses menjadi salah satu fungsi inti dari sistem operasi modern.

Pada masa awal komputasi, komputer hanya mampu menjalankan satu proses dalam satu waktu (single-tasking). Dengan munculnya konsep **time-sharing** pada tahun 1960-an, komputer mulai dapat menjalankan beberapa proses secara bergantian, memberikan ilusi multitasking kepada pengguna. Sejak saat itu, manajemen proses telah menjadi semakin kompleks, dengan sistem operasi modern mendukung **multitasking**, **multithreading**, dan **multiprocessing**. Pengembangan ini memungkinkan komputer untuk menjalankan banyak proses secara bersamaan, memanfaatkan sumber daya secara optimal, dan memberikan kinerja yang lebih baik.

B. Pengertian Proses

Secara sederhana, proses adalah program dalam status eksekusi. Saat program dimuat ke dalam memori dan mulai berjalan, program tersebut berubah status menjadi proses. Namun, proses bukan hanya sekedar kode program; ia juga mencakup ruang alamat yang terdiri dari:

1. **Text Segment**: Berisi instruksi program yang dieksekusi oleh CPU.
2. **Data Segment**: Berisi variabel global dan statis yang digunakan oleh program.
3. **Heap**: Area memori yang dialokasikan secara dinamis selama runtime program.
4. **Stack**: Menyimpan data lokal, parameter fungsi, dan alamat return untuk setiap fungsi yang dipanggil.

Manajemen proses mengacu pada serangkaian fungsi yang dilakukan oleh sistem operasi untuk menciptakan, menjadwalkan, dan menghentikan proses. Manajemen ini mencakup pengalokasian sumber daya seperti CPU, memori, dan perangkat input/output. Sistem operasi harus mampu

mengelola ribuan proses secara efisien dalam sistem yang kompleks, sambil tetap memberikan pengalaman pengguna yang responsif.

C. Peran Manajemen Proses dalam Sistem Operasi

Manajemen proses merupakan inti dari operasi sistem yang kompleks, dengan fungsi utama sebagai berikut:

1. **Pembuatan dan Penghentian Proses:**
    a. Sistem operasi menyediakan mekanisme untuk menciptakan proses baru ketika program dimulai, dan menghentikan proses saat program selesai dieksekusi. Proses baru dapat diciptakan melalui panggilan sistem seperti fork() di UNIX atau CreateProcess() di Windows.
    b. Penghentian proses terjadi saat program selesai menjalankan instruksi atau terjadi error. Sistem operasi harus membersihkan memori dan sumber daya yang digunakan oleh proses tersebut untuk mencegah kebocoran memori.
2. **Penjadwalan Proses:**
    a. Penjadwalan adalah teknik yang digunakan oleh sistem operasi untuk menentukan urutan eksekusi proses. Penjadwalan ini penting untuk memastikan bahwa setiap proses mendapatkan waktu CPU yang cukup, serta mencegah satu proses mendominasi sumber daya.
    b. Ada berbagai algoritma penjadwalan yang digunakan, seperti **First-Come, First-Served (FCFS)**, **Shortest Job First (SJF)**, dan **Round Robin (RR)**. Algoritma penjadwalan harus dipilih berdasarkan konteks sistem, apakah untuk server, desktop, atau perangkat embedded.
3. **Sinkronisasi dan Komunikasi Antar Proses:**
    a. Dalam sistem multitasking, beberapa proses mungkin harus bekerja bersama dan berbagi data. Sinkronisasi diperlukan untuk menghindari konflik saat proses-proses ini mengakses

data bersama. Teknik sinkronisasi seperti **semaphore**, **mutex**, dan **monitor** digunakan untuk mengelola akses ke data bersama.

b. Komunikasi antar proses (IPC) memungkinkan proses untuk bertukar pesan atau data. IPC dapat dilakukan melalui mekanisme seperti **pipes**, **message queues**, atau **shared memory**.

4. **Penanganan Deadlock:**

   a. Deadlock terjadi ketika dua atau lebih proses saling menunggu untuk melepaskan sumber daya yang sedang digunakan oleh proses lain, sehingga tidak ada proses yang dapat melanjutkan. Sistem operasi harus memiliki strategi untuk mendeteksi dan menangani deadlock, seperti **deadlock prevention**, **deadlock avoidance**, dan **deadlock detection**.

D. Evolusi Manajemen Proses

Manajemen proses telah mengalami banyak perubahan sejak awal perkembangan sistem operasi. Pada sistem operasi awal yang hanya mendukung batch processing, proses dieksekusi satu per satu tanpa interaksi pengguna. Dengan munculnya time-sharing, sistem operasi mulai mendukung eksekusi beberapa proses secara bersamaan, memungkinkan multitasking.

Pada tahun 1980-an, sistem operasi mulai mendukung **multithreading**, di mana satu proses dapat memiliki beberapa thread yang berjalan secara paralel. Thread adalah unit eksekusi terkecil dalam proses, memungkinkan pemrogram untuk membagi tugas besar menjadi beberapa sub-tugas yang dapat dijalankan secara paralel. Dengan multithreading, aplikasi dapat memanfaatkan prosesor multi-core untuk meningkatkan kinerja.

Sistem operasi modern, seperti Windows, Linux, dan macOS, mendukung **multiprocessing**, di mana beberapa prosesor atau inti CPU digunakan secara bersamaan untuk menjalankan banyak proses. Multiprocessing memungkinkan distribusi beban kerja yang lebih baik dan meningkatkan efisiensi sistem secara keseluruhan.

E. Tantangan dalam Manajemen Proses

Meskipun manajemen proses telah berkembang pesat, masih ada beberapa tantangan yang harus dihadapi oleh sistem operasi:

1. Efisiensi dan Performa:
   Sistem operasi harus mengalokasikan CPU dan memori dengan efisien untuk menghindari bottleneck. Penjadwalan yang buruk dapat menyebabkan penurunan performa atau masalah seperti thrashing, di mana CPU sibuk mengganti halaman memori alih-alih mengeksekusi proses.
2. Keamanan dan Isolasi Proses:
   Dalam sistem multitasking, sistem operasi harus menjaga isolasi antar proses. Jika satu proses dapat mengakses data proses lain, ini dapat menimbulkan masalah keamanan dan privasi. Teknik seperti **sandboxing** dan **protection rings** digunakan untuk menjaga isolasi proses.
3. Manajemen Proses dalam Lingkungan Terdistribusi:
   Dalam lingkungan komputasi terdistribusi, manajemen proses menjadi lebih kompleks karena proses dapat berjalan di beberapa perangkat yang terhubung dalam jaringan. Sinkronisasi dan komunikasi antar proses menjadi lebih menantang dalam sistem terdistribusi.
4. Penggunaan Energi pada Perangkat Mobile:
   Pada perangkat mobile, sistem operasi harus mengelola proses dengan efisiensi daya tinggi untuk memperpanjang umur baterai. Pengelolaan aplikasi latar belakang dan pengaturan sleep mode sangat penting untuk menghemat energi.

F. Pengertian dan Siklus Hidup Proses

1. **Definisi Proses**: Proses adalah program yang sedang dieksekusi. Sebuah program bisa saja terdiri dari satu atau beberapa proses, terutama jika menggunakan model thread atau proses paralel. Setiap proses memiliki identitas unik yang disebut **Process ID (PID)** dan

ruang alamat yang memuat kode program, data, dan stack untuk eksekusi.

2. **Siklus Hidup Proses**: Setiap proses memiliki siklus hidup yang terdiri dari beberapa tahapan, yang diatur dan dikelola oleh OS. Tahapan siklus hidup meliputi:
   a. **New**: Proses baru dibuat dan dimasukkan ke antrian siap untuk dieksekusi.
   b. **Ready**: Proses telah diinisialisasi dan siap untuk dieksekusi, menunggu penjadwalan dari CPU.
   c. **Running**: Proses sedang dieksekusi oleh CPU.
   d. **Waiting**: Proses tidak dapat melanjutkan eksekusi karena menunggu input, waktu, atau sinyal tertentu.
   e. **Terminated**: Proses telah selesai dieksekusi dan keluar dari sistem.

Gambar 13. Diagram State Siklus Hidup

G. Penjadwalan Proses

Penjadwalan proses adalah teknik untuk menentukan urutan eksekusi proses agar sumber daya sistem dimanfaatkan secara optimal. Penjadwalan dilakukan oleh komponen sistem operasi yang disebut **scheduler**.

Jenis Penjadwalan

1. Penjadwalan Jangka Panjang (Long-Term Scheduling)
   a. Bertugas memilih proses dari antrian masuk untuk dimuat ke dalam memori dan siap dieksekusi.
   b. Tujuannya adalah mengatur tingkat multiprogramming, yaitu jumlah proses yang aktif dalam memori pada saat yang sama.
2. Penjadwalan Jangka Menengah (Medium-Term Scheduling)
   a. Mengontrol proses yang berada di memori aktif dengan memutus sementara proses yang tidak aktif (swapping).
   b. Memungkinkan penggunaan memori yang lebih efisien dengan mengatur proses yang saat ini tidak memerlukan CPU.
3. Penjadwalan Jangka Pendek (Short-Term Scheduling)
   a. Memilih proses dari antrian siap untuk dieksekusi berikutnya.
   b. Scheduler ini menentukan proses mana yang akan mendapatkan jatah CPU selanjutnya, sering kali berdasarkan algoritma penjadwalan seperti Round Robin atau Shortest Job First.

Algoritma Penjadwalan

1. First-Come, First-Served (FCFS):
   Proses yang pertama datang akan dijalankan terlebih dahulu. Sederhana, tetapi dapat menyebabkan **convoy effect**, di mana proses yang lebih lama dapat menghambat proses berikutnya.
2. Shortest Job First (SJF):
   Memprioritaskan proses dengan waktu eksekusi terpendek. Efektif untuk meminimalkan waktu tunggu rata-rata, tetapi sulit diterapkan dalam situasi di mana waktu eksekusi sulit diprediksi.

3. Round Robin (RR):
    a. Setiap proses diberi waktu eksekusi dalam interval tetap atau **quantum**. Jika proses tidak selesai dalam waktu tersebut, ia akan dipindahkan ke antrian belakang, dan proses berikutnya akan mendapatkan giliran.
    b. Cocok untuk sistem multitasking karena memberikan jatah CPU yang adil kepada setiap proses.
4. Priority Scheduling:
    a. Setiap proses memiliki prioritas tertentu, dan CPU dialokasikan ke proses dengan prioritas tertinggi. Algoritma ini memungkinkan interupsi dari proses berprioritas lebih tinggi.
    b. Masalah utama adalah **starvation** di mana proses dengan prioritas rendah bisa saja menunggu terlalu lama.

H. Manajemen Deadlock

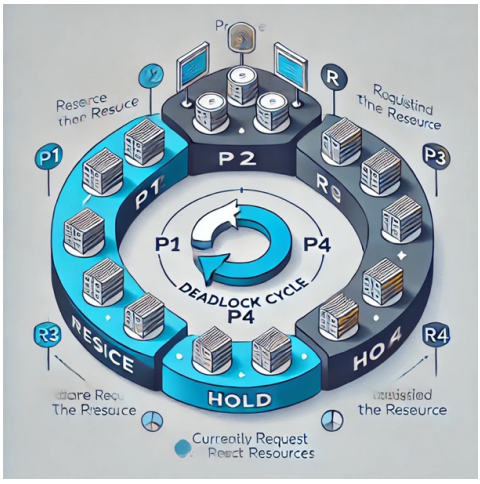
Deadlock terjadi ketika dua atau lebih proses saling menunggu sumber daya yang sedang digunakan oleh proses lain, sehingga tak ada proses yang bisa maju. Dalam manajemen deadlock, ada empat kondisi utama yang dapat menyebabkan deadlock:

1. **Mutual Exclusion**: Setidaknya satu sumber daya harus dalam keadaan eksklusif, yang artinya hanya satu proses yang dapat mengakses sumber daya tersebut dalam satu waktu.
2. **Hold and Wait**: Proses yang telah memiliki sumber daya tertentu dapat meminta sumber daya tambahan dan menunggu tanpa melepaskan sumber daya yang telah dimilikinya.
3. **No Preemption**: Sumber daya yang telah diberikan tidak dapat diambil secara paksa dari proses sampai proses tersebut melepaskannya secara sukarela.
4. **Circular Wait**: Ada serangkaian proses yang saling menunggu secara siklis.

Penanganan Deadlock

Ada beberapa metode untuk menangani deadlock:

1. Pencegahan Deadlock:
   Mencegah satu atau lebih kondisi deadlock terjadi. Contohnya adalah dengan memastikan tidak ada kondisi hold and wait atau circular wait.
2. Penghindaran Deadlock:
   Algoritma seperti **Banker's Algorithm** mengecek apakah alokasi sumber daya tertentu dapat menyebabkan deadlock sebelum diberikan ke suatu proses.
3. Deteksi Deadlock dan Pemulihan:
   a. Sistem secara berkala melakukan pengecekan kondisi deadlock menggunakan graf alokasi sumber daya.
   b. Jika deadlock terdeteksi, sistem dapat melakukan pemulihan dengan membatalkan proses tertentu atau melepaskan sumber daya.
4. Pendekatan Ignorance (Pengabaian):
   Dalam banyak sistem seperti OS konsumen, deadlock tidak dicegah atau dihindari secara eksplisit, dengan asumsi bahwa deadlock jarang terjadi.



Gambar 14. Diagram Siklus Deadlock

I. Concurrency dan Sinkronisasi Proses

Concurrency (keserentakan) terjadi ketika beberapa proses atau thread berjalan bersamaan, baik pada CPU tunggal atau multi-core. Pengelolaan concurrency membutuhkan sinkronisasi untuk menghindari konflik akses ke data yang dibagi.

1. **Critical Section Problem**
   Critical Section adalah bagian dari kode di mana beberapa proses atau thread mungkin mengakses data bersama secara bersamaan. Tanpa sinkronisasi, ini dapat menyebabkan **race condition**, di mana hasil akhir bergantung pada urutan eksekusi proses atau thread.
2. **Teknik Sinkronisasi**
   a. Locks:
      Mekanisme sederhana untuk mencegah proses lain mengakses critical section. Hanya satu proses yang dapat mengunci sumber daya dalam satu waktu.
   b. Semaphores:
      Semaphores adalah variabel integer yang mengatur akses ke sumber daya terbatas. Ada dua jenis semaphore:
      1) **Counting Semaphore**: Mengatur sejumlah akses ke sumber daya terbatas.
      2) **Binary Semaphore**: Mirip dengan lock, mengizinkan hanya satu proses dalam critical section.
   c. Monitors:
      Monitor adalah mekanisme yang mengelola akses ke variabel bersama. Proses yang ingin mengakses variabel di dalam monitor harus menggunakan **sinyal kondisi** untuk memasuki critical section.
   d. Barriers:
      Digunakan dalam sistem multiproses di mana beberapa proses atau thread harus menunggu di titik tertentu sebelum melanjutkan eksekusi.

3. **Masalah Klasik Sinkronisasi**

   Beberapa masalah klasik sinkronisasi yang sering diajarkan adalah:

   a. **The Producer-Consumer Problem**: Menangani masalah buffer dengan sinkronisasi antara produsen yang menambahkan data dan konsumen yang mengambil data.
   b. **The Readers-Writers Problem**: Mengatur sinkronisasi antara pembaca dan penulis agar pembaca tidak mengganggu penulis dan sebaliknya.
   c. **The Dining Philosophers Problem**: Contoh klasik yang menggambarkan deadlock dan starvation. Setiap filosof membutuhkan dua garpu untuk makan, tetapi hanya satu yang dapat digunakan dalam satu waktu.

Bab 4

MANAJEMEN MEMORI

A. Manajemen Memori

Memori adalah komponen vital dalam sistem komputasi yang berfungsi sebagai tempat penyimpanan sementara bagi data dan instruksi yang sedang dieksekusi oleh CPU. Dalam arsitektur komputer modern, memori sering kali disebut sebagai **memori utama** atau **Random Access Memory (RAM)**. Memori memainkan peran penting dalam menjalankan program dan aplikasi, karena data dan instruksi yang dibutuhkan CPU harus

dimuat ke memori sebelum dapat diproses. Namun, memori fisik memiliki keterbatasan kapasitas, dan sistem operasi bertanggung jawab untuk mengelola memori ini dengan efisien. Inilah yang menjadi dasar dari **manajemen memori** dalam sistem operasi.

Manajemen memori merupakan salah satu fungsi inti dari sistem operasi yang memastikan bahwa setiap aplikasi memiliki akses ke ruang memori yang diperlukan tanpa mengganggu aplikasi lain. Pada sistem operasi modern, manajemen memori melibatkan berbagai teknik seperti alokasi memori, paging, segmentasi, dan memori virtual. Sistem operasi harus dapat mengatur memori dengan cerdas, terutama pada sistem multitasking di mana banyak proses berjalan secara bersamaan dan memperebutkan akses ke memori yang terbatas.

Seiring perkembangan teknologi, kebutuhan akan memori yang lebih besar dan teknik pengelolaan memori yang lebih canggih terus meningkat. Dari komputer mainframe pada era 1960-an hingga komputer personal dan perangkat mobile saat ini, sistem operasi harus terus beradaptasi untuk mengelola memori secara lebih efisien dan mendukung aplikasi yang semakin kompleks. Pada bab ini, kita akan mengeksplorasi berbagai teknik manajemen memori dan bagaimana sistem operasi memastikan kinerja yang optimal serta stabilitas sistem.

B. Konsep Dasar Manajemen Memori

Secara umum, memori adalah tempat penyimpanan data yang dapat diakses oleh CPU. Memori utama atau RAM memiliki sifat volatile, artinya data akan hilang saat daya mati. Oleh karena itu, memori utama digunakan untuk menyimpan data sementara saat aplikasi sedang dijalankan. Manajemen memori mencakup pengalokasian ruang memori untuk proses, pelacakan ruang memori yang digunakan dan yang bebas, serta pengelolaan data yang harus disimpan dalam memori.

Ada beberapa konsep dasar yang menjadi inti dari manajemen memori, yaitu:

1. Alokasi Memori:

Proses pengalokasian memori mengacu pada penetapan ruang memori untuk proses yang baru dibuat. Sistem operasi harus dapat memberikan memori yang cukup kepada setiap proses, serta menghindari konflik penggunaan memori antara proses yang berjalan secara bersamaan.

2. Fragmentasi:
   Fragmentasi adalah masalah yang muncul ketika memori terbagi menjadi blok-blok kecil yang tidak berdekatan, sehingga sulit untuk dialokasikan pada proses baru. Ada dua jenis fragmentasi:
   a. **Fragmentasi Internal**: Terjadi ketika memori yang dialokasikan lebih besar dari yang dibutuhkan oleh proses, sehingga sebagian memori terbuang.
   b. **Fragmentasi Eksternal**: Terjadi ketika blok-blok memori yang tersedia tidak cukup besar untuk menampung proses baru meskipun total memori bebas cukup.
3. Swapping:
   Swapping adalah teknik di mana sistem operasi memindahkan proses dari memori utama ke penyimpanan sekunder (disk) untuk mengosongkan ruang memori. Proses ini dilakukan ketika memori utama penuh dan memerlukan ruang untuk proses baru.
4. Memori Virtual:
   Memori virtual adalah teknik yang memungkinkan sistem operasi memperluas kapasitas memori dengan menggunakan penyimpanan sekunder sebagai tambahan memori. Dengan memori virtual, aplikasi dapat mengakses ruang memori yang lebih besar dari kapasitas fisik RAM.

C. Fungsi Manajemen Memori dalam Sistem Operasi

Sistem operasi memiliki beberapa fungsi utama dalam manajemen memori:

1. Pelacakan Memori:
   Sistem operasi harus dapat melacak ruang memori yang digunakan oleh proses dan ruang memori yang bebas. Untuk melakukannya, OS menggunakan struktur data seperti **bitmap** atau **linked list** yang mencatat status setiap blok memori.
2. Alokasi Memori:
   Sistem operasi mengalokasikan memori kepada proses berdasarkan permintaan. Ada dua metode utama alokasi memori, yaitu **alokasi statis** dan **alokasi dinamis**. Alokasi statis dilakukan pada saat program dimulai, sedangkan alokasi dinamis dilakukan selama runtime.
3. Manajemen Fragmentasi:
   Untuk mengurangi fragmentasi, sistem operasi menggunakan teknik seperti **compaction** (menggabungkan blok-blok memori yang bebas) atau teknik pengelolaan memori seperti paging dan segmentasi yang menghindari fragmentasi eksternal.
4. Proteksi Memori:
   Proteksi memori memastikan bahwa satu proses tidak dapat mengakses memori milik proses lain. Sistem operasi menggunakan mekanisme seperti **address space** dan **paging table** untuk memisahkan ruang memori proses.

D. Teknik Manajemen Memori: Paging dan Segmentasi

Dua teknik utama dalam manajemen memori yang digunakan oleh sistem operasi adalah **paging** dan **segmentasi**.

1. Paging:
   a. Paging adalah teknik yang membagi memori fisik dan memori virtual menjadi blok-blok kecil yang disebut **pages** dan **frames**. Setiap page dari memori virtual dipetakan ke frame di memori

fisik. Teknik ini menghindari fragmentasi eksternal karena ukuran page yang tetap.

b. **Page Table** digunakan oleh sistem operasi untuk mencatat pemetaan antara pages dan frames. Setiap proses memiliki page table yang mencatat alamat fisik setiap page yang digunakan.

2. Segmentasi:
   a. Segmentasi adalah teknik yang membagi program menjadi segmen-segmen logis berdasarkan unit program, seperti kode, data, dan stack. Setiap segmen memiliki ukuran yang berbeda sesuai dengan kebutuhan.
   b. Sistem operasi menggunakan **segment table** untuk mencatat alamat awal dan panjang setiap segmen, memungkinkan akses langsung ke data yang diperlukan.

E. Memori Virtual: Ekspansi Kapasitas Memori

Memori virtual adalah teknik yang memungkinkan aplikasi untuk menggunakan lebih banyak memori daripada yang tersedia secara fisik di RAM. Sistem operasi menggunakan **paging** dan **swapping** untuk mengelola memori virtual, memungkinkan proses yang tidak aktif untuk disimpan sementara di disk.

Keuntungan dari memori virtual meliputi:

1. **Eksekusi Program Besar**: Memori virtual memungkinkan program yang membutuhkan lebih banyak memori daripada kapasitas RAM untuk tetap dijalankan.
2. **Isolasi Proses**: Setiap proses memiliki ruang memori virtual tersendiri, sehingga mencegah konflik antara proses.
3. **Efisiensi Penggunaan Memori**: Memori virtual memungkinkan pemanfaatan memori secara lebih efisien, hanya memuat bagian dari program yang sedang dibutuhkan.

F. Tantangan dalam Manajemen Memori

Meski manajemen memori telah mengalami banyak perkembangan, beberapa tantangan masih harus dihadapi:

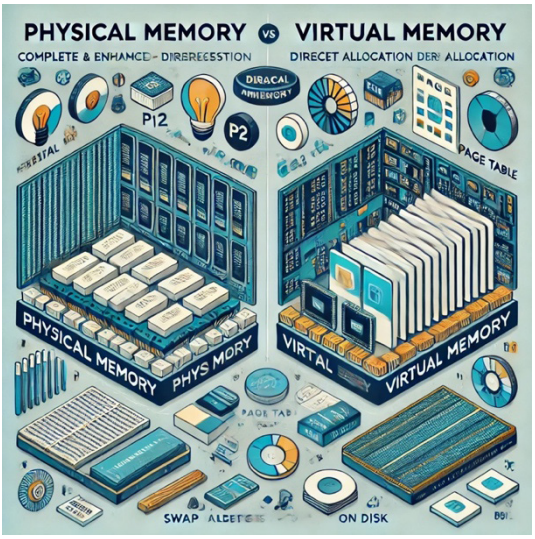
1. Thrashing:
   Thrashing terjadi ketika sistem sibuk memindahkan data antara memori utama dan disk, sehingga mengurangi waktu yang tersedia untuk eksekusi program. Thrashing biasanya disebabkan oleh penggunaan memori yang berlebihan dan algoritma page replacement yang tidak efisien.
2. Keamanan dan Proteksi Memori:
   Sistem operasi harus melindungi memori dari akses tidak sah, baik dari aplikasi jahat maupun dari kesalahan program. Teknik proteksi seperti **sandboxing** dan **ASLR (Address Spac Layout Randomization)** digunakan untuk mencegah serangan.
3. Manajemen Memori pada Perangkat Mobile:
   Pada perangkat mobile, sistem operasi harus lebih hemat dalam penggunaan memori karena kapasitas RAM yang terbatas. OS mobile seperti Android dan iOS menggunakan teknik **memory compaction** dan **background task management** untuk mengoptimalkan memori.

G. Konsep Memori Fisik dan Virtual

1. **Memori Fisik**: Memori fisik adalah memori yang terdapat dalam perangkat keras komputer, yaitu RAM. RAM memiliki kapasitas terbatas, dan sistem operasi harus mengelola alokasi ruang untuk memastikan aplikasi yang berjalan memiliki cukup memori tanpa menghabiskan sumber daya.
2. **Memori Virtual**: Memori virtual adalah teknik yang memungkinkan sistem operasi menggunakan sebagian dari penyimpanan sekunder (misalnya, hard disk) sebagai memori tambahan. Dengan memori virtual, sistem dapat menjalankan program yang membutuhkan lebih banyak memori daripada kapasitas fisik yang tersedia di RAM.

Fungsi Memori Virtual

1. **Ekspansi Memori**: Memori virtual memungkinkan sistem untuk menjalankan program besar yang tidak cukup hanya dengan memori fisik.
2. **Isolasi Proses**: Memori virtual memberikan setiap proses ruang memori yang terisolasi, menghindari konflik antar proses.
3. **Keamanan**: Memori virtual dapat mengontrol akses ke memori, sehingga proses tidak bisa membaca atau menulis data proses lain.



Gambar 15. Ilustrasi perbandingan memori fisik dan memori virtual

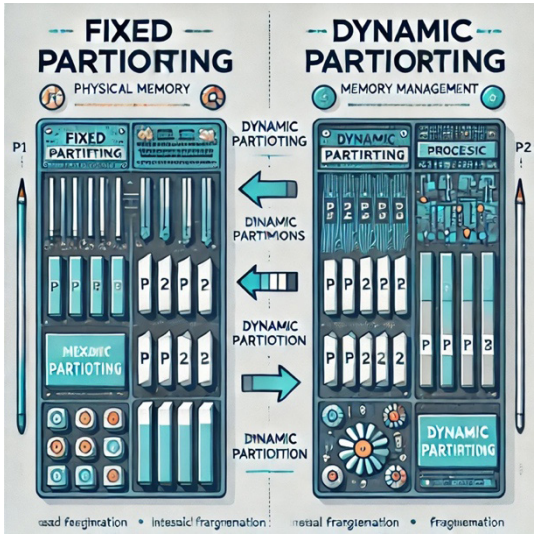
H. Alokasi Memori

Sistem operasi memiliki beberapa metode untuk mengalokasikan memori yang efisien agar dapat menghindari fragmentasi, baik internal maupun eksternal. Berikut adalah penjelasan beberapa metode alokasi memori:

1. **Fixed Partitioning (Partisi Tetap)**
   a. Memori dibagi menjadi partisi-partisi dengan ukuran tetap. Setiap partisi hanya bisa ditempati satu proses, tanpa memperhatikan kebutuhan ukuran proses.
   b. **Kelebihan**: Sederhana dan cepat dalam implementasi.

c. **Kekurangan**: Terjadi fragmentasi internal (sisa memori dalam partisi yang tidak terpakai oleh proses).

2. **Dynamic Partitioning (Partisi Dinamis)**
   a. Memori dialokasikan secara dinamis sesuai dengan ukuran proses yang masuk. Setiap proses mendapatkan ruang sesuai kebutuhannya.
   b. **Kelebihan**: Mengurangi fragmentasi internal karena ukuran partisi mengikuti kebutuhan proses.
   c. **Kekurangan**: Dapat menyebabkan fragmentasi eksternal, yaitu ruang kosong kecil yang tersebar di memori dan sulit digunakan.



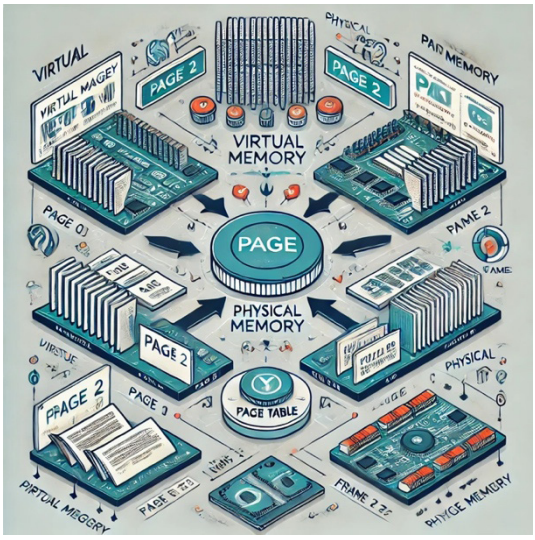
Gambar 16. Ilustrasi fixed dan dynamic partitioning

I. Pengelolaan Memori Virtual: Paging dan Segmentation

1. **Paging**
   a. Paging adalah teknik untuk membagi memori fisik dan memori virtual menjadi blok-blok berukuran tetap yang disebut **halaman (pages)**.
   b. Setiap halaman dari proses dipetakan ke dalam **frame** di memori fisik.

c. **Keuntungan**:
   1) Menghilangkan fragmentasi eksternal, karena setiap halaman memiliki ukuran yang sama.
   2) Proses dapat dialokasikan pada blok memori yang terpisah, memungkinkan lebih banyak proses dijalankan bersamaan.

d. **Page Table**: Setiap proses memiliki page table yang mencatat pemetaan antara halaman virtual dengan frame fisik. Page table membantu OS dalam melakukan translasi alamat dari alamat virtual ke alamat fisik.



Gambar 17. Diagram paging

2. Segmentation

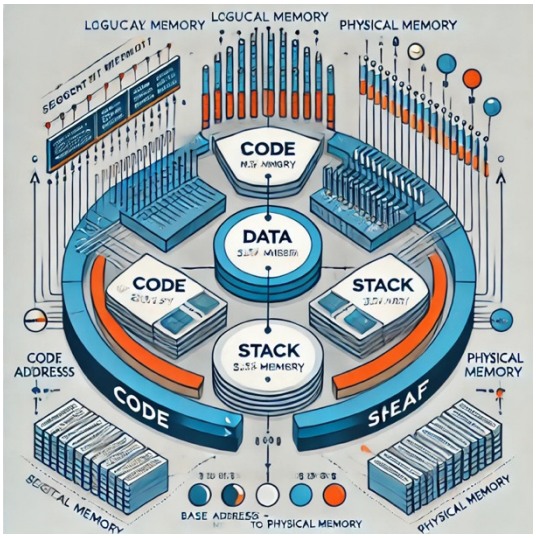
a. Segmentation adalah teknik pengelolaan memori di mana program dibagi menjadi segmen-segmen yang mencerminkan unit logis dari program, seperti **kode**, **data**, dan **stack**.

b. Setiap segmen dialokasikan dalam blok yang terpisah di memori, memungkinkan proses menggunakan segmen yang berbeda untuk tugas yang berbeda.

c. **Keuntungan**:

1) Meningkatkan efisiensi karena setiap segmen hanya memerlukan ruang sebesar kebutuhan proses.
2) Segmen logis membantu pemrogram untuk mengelola memori secara lebih teratur.

d. **Segment Table**: Seperti page table, segment table menyimpan informasi mengenai alamat awal dan panjang setiap segmen untuk membantu OS dalam pemetaan alamat virtual ke fisik.



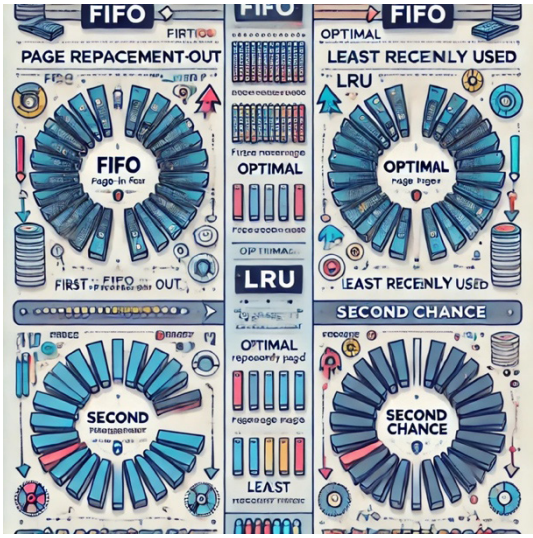
Gambar 18. Diagram Segmentation

J. Page Replacement dan Thrashing

Paging memungkinkan OS untuk menggunakan memori virtual secara efisien, tetapi kadang-kadang, halaman yang diperlukan oleh proses tidak ada di memori fisik dan harus dimuat dari disk. Situasi ini disebut **page fault**.

1. **Algoritma Page Replacement**
   Ketika memori fisik penuh, OS harus mengganti halaman yang ada dengan halaman baru yang dibutuhkan proses. Beberapa algoritma penggantian halaman adalah:

a. FIFO (First-In, First-Out):
    1) Halaman yang pertama kali dimuat adalah halaman yang pertama kali diganti.
    2) Kelemahan: Tidak efisien karena halaman yang sering dibutuhkan bisa saja diganti.
b. Optimal Page Replacement:
    1) Mengganti halaman yang tidak akan digunakan dalam waktu terpanjang di masa depan.
    2) Ideal namun sulit diterapkan karena OS tidak bisa memprediksi halaman yang dibutuhkan di masa depan.
c. LRU (Least Recently Used):
    1) Mengganti halaman yang paling lama tidak digunakan.
    2) Kelebihan: Lebih efisien daripada FIFO dalam banyak kasus, karena mempertimbangkan halaman yang sering digunakan.
    3) Kelemahan: Memerlukan pencatatan waktu penggunaan setiap halaman, sehingga menambah overhead.
d. Second Chance (Clock Algorithm):
    1) Modifikasi dari FIFO dengan memberikan "kesempatan kedua" pada halaman yang sudah lama tidak digunakan tetapi baru saja diakses.
    2) Meningkatkan efisiensi dan mengurangi penggantian halaman yang masih aktif.



Gambar 19. Ilustrasi setiap algoritma page replacement (FIFO, Optimal, LRU, Second Chance) untuk menunjukkan bagaimana algoritma memutuskan halaman yang harus diganti.

2. **Thrashing**
   a. **Thrashing** adalah kondisi di mana sistem terlalu sibuk memuat dan mengganti halaman sehingga waktu eksekusi aplikasi sangat lambat atau bahkan terhenti.
   b. **Penyebab Thrashing**:
      1) Jumlah proses terlalu banyak dan membutuhkan lebih banyak memori daripada yang tersedia.
      2) Algoritma page replacement tidak efektif dalam mengelola halaman yang sering digunakan.
   c. **Solusi untuk Thrashing**:
      1) Mengurangi multiprogramming dengan menunda proses tertentu atau menurunkan prioritasnya.
      2) Menggunakan algoritma page replacement yang lebih efisien, seperti LRU atau second chance.

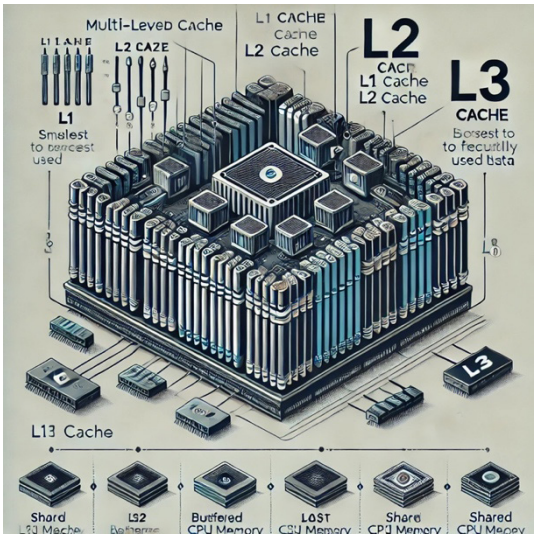
Gambar 20. Diagram thrashing yang menunjukkan situasi ketika page fault meningkat secara signifikan, memperlambat performa sistem

K. Memori Cache

Cache adalah memori berukuran kecil dengan kecepatan tinggi yang digunakan untuk menyimpan data sementara dari memori utama atau perangkat penyimpanan sekunder. Cache sangat penting dalam meningkatkan kinerja sistem.

1. **Tingkatan Memori Cache**
   a. **L1 Cache**: Cache yang terintegrasi langsung dalam CPU dan memiliki kecepatan sangat tinggi. Kapasitasnya kecil, biasanya beberapa kilobyte.
   b. **L2 Cache**: Cache dengan kapasitas yang lebih besar dari L1, sering berada di antara CPU dan memori utama. L2 memiliki kecepatan sedikit di bawah L1.
   c. **L3 Cache**: Cache dengan kapasitas yang lebih besar lagi dan digunakan untuk menyimpan data dari beberapa core CPU dalam sistem multi-core.

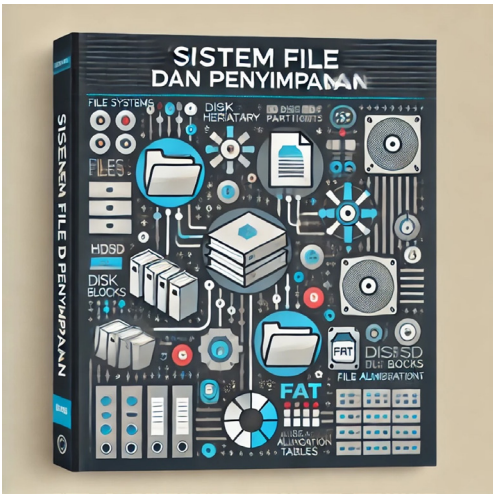
2. **Algoritma Cache Replacement**
   Mirip dengan page replacement, cache juga membutuhkan algoritma untuk mengganti data lama saat ruang cache penuh. Beberapa algoritma yang umum digunakan adalah:
   a. **Least Recently Used (LRU)**: Mengganti data yang paling lama tidak diakses.
   b. **Most Recently Used (MRU)**: Mengganti data yang paling sering digunakan, sering digunakan dalam kasus tertentu.
   c. **Random Replacement**: Mengganti data secara acak, biasanya digunakan untuk cache kecil.
3. **Keuntungan Memori Cache**
   a. **Waktu Akses Cepat**: Cache memiliki kecepatan akses lebih cepat daripada RAM, mengurangi waktu yang dibutuhkan untuk mengambil data yang sering diakses.
   b. **Meningkatkan Kinerja CPU**: Dengan mengurangi waktu yang dibutuhkan untuk mengambil data, cache membantu CPU dalam menjalankan instruksi lebih cepat.



Gambar 21. Diagram hierarki cache (L1, L2, L3) dan peran masing-masing dalam meningkatkan kinerja.

Bab 5

SISTEM FILE DAN PENYIMPANAN



Sistem file dan penyimpanan adalah komponen kunci dalam sistem operasi yang berfungsi untuk mengatur dan mengelola data. Setiap kali kita menyimpan dokumen, membuka aplikasi, atau memutar video, kita berinteraksi dengan sistem file yang ada di balik layar. Sistem file memungkinkan sistem operasi untuk menyimpan, mengakses, dan mengatur data dengan cara yang efisien dan terstruktur. Tanpa sistem file, data yang disimpan di perangkat penyimpanan hanya berupa kumpulan

bit yang tidak terorganisir, sehingga sulit untuk diakses dan dikelola oleh pengguna.

Perangkat penyimpanan adalah media fisik yang digunakan untuk menyimpan data secara permanen atau sementara. Penyimpanan modern mencakup berbagai jenis perangkat, seperti hard disk drive (HDD), solid-state drive (SSD), dan penyimpanan berbasis cloud. Setiap perangkat penyimpanan memiliki karakteristik unik dalam hal kecepatan, kapasitas, dan metode akses data. Sistem operasi harus mampu mengelola berbagai jenis penyimpanan ini dengan efisien untuk memastikan kinerja yang optimal.

A. Latar Belakang dan Evolusi Sistem File

Pada awal perkembangan komputer, penyimpanan data dilakukan secara sederhana dengan format yang tidak terstruktur. File disimpan secara langsung di perangkat penyimpanan tanpa ada pengorganisasian yang jelas. Namun, dengan meningkatnya kompleksitas program dan kebutuhan akan penyimpanan data yang lebih besar, sistem operasi mulai mengembangkan sistem file yang lebih terstruktur.

Sistem file pertama yang diperkenalkan pada komputer mainframe hanyalah kumpulan file dalam satu direktori. Namun, ketika komputer personal menjadi umum, sistem file berkembang menjadi lebih kompleks dengan dukungan untuk hierarki direktori, izin akses, dan manajemen metadata. Kemajuan dalam teknologi penyimpanan, seperti penggunaan SSD dan teknologi cloud, juga telah mendorong pengembangan sistem file yang lebih cepat, lebih aman, dan lebih fleksibel.

B. Fungsi Utama Sistem File

Sistem file bertindak sebagai antarmuka antara pengguna dan perangkat penyimpanan, menyediakan cara terorganisir untuk menyimpan dan mengambil data. Beberapa fungsi utama sistem file meliputi:

1. Manajemen Penyimpanan Data:
   Sistem file mengelola alokasi ruang di perangkat penyimpanan, memastikan bahwa data disimpan secara efisien tanpa fragmentasi yang berlebihan. Sistem operasi menggunakan berbagai metode, seperti alokasi kontigu, linked allocation, dan indexed allocation untuk mengelola ruang penyimpanan.
2. Organisasi File dan Direktori:
   Sistem file menyediakan struktur hierarki yang memungkinkan pengguna untuk mengatur file dalam folder atau direktori. Struktur ini memudahkan pengguna untuk menemukan dan mengelola data. Struktur direktori dapat berupa single-level, two-level, tree-structured, atau bahkan acyclic graph.
3. Manajemen Metadata:
   Metadata adalah informasi tentang file, seperti nama file, ukuran, waktu pembuatan, dan izin akses. Sistem file menyimpan metadata ini untuk memungkinkan pengguna mengakses file dengan cepat dan memastikan keamanan serta integritas data.
4. Keamanan dan Proteksi:
   Sistem file harus memastikan bahwa data dilindungi dari akses yang tidak sah. Izin akses dan kontrol keamanan, seperti Access Control List (ACL) dan enkripsi file, digunakan untuk melindungi file dari modifikasi atau penghapusan yang tidak diinginkan.

C. Tipe-tipe Sistem File

Ada berbagai jenis sistem file yang dikembangkan untuk memenuhi kebutuhan dan karakteristik perangkat yang berbeda:

1. FAT (File Allocation Table):
   FAT adalah salah satu sistem file paling awal yang dikembangkan untuk sistem DOS dan Windows. Tipe FAT mencakup FAT12, FAT16, dan FAT32, yang digunakan terutama pada perangkat penyimpanan eksternal seperti USB flash drive.

2. NTFS (New Technology File System):
   NTFS adalah sistem file modern yang digunakan oleh Windows, menawarkan fitur-fitur canggih seperti journaling, enkripsi, dan dukungan untuk file besar. NTFS dirancang untuk keandalan dan keamanan data yang lebih baik dibandingkan FAT.
3. EXT (Extended File System):
   EXT digunakan pada sistem Linux dan memiliki beberapa versi, seperti ext2, ext3, dan ext4. EXT4 mendukung fitur journaling dan dapat mengelola file dengan ukuran yang sangat besar, menjadikannya pilihan populer untuk server dan perangkat berbasis Linux.
4. APFS (Apple File System):
   APFS adalah sistem file yang dikembangkan oleh Apple untuk perangkat macOS dan iOS. APFS dirancang untuk kecepatan dan efisiensi pada SSD, dengan dukungan untuk enkripsi, snapshot, dan fitur lainnya yang meningkatkan performa dan keamanan.

D. Teknologi Penyimpanan: Dari Hard Disk Hingga Cloud

Perangkat penyimpanan telah mengalami perkembangan yang signifikan, mulai dari hard disk tradisional hingga teknologi penyimpanan berbasis cloud yang memungkinkan akses data dari mana saja. Beberapa teknologi penyimpanan yang umum digunakan meliputi:

1. Hard Disk Drive (HDD):
   HDD adalah perangkat penyimpanan yang menggunakan piringan magnetik untuk menyimpan data. Meskipun memiliki kapasitas besar, HDD lebih lambat dibandingkan SSD karena menggunakan komponen mekanis.
2. Solid-State Drive (SSD):
   SSD menggunakan flash memory untuk menyimpan data, yang memungkinkan akses data yang lebih cepat dan konsumsi daya yang lebih rendah dibandingkan HDD. SSD banyak digunakan pada laptop, desktop, dan server modern.

3. Cloud Storage:
   Penyimpanan berbasis cloud memungkinkan pengguna untuk menyimpan data di server remote yang diakses melalui internet. Cloud storage menawarkan fleksibilitas, skalabilitas, dan kemudahan akses, tetapi juga memerlukan langkah-langkah keamanan tambahan untuk melindungi data.

E. Keamanan dalam Sistem File dan Penyimpanan

Keamanan data menjadi salah satu perhatian utama dalam manajemen sistem file dan penyimpanan, terutama dengan meningkatnya serangan siber dan risiko kehilangan data. Sistem operasi modern menawarkan berbagai fitur keamanan untuk melindungi data, seperti:

1. Enkripsi File dan Disk:
   Enkripsi adalah teknik yang mengubah data menjadi format yang tidak dapat dibaca tanpa kunci enkripsi. Sistem operasi seperti Windows dan macOS mendukung fitur enkripsi seperti BitLocker dan FileVault untuk melindungi data di perangkat penyimpanan.
2. Journaling File System:
   Sistem file dengan fitur journaling mencatat setiap perubahan sebelum dilakukan secara permanen. Hal ini membantu mencegah kerusakan data jika terjadi kegagalan sistem atau mati listrik mendadak.
3. Backup dan Recovery:
   Sistem operasi menyediakan mekanisme backup yang memungkinkan pengguna untuk membuat salinan data penting. Backup membantu memulihkan data jika terjadi kerusakan sistem atau kehilangan data akibat serangan malware.

F. Tantangan dalam Manajemen Sistem File dan Penyimpanan

Meskipun sistem file dan teknologi penyimpanan telah berkembang pesat, masih ada beberapa tantangan yang harus dihadapi:

1. Fragmentasi:
   Fragmentasi terjadi ketika file disimpan dalam blok-blok yang tidak berurutan, sehingga mengurangi efisiensi akses data. Sistem operasi harus menggunakan teknik defragmentasi atau menggunakan sistem file seperti EXT4 dan APFS yang dirancang untuk meminimalkan fragmentasi.
2. Kapasitas Penyimpanan:
   Dengan meningkatnya ukuran file multimedia dan data aplikasi, kebutuhan akan kapasitas penyimpanan yang besar terus meningkat. Solusi seperti cloud storage dan storage virtualization digunakan untuk mengatasi masalah ini.
3. Keamanan dan Privasi:
   Dalam era digital saat ini, menjaga keamanan dan privasi data menjadi tantangan utama. Sistem operasi harus menerapkan protokol keamanan yang kuat dan enkripsi untuk melindungi data pengguna dari akses yang tidak sah.

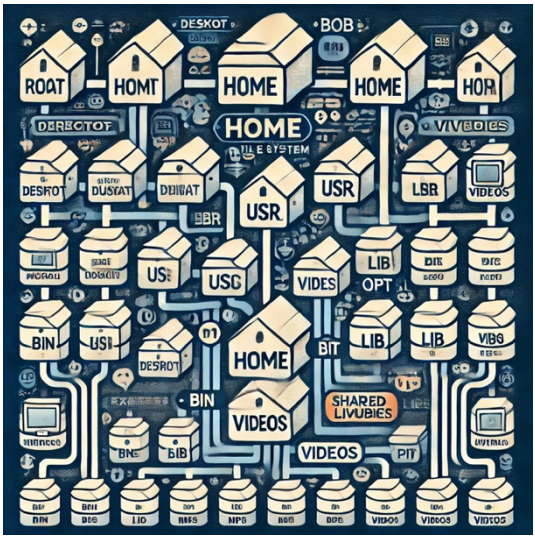
G. Struktur Sistem File

Sistem file adalah mekanisme yang disediakan oleh OS untuk menyimpan dan mengorganisir file di perangkat penyimpanan seperti hard disk, SSD, atau penyimpanan eksternal. Struktur sistem file mencakup beberapa elemen:

1. **File**: Unit dasar dari data yang disimpan, dapat berupa file teks, gambar, audio, program, dll. File memiliki nama dan ekstensi yang membedakan jenisnya (misalnya, .txt, .jpg, .mp3).
2. **Direktori**: Struktur hierarki yang memungkinkan pengelompokan file dalam folder untuk organisasi yang lebih baik.
3. **Path**: Rangkaian direktori yang menuju ke suatu file, baik berupa **absolute path** (path lengkap dari root) atau **relative path** (path dari direktori saat ini).

Jenis Sistem File

1. **FAT (File Allocation Table)**: Digunakan pada OS lama dan perangkat penyimpanan eksternal. Terdiri dari FAT16, FAT32, dan exFAT.
2. **NTFS (New Technology File System)**: Sistem file Windows yang mendukung fitur keamanan dan enkripsi.
3. **EXT (Extended File System)**: Sistem file Linux, seperti ext2, ext3, dan ext4, yang mendukung fitur journaling.
4. **APFS (Apple File System)**: Digunakan pada macOS dengan dukungan enkripsi dan snapshot.



Gambar 22. Ilustrasi hierarki sistem file yang menunjukkan contoh struktur direktori dan file

H. Operasi Dasar pada Sistem File

OS menyediakan operasi dasar untuk pengelolaan file dan direktori:

1. **Create**: Membuat file atau direktori baru.
2. **Open**: Membuka file untuk membaca, menulis, atau memodifikasi.
3. **Read**: Membaca isi dari file yang telah dibuka.
4. **Write**: Menulis atau memodifikasi isi file.

5. **Delete**: Menghapus file atau direktori dari sistem.
6. **Seek**: Memindahkan pointer ke lokasi tertentu dalam file.

Operasi ini memungkinkan aplikasi berinteraksi dengan data yang disimpan di perangkat penyimpanan. Misalnya, aplikasi pengolah kata membuka file teks, membaca, dan menulis data saat pengguna mengedit dokumen.

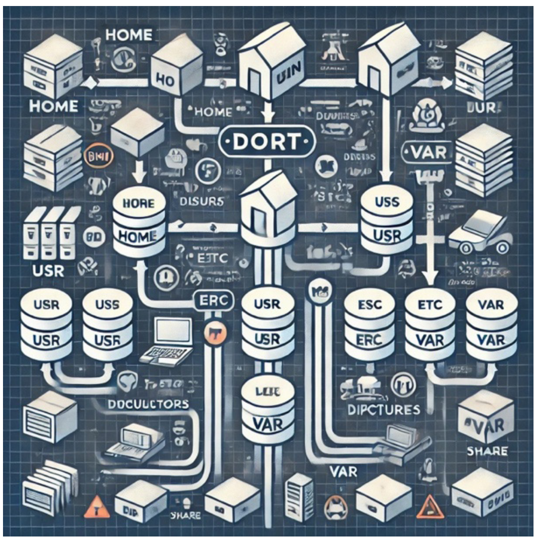
Gambar 23. Diagram operasi dasar pada file, termasuk open, read, write, dan delete

I. Manajemen Struktur Direktori

Struktur direktori adalah organisasi logis yang memungkinkan file diorganisir dalam bentuk hierarki. Ada beberapa jenis struktur direktori:

1. Single-Level Directory:
   Semua file berada dalam satu direktori. Mudah diimplementasikan, namun sulit untuk mengelola file dalam jumlah besar.
2. Two-Level Directory:
   Memisahkan direktori untuk setiap pengguna. Setiap pengguna memiliki direktori utamanya sendiri, memungkinkan keamanan dan isolasi file.

3. Tree-Structured Directory:
   Direktori utama (root) memiliki cabang direktori yang dapat berisi sub-direktori. Struktur ini memungkinkan organisasi file yang kompleks.
4. Acyclic Graph Directory:
   Memungkinkan file atau direktori memiliki beberapa parent, memungkinkan akses data secara bersama dari beberapa lokasi.
5. General Graph Directory:
   Struktur direktori yang paling fleksibel, mendukung banyak cabang dan link antar direktori. Namun, diperlukan mekanisme untuk menghindari loop.



Gambar 24. Diagram struktur direktori

J. Pengelolaan Ruang Penyimpanan

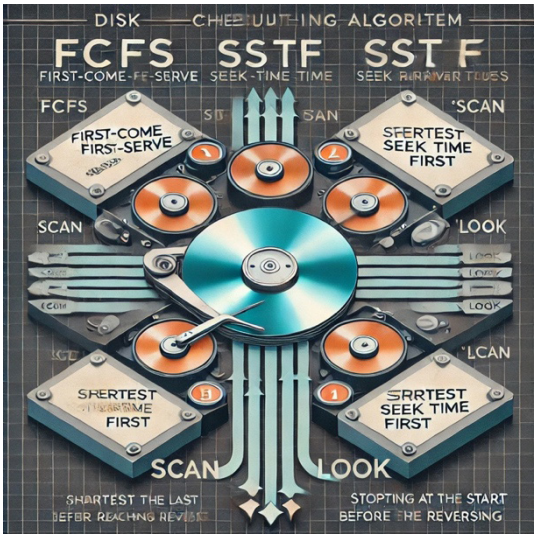
Sistem operasi mengelola ruang penyimpanan untuk memastikan data disimpan secara efisien dan menghindari fragmentasi. Ada beberapa metode alokasi ruang penyimpanan:

1. Contiguous Allocation:
   File disimpan dalam blok-blok yang berurutan. Mudah dalam akses sekuensial, tetapi rentan terhadap fragmentasi eksternal.
2. Linked Allocation:
   File disimpan dalam blok-blok yang terhubung secara logis menggunakan pointer. Menghindari fragmentasi eksternal, tetapi akses acak menjadi lambat.
3. Indexed Allocation:
   Setiap file memiliki index block yang mencatat semua blok yang dialokasikan untuk file tersebut. Memberikan fleksibilitas untuk akses sekuensial dan acak.

K. Teknik Pengelolaan Disk

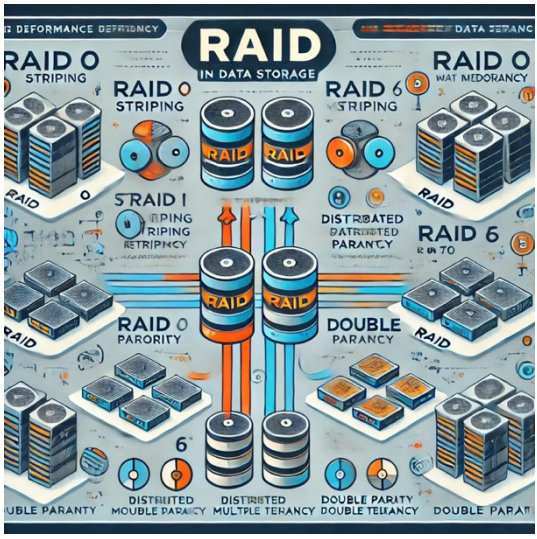
Disk adalah salah satu perangkat penyimpanan sekunder yang sering digunakan, dan OS mengelola disk untuk memastikan efisiensi dalam membaca dan menulis data.

1. Disk Scheduling:
   Karena permintaan I/O sering kali lebih banyak daripada yang dapat diakomodasi disk, OS menggunakan algoritma disk scheduling untuk memprioritaskan permintaan. Beberapa algoritma umum adalah:
   a. **FCFS (First-Come, First-Served)**: Permintaan diproses dalam urutan kedatangannya.
   b. **SSTF (Shortest Seek Time First)**: Memproses permintaan yang paling dekat dengan posisi head saat ini.
   c. **SCAN (Elevator Algorithm)**: Head disk bergerak maju-mundur seperti elevator, memproses permintaan sesuai arah.
   d. **C-SCAN (Circular SCAN)**: Head disk bergerak satu arah sampai akhir, lalu kembali ke awal tanpa memproses permintaan.



Gambar 25. Ilustrasi algoritma disk scheduling

2. RAID (Redundant Array of Independent Disks):
   RAID adalah teknik yang menggabungkan beberapa disk untuk meningkatkan kinerja, keandalan, atau keduanya. Tipe RAID meliputi:
   a. **RAID 0**: Pembagian data (striping) tanpa redundansi, untuk peningkatan kinerja.
   b. **RAID 1**: Mirroring, setiap data disalin ke dua atau lebih disk untuk meningkatkan keandalan.
   c. **RAID 5**: Striping dengan parity, memungkinkan data dipulihkan jika satu disk gagal.
   d. **RAID 10**: Kombinasi striping dan mirroring untuk kinerja dan keandalan.



Gambar 26. Diagram tipe RAID 0

L. Keamanan Sistem File

Keamanan sistem file penting untuk melindungi data dari akses tidak sah, manipulasi, atau kerusakan. Sistem operasi menerapkan berbagai metode keamanan untuk menjaga integritas dan kerahasiaan file.

1. Permissions dan Akses Kontrol
   a. OS menyediakan **permissions** untuk file dan direktori, memungkinkan pengguna untuk mengatur siapa yang bisa membaca, menulis, atau mengeksekusi file.
   b. **Access Control Lists (ACLs)** memungkinkan aturan akses lebih spesifik pada file, berdasarkan user ID atau group ID.
2. Enkripsi File
   a. Enkripsi adalah teknik yang mengamankan data dengan mengubahnya menjadi format yang tidak bisa dibaca tanpa kunci enkripsi. Contoh teknologi enkripsi pada sistem file adalah **EFS (Encrypting File System)** pada Windows dan **FileVault** pada macOS.

3. Backup dan Recovery
    a. OS menyediakan opsi backup untuk mencegah hilangnya data. Backup dapat dilakukan secara otomatis atau manual ke lokasi lain (lokal atau cloud).
    b. **Recovery** mengacu pada proses memulihkan data dari backup setelah data asli hilang atau rusak.
4. Journaling

    Beberapa sistem file modern, seperti NTFS dan ext4, mendukung journaling yang mencatat perubahan pada file sebelum disimpan secara permanen. Journaling membantu mengurangi kemungkinan data korupsi jika terjadi kegagalan sistem mendadak.

Gambar 27. Ilustrasi sistem pengaturan permissions, ACL, enkripsi, dan mekanisme journaling dalam sistem file.

Bab 6

SISTEM I/O (INPUT/OUTPUT)

Sistem Input/Output (I/O) merupakan salah satu komponen penting dalam sistem operasi yang memungkinkan interaksi antara perangkat keras dan perangkat lunak. Melalui sistem I/O, data dapat dikirimkan dari perangkat input seperti keyboard, mouse, atau sensor, menuju sistem komputasi, dan sebaliknya, dari sistem ke perangkat output seperti monitor, printer, atau speaker. Tanpa sistem I/O yang efisien, komunikasi antara

pengguna, aplikasi, dan perangkat keras akan terhambat, menyebabkan sistem tidak dapat berfungsi dengan optimal.

Dalam arsitektur komputer modern, sistem I/O berperan sebagai perantara antara perangkat keras dan CPU, mengatur aliran data yang masuk dan keluar dari sistem. Komponen ini mencakup berbagai perangkat mulai dari perangkat input sederhana seperti keyboard dan mouse hingga perangkat penyimpanan sekunder seperti hard disk drive (HDD), solid-state drive (SSD), serta perangkat jaringan. Setiap perangkat memiliki karakteristik dan protokol komunikasi yang berbeda, sehingga sistem operasi harus mampu mengelola berbagai jenis perangkat secara bersamaan.

A. Fungsi Utama Sistem Input/Output

Sistem I/O bertanggung jawab untuk mengelola operasi input dan output yang dilakukan oleh aplikasi dan perangkat keras. Beberapa fungsi utama sistem I/O meliputi:

1. Manajemen Perangkat I/O:
   Sistem operasi harus mengenali dan mengelola berbagai perangkat I/O yang terhubung ke sistem. Untuk itu, OS menggunakan device drivers, yaitu perangkat lunak yang berfungsi sebagai antarmuka antara sistem operasi dan perangkat keras. Driver mengizinkan OS untuk mengirim dan menerima data dari perangkat tanpa perlu mengetahui detail teknis perangkat tersebut.
2. Sinkronisasi Operasi I/O:
   Dalam sistem multitasking, beberapa aplikasi atau proses mungkin membutuhkan akses ke perangkat I/O yang sama secara bersamaan. Sistem I/O harus mampu mengatur antrian permintaan ini dan memastikan bahwa setiap proses mendapat giliran akses sesuai dengan prioritas yang telah ditentukan.
3. Pengelolaan Buffering:
   Buffering adalah teknik yang digunakan untuk menampung data sementara sebelum diteruskan ke perangkat I/O atau aplikasi.

Buffering membantu mengurangi waktu tunggu antara perangkat dengan kecepatan yang berbeda, sehingga data dapat diproses lebih efisien.

4. Penanganan Interrupt:
   Interrupt adalah sinyal dari perangkat keras yang memberitahu CPU bahwa perangkat memerlukan perhatian. Sistem operasi menggunakan interrupt untuk menangani permintaan I/O secara efisien tanpa harus terus-menerus memeriksa status perangkat (polling). Interrupt-driven I/O memungkinkan CPU untuk tetap bekerja pada tugas lain hingga perangkat I/O siap untuk mengirim atau menerima data.

B. Teknik Pengelolaan I/O

Untuk mengoptimalkan performa dan efisiensi, sistem operasi menggunakan beberapa teknik pengelolaan I/O yang berbeda, yaitu:

1. Polling:
   Teknik polling melibatkan pengecekan status perangkat secara berkala oleh CPU untuk menentukan apakah perangkat siap mengirim atau menerima data. Meskipun sederhana, polling dapat menyebabkan inefisiensi karena CPU harus sering memeriksa status perangkat.
2. Interrupt-Driven I/O:
   Interrupt-driven I/O memungkinkan perangkat untuk mengirim sinyal interrupt ke CPU ketika siap beroperasi, sehingga CPU tidak perlu melakukan polling. Teknik ini mengurangi overhead CPU dan lebih efisien untuk sistem multitasking.
3. Direct Memory Access (DMA):
   DMA adalah teknik di mana perangkat I/O dapat mengakses memori utama secara langsung tanpa melibatkan CPU dalam transfer data. DMA sangat berguna untuk transfer data besar seperti pada hard drive atau kartu jaringan, karena memungkinkan CPU untuk mengerjakan tugas lain sementara data dipindahkan.

C. Struktur Hierarki Sistem I/O

Sistem I/O dalam sistem operasi modern sering diorganisasikan dalam struktur hierarki yang mencakup beberapa lapisan, yaitu:

1. Hardware Layer:
   Lapisan terendah yang berisi perangkat keras fisik, seperti hard drive, keyboard, monitor, dan kartu jaringan. Setiap perangkat memiliki protokol komunikasi dan fitur yang berbeda, sehingga sistem operasi harus menyediakan driver khusus untuk setiap perangkat.
2. Device Driver Layer:
   Device drivers adalah modul perangkat lunak yang bertanggung jawab untuk mengendalikan perangkat keras dan menyediakan antarmuka standar bagi sistem operasi. Driver ini memungkinkan OS untuk mengirim perintah dan menerima data dari perangkat tanpa mengetahui detail perangkat tersebut.
3. Operating System Layer:
   Pada lapisan ini, sistem operasi mengelola antrian permintaan I/O dari berbagai aplikasi dan proses, serta melakukan penjadwalan I/O berdasarkan prioritas dan kebutuhan sumber daya. Sistem operasi juga menangani penanganan error yang mungkin terjadi selama operasi I/O.
4. Application Layer:
   Lapisan tertinggi di mana aplikasi berinteraksi dengan sistem I/O melalui API (Application Programming Interface) atau sistem panggilan (system calls). Aplikasi tidak perlu berinteraksi langsung dengan perangkat keras, melainkan menggunakan fungsi yang disediakan oleh OS untuk melakukan operasi I/O.

D. Tantangan dalam Manajemen Sistem I/O

Manajemen sistem I/O merupakan salah satu aspek paling kompleks dalam desain sistem operasi, karena harus menangani berbagai jenis

perangkat dengan karakteristik yang berbeda. Beberapa tantangan utama dalam manajemen I/O adalah:

1. Heterogenitas Perangkat:
   Sistem operasi harus dapat mengelola berbagai perangkat yang berbeda, mulai dari perangkat input sederhana hingga perangkat penyimpanan dan jaringan yang kompleks. Setiap perangkat memerlukan driver khusus dan protokol komunikasi yang berbeda.
2. Perbedaan Kecepatan Perangkat:
   Kecepatan perangkat I/O sering kali jauh lebih lambat daripada kecepatan CPU dan memori utama. Misalnya, akses data dari hard drive membutuhkan waktu yang lebih lama dibandingkan dengan akses data di memori. Sistem operasi harus mengatasi perbedaan ini dengan teknik buffering dan caching.
3. Sinkronisasi Antar Proses:
   Dalam sistem multitasking, beberapa proses mungkin membutuhkan akses ke perangkat I/O yang sama pada saat yang bersamaan. Sistem operasi harus melakukan sinkronisasi dan penjadwalan untuk menghindari konflik dan memastikan bahwa setiap proses mendapatkan waktu akses yang adil.
4. Manajemen Error:
   Kesalahan I/O, seperti kegagalan membaca disk atau perangkat yang tidak terhubung, harus ditangani dengan benar oleh sistem operasi. OS harus dapat mendeteksi, melaporkan, dan mencoba memulihkan dari error I/O tanpa mengganggu aplikasi yang sedang berjalan.

E. Perkembangan Sistem I/O di Era Modern

Teknologi I/O telah berkembang pesat dengan munculnya perangkat penyimpanan baru seperti SSD dan teknologi jaringan seperti Wi-Fi 6 dan 5G. Sistem operasi modern harus terus beradaptasi dengan teknologi ini untuk memanfaatkan kecepatan yang lebih tinggi dan kapasitas yang lebih besar. Selain itu, dengan berkembangnya Internet of Things (IoT),

sistem operasi kini juga harus mampu mengelola data dari sensor dan perangkat I/O dalam jumlah besar yang tersebar di jaringan.

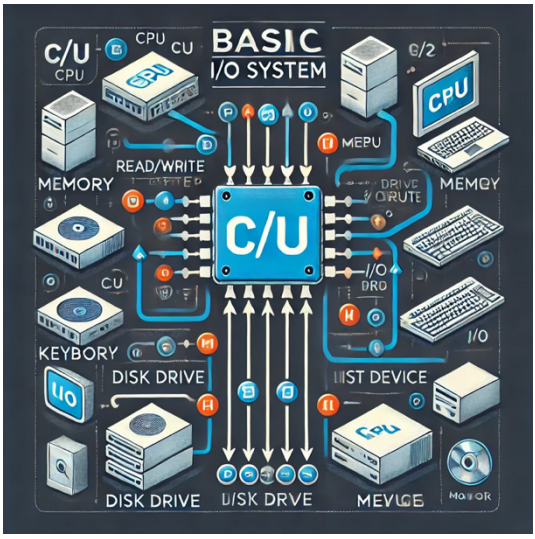
Virtualisasi dan cloud computing juga menghadirkan tantangan baru dalam manajemen I/O. Dengan virtualisasi, beberapa mesin virtual dapat berbagi perangkat I/O yang sama, sehingga sistem operasi harus memastikan bahwa perangkat I/O dapat diakses secara efisien dan aman oleh beberapa VM sekaligus.

F. Konsep Dasar I/O

Sistem I/O (Input/Output) adalah komponen sistem operasi yang memungkinkan komunikasi antara perangkat keras dan perangkat lunak. Komponen I/O menangani semua perangkat yang berinteraksi dengan lingkungan eksternal, seperti keyboard, mouse, monitor, printer, dan hard drive. Sistem operasi mengelola I/O untuk memproses data yang masuk dan keluar melalui berbagai perangkat ini.

Tipe Operasi I/O:

1. **Input**: Memasukkan data dari perangkat eksternal ke dalam sistem (misalnya, keyboard atau sensor).
2. **Output**: Mengeluarkan data dari sistem ke perangkat eksternal (misalnya, monitor atau printer).
3. **Read/Write**: Membaca data dari atau menulis data ke perangkat penyimpanan sekunder, seperti hard drive.



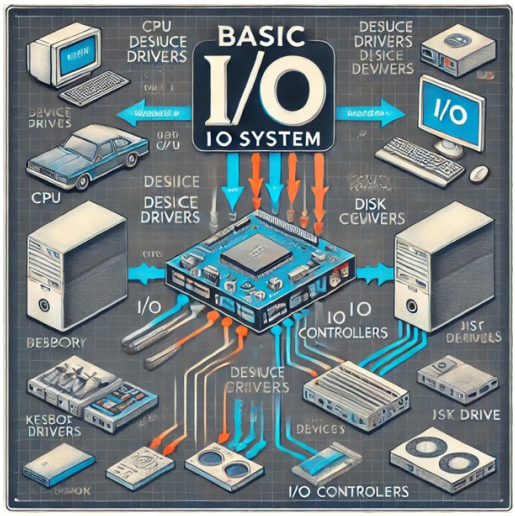
Gambar 28. Diagram dasar sistem I/O yang menunjukkan aliran data antara CPU, perangkat I/O, dan memori.

G. Manajemen Perangkat I/O

Sistem operasi mengelola perangkat I/O dengan mengatur bagaimana data dikirim dan diterima, serta bagaimana perangkat berinteraksi satu sama lain. Manajemen perangkat I/O mencakup:

1. Device Drivers (Pengendali Perangkat)
    a. Device driver adalah perangkat lunak khusus yang berfungsi sebagai perantara antara OS dan perangkat keras. Driver memungkinkan OS untuk berinteraksi dengan perangkat tertentu tanpa harus mengetahui detail teknis perangkat tersebut.
    b. Setiap perangkat keras memiliki driver yang sesuai, misalnya driver printer, driver GPU, atau driver audio.
2. I/O Controller
    a. I/O Controller adalah komponen perangkat keras yang mengatur komunikasi antara perangkat I/O dan CPU. Controller ini mengelola operasi dasar seperti transfer data dan pengaturan perangkat.

b. Contoh I/O Controller termasuk **disk controller** untuk hard drive dan **video controller** untuk monitor.



Gambar 29. Ilustrasi yang menunjukkan peran device driver dan I/O controller dalam aliran komunikasi antara perangkat keras dan sistem operasi

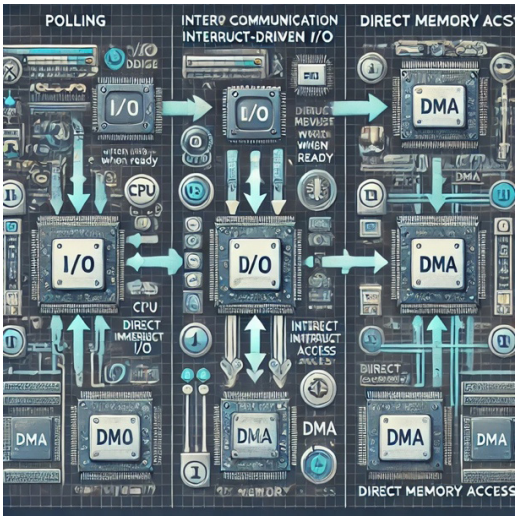
H. Teknik Pengelolaan I/O

Ada beberapa teknik pengelolaan I/O yang digunakan OS untuk memastikan bahwa proses komunikasi antara perangkat berjalan efisien:

1. Polling:
   a. Teknik polling melibatkan CPU yang secara berkala memeriksa status perangkat I/O untuk mengetahui apakah perangkat siap mengirim atau menerima data.
   b. Kelemahan: Teknik ini tidak efisien karena CPU harus selalu mengecek status perangkat I/O, sehingga menambah overhead.
2. Interrupt-Driven I/O:
   a. Pada teknik ini, perangkat I/O akan mengirim sinyal **interrupt** ke CPU saat mereka siap mengirim atau menerima data. Setelah menerima interrupt, CPU akan menunda tugas lain untuk menangani permintaan I/O.

b. Keuntungan: Mengurangi overhead CPU karena CPU hanya akan mengakses perangkat I/O saat diperlukan.

3. Direct Memory Access (DMA):
   a. DMA memungkinkan perangkat I/O mengakses memori utama secara langsung tanpa campur tangan CPU. Dengan DMA, data dapat ditransfer dengan cepat ke dan dari memori tanpa membebani CPU.
   b. Keuntungan: Sangat efisien, terutama untuk transfer data besar seperti transfer file antara hard drive dan RAM.



Gambar 30. Diagram polling, interrupt-driven I/O, dan DMA untuk menunjukkan cara kerja masing-masing teknik dalam komunikasi I/O.

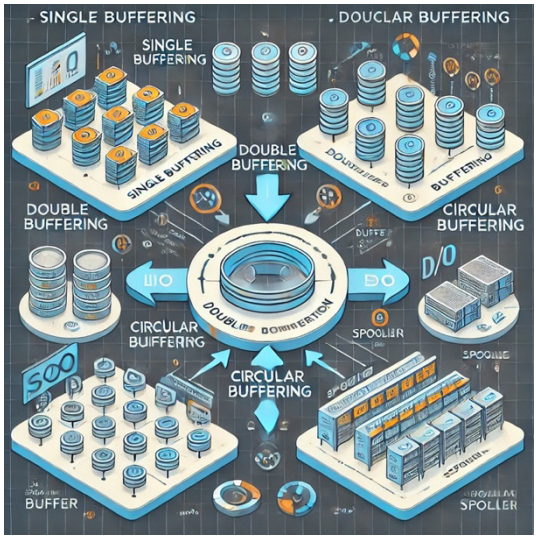
I. Buffering dan Spooling

Buffering dan spooling adalah teknik yang digunakan untuk mengoptimalkan proses I/O agar lebih efisien, terutama saat menghadapi perbedaan kecepatan antara perangkat I/O dan CPU.

1. Buffering
   a. Buffering adalah teknik menyimpan data sementara dalam buffer (memori sementara) sebelum data diproses lebih lanjut atau dikirim ke perangkat output.

b. **Single Buffering**: Memiliki satu buffer untuk menyimpan data sementara. Saat buffer penuh, data akan diproses dan dikosongkan untuk data selanjutnya.
c. **Double Buffering**: Memiliki dua buffer sehingga satu buffer dapat digunakan untuk pemrosesan sementara buffer lain diisi data baru, memungkinkan proses berjalan lebih cepat.
d. **Circular Buffering**: Menggunakan beberapa buffer dalam bentuk lingkaran, cocok untuk aplikasi yang membutuhkan transfer data terus-menerus.

2. Spooling (Simultaneous Peripheral Operation On-Line)
   a. Spooling adalah teknik menyimpan data untuk perangkat I/O dalam antrian hingga perangkat tersebut siap untuk memproses data. Misalnya, pada pencetakan, data disimpan dalam antrian spooling hingga printer siap.
   b. Keuntungan spooling adalah memungkinkan perangkat I/O menangani data secara bertahap tanpa mengganggu proses lain yang sedang berjalan di CPU.



Gambar 31. Ilustrasi buffering (single, double, circular) dan spooling, menunjukkan bagaimana data diatur untuk mengoptimalkan komunikasi I/O.

J. Sistem File Networked dan Sistem I/O Terdistribusi

Dalam lingkungan terdistribusi, sistem operasi harus dapat menangani perangkat I/O dan file yang tidak hanya berada pada perangkat lokal, tetapi juga tersebar di jaringan.

1. Network File System (NFS)
    a. NFS memungkinkan akses file yang tersimpan di komputer lain melalui jaringan, seolah-olah file tersebut ada di perangkat lokal. Dengan NFS, pengguna dapat membaca dan menulis file di server remote tanpa perlu menyalin file ke perangkat lokal.
    b. NFS mengandalkan protokol jaringan untuk mengatur komunikasi dan sinkronisasi file antar perangkat di jaringan.
2. Remote Procedure Call (RPC)
    a. RPC adalah teknik yang memungkinkan suatu program untuk mengeksekusi prosedur atau fungsi yang berada di sistem lain. RPC banyak digunakan dalam sistem terdistribusi untuk komunikasi antar proses di perangkat yang berbeda.
    b. RPC memberikan ilusi bahwa prosedur remote berjalan seolah-olah berada di sistem lokal.
3. Sinkronisasi dan Keamanan dalam Sistem I/O Terdistribusi
    a. Sinkronisasi diperlukan untuk mencegah konflik saat beberapa pengguna mencoba mengakses file atau perangkat yang sama di jaringan. OS mengelola sinkronisasi ini dengan menggunakan mekanisme seperti locking.
    b. Keamanan dalam I/O terdistribusi melibatkan enkripsi dan otorisasi untuk memastikan bahwa hanya pengguna yang sah yang dapat mengakses perangkat atau file tertentu di jaringan.

Gambar 32. Diagram arsitektur NFS dan RPC yang menunjukkan cara kerja akses file remote dan komunikasi antar sistem dalam jaringan.

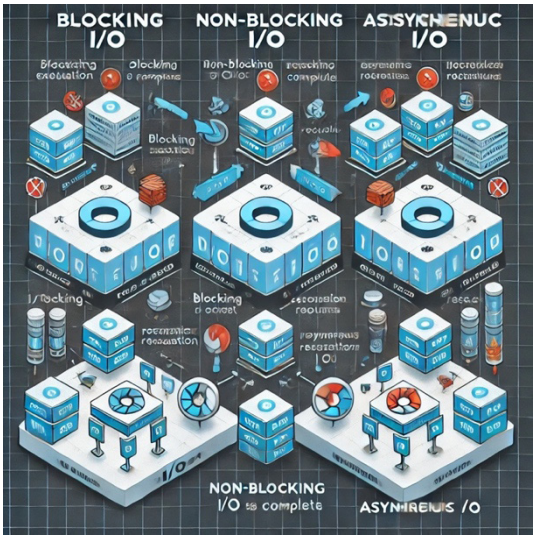
K. Manajemen Sinkronisasi I/O

Sinkronisasi sangat penting dalam manajemen I/O, terutama saat beberapa proses atau perangkat berinteraksi dengan satu perangkat I/O yang sama.

1. Blocking I/O:
    a. Pada blocking I/O, proses akan berhenti sementara hingga operasi I/O selesai. Blocking I/O biasanya digunakan saat operasi membutuhkan kecepatan tinggi tanpa banyak proses lain yang perlu dijalankan.
    b. **Keuntungan**: Sederhana dan cocok untuk aplikasi yang tidak memerlukan multitasking intensif.
    c. **Kekurangan**: Membuat proses lain tertunda saat menunggu operasi I/O selesai.
2. Non-Blocking I/O:
    a. Pada non-blocking I/O, proses akan tetap berjalan sambil menunggu operasi I/O selesai. OS akan memberikan sinyal saat operasi I/O selesai dan data dapat diakses.

b. **Keuntungan**: Memungkinkan proses lain untuk berjalan selama operasi I/O berlangsung.
    - **Kekurangan**: Kompleksitas yang lebih tinggi karena memerlukan pemrograman tambahan untuk memeriksa status operasi I/O.

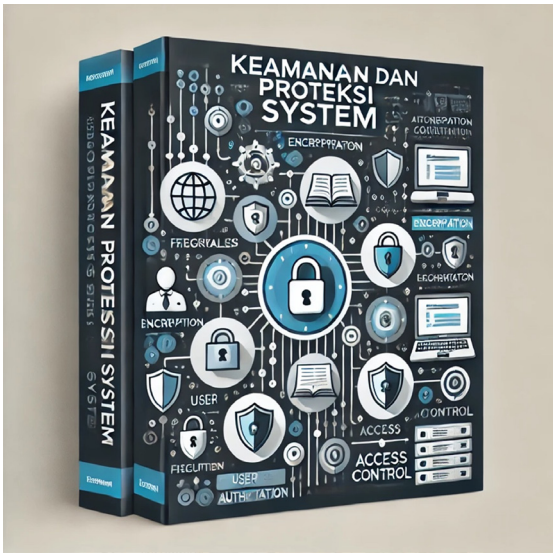
3. Asynchronous I/O:
    a. Asynchronous I/O memungkinkan aplikasi untuk memulai operasi I/O dan melanjutkan proses tanpa harus menunggu operasi selesai.
    b. **Keuntungan**: Sangat efisien dan cocok untuk aplikasi yang membutuhkan kinerja tinggi, seperti server web.
    c. **Kekurangan**: Kompleksitas yang lebih tinggi dalam mengelola komunikasi antar proses.



Gambar 33. Diagram blocking I/O, non-blocking I/O, dan asynchronous I/O untuk menunjukkan perbedaan pendekatan dalam sinkronisasi operasi I/O.

Bab 7

KEAMANAN DAN PROTEKSI SISTEM



Keamanan dan proteksi sistem operasi merupakan aspek yang sangat penting dalam dunia komputasi modern. Di era digital saat ini, sistem operasi tidak hanya mengatur dan mengelola sumber daya perangkat keras dan perangkat lunak, tetapi juga bertanggung jawab untuk melindungi data, proses, dan komunikasi dari ancaman internal dan eksternal. Setiap sistem operasi, baik itu Windows, Linux, macOS, maupun sistem operasi

mobile seperti Android dan iOS, dirancang dengan berbagai mekanisme keamanan untuk menjaga integritas, kerahasiaan, dan ketersediaan informasi.

Pada awal perkembangan komputer, sistem operasi tidak banyak menghadapi ancaman keamanan karena digunakan dalam lingkungan yang terbatas dan terisolasi. Namun, seiring dengan berkembangnya teknologi jaringan, internet, dan perangkat mobile, sistem operasi menjadi semakin rentan terhadap berbagai serangan seperti malware, virus, ransomware, dan serangan siber lainnya. Serangan-serangan ini dapat merusak data, mencuri informasi pribadi, dan mengganggu kinerja sistem, yang dapat menyebabkan kerugian besar baik bagi pengguna individu maupun organisasi.

A. Konsep Dasar Keamanan Sistem Operasi

Keamanan sistem operasi dapat dipahami melalui tiga pilar utama, yang dikenal dengan konsep CIA Triad:

1. Confidentiality (Kerahasiaan):
   Kerahasiaan bertujuan untuk melindungi data dari akses yang tidak sah. Hanya pengguna atau aplikasi yang memiliki izin yang dapat mengakses informasi sensitif. Sistem operasi menggunakan teknik seperti enkripsi, kontrol akses, dan autentikasi untuk menjaga kerahasiaan data.
2. Integrity (Integritas):
   Integritas memastikan bahwa data tidak diubah secara tidak sah selama proses penyimpanan, pengiriman, atau eksekusi. Sistem operasi menggunakan mekanisme seperti hashing, digital signature, dan journaling file system untuk mendeteksi dan mencegah perubahan data yang tidak sah.
3. Availability (Ketersediaan):
   Ketersediaan memastikan bahwa sistem dan data selalu dapat diakses oleh pengguna yang sah ketika dibutuhkan. Sistem operasi harus

mampu menangani serangan seperti Distributed Denial of Service (DDoS)yang bertujuan untuk mengganggu ketersediaan layanan.

B. Ancaman Keamanan pada Sistem Operasi

Sistem operasi menghadapi berbagai jenis ancaman yang dapat mengganggu fungsionalitas dan keamanan sistem, antara lain:

1. Malware:
   Malware, atau malicious software, adalah perangkat lunak berbahaya yang dirancang untuk merusak, mencuri, atau mengambil alih kendali sistem. Contoh malware meliputi virus, worm, trojan, ransomware, dan spyware. Malware dapat masuk ke sistem melalui unduhan file yang tidak aman, lampiran email berbahaya, atau eksploitasi kerentanan perangkat lunak.
2. Serangan Phishing:
   Phishing adalah teknik serangan di mana penyerang mencoba mencuri informasi sensitif seperti kata sandi dan nomor kartu kredit dengan menyamar sebagai entitas yang tepercaya. Sistem operasi harus memberikan perlindungan terhadap serangan ini melalui pengelolaan sertifikat keamanan dan otentikasi yang kuat.
3. Privilege Escalation:
   Privilege escalation terjadi ketika penyerang mendapatkan akses dengan hak istimewa yang lebih tinggi daripada yang dimilikinya secara sah. Dengan akses ini, penyerang dapat mengubah sistem, menginstal malware, atau mencuri data sensitif. Sistem operasi menggunakan mekanisme kontrol akses dan isolasi proses untuk mencegah privilege escalation.
4. Exploits dan Zero-Day Vulnerabilities:
   Exploits adalah serangan yang memanfaatkan kerentanan dalam sistem operasi atau aplikasi. Zero-day vulnerabilities adalah kerentanan yang belum diketahui oleh pengembang perangkat lunak, sehingga tidak ada patch yang tersedia. Sistem operasi harus selalu diperbarui dengan patch keamanan untuk mengurangi risiko dari exploits.

C. Mekanisme Keamanan dalam Sistem Operasi

Gambar 34. CIA Triad

Sistem operasi modern dilengkapi dengan berbagai mekanisme keamanan yang dirancang untuk melindungi data dan proses. Beberapa mekanisme utama meliputi:

1. Autentikasi:
   Autentikasi adalah proses verifikasi identitas pengguna sebelum mereka diberikan akses ke sistem. Sistem operasi menggunakan berbagai metode autentikasi, termasuk kata sandi, token, dan autentikasi biometrik seperti sidik jari atau pengenalan wajah.
2. Kontrol Akses:
   Sistem operasi menerapkan kontrol akses untuk menentukan siapa yang dapat mengakses sumber daya dan data. Model kontrol akses yang umum meliputi:
   a. Discretionary Access Control (DAC): Pemilik data dapat menentukan siapa yang memiliki izin untuk mengakses file atau direktori.
   b. Mandatory Access Control (MAC): Sistem operasi menentukan akses berdasarkan kebijakan keamanan yang ketat.

c. Role-Based Access Control (RBAC): Akses diberikan berdasarkan peran pengguna dalam organisasi.

3. Enkripsi:
   Enkripsi digunakan untuk melindungi data saat disimpan (at-rest) dan saat dikirimkan (in-transit). Sistem operasi menyediakan fitur enkripsi file dan disk, seperti BitLocker pada Windows dan FileVault pada macOS, untuk melindungi data dari akses yang tidak sah.
4. Sandboxing dan Virtualisasi:
   a. Sandboxing adalah teknik isolasi yang digunakan untuk menjalankan aplikasi dalam lingkungan terbatas, mencegah akses ke bagian sistem yang lebih luas. Sistem operasi seperti iOS dan Android menggunakan sandboxing untuk membatasi akses aplikasi.
   b. Virtualisasi memungkinkan sistem operasi menjalankan beberapa mesin virtual yang terisolasi, meningkatkan keamanan dengan memisahkan lingkungan eksekusi.
5. Firewall dan IDS/IPS:
   Firewall adalah perangkat atau perangkat lunak yang mengontrol lalu lintas jaringan masuk dan keluar berdasarkan aturan keamanan yang ditentukan. Intrusion Detection Systems (IDS) dan Intrusion Prevention Systems (IPS) digunakan untuk mendeteksi dan mencegah serangan siber sebelum mencapai sistem operasi.

D. Peran Patch dan Pembaruan Sistem dalam Keamanan

Patch keamanan adalah pembaruan perangkat lunak yang dirilis oleh pengembang sistem operasi untuk memperbaiki kerentanan atau bug yang ditemukan. Pembaruan sistem sangat penting untuk menjaga keamanan karena ancaman baru terus berkembang, dan setiap kerentanan yang ditemukan dapat dieksploitasi oleh penyerang. Sistem operasi modern biasanya memiliki mekanisme pembaruan otomatis untuk memastikan bahwa sistem selalu dilindungi dengan patch terbaru.

E. Tantangan Keamanan di Era Cloud dan IoT

Dengan berkembangnya teknologi cloud computing dan Internet of Things (IoT), tantangan keamanan bagi sistem operasi semakin kompleks. Sistem operasi kini harus mampu mengelola data yang tersebar di banyak server cloud dan perangkat IoT yang sering kali memiliki sumber daya terbatas. Keamanan menjadi tantangan besar karena serangan dapat datang dari berbagai titik dan vektor serangan semakin beragam.

1. Keamanan Data di Cloud:
   Penyimpanan data di cloud memerlukan enkripsi yang kuat dan manajemen akses yang ketat. Sistem operasi di cloud harus memastikan bahwa data tetap aman selama penyimpanan dan transmisi.
2. Keamanan pada Perangkat IoT:
   Perangkat IoT sering kali memiliki sistem operasi ringan dengan fitur keamanan yang terbatas. Hal ini menjadikan perangkat IoT lebih rentan terhadap serangan siber. Sistem operasi untuk IoT harus dirancang dengan fokus pada proteksi data dan komunikasi aman.

F. Model Proteksi dan Akses Kontrol

Model proteksi adalah mekanisme yang digunakan OS untuk mengendalikan akses pengguna terhadap sumber daya. Model proteksi ini bertujuan untuk mengatur siapa yang dapat mengakses, mengubah, atau mengeksekusi suatu sumber daya.

1. Access Control Matrix (Matriks Kontrol Akses)
   a. Matriks kontrol akses adalah model yang menggambarkan hak akses dari setiap pengguna untuk setiap sumber daya di dalam sistem. Dalam matriks ini, baris menunjukkan pengguna, kolom menunjukkan sumber daya, dan setiap sel menunjukkan hak akses.
   b. **Kelebihan**: Memberikan kontrol akses yang rinci dan spesifik.
   c. **Kekurangan**: Matriks ini dapat menjadi sangat besar dan kompleks pada sistem besar dengan banyak pengguna dan sumber daya.

2. Access Control Lists (ACLs)
    a. ACL adalah daftar yang melekat pada setiap sumber daya, yang mencantumkan pengguna atau kelompok pengguna yang memiliki izin tertentu. Misalnya, file di Linux dapat diatur dengan ACL untuk menentukan siapa yang dapat membaca, menulis, atau mengeksekusi file tersebut.
    b. **Kelebihan**: Memberikan kontrol yang lebih fleksibel dan lebih mudah diimplementasikan daripada matriks kontrol akses.
3. Capabilities
    a. Capabilities adalah hak akses yang diberikan kepada pengguna atau proses tertentu untuk mengakses sumber daya tanpa daftar akses langsung. Misalnya, pada sistem UNIX, file descriptor digunakan sebagai capability yang menunjukkan izin pengguna untuk mengakses file.
    b. **Keunggulan**: Membatasi akses dan mencegah penyalahgunaan, serta mempermudah distribusi hak akses dalam lingkungan terdistribusi.

Gambar 35. Ilustrasi matriks kontrol akses, ACL, dan capabilities untuk menunjukkan perbedaan dalam mengatur hak akses.

G. Autentikasi dan Autorisasi

Autentikasi dan otorisasi adalah langkah-langkah penting untuk mengidentifikasi pengguna dan menetapkan hak akses mereka di sistem.

1. Autentikasi (Authentication)
   Autentikasi adalah proses untuk memverifikasi identitas pengguna. OS menggunakan berbagai metode autentikasi, antara lain:
   a. **Password**: Bentuk autentikasi yang paling umum, di mana pengguna memasukkan kata sandi yang dikenal oleh sistem.
   b. **Token-Based Authentication**: Pengguna diberikan token (misalnya kartu, atau perangkat OTP) yang berfungsi sebagai bukti identitas.
   c. **Biometric Authentication**: Menggunakan karakteristik biologis pengguna, seperti sidik jari, retina, atau suara.
   d. **Two-Factor Authentication (2FA)**: Menggabungkan dua metode autentikasi untuk meningkatkan keamanan, seperti kombinasi password dan OTP.

2. Autorisasi (Authorization)
   Autorisasi adalah proses menentukan hak akses setelah autentikasi berhasil. Sistem menentukan sumber daya apa yang dapat diakses oleh pengguna atau aplikasi, sesuai dengan hak akses yang dimiliki.

Gambar 36. Diagram autentikasi dan otorisasi, menunjukkan alur dari verifikasi identitas hingga penetapan hak akses.

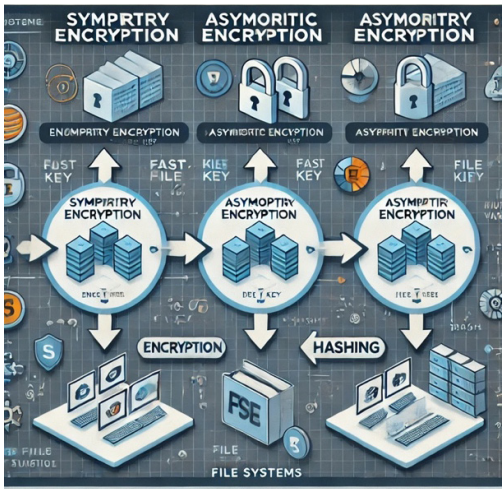
H. Enkripsi dan Keamanan Data

Enkripsi adalah proses mengamankan data dengan mengubahnya menjadi format yang hanya bisa dibaca oleh pihak yang memiliki kunci enkripsi. Enkripsi sangat penting dalam menjaga kerahasiaan dan integritas data.

1. Teknik Enkripsi
    a. **Enkripsi Simetris (Symmetric Encryption)**: Menggunakan satu kunci yang sama untuk enkripsi dan dekripsi data. Contoh algoritma: AES (Advanced Encryption Standard).
    b. **Enkripsi Asimetris (Asymmetric Encryption)**: Menggunakan sepasang kunci – satu untuk enkripsi (public key) dan satu lagi untuk dekripsi (private key). Contoh algoritma: RSA.
    c. **Hashing**: Proses mengubah data menjadi serangkaian karakter pendek yang merepresentasikan data tersebut. Digunakan untuk verifikasi integritas data. Contoh: SHA-256.
2. Enkripsi Sistem File
    a. Sistem operasi mendukung enkripsi pada level sistem file, seperti **EFS (Encrypting File System)** pada Windows dan **FileVault**

pada macOS, yang mengenkripsi seluruh disk untuk keamanan data.

b. **Full Disk Encryption (FDE)**: Mengenkripsi seluruh isi hard drive, termasuk sistem operasi, data pengguna, dan aplikasi.



Gambar 37. Ilustrasi teknik enkripsi (simetris, asimetris, hashing) dan contoh penggunaannya pada sistem file.

I. Pengamanan Sistem: Firewall, Antivirus, dan IDS/IPS

Pengamanan sistem operasi tidak hanya tentang kontrol akses tetapi juga tentang perlindungan dari ancaman eksternal.

1. Firewall
   a. Firewall adalah perangkat atau perangkat lunak yang mengontrol lalu lintas jaringan masuk dan keluar berdasarkan aturan keamanan yang telah ditetapkan. Firewall berfungsi untuk memblokir akses tidak sah dan melindungi sistem dari serangan berbasis jaringan.

b. **Jenis Firewall**:
    1) **Packet Filtering Firewall**: Menganalisis setiap paket jaringan dan menentukan apakah akan mengizinkan atau menolak berdasarkan alamat IP dan port.
    2) **Stateful Inspection Firewall**: Melacak status koneksi jaringan dan hanya mengizinkan lalu lintas yang sesuai dengan koneksi yang sudah ada.
    3) **Application Layer Firewall**: Menganalisis data yang masuk pada level aplikasi, seperti HTTP atau FTP, untuk keamanan yang lebih mendalam.

2. Antivirus
    a. Antivirus adalah perangkat lunak yang mendeteksi dan menghapus program berbahaya seperti virus, worm, dan malware. Antivirus bekerja dengan memindai file atau program dan mencocokkannya dengan database virus yang dikenal.
    b. Antivirus modern juga dapat mendeteksi malware baru menggunakan teknik heuristik dan pemindaian berbasis perilaku.
3. Intrusion Detection System (IDS) dan Intrusion Prevention System (IPS)
    a. **IDS**: Sistem yang memantau aktivitas jaringan atau sistem untuk mendeteksi serangan atau pelanggaran keamanan. IDS tidak melakukan tindakan pencegahan, tetapi memberi tahu administrator tentang potensi ancaman.
    b. **IPS**: Berfungsi seperti IDS, tetapi juga dapat memblokir lalu lintas yang mencurigakan secara otomatis untuk mencegah serangan.

Gambar 38. Diagram firewall, antivirus, IDS, dan IPS, serta cara mereka bekerja dalam sistem untuk mendeteksi dan mencegah ancaman.

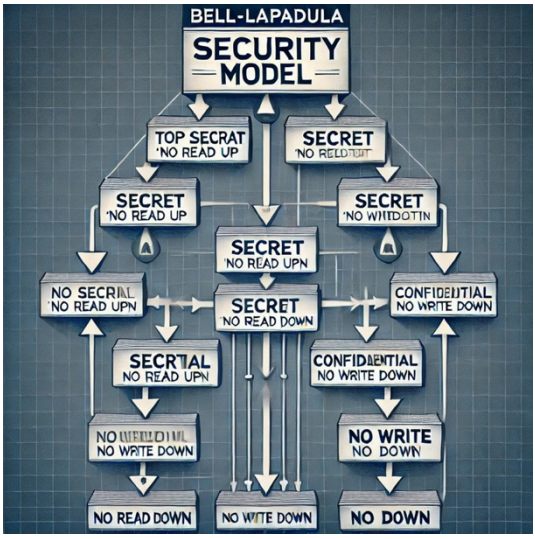
J. Model Keamanan dalam Sistem Operasi

Beberapa model keamanan umum digunakan oleh OS untuk menjaga integritas data dan kontrol akses:

1. Bell-LaPadula Model
   Fokus pada **confidentiality**. Model ini memungkinkan akses berdasarkan tingkat keamanan data. Pengguna hanya dapat mengakses data di level atau di bawah izin mereka, tetapi tidak di atasnya (no read up).
2. Biba Model
   Fokus pada **integrity**. Kebalikan dari Bell-LaPadula, model ini mencegah pengguna untuk menulis ke tingkat data yang lebih rendah (no write down) untuk menjaga integritas data.

3. Clark-Wilson Model

   Menggabungkan integritas dan kontrol akses, model ini mendefinisikan aturan-aturan untuk memastikan bahwa hanya transaksi yang valid yang dapat dilakukan oleh pengguna yang berwenang.



Gambar 39. Ilustrasi model keamanan Bell-LaPadula

K. Mitigasi Ancaman dan Respon Insiden

Sistem operasi harus siap untuk menghadapi ancaman dan merespon insiden keamanan yang mungkin terjadi.

1. Patching dan Update

   Patching adalah proses memperbarui sistem untuk menutup celah keamanan atau memperbaiki bug. OS secara berkala merilis patch keamanan untuk melindungi sistem dari ancaman yang diketahui.

2. Backup dan Recovery
   a. Backup adalah proses membuat salinan data yang dapat digunakan untuk memulihkan sistem jika terjadi insiden, seperti serangan ransomware atau kerusakan data.
   b. Recovery melibatkan pemulihan data dari backup untuk meminimalkan gangguan operasional.

3. Logging dan Monitoring
    a. Logging adalah pencatatan aktivitas yang terjadi di sistem, seperti akses file, perubahan konfigurasi, dan login pengguna. Monitoring melibatkan pemantauan log untuk mendeteksi aktivitas yang mencurigakan.
    b. OS dapat menggunakan tool monitoring seperti **Syslog** di Linux atau **Event Viewer** di Windows untuk mengidentifikasi dan merespon ancaman.

Bab 8

SISTEM OPERASI TERDISTRIBUSI DAN JARINGAN

Dalam dunia komputasi modern, sistem tidak lagi berdiri sendiri sebagai entitas terisolasi. Sebaliknya, berbagai perangkat komputasi saling terhubung dalam jaringan, memungkinkan kolaborasi dan pemanfaatan sumber daya secara lebih efisien. Konsep sistem operasi terdistribusi lahir dari kebutuhan untuk mengelola kumpulan perangkat yang tersebar dalam jaringan, sehingga dapat berfungsi sebagai satu kesatuan yang terpadu.

Sistem operasi terdistribusi memungkinkan beberapa komputer bekerja bersama untuk menyelesaikan tugas yang kompleks, berbagi data, dan mengelola sumber daya dengan lebih efisien. Teknologi ini menjadi dasar bagi berbagai aplikasi modern, mulai dari layanan cloud computing hingga Internet of Things (IoT).

Pada awalnya, komputer hanya digunakan dalam lingkungan yang tertutup dan terisolasi. Namun, dengan berkembangnya teknologi jaringan pada tahun 1970-an dan 1980-an, muncul kebutuhan akan kolaborasi antar komputer. Sistem operasi tradisional yang dirancang untuk satu mesin tunggal tidak lagi memadai untuk memenuhi kebutuhan ini. Oleh karena itu, sistem operasi terdistribusi dikembangkan untuk mengatasi tantangan ini, menyediakan mekanisme yang memungkinkan perangkat terhubung dalam jaringan untuk berbagi sumber daya, seperti CPU, memori, dan perangkat penyimpanan.

A. Apa Itu Sistem Operasi Terdistribusi?

Sistem operasi terdistribusi adalah sistem yang mengelola beberapa komputer yang terhubung dalam jaringan sebagai satu sistem tunggal yang terpadu. Dalam sistem terdistribusi, pengguna tidak perlu mengetahui di mana data atau proses dieksekusi, karena sistem operasi menyembunyikan kompleksitas di balik jaringan dan mengabstraksikan sumber daya secara transparan. Pengguna hanya melihat satu sistem komputasi, meskipun sebenarnya terdiri dari banyak node atau komputer yang bekerja bersama.

Beberapa karakteristik utama dari sistem operasi terdistribusi meliputi:

1. Transparansi: Sistem operasi terdistribusi dirancang untuk memberikan ilusi bahwa pengguna berinteraksi dengan satu sistem tunggal, meskipun terdiri dari banyak perangkat. Transparansi mencakup akses data, lokasi, migrasi proses, serta replikasi sumber daya.
2. Reliabilitas: Sistem terdistribusi menawarkan keandalan yang lebih tinggi karena memiliki redundansi; jika satu node mengalami

kegagalan, node lainnya dapat mengambil alih tugas tersebut tanpa mengganggu kinerja sistem.

3. Scalability (Dapat Diskalakan): Sistem terdistribusi dapat dengan mudah diperluas dengan menambahkan lebih banyak node atau perangkat ke jaringan, tanpa mengurangi kinerja keseluruhan sistem.
4. Resource Sharing (Berbagi Sumber Daya): Sistem operasi terdistribusi memungkinkan berbagi sumber daya, seperti memori, CPU, dan perangkat penyimpanan antar node, meningkatkan efisiensi dan utilisasi.

B. Peran Jaringan dalam Sistem Operasi Terdistribusi

Teknologi jaringan merupakan fondasi dari sistem operasi terdistribusi. Jaringan memungkinkan beberapa komputer yang tersebar secara geografis untuk berkomunikasi dan bekerja bersama. Ada beberapa jenis jaringan yang digunakan dalam sistem terdistribusi:

1. Local Area Network (LAN):
   LAN adalah jaringan dengan jangkauan terbatas, biasanya dalam satu gedung atau kampus. LAN memiliki kecepatan tinggi dan latency rendah, cocok untuk sistem terdistribusi dengan node yang berdekatan.
2. Wide Area Network (WAN):
   WAN mencakup area geografis yang lebih luas, memungkinkan komunikasi antar node yang berjauhan. WAN sering digunakan untuk menghubungkan beberapa LAN dalam satu jaringan terdistribusi yang besar.
3. Wireless Networks dan IoT:
   Jaringan nirkabel memungkinkan komunikasi antar perangkat tanpa menggunakan kabel fisik, mendukung fleksibilitas dan mobilitas. Dalam sistem terdistribusi modern, jaringan IoT memungkinkan ribuan perangkat untuk saling berkomunikasi dan berbagi data secara real-time.

C. Keuntungan Sistem Operasi Terdistribusi

Sistem operasi terdistribusi menawarkan beberapa keuntungan dibandingkan dengan sistem operasi tradisional:

1. Efisiensi Penggunaan Sumber Daya:
   Dengan berbagi sumber daya antar node, sistem terdistribusi dapat memanfaatkan perangkat keras dengan lebih baik. Proses yang memerlukan banyak memori atau daya komputasi dapat dialokasikan ke node yang memiliki kapasitas lebih besar.
2. Keandalan dan Toleransi Kesalahan:
   Karena sistem terdistribusi memiliki banyak node yang berfungsi bersama, kegagalan pada satu node tidak akan menyebabkan kegagalan sistem secara keseluruhan. Sistem operasi terdistribusi dapat mendeteksi kegagalan dan melakukan failover, yaitu mengalihkan tugas ke node lain yang masih aktif.
3. Skalabilitas Tinggi:
   Sistem terdistribusi dapat dengan mudah diperluas dengan menambah node baru ke jaringan. Ini memungkinkan sistem untuk menangani beban kerja yang lebih besar dan meningkatkan kinerja secara keseluruhan.

D. Tantangan dalam Sistem Operasi Terdistribusi

Meskipun menawarkan banyak keuntungan, sistem operasi terdistribusi juga menghadapi beberapa tantangan, di antaranya:

1. Sinkronisasi dan Koordinasi:
   Dalam sistem terdistribusi, beberapa node mungkin bekerja pada tugas yang sama atau berbagi data yang sama. Sinkronisasi diperlukan untuk menghindari konflik atau inkonsistensi data. Mekanisme seperti logical clocks dan distributed mutual exclusion digunakan untuk mengoordinasikan akses ke sumber daya bersama.

2. Keamanan dan Privasi:
   Sistem terdistribusi lebih rentan terhadap serangan karena data dan proses tersebar di banyak node yang terhubung melalui jaringan. Sistem operasi harus menerapkan mekanisme keamanan yang kuat, seperti enkripsi, otentikasi, dan kontrol akses untuk melindungi data.
3. Keterlambatan dan Latensi:
   Dalam jaringan yang luas, keterlambatan dalam komunikasi antar node dapat memengaruhi kinerja sistem. Latensi tinggi dapat menyebabkan inkonsistensi data dan memperlambat respon sistem.

E. Aplikasi Sistem Operasi Terdistribusi

Sistem operasi terdistribusi banyak digunakan dalam berbagai aplikasi modern, seperti:

1. Cloud Computing:
   Cloud computing menggunakan konsep sistem terdistribusi untuk menyediakan sumber daya komputasi yang skalabel dan elastis. Layanan seperti Amazon Web Services (AWS) dan Google Cloud Platform (GCP) memungkinkan pengguna untuk menjalankan aplikasi di jaringan terdistribusi tanpa perlu mengelola perangkat keras fisik.
2. Internet of Things (IoT):
   IoT terdiri dari banyak perangkat pintar yang saling berkomunikasi melalui jaringan. Sistem operasi terdistribusi memungkinkan koordinasi dan pengelolaan data dari ribuan sensor dan perangkat dalam jaringan IoT.
3. Sistem P2P (Peer-to-Peer):
   Sistem P2P seperti BitTorrent atau aplikasi blockchain menggunakan konsep terdistribusi di mana setiap node berfungsi sebagai client dan server, berbagi data dan layanan secara langsung tanpa pusat kontrol.

F. Konsep Dasar Sistem Operasi Terdistribusi

Sistem operasi terdistribusi adalah OS yang memungkinkan beberapa komputer yang terhubung dalam jaringan untuk bekerja sama seolah-olah mereka adalah satu sistem. Sistem ini mengelola sumber daya dan data yang tersebar di berbagai perangkat untuk memastikan koordinasi, efisiensi, dan transparansi.

Karakteristik Sistem Operasi Terdistribusi:

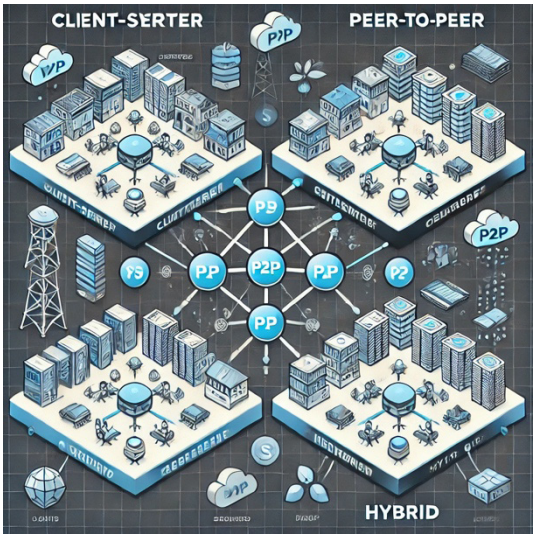
1. **Transparansi**: Pengguna dan aplikasi tidak perlu mengetahui lokasi fisik sumber daya atau data. OS membuat semua data terlihat seolah-olah berada dalam satu sistem.
2. **Reliability (Keandalan)**: Sistem terdistribusi memiliki redundansi data yang memungkinkan pemulihan jika salah satu komponen gagal.
3. **Scalability (Dapat Diskalakan)**: Sistem dapat dengan mudah ditambah kapasitasnya dengan menambahkan perangkat baru.
4. **Resource Sharing (Berbagi Sumber Daya)**: Sumber daya seperti penyimpanan, data, dan printer dapat digunakan bersama oleh berbagai pengguna dan aplikasi di jaringan.

Gambar 40. Diagram arsitektur sistem terdistribusi yang menunjukkan komputer terhubung dalam jaringan dan berbagi sumber daya

G. Arsitektur Sistem Operasi Terdistribusi

Ada beberapa model arsitektur dalam sistem terdistribusi yang umum digunakan, yaitu:

1. Client-Server Model
    a. Dalam model ini, ada dua entitas utama: **client** yang meminta layanan dan **server** yang menyediakan layanan. Server bertanggung jawab untuk menyediakan sumber daya atau melakukan pemrosesan yang diminta oleh client.
    b. **Contoh**: Web server yang menyediakan halaman web untuk pengguna di seluruh dunia.
2. Peer-to-Peer Model (P2P)
    a. Setiap node di jaringan berfungsi sebagai client dan server. Dalam P2P, tidak ada satu node pusat; semua node dapat saling berinteraksi langsung.
    b. **Contoh**: Torrent dan aplikasi berbagi file yang menggunakan jaringan P2P untuk berbagi data.
3. Hybrid Model
    a. Menggabungkan fitur dari model client-server dan P2P untuk memanfaatkan kelebihan dari kedua model. Beberapa aplikasi terdistribusi, seperti aplikasi messaging, menggunakan model hybrid untuk mendukung akses cepat dan fleksibel.



Gambar 41. Diagram arsitektur client-server, P2P, dan hybrid yang menunjukkan bagaimana setiap model berinteraksi dalam sistem terdistribusi.

H. Komunikasi Antar Proses di Sistem Terdistribusi

Komunikasi antar proses adalah kunci dalam sistem terdistribusi, memungkinkan proses di perangkat yang berbeda berkomunikasi dan bekerja bersama. Ada beberapa metode utama yang digunakan:

1. Remote Procedure Call (RPC)
    a. RPC memungkinkan suatu proses untuk menjalankan prosedur di perangkat lain seolah-olah prosedur itu ada di perangkat lokal. Dengan RPC, proses dapat meminta layanan dari proses di perangkat lain melalui jaringan.
    b. **Keuntungan**: Mempermudah pemrograman terdistribusi dengan cara yang transparan.
    c. **Kekurangan**: Overhead tambahan dari proses komunikasi jarak jauh.
2. Message Passing
    a. Message passing adalah metode di mana proses saling bertukar pesan untuk berkomunikasi. Proses dapat mengirim dan

menerima pesan untuk sinkronisasi atau untuk mengirim data tertentu.

b. **Keuntungan**: Efisien untuk pertukaran data kecil dan komunikasi sinkron.

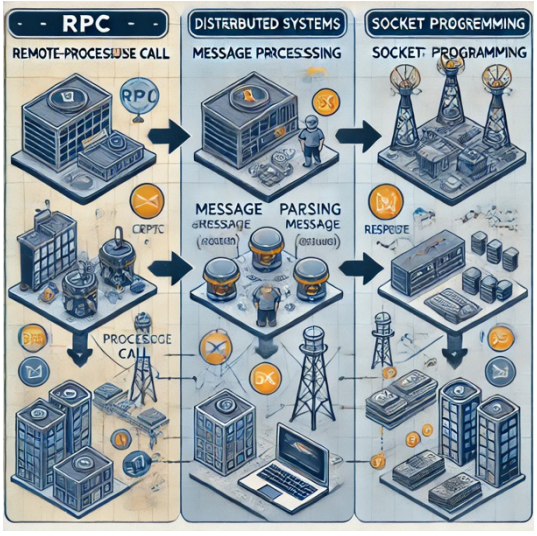
c. **Kekurangan**: Memerlukan protokol tambahan untuk mengatur format dan penerimaan pesan.

3. Socket Programming

a. Sockets memungkinkan dua proses untuk saling berkomunikasi melalui protokol jaringan, seperti TCP atau UDP. Socket menyediakan antarmuka untuk mengirim dan menerima data secara langsung.

b. **Keuntungan**: Memberikan kontrol yang baik terhadap komunikasi jaringan.

c. **Kekurangan**: Kompleks untuk dikonfigurasi dan memerlukan pemahaman lebih mendalam tentang protokol jaringan.

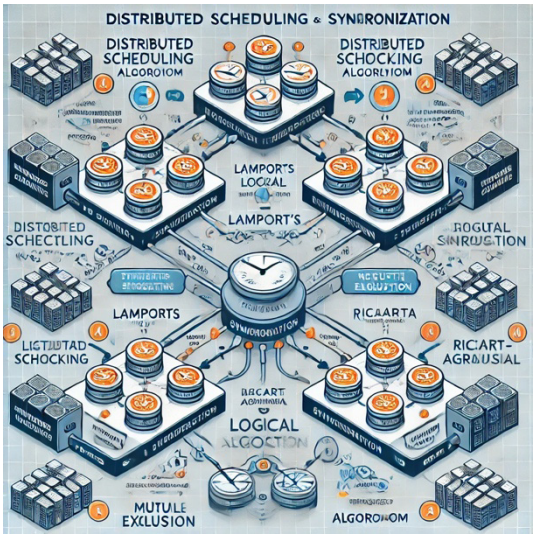


Gambar 42. Diagram RPC, message passing, dan socket programming yang menunjukkan alur komunikasi antar proses di sistem terdistribusi

I. Penjadwalan Terdistribusi dan Sinkronisasi

Dalam sistem terdistribusi, sinkronisasi antar proses diperlukan untuk menghindari konflik dan memastikan proses berjalan dengan baik.

1. Penjadwalan Terdistribusi
    a. Penjadwalan terdistribusi bertujuan untuk mendistribusikan tugas di berbagai perangkat dalam jaringan. Algoritma penjadwalan terdistribusi mempertimbangkan faktor seperti beban perangkat, waktu pemrosesan, dan kapasitas jaringan.
    b. **Algoritma Penjadwalan**:
        1) **Load Balancing**: Mendistrubusikan beban secara merata untuk mencegah perangkat tertentu bekerja terlalu berat.
        2) **Round Robin**: Menjadwalkan tugas secara bergiliran di semua perangkat.
        3) **Priority-Based Scheduling**: Menjadwalkan tugas berdasarkan prioritas untuk tugas yang mendesak.
2. Sinkronisasi Terdistribusi
    a. Dalam sistem terdistribusi, sinkronisasi digunakan untuk mengatur akses ke sumber daya bersama oleh beberapa perangkat. Beberapa metode sinkronisasi yang umum adalah:
    b. **Logical Clock (Algoritma Lamport)**: Menyinkronkan proses berdasarkan peristiwa di sistem untuk menciptakan urutan logis tanpa perlu sinkronisasi waktu yang sebenarnya.
    c. **Distributed Mutual Exclusion**: Mengontrol akses proses ke sumber daya bersama dalam jaringan, seperti menggunakan algoritma Ricart-Agrawala.



Gambar 43. Diagram penjadwalan terdistribusi dan sinkronisasi, termasuk algoritma Lamport dan Ricart-Agrawala.

J. Manajemen Replikasi dan Konsistensi Data

Replikasi data dalam sistem terdistribusi diperlukan untuk meningkatkan keandalan dan aksesibilitas data. Namun, ini juga menimbulkan tantangan dalam menjaga konsistensi data di seluruh node.

1. Teknik Replikasi Data
    a. **Full Replication**: Seluruh data direplikasi di semua node. Keuntungan dari teknik ini adalah keandalan yang tinggi, tetapi membutuhkan banyak ruang penyimpanan.
    b. **Partial Replication**: Hanya sebagian data yang direplikasi di node tertentu, mengurangi kebutuhan ruang penyimpanan dan bandwidth.
    c. **Dynamic Replication**: Jumlah dan lokasi replikasi dapat disesuaikan dengan kebutuhan, seperti saat permintaan terhadap data meningkat atau berkurang.
2. Konsistensi Data
    a. **Strong Consistency**: Memastikan bahwa semua replika selalu memiliki data yang sama. Dalam model ini, perubahan pada

satu replika langsung diterapkan pada semua replika lain, tetapi memerlukan overhead tinggi.

b. **Eventual Consistency**: Membiarkan setiap replika mencapai konsistensi setelah beberapa waktu. Cocok untuk aplikasi yang toleran terhadap data yang tidak selalu up-to-date, seperti dalam sistem P2P.

c. **Causal Consistency**: Perubahan data di satu node dapat dilihat oleh semua node dengan urutan yang sama, menjaga hubungan sebab-akibat antar data.

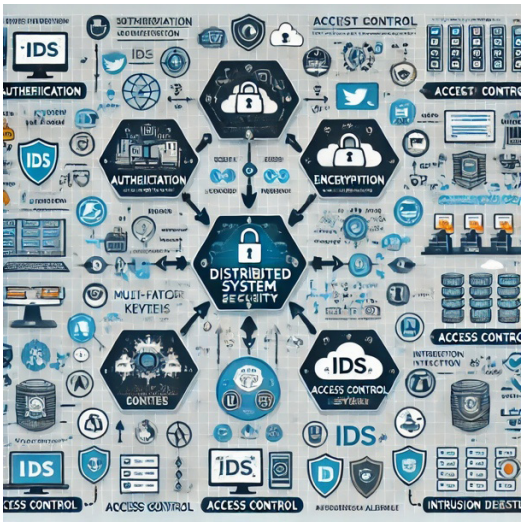
Gambar 44. Ilustrasi replikasi data (full, partial, dynamic) dan model konsistensi (strong, eventual, causal).

K. Keamanan dalam Sistem Terdistribusi

Sistem terdistribusi rentan terhadap serangan keamanan karena data dan proses tersebar di banyak perangkat dalam jaringan. Berikut adalah beberapa pendekatan keamanan dalam sistem terdistribusi:

1. Autentikasi Terdistribusi
   a. Mengidentifikasi pengguna yang sah di seluruh jaringan. Teknik seperti **Kerberos** digunakan untuk mengelola autentikasi di lingkungan terdistribusi melalui sistem tiket, yang memberikan otentikasi sekali dan dapat digunakan di seluruh jaringan.

2. Enkripsi Data
   Data dalam sistem terdistribusi sering kali melewati jaringan publik. Enkripsi data memastikan bahwa data hanya bisa dibaca oleh penerima yang sah. Protokol seperti **SSL/TLS** digunakan untuk mengenkripsi data selama transmisi.
3. Distributed Access Control
   Sistem terdistribusi memerlukan kontrol akses yang dapat diterapkan di seluruh node dalam jaringan. Pendekatan ini dapat menggunakan **Access Control Lists (ACLs)** yang diperluas di semua node, atau **Role-Based Access Control (RBAC)**, di mana akses diberikan berdasarkan peran pengguna.
4. Intrusion Detection System (IDS) Terdistribusi
   IDS terdistribusi dapat mendeteksi ancaman dan serangan dari luar, serta mengidentifikasi aktivitas yang mencurigakan di dalam jaringan. IDS terdistribusi dapat mengumpulkan log dan memantau jaringan secara keseluruhan untuk mendeteksi perilaku anomali.



Gambar 45. Diagram keamanan terdistribusi, termasuk autentikasi, enkripsi, akses kontrol, dan IDS untuk perlindungan dalam sistem terdistribusi.

Bab 9

VIRTUALISASI DAN CLOUD COMPUTING

Dalam beberapa dekade terakhir, dunia komputasi telah mengalami transformasi besar melalui perkembangan teknologi virtualisasi dan cloud computing. Virtualisasi memungkinkan satu perangkat keras fisik untuk menjalankan beberapa sistem operasi secara bersamaan, memberikan fleksibilitas, efisiensi, dan pemanfaatan sumber daya yang lebih optimal. Di sisi lain, cloud computing telah mengubah cara kita mengakses dan

menggunakan sumber daya komputasi, memungkinkan akses layanan yang elastis dan skalabel melalui internet. Kedua teknologi ini telah menjadi fondasi bagi banyak layanan modern, mulai dari penyimpanan data hingga aplikasi berbasis cloud, yang kini menjadi bagian integral dari kehidupan sehari-hari.

Virtualisasi dimulai dengan kebutuhan untuk memanfaatkan sumber daya perangkat keras yang lebih baik. Pada masa awal komputasi, perangkat keras sering kali digunakan secara tidak efisien karena hanya mampu menjalankan satu sistem operasi atau aplikasi pada satu waktu. Dengan munculnya teknologi virtualisasi, satu server fisik dapat dipecah menjadi beberapa virtual machines (VMs), masing-masing dengan sistem operasi dan aplikasi tersendiri. Virtualisasi memungkinkan penghematan biaya, meningkatkan fleksibilitas, dan mempermudah pengelolaan sistem, terutama dalam lingkungan server dan pusat data.

Cloud computing, yang dibangun di atas teknologi virtualisasi, membawa konsep ini lebih jauh dengan menyediakan akses ke sumber daya komputasi secara on-demand melalui jaringan. Pengguna dapat menyewa layanan seperti penyimpanan, daya komputasi, atau perangkat lunak tanpa harus membeli dan mengelola perangkat keras sendiri. Cloud computing menawarkan model layanan seperti Infrastructure as a Service (IaaS), Platform as a Service (PaaS), dan Software as a Service (SaaS), yang memungkinkan pengguna memilih layanan sesuai kebutuhan mereka. Teknologi ini telah mendorong adopsi aplikasi skala besar, meningkatkan efisiensi operasional, dan memungkinkan inovasi yang lebih cepat.

A. Pengertian Virtualisasi

Virtualisasi adalah teknologi yang memungkinkan pemisahan antara perangkat keras fisik dan sistem operasi, sehingga memungkinkan satu perangkat keras untuk menjalankan beberapa sistem operasi atau aplikasi secara bersamaan. Teknologi ini dicapai melalui perangkat lunak khusus yang disebut hypervisor, yang mengelola virtual machines dan

mengalokasikan sumber daya fisik secara dinamis. Ada dua jenis utama hypervisor:

1. Type 1 (Bare-Metal Hypervisor):
   Hypervisor yang berjalan langsung di atas perangkat keras tanpa sistem operasi tambahan. Contoh: VMware ESXi, Microsoft Hyper-V. Type 1 hypervisor banyak digunakan di pusat data karena performanya yang tinggi.
2. Type 2 (Hosted Hypervisor):
   Hypervisor yang berjalan di atas sistem operasi host. Contoh: VMware Workstation, Oracle VirtualBox. Type 2 hypervisor umumnya digunakan untuk pengujian dan pengembangan di lingkungan desktop.

Manfaat utama dari virtualisasi meliputi:

1. Penggunaan Sumber Daya yang Efisien: Memungkinkan pemanfaatan perangkat keras yang lebih baik dengan menjalankan beberapa VM pada satu server fisik.
2. Isolasi dan Keamanan: Setiap VM terisolasi, sehingga kegagalan atau serangan pada satu VM tidak memengaruhi VM lain.
3. Fleksibilitas dan Skalabilitas: Memungkinkan penyebaran aplikasi dengan cepat dan mudah, serta mendukung pengelolaan sumber daya yang lebih dinamis.

B. Pengertian Cloud Computing

Cloud computing adalah model komputasi yang menyediakan layanan komputasi (seperti penyimpanan, pemrosesan, jaringan) melalui internet. Cloud computing memungkinkan pengguna untuk mengakses sumber daya yang skalabel tanpa perlu mengelola infrastruktur fisik secara langsung. Model ini sering digambarkan sebagai layanan on-demand, di mana pengguna hanya membayar sesuai penggunaan, mirip dengan layanan utilitas seperti listrik atau air.

Cloud computing menawarkan beberapa model layanan utama:

1. Infrastructure as a Service (IaaS):
   Menyediakan infrastruktur dasar seperti server virtual, penyimpanan, dan jaringan. Pengguna memiliki kontrol penuh atas sistem operasi dan aplikasi yang dijalankan di atas infrastruktur ini. Contoh: Amazon EC2, Google Compute Engine.
2. Platform as a Service (PaaS):
   Menyediakan platform pengembangan yang mencakup sistem operasi, basis data, dan runtime aplikasi. PaaS memungkinkan pengembang untuk membangun, menguji, dan mengelola aplikasi tanpa mengkhawatirkan manajemen infrastruktur. Contoh: Google App Engine, Microsoft Azure App Services.
3. Software as a Service (SaaS):
   Menyediakan aplikasi perangkat lunak yang dapat diakses melalui internet tanpa perlu instalasi lokal. Pengguna hanya mengakses aplikasi melalui browser web, sementara penyedia cloud mengelola semua aspek teknis. Contoh: Google Workspace, Microsoft Office 365, Salesforce.

C. Manfaat Virtualisasi dan Cloud Computing

Virtualisasi dan cloud computing memberikan sejumlah manfaat yang signifikan bagi pengguna dan organisasi, antara lain:

1. Penghematan Biaya:
   Dengan virtualisasi, perusahaan dapat mengurangi jumlah perangkat keras yang diperlukan, menghemat biaya peralatan dan pemeliharaan. Cloud computing memungkinkan pengguna hanya membayar untuk layanan yang mereka gunakan, menghilangkan biaya investasi awal yang besar.
2. Skalabilitas dan Fleksibilitas:
   Cloud computing memungkinkan perusahaan untuk menambah atau mengurangi sumber daya dengan cepat sesuai kebutuhan.

Virtualisasi mempermudah penyebaran VM baru dalam hitungan menit, mendukung kelincahan bisnis yang tinggi.

3. Keandalan dan Ketersediaan:
   Cloud computing menawarkan redundansi data dan failover otomatis, yang meningkatkan keandalan dan ketersediaan layanan. Virtualisasi memungkinkan migrasi VM tanpa downtime (live migration), meminimalkan gangguan selama pemeliharaan.
4. Pengelolaan yang Mudah:
   Virtualisasi memungkinkan administrator untuk mengelola beberapa VM dari satu antarmuka, mempermudah monitoring, backup, dan pemeliharaan. Cloud computing menawarkan alat manajemen otomatis yang membantu organisasi mengelola sumber daya dengan efisien.

D. Tantangan dalam Virtualisasi dan Cloud Computing

Meskipun menawarkan banyak keuntungan, virtualisasi dan cloud computing juga menghadapi sejumlah tantangan:

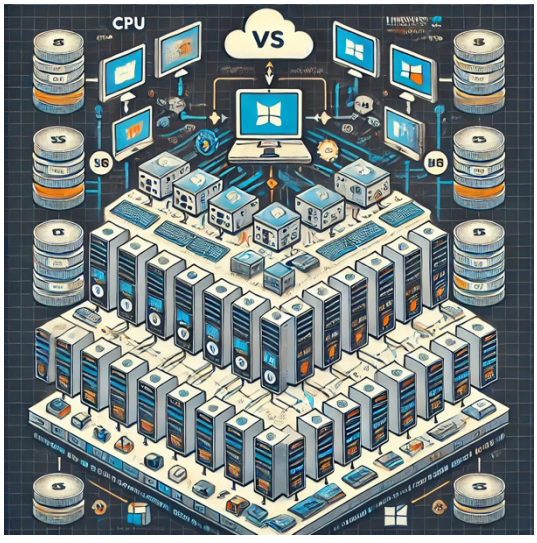
1. Keamanan dan Privasi:
   Dalam lingkungan cloud, data sering kali disimpan di server yang dikelola oleh pihak ketiga. Risiko akses tidak sah dan kebocoran data menjadi perhatian utama. Enkripsi data, otentikasi yang kuat, dan manajemen identitas adalah beberapa langkah untuk mengatasi masalah ini.
2. Kompleksitas Manajemen:
   Mengelola banyak VM dan layanan cloud memerlukan keahlian teknis yang tinggi. Sistem operasi harus menyediakan alat dan fitur yang mempermudah manajemen, monitoring, dan optimasi sumber daya.
3. Kinerja:
   Virtualisasi dapat menambah overhead kinerja karena perangkat keras dibagi di antara beberapa VM. Penggunaan cloud juga dapat menyebabkan latensi, terutama jika server berada jauh dari pengguna.

E. Konsep Dasar Virtualisasi

Virtualisasi adalah teknologi yang memungkinkan satu perangkat keras untuk menjalankan beberapa lingkungan komputasi yang terisolasi, disebut sebagai **virtual machines (VMs)**. Sistem operasi mendukung virtualisasi untuk meningkatkan efisiensi dan fleksibilitas dalam penggunaan sumber daya.

Manfaat Virtualisasi:

1. **Penggunaan Sumber Daya yang Efisien**: Beberapa VM dapat berjalan pada satu perangkat keras fisik, meningkatkan pemanfaatan CPU, memori, dan penyimpanan.
2. **Isolasi**: Setiap VM terisolasi sehingga aplikasi dan data pada satu VM tidak dapat memengaruhi VM lain.
3. **Portabilitas**: VM dapat dengan mudah dipindahkan dari satu host ke host lain, memungkinkan disaster recovery dan maintenance yang lebih fleksibel.



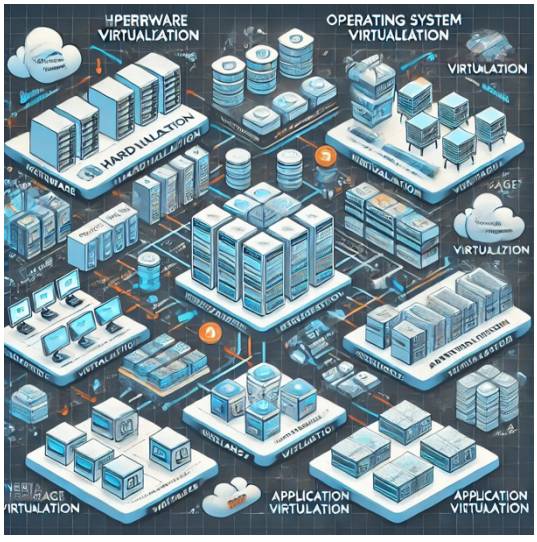
Gambar 46. Diagram yang menunjukkan satu perangkat keras menjalankan beberapa VM, masing-masing dengan OS dan aplikasi sendiri.

F. Jenis-jenis Virtualisasi

Ada beberapa jenis virtualisasi yang didukung oleh sistem operasi:

1. Virtualisasi Server
    a. Membagi satu server fisik menjadi beberapa server virtual, yang masing-masing dapat menjalankan OS dan aplikasi secara independen. Virtualisasi server adalah basis dari infrastruktur cloud.
    b. **Hypervisor**: Komponen utama dalam virtualisasi server. Hypervisor adalah perangkat lunak yang mengelola VM pada perangkat keras fisik dan memungkinkan beberapa OS berjalan di satu perangkat. Hypervisor terdiri dari:
        1) **Type 1 (Bare-Metal Hypervisor)**: Hypervisor yang berjalan langsung di atas perangkat keras, misalnya, VMware ESXi, Microsoft Hyper-V.
        2) **Type 2 (Hosted Hypervisor)**: Hypervisor yang berjalan di atas OS, seperti VirtualBox atau VMware Workstation.
2. Virtualisasi Desktop
    a. Mengizinkan desktop virtual yang dapat diakses dari mana saja. Virtualisasi desktop memungkinkan pengguna mengakses sistem operasi mereka dari jarak jauh, melalui perangkat yang berbeda.
    b. **Contoh Penggunaan**: Desktop as a Service (DaaS), di mana desktop pengguna di-host di server cloud.
3. **Virtualisasi Jaringan**
    a. Memisahkan perangkat keras jaringan dari fungsionalitas perangkat lunaknya, memungkinkan beberapa jaringan virtual berjalan di atas satu infrastruktur fisik.
    b. **Contoh Penggunaan**: Software-Defined Networking (SDN) yang memungkinkan jaringan dikelola melalui perangkat lunak, mengurangi kebutuhan perangkat keras fisik yang banyak.

4. **Virtualisasi Penyimpanan**
    a. Menggabungkan beberapa perangkat penyimpanan fisik menjadi satu perangkat penyimpanan virtual yang terlihat sebagai satu entitas.
    b. **Contoh Penggunaan**: Storage Area Network (SAN) atau Network-Attached Storage (NAS) yang memungkinkan banyak server untuk berbagi penyimpanan secara terpusat.



Gambar 47. Diagram tipe-tipe virtualisasi

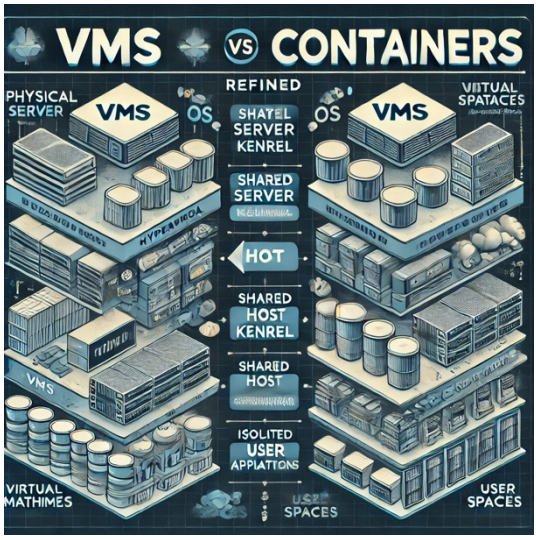
G. Containerization

Containerization adalah teknologi virtualisasi pada level aplikasi yang memungkinkan aplikasi dan semua dependensinya dikemas dalam satu paket yang ringan, disebut **container**. Tidak seperti VM, container berbagi kernel OS yang sama tetapi tetap terisolasi satu sama lain.

1. Manfaat Containerization
    a. **Ringan dan Cepat**: Container menggunakan lebih sedikit sumber daya dibandingkan VM karena berbagi kernel OS.

b. **Portabilitas**: Aplikasi yang dikemas dalam container dapat berjalan di berbagai lingkungan (on-premises, cloud) tanpa perubahan.
c. **Isolasi**: Meskipun berbagi kernel yang sama, container terisolasi dari container lain, memungkinkan aplikasi berjalan tanpa konflik.

2. Docker dan Kubernetes
   a. **Docker**: Platform containerization populer yang memungkinkan pembuatan, deployment, dan pengelolaan container secara mudah.
   b. **Kubernetes**: Platform orkestrasi container yang mengelola scaling, load balancing, dan penempatan container pada kluster, sangat cocok untuk aplikasi terdistribusi di cloud.



Gambar 48. Diagram perbandingan VM dan container, menunjukkan bagaimana container berbagi kernel OS sementara VM memiliki OS sendiri.

H. Cloud Computing

Cloud computing adalah model komputasi yang menyediakan sumber daya seperti CPU, penyimpanan, dan aplikasi melalui internet. Cloud

computing memungkinkan pengguna untuk mengakses dan menggunakan sumber daya tanpa harus mengelola infrastruktur fisik.

Model Layanan Cloud:

1. Infrastructure as a Service (IaaS):
    a. Menyediakan infrastruktur dasar seperti server, penyimpanan, dan jaringan. Pengguna memiliki kontrol penuh atas sistem operasi dan aplikasi yang berjalan di atas infrastruktur ini.
    b. **Contoh**: Amazon EC2, Google Compute Engine, Microsoft Azure Virtual Machines.
2. Platform as a Service (PaaS):
    a. Menyediakan platform bagi pengembang untuk membuat, menguji, dan mengelola aplikasi tanpa harus mengelola infrastruktur. PaaS mencakup sistem operasi, basis data, dan runtime aplikasi.
    b. **Contoh**: Google App Engine, Microsoft Azure App Services, Heroku.
3. Software as a Service (SaaS):
    a. Menyediakan aplikasi lengkap yang dapat diakses melalui internet. Pengguna hanya menggunakan aplikasi tanpa perlu memikirkan infrastruktur atau platform di baliknya.
    b. **Contoh**: Google Workspace, Microsoft Office 365, Salesforce.

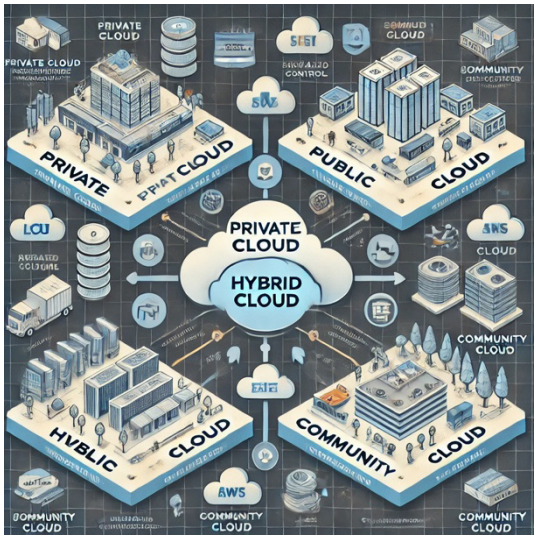
Gambar 49. Diagram model layanan cloud (IaaS, PaaS, SaaS) yang menunjukkan perbedaan level kontrol pengguna pada infrastruktur, platform, dan aplikasi

I. Model Implementasi Cloud

Ada beberapa model implementasi cloud yang umum digunakan, tergantung pada kebutuhan dan kontrol atas data serta aplikasi:

1. Private Cloud
    a. Cloud yang dioperasikan oleh dan untuk satu organisasi. Private cloud memberikan kontrol penuh atas data dan keamanan, cocok untuk organisasi yang memiliki persyaratan kepatuhan yang ketat.
    b. **Contoh**: Infrastruktur cloud di dalam perusahaan besar yang dioperasikan secara internal.
2. Public Cloud
    a. Cloud yang dikelola oleh penyedia layanan dan tersedia untuk banyak pengguna. Public cloud menawarkan skala besar dengan biaya rendah karena berbagi infrastruktur.
    b. **Contoh**: Amazon Web Services (AWS), Google Cloud Platform, Microsoft Azure.

3. Hybrid Cloud
    a. Kombinasi dari private dan public cloud, memungkinkan data atau aplikasi untuk berpindah antara private dan public cloud sesuai kebutuhan.
    b. **Contoh Penggunaan**: Perusahaan menggunakan public cloud untuk beban kerja non-sensitif tetapi memanfaatkan private cloud untuk data sensitif.
4. Community Cloud
    a. Cloud yang dibagikan oleh beberapa organisasi yang memiliki kebutuhan yang sama. Community cloud memungkinkan kolaborasi antar organisasi dengan skema keamanan yang konsisten.
    b. **Contoh Penggunaan**: Cloud yang digunakan oleh organisasi pemerintah untuk mengelola data bersama.

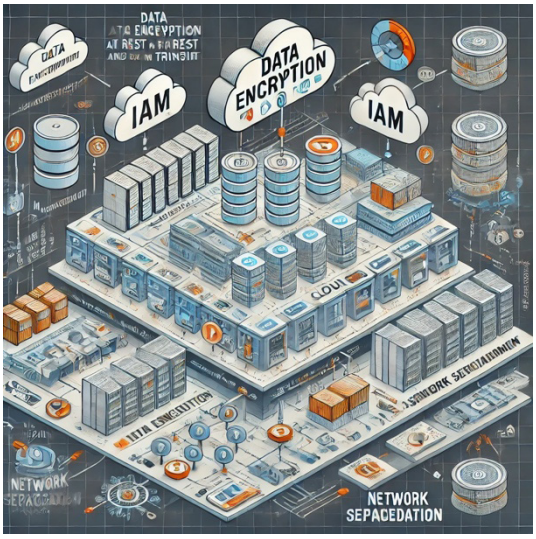


Gambar 50. Diagram model implementasi cloud (private, public, hybrid, community) dengan contoh kasus penggunaan masing-masing.

J. Keamanan dalam Virtualisasi dan Cloud Computing

Keamanan merupakan komponen penting dalam virtualisasi dan cloud computing, terutama karena data dan aplikasi mungkin tersebar di berbagai lingkungan dan dikelola oleh penyedia layanan pihak ketiga.

1. Keamanan Virtualisasi
    a. **Isolation of VMs and Containers**: Mencegah VM atau container saling memengaruhi satu sama lain. Hypervisor harus menjaga isolasi yang ketat untuk menghindari kebocoran data antar VM.
    b. **Hypervisor Security**: Hypervisor harus dipastikan aman dari eksploitasi karena jika terkompromikan, seluruh VM yang berjalan di atasnya bisa terkena dampak.
2. Keamanan Data di Cloud
    a. **Enkripsi Data**: Data yang disimpan di cloud perlu dienkripsi baik saat di-at-rest (disimpan) maupun in-transit (dalam perjalanan).
    b. **Identity and Access Management (IAM)**: Sistem autentikasi dan otorisasi untuk mengelola siapa yang memiliki akses ke data atau aplikasi di cloud.
    c. **Compliance and Regulatory Requirements**: Banyak organisasi harus memenuhi persyaratan kepatuhan, seperti GDPR, HIPAA, atau PCI-DSS, dalam pengelolaan data di cloud.
3. Keamanan Jaringan di Cloud
    a. **Firewall dan Network Segmentation**: Firewall melindungi cloud dari akses jaringan yang tidak sah, sementara segmentasi jaringan mengisolasi bagian-bagian dari cloud untuk keamanan yang lebih baik.
    b. **Distributed Denial of Service (DDoS) Protection**: Sistem yang memitigasi serangan DDoS, yang dapat mengganggu akses pengguna ke layanan cloud.

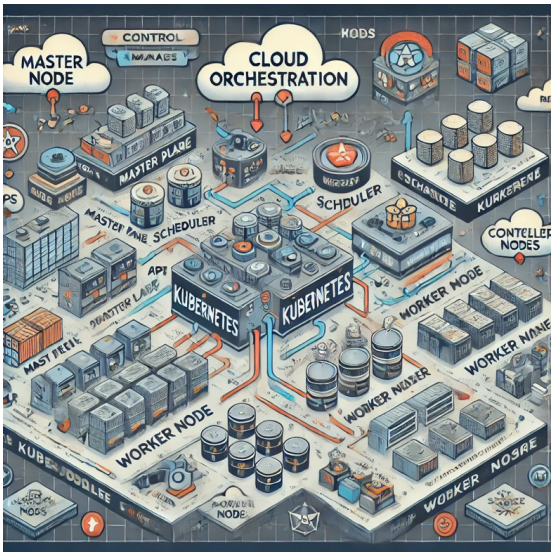


Gambar 51. Diagram keamanan dalam virtualisasi dan cloud, termasuk enkripsi data, IAM, dan segmentasi jaringan.

K. Orkestrasi dan Manajemen Sumber Daya dalam Cloud

Orkestrasi cloud memungkinkan manajemen otomatis dari sumber daya, seperti VM, container, dan jaringan, untuk meningkatkan efisiensi dan skalabilitas.

1. **Kubernetes**: Platform orkestrasi yang mengelola container dalam skala besar, memungkinkan deployment, scaling, dan monitoring aplikasi terdistribusi.
2. **OpenStack**: Sistem open-source untuk mengelola cloud pribadi, menyediakan layanan untuk komputasi, penyimpanan, dan jaringan.
3. **Autoscaling**: Kemampuan untuk menambah atau mengurangi sumber daya berdasarkan beban kerja, memastikan penggunaan sumber daya yang efisien.



Gambar 52. Diagram orkestrasi cloud menggunakan Kubernetes

Bab 10

TOPIK LANJUTAN DALAM SISTEM OPERASI

A. Sistem Operasi untuk Internet of Things (IoT)

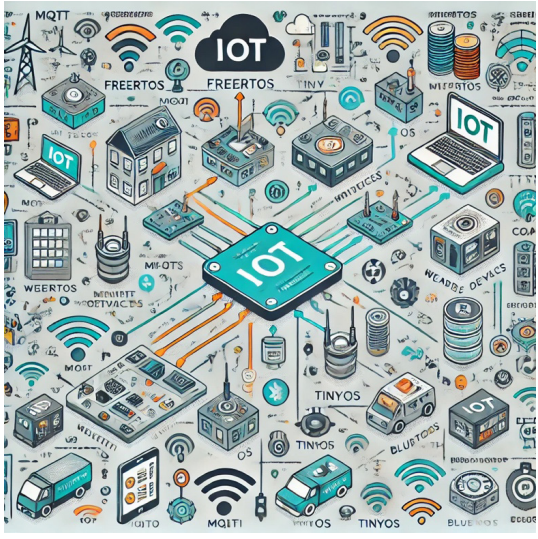
Internet of Things (IoT) menghubungkan perangkat fisik seperti sensor, alat rumah tangga, dan mesin industri ke internet, memungkinkan mereka saling berkomunikasi. Sistem operasi untuk perangkat IoT dirancang khusus agar sesuai dengan keterbatasan sumber daya dan kebutuhan jaringan yang rendah.

Karakteristik Sistem Operasi IoT:

1. **Ringan dan Efisien**: Perangkat IoT umumnya memiliki sumber daya yang terbatas, seperti prosesor berdaya rendah dan memori kecil. OS IoT harus ringan agar dapat berjalan pada perangkat ini.
2. **Jaringan yang Terintegrasi**: OS IoT harus mendukung protokol jaringan, seperti MQTT, CoAP, dan HTTP, yang memungkinkan komunikasi antar perangkat.
3. **Manajemen Daya**: OS IoT harus dapat menghemat daya untuk memperpanjang umur baterai perangkat, misalnya dengan sleep mode dan pengelolaan konsumsi energi.

Contoh Sistem Operasi IoT:

1. **RIOT OS**: OS open-source yang dirancang khusus untuk IoT, dengan dukungan pada perangkat berdaya rendah.
2. **Contiki OS**: OS dengan fitur protokol jaringan IPv6 dan IPv4 yang cocok untuk perangkat IoT.
3. **FreeRTOS**: OS ringan yang mendukung real-time dan digunakan dalam perangkat IoT dengan kontrol waktu yang presisi.



Gambar 53. Ilustrasi perangkat IoT yang menjalankan OS ringan, menunjukkan komunikasi antar perangkat IoT.

B. Sistem Operasi pada Perangkat Mobile

Perangkat mobile memiliki kebutuhan khusus yang berbeda dari komputer tradisional. OS mobile harus mampu mengelola antarmuka sentuh, manajemen energi, dan konektivitas jaringan.

Karakteristik Sistem Operasi Mobile:

1. **Antarmuka Pengguna yang Responsif**: OS mobile harus mendukung pengalaman pengguna yang intuitif dengan antarmuka berbasis sentuhan.
2. **Manajemen Daya**: Perangkat mobile mengandalkan baterai, sehingga OS harus mengoptimalkan konsumsi daya melalui pengelolaan aplikasi latar belakang dan mode hemat daya.
3. **Keamanan Aplikasi**: OS mobile harus memberikan keamanan yang ketat untuk mengelola izin aplikasi dan melindungi data pengguna dari akses yang tidak sah.

Contoh Sistem Operasi Mobile:

1. **Android**: OS berbasis Linux yang dikembangkan oleh Google, mendukung berbagai perangkat dan memiliki ekosistem aplikasi yang luas.
2. **iOS**: OS tertutup yang dikembangkan oleh Apple untuk perangkat seperti iPhone dan iPad, dengan fitur keamanan dan privasi yang ketat.
3. **HarmonyOS**: Dikembangkan oleh Huawei untuk perangkat IoT dan mobile, dengan dukungan untuk lingkungan yang terhubung.

C. Sistem Operasi Real-Time (RTOS)

Sistem operasi real-time (RTOS) dirancang untuk menangani aplikasi yang membutuhkan respon dalam waktu yang ketat, seperti sistem kendali industri, alat medis, dan sistem otomotif. RTOS memastikan bahwa setiap proses dijalankan tepat waktu dan sesuai dengan prioritas.

Karakteristik RTOS:

1. **Determinisme**: RTOS harus memberikan respon dalam waktu yang sudah ditentukan, baik untuk proses yang sudah dijadwalkan maupun untuk tugas mendadak.
2. **Prioritas Tinggi**: RTOS menggunakan skema penjadwalan berbasis prioritas, memastikan bahwa proses penting mendapat prioritas tertinggi.
3. **Ringan dan Efisien**: Seperti OS IoT, RTOS sering digunakan pada perangkat dengan sumber daya terbatas.

Tipe-tipe RTOS:

1. **Hard Real-Time Systems**: Sistem di mana setiap tugas harus selesai dalam batas waktu tertentu atau dapat menimbulkan kegagalan, misalnya pada alat kontrol penerbangan.
2. **Soft Real-Time Systems**: Sistem yang tetap dapat berfungsi jika beberapa tugas tidak selesai tepat waktu, tetapi performa akan berkurang. Contohnya adalah streaming video.

Contoh RTOS:

1. **FreeRTOS**: Sistem operasi open-source yang digunakan di berbagai perangkat embedded, dengan dukungan real-time.
2. **QNX**: RTOS komersial yang digunakan pada aplikasi industri dan otomotif, terkenal karena ketahanannya terhadap kesalahan.
3. **VxWorks**: RTOS yang sering digunakan dalam aplikasi aerospace dan pertahanan, menawarkan keandalan dan fleksibilitas yang tinggi.

D. Manajemen Energi dalam Sistem Operasi

Manajemen energi adalah aspek penting dalam OS untuk perangkat portabel, IoT, dan perangkat lain yang bergantung pada baterai. OS harus mampu mengurangi konsumsi daya sambil tetap memberikan performa yang optimal.

Teknik Manajemen Energi dalam OS:

1. **Dynamic Voltage and Frequency Scaling (DVFS)**: Teknik yang menyesuaikan tegangan dan frekuensi CPU sesuai beban kerja, menghemat daya saat beban rendah.
2. **Power States (Sleep, Hibernate, Suspend)**: OS menggunakan berbagai power states untuk menghemat energi saat perangkat tidak aktif. Mode sleep menghemat daya dengan menjaga status aplikasi di memori.
3. **Pengelolaan Aplikasi Latar Belakang**: Pada OS mobile, pengelolaan aplikasi latar belakang sangat penting untuk mengurangi konsumsi daya. Aplikasi di latar belakang diatur untuk tidak menghabiskan banyak sumber daya.

E. Sistem Operasi Tertanam (Embedded OS)

Sistem operasi tertanam adalah OS yang dirancang untuk perangkat khusus yang menjalankan fungsi tertentu. Perangkat embedded memiliki keterbatasan dalam hal memori, prosesor, dan sumber daya lainnya, sehingga OS tertanam harus sangat efisien.

Karakteristik Sistem Operasi Tertanam:

1. **Ringan dan Cepat**: OS harus dapat dijalankan pada perangkat dengan sumber daya terbatas.
2. **Real-Time Support**: Sebagian besar perangkat embedded memerlukan RTOS agar dapat merespons input secara cepat dan tepat waktu.
3. **Reliability (Keandalan)**: Perangkat embedded seringkali menjalankan tugas yang kritis, seperti kendali mesin atau perangkat medis. OS harus sangat andal dan tahan terhadap kegagalan.

Contoh Embedded OS:

1. **FreeRTOS**: Digunakan pada perangkat yang memerlukan real-time dengan konsumsi daya rendah.
2. **TinyOS**: OS yang didesain untuk perangkat jaringan sensor, seperti sensor pada perangkat IoT.

3. **Embedded Linux**: Versi ringan dari Linux yang diadaptasi untuk perangkat embedded, banyak digunakan dalam router, sistem kontrol industri, dan perangkat rumah pintar.

F. Sistem Operasi di Komputasi Edge

Komputasi edge adalah model di mana data diproses di dekat sumbernya, yaitu pada perangkat edge, sebelum dikirim ke pusat data atau cloud. OS pada perangkat edge harus mendukung pemrosesan lokal dan komunikasi cepat dengan perangkat lain.

Karakteristik OS untuk Komputasi Edge:

1. **Latency Rendah**: Pemrosesan data secara lokal memungkinkan respon cepat untuk aplikasi yang sensitif terhadap waktu, seperti analisis video atau AI.
2. **Konektivitas**: OS pada perangkat edge harus mendukung berbagai protokol komunikasi untuk interaksi antar perangkat dan server.
3. **Keamanan**: Mengingat perangkat edge sering diimplementasikan di lapangan, keamanan menjadi penting untuk melindungi data yang dikumpulkan dan mencegah akses tidak sah.

Contoh OS Edge:

1. **EdgeX Foundry**: Platform open-source yang menyediakan framework untuk edge computing.
2. **Ubuntu Core**: OS berbasis Linux yang didesain untuk perangkat edge dan IoT, dengan fitur keamanan yang kuat.
3. **AWS IoT Greengrass**: OS yang memungkinkan perangkat IoT untuk menjalankan AWS Lambda, mengelola perangkat, dan berkomunikasi dengan cloud.

G. Tantangan Masa Depan dalam Sistem Operasi

Sistem operasi terus berkembang mengikuti kemajuan teknologi dan kebutuhan baru. Beberapa tantangan masa depan dalam pengembangan OS meliputi:

1. **Integrasi AI dan Machine Learning**: Sistem operasi masa depan mungkin akan mengintegrasikan kecerdasan buatan untuk otomatisasi, optimasi sumber daya, dan peningkatan keamanan.
2. **Quantum Computing**: Dengan hadirnya komputasi kuantum, OS perlu beradaptasi untuk mendukung algoritma dan arsitektur kuantum, yang sangat berbeda dari komputasi tradisional.
3. **Keamanan yang Lebih Kuat**: Dalam era yang penuh dengan ancaman keamanan siber, OS harus dilengkapi dengan sistem keamanan yang lebih kompleks, seperti otentikasi berbasis biometrik, deteksi intrusi berbasis AI, dan enkripsi yang lebih kuat.
4. **Adaptasi untuk Komputasi Terdesentralisasi**: Dengan kemunculan teknologi blockchain dan peer-to-peer, OS akan diharapkan untuk mendukung ekosistem terdesentralisasi yang memungkinkan kolaborasi tanpa server pusat.

Bab 11

STUDI KASUS SISTEM OPERASI POPULER

Sistem operasi adalah perangkat lunak inti yang menjadi penghubung antara perangkat keras dan aplikasi, memungkinkan pengguna untuk berinteraksi dengan komputer secara efisien. Seiring dengan perkembangan teknologi, sistem operasi telah berkembang menjadi lebih kompleks, mencakup berbagai fitur dan kemampuan yang dirancang

untuk memenuhi kebutuhan yang beragam, mulai dari komputer pribadi hingga server cloud, perangkat mobile, dan embedded systems. Pada bab ini, kita akan melakukan studi kasus terhadap beberapa sistem operasi populer yang telah memainkan peran penting dalam dunia komputasi modern, seperti Windows, Linux, macOS, Android, dan iOS.

Setiap sistem operasi memiliki sejarah, arsitektur, dan fitur unik yang mencerminkan filosofi desain serta tujuan penggunaannya. Misalnya, Windows dikenal dengan antarmuka grafis yang ramah pengguna dan dukungan yang luas untuk perangkat keras serta perangkat lunak komersial. Linux, di sisi lain, menawarkan fleksibilitas dan kustomisasi tinggi sebagai sistem operasi open-source, menjadikannya pilihan utama bagi server, superkomputer, dan pengembangan aplikasi. macOS, dengan desain yang elegan dan integrasi kuat dengan ekosistem Apple, difokuskan pada pengalaman pengguna dan keamanan yang solid. Di sisi lain, Android dan iOS mendominasi pasar perangkat mobile dengan pendekatan yang berbeda dalam hal fleksibilitas, keamanan, dan ekosistem aplikasi.

A. Tujuan Studi Kasus

Studi kasus sistem operasi populer bertujuan untuk memberikan pemahaman mendalam tentang bagaimana masing-masing OS dirancang, diimplementasikan, dan dioptimalkan untuk memenuhi kebutuhan pengguna yang berbeda. Dengan mempelajari beberapa contoh sistem operasi yang sukses, kita dapat melihat bagaimana berbagai fitur dan teknik pengelolaan sumber daya diterapkan dalam konteks nyata. Selain itu, memahami kekuatan dan kelemahan dari setiap OS akan membantu mahasiswa mengembangkan wawasan kritis dalam memilih atau mengembangkan sistem operasi yang sesuai untuk proyek dan aplikasi tertentu.

Studi kasus ini akan mencakup aspek-aspek berikut:

1. Sejarah dan Evolusi: Menjelajahi latar belakang pengembangan sistem operasi, peristiwa penting, dan versi-versi utama yang telah dirilis.

2. Arsitektur dan Desain: Menggambarkan komponen inti, seperti kernel, sistem file, dan subsistem yang membentuk dasar dari setiap OS.
3. Fitur dan Fungsionalitas: Menyoroti fitur unggulan dan inovasi yang ditawarkan oleh setiap sistem operasi.
4. Penggunaan dan Penerapan: Menjelaskan konteks penggunaan OS, mulai dari desktop dan server hingga perangkat mobile dan embedded systems.
5. Keunggulan dan Kelemahan: Menilai kekuatan dan kelemahan dari setiap sistem operasi berdasarkan kriteria seperti keamanan, kinerja, stabilitas, dan kustomisasi.

B. Signifikansi dan Peran Sistem Operasi Populer

Sistem operasi populer seperti Windows, Linux, macOS, Android, dan iOS tidak hanya memimpin pasar dalam kategori mereka masing-masing, tetapi juga mempengaruhi bagaimana perangkat keras dan perangkat lunak berkembang. Windows, misalnya, telah menjadi standar de facto untuk komputer pribadi, menawarkan antarmuka grafis yang user-friendly dan dukungan ekosistem aplikasi yang luas. Linux, dengan sifat open-source dan fleksibilitas tinggi, telah menjadi fondasi dari sebagian besar server, superkomputer, dan perangkat IoT. macOS menawarkan integrasi yang mulus dengan produk Apple lainnya dan memberikan pengalaman pengguna premium yang dioptimalkan untuk performa dan keamanan. Di dunia mobile, Android memberikan kebebasan kustomisasi kepada produsen dan pengguna, sementara iOS dikenal dengan keamanan yang ketat dan kontrol kualitas aplikasi yang tinggi.

Setiap sistem operasi ini telah berkembang dan beradaptasi dengan perubahan teknologi serta kebutuhan pasar yang dinamis. Kemajuan dalam virtualisasi, cloud computing, serta Internet of Things (IoT) juga mendorong sistem operasi untuk terus berinovasi. Melalui studi kasus ini, kita dapat mempelajari bagaimana sistem operasi beradaptasi terhadap tren dan tantangan baru, serta bagaimana desain dan fitur mereka terus berevolusi untuk memenuhi ekspektasi pengguna yang terus berubah.

C. Pendekatan dalam Studi Kasus

Pendekatan dalam studi kasus ini akan dimulai dengan menggambarkan karakteristik unik dari setiap sistem operasi, diikuti dengan analisis arsitektur dan desainnya. Kita akan membahas bagaimana OS mengelola sumber daya seperti CPU, memori, dan perangkat I/O, serta bagaimana mereka menangani keamanan dan proteksi data. Bab ini juga akan mengeksplorasi bagaimana sistem operasi diterapkan dalam berbagai konteks, seperti desktop, server, mobile, dan cloud computing.

Selain itu, kita akan membandingkan fitur dan kinerja dari masing-masing sistem operasi dalam berbagai kategori, seperti:

1. Manajemen Proses: Bagaimana setiap OS mengelola proses dan penjadwalan, serta mendukung multitasking dan multithreading.
2. Manajemen Memori: Teknik yang digunakan untuk mengalokasikan, mengoptimalkan, dan melindungi memori.
3. Sistem File dan Penyimpanan: Struktur dan teknik sistem file yang digunakan, serta dukungan untuk perangkat penyimpanan modern.
4. Keamanan dan Proteksi: Mekanisme keamanan yang diterapkan, seperti kontrol akses, enkripsi, dan fitur proteksi terhadap malware.

D. Linux

Linux adalah sistem operasi open-source berbasis Unix yang sangat populer untuk server, superkomputer, dan perangkat embedded. Linux memiliki banyak distribusi (distro) yang dapat disesuaikan sesuai kebutuhan, seperti Ubuntu, Fedora, Debian, dan CentOS.

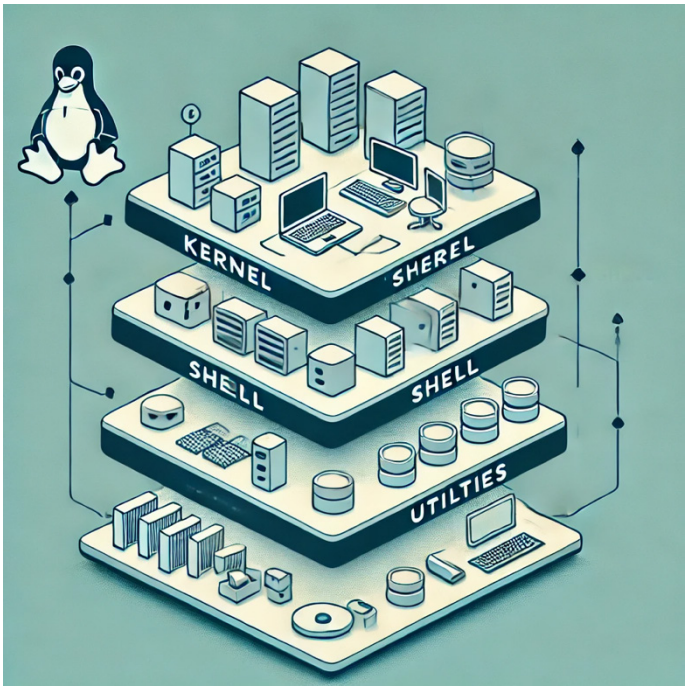
Karakteristik Linux:

1. **Open-Source dan Bebas**: Linux dikembangkan dengan lisensi open-source, sehingga kode sumbernya tersedia secara bebas untuk dimodifikasi dan digunakan oleh siapa saja.
2. **Stabil dan Aman**: Linux dikenal stabil dan jarang mengalami crash, sehingga cocok untuk server dan lingkungan kritis.
3. **Kustomisasi Tinggi**: Pengguna dapat menyesuaikan hampir setiap aspek dari Linux, dari kernel hingga antarmuka pengguna.

4. **Multi-User dan Multi-Tasking**: Linux mendukung banyak pengguna dan dapat menjalankan beberapa proses secara bersamaan.

Arsitektur Linux:

1. **Kernel**: Inti dari Linux, yang mengelola proses, memori, dan perangkat I/O.
2. **Shell**: Antarmuka command-line yang memungkinkan pengguna untuk menjalankan perintah dan skrip.
3. **System Libraries dan Utilities**: Menyediakan fungsi dasar yang dapat digunakan oleh aplikasi dan script.



Gambar 54. Diagram arsitektur Linux yang menunjukkan kernel, shell, dan utilities

Contoh Penggunaan:

1. **Server**: Banyak web server dan cloud platform berjalan di Linux karena stabilitasnya.

2. **Embedded Systems**: Versi ringan dari Linux sering digunakan dalam perangkat embedded seperti router, TV pintar, dan sistem otomotif.
3. **Pengembangan Software**: Linux adalah OS favorit untuk pengembang software karena mendukung berbagai alat pemrograman.

E. Windows

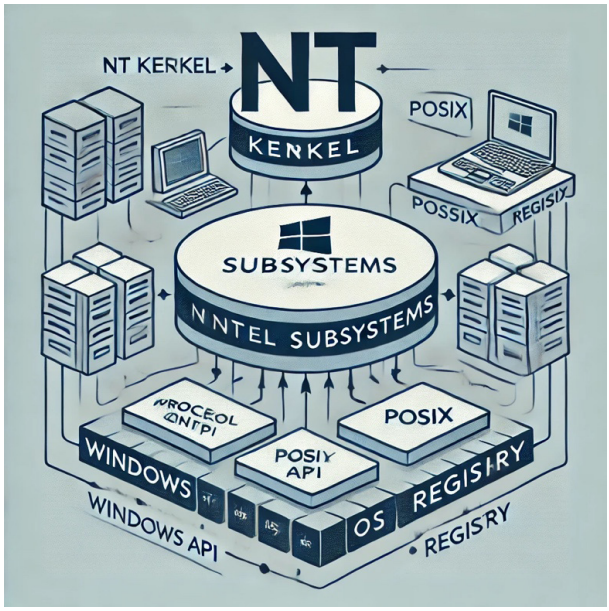
Windows adalah sistem operasi komersial yang dikembangkan oleh Microsoft dan paling banyak digunakan pada komputer pribadi. Windows terkenal dengan antarmuka grafisnya yang ramah pengguna dan dukungan perangkat lunak yang luas.

Karakteristik Windows:

1. **Antarmuka Grafis (GUI)**: Windows dirancang untuk memberikan pengalaman visual yang intuitif, dengan menu Start, taskbar, dan pengelolaan jendela.
2. **Kompatibilitas Aplikasi**: Windows mendukung berbagai aplikasi dan perangkat lunak komersial, dari aplikasi produktivitas hingga perangkat lunak gaming.
3. **Fokus pada Pengguna Perorangan**: Windows banyak digunakan di kalangan konsumen individu dan lingkungan bisnis.
4. **Pembaruan Berkala**: Microsoft merilis update reguler untuk keamanan dan peningkatan fitur.

Arsitektur Windows:

1. **Kernel NT**: Basis dari Windows, yang mengelola proses, memori, dan keamanan.
2. **Subsystem**: Menyediakan lingkungan komputasi yang berbeda, termasuk untuk aplikasi 32-bit, 64-bit, dan dukungan untuk aplikasi lama.
3. **Registry**: Basis data konfigurasi sistem Windows, menyimpan informasi tentang pengaturan perangkat keras, perangkat lunak, dan pengguna.



Gambar 55. Diagram arsitektur Windows yang menunjukkan kernel NT, subsistem, dan registry.

Contoh Penggunaan:

1. **Komputer Pribadi (PC)**: Windows sangat populer di kalangan pengguna PC di rumah dan kantor.
2. **Gaming**: Windows adalah platform yang banyak didukung oleh pengembang game dan memiliki ekosistem gaming yang kuat.
3. **Lingkungan Perusahaan**: Banyak perusahaan menggunakan Windows karena integrasinya dengan Microsoft Office dan alat produktivitas lainnya.

F. macOS

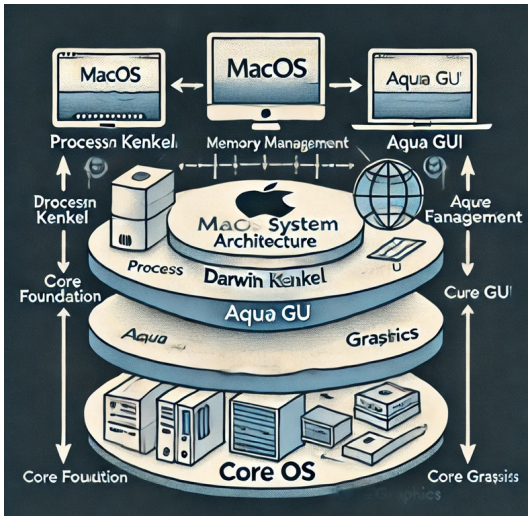
macOS adalah sistem operasi berbasis Unix yang dikembangkan oleh Apple untuk digunakan pada perangkat Macintosh (Mac). macOS dikenal karena desain antarmuka yang elegan dan pengalaman pengguna yang terintegrasi dengan baik dengan ekosistem Apple.

Karakteristik macOS:

1. **Antarmuka yang Bersih dan Intuitif**: macOS memiliki desain antarmuka grafis yang estetis dan intuitif, serta mudah digunakan.
2. **Keamanan yang Tinggi**: macOS dilengkapi dengan berbagai fitur keamanan, termasuk Gatekeeper dan XProtect, yang melindungi sistem dari malware.
3. **Integrasi dengan Ekosistem Apple**: macOS terintegrasi dengan iCloud, iPhone, iPad, dan perangkat Apple lainnya untuk kemudahan transfer data dan sinkronisasi.
4. **Stabilitas dan Performa**: macOS dikenal stabil dan efisien dalam pengelolaan memori dan sumber daya.

Arsitektur macOS:

1. **Darwin Kernel**: Kernel berbasis Unix yang mengelola proses, memori, dan perangkat keras.
2. **Aqua GUI**: Antarmuka grafis berbasis GUI yang menyediakan tampilan visual elegan dan animasi halus.
3. **Core OS dan Framework**: Lapisan sistem yang menyediakan fungsi dasar dan framework untuk aplikasi.



Gambar 56. Diagram arsitektur macOS, menunjukkan Darwin kernel, Aqua GUI, dan Core OS.

Contoh Penggunaan:

1. **Desain dan Kreativitas**: macOS banyak digunakan dalam industri kreatif untuk desain grafis, produksi audio, dan pengeditan video.
2. **Pengembangan iOS dan macOS**: Pengembang aplikasi Apple menggunakan macOS untuk mengembangkan aplikasi iOS dan macOS.
3. **Pengguna Profesional**: Banyak profesional memilih macOS untuk kestabilan dan fitur keamanan yang terintegrasi.

G. Android

Android adalah sistem operasi mobile open-source yang dikembangkan oleh Google. Android mendominasi pasar smartphone global dan dikenal dengan fleksibilitasnya yang tinggi serta kompatibilitas dengan berbagai perangkat.

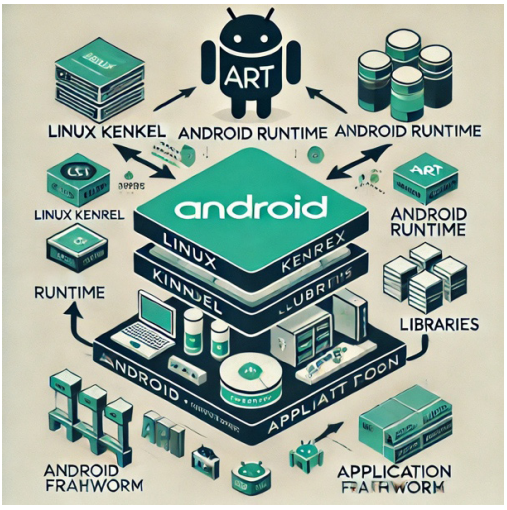
Karakteristik Android:

1. **Berbasis Open Source**: Android dikembangkan menggunakan lisensi open-source, memungkinkan produsen untuk memodifikasi dan mengadaptasi OS sesuai kebutuhan.
2. **Ekosistem Aplikasi yang Luas**: Android mendukung jutaan aplikasi melalui Google Play Store, menawarkan banyak pilihan aplikasi untuk pengguna.
3. **Kustomisasi Tinggi**: Android memungkinkan pengguna untuk menyesuaikan antarmuka, widget, dan fungsi lainnya.
4. **Dukungan Berbagai Perangkat**: Android digunakan pada berbagai perangkat, mulai dari smartphone dan tablet hingga perangkat IoT.

Arsitektur Android:

1. **Linux Kernel**: Kernel Linux digunakan sebagai basis untuk manajemen proses, memori, dan keamanan.
2. **Libraries**: Layer yang menyediakan fungsi dasar, seperti SQLite untuk database dan OpenGL untuk grafis.
3. **Android Runtime (ART)**: Menjalankan aplikasi Android dalam lingkungan runtime yang efisien.

4. **Application Framework**: Menyediakan API untuk pengembangan aplikasi, termasuk notifikasi, layanan lokasi, dan manajemen UI.



Gambar 57. Diagram arsitektur Android yang menunjukkan Linux kernel, Android Runtime, libraries, dan application framework.

Contoh Penggunaan:

1. **Smartphone dan Tablet**: Android adalah OS utama di sebagian besar perangkat mobile global.
2. **Smart TV**: Android TV menawarkan pengalaman TV pintar dengan dukungan aplikasi dan game.
3. **Perangkat IoT**: Android Things digunakan pada perangkat IoT, seperti perangkat smart home dan sensor.

H. iOS

iOS adalah sistem operasi mobile yang dikembangkan oleh Apple untuk perangkat iPhone dan iPad. iOS terkenal dengan keamanan, performa tinggi, dan integrasi dengan ekosistem Apple lainnya.

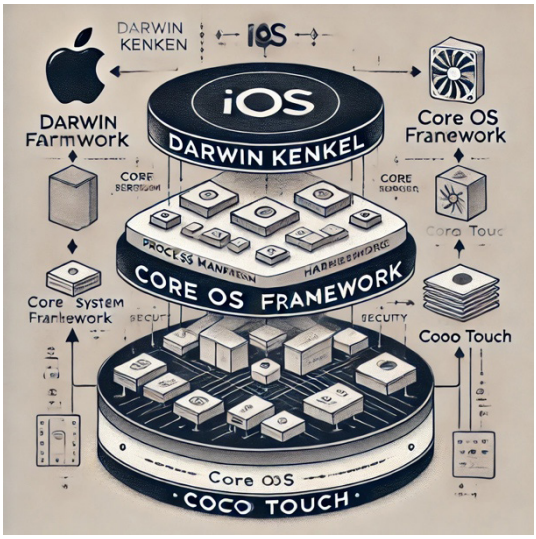
Karakteristik iOS:

1. **Desain UI yang Konsisten dan Intuitif**: iOS menawarkan antarmuka pengguna yang mudah dinavigasi dan konsisten pada semua perangkat Apple.

2. **Keamanan yang Ketat**: Apple menggunakan enkripsi, sandboxing, dan kontrol akses untuk melindungi data pengguna pada perangkat iOS.
3. **App Store yang Terpusat**: Apple mengontrol aplikasi di App Store dengan ketat untuk memastikan keamanan dan kualitas.
4. **Integrasi dengan Ekosistem Apple**: iOS terintegrasi dengan perangkat dan layanan Apple lainnya, seperti iCloud, iMessage, dan AirDrop.

Arsitektur iOS:

1. **Darwin Kernel**: Kernel berbasis Unix yang digunakan oleh macOS dan iOS, mengelola sumber daya perangkat keras.
2. **Core OS, Core Services, Media, Cocoa Touch**: Stack framework yang menyediakan fungsionalitas dasar, layanan data, media, dan antarmuka pengguna.
3. **App Sandbox**: Memastikan setiap aplikasi berjalan di lingkungan terisolasi, melindungi data pengguna dari akses aplikasi lain.



Gambar 58. Diagram arsitektur iOS, menunjukkan Darwin kernel, framework Core OS, dan lapisan Cocoa Touch untuk antarmuka

Contoh Penggunaan:

1. **Smartphone (iPhone)**: iOS adalah OS eksklusif untuk iPhone, dengan pengalaman pengguna yang sangat konsisten.
2. **Tablet (iPad)**: iPadOS adalah varian dari iOS yang dioptimalkan untuk layar besar dan multitasking.
3. **Wearables**: iOS terintegrasi dengan watchOS di Apple Watch dan digunakan untuk pengelolaan perangkat wearable lainnya.
4.

PENUTUP

Dengan berakhirnya buku "Sistem Operasi: Teori dan Konsep Dasar", kami berharap pembaca telah mendapatkan pemahaman yang lebih baik tentang fondasi sistem operasi dan bagaimana konsep-konsep ini diaplikasikan dalam dunia nyata. Mulai dari dasar-dasar manajemen memori, penjadwalan proses, hingga perkembangan teknologi terbaru seperti virtualisasi dan cloud computing, buku ini dirancang untuk memberikan gambaran menyeluruh tentang peran vital sistem operasi dalam ekosistem komputasi modern.

Kami menyadari bahwa teknologi sistem operasi terus berkembang, seiring dengan inovasi di bidang perangkat keras dan perangkat lunak. Oleh karena itu, kami mengajak pembaca untuk terus memperdalam pengetahuan dan mengikuti perkembangan terbaru dalam dunia sistem operasi, baik melalui studi lanjut maupun praktik di lapangan.

Kami berterima kasih kepada para pembaca yang telah meluangkan waktu untuk mempelajari materi dalam buku ini. Semoga buku ini dapat menjadi referensi yang bermanfaat, baik bagi mahasiswa, pengajar, maupun praktisi yang berkecimpung dalam bidang teknologi informasi. Kritik dan saran dari pembaca sangat kami nantikan untuk penyempurnaan edisi-edisi selanjutnya.

Akhir kata, semoga buku ini dapat membantu pembaca dalam perjalanan pembelajaran, penelitian, dan pengembangan aplikasi di masa depan. Teruslah belajar, berinovasi, dan berkontribusi bagi kemajuan teknologi.

INDEKS

DAFTAR PUSTAKA

Abusaimeh, H (2020). Virtual machine escape in cloud computing services. International Journal of Ad*vanced Computer Science …, academi*a.edu, https://www.academia.edu/download/101585126/Paper_43-Virtual_Machine_Escape_in_Cloud_Computing_Services.pdf

Akbar, H, Zubair, M, & Malik, MS (2023). The security issues and challenges in cloud computing. International Journal for Electronic …, *ijeci.lgu.edu.pk,* http://ijeci.lgu.edu.pk/index.php/ijeci/article/view/125

Akhtar, ZB (2024). Securing operating systems (OS): a comprehensive approach to security with best practices and techniques., repository.cam.ac.uk, https://www.repository.cam.ac.uk/ite*ms/d2f7bd67-f8cc-44c*4-b2f5-2b993e738797

Alemami, Y, Al-Ghonmein, AM, & ... (2023). Cloud data security and various cryptographic algorithms. … and Computer …, researchgate.net, https://www.researchga*te.net/profile/Ali-Al-Ghonme*in/publication/365824222_Cloud_data_security_and_various_cryptographic_algorithms_*Corresponding_Author/links/6385dcf5c2cb154d293c0f2b/Cloud-data-security-and-various-*cryptographic-algorithms-Corresponding-Author.pdf

Arogundade, OR, & Palla, K (2023). Virtualization revolution: Transforming cloud computing with scalability and agility., academia.edu, https://www.academia.edu/download/1*15018998/iarjset.2023.pdf*

Bal, PK, Mohapatra, SK, Das, TK, Srinivasan, K, & Hu, YC (2022). A joint resource allocation, security with efficient task scheduling in cloud

computing using hybrid machine learning techniques. Sensors, mdpi.com, https://www.mdpi.com/1424-8220/22/3/1242

Balen, J, Vajak, D, & Salah, K (2020). Comparative performance evaluation of popular virtual private servers. Journal of Internet Technology, jit.ndhu.edu.tw, https://jit.ndhu.edu.tw/article/viewFile/2256/2269

Bhardwaj, A, & Krishna, CR (2021). Virtualization in cloud computing: Moving from hypervisor to containerization—a survey. Arabian Journal for Science and Engineering, Springer, https://doi.org/10.1007/s13369-021-05553-3

Chen, L, Xian, M, Liu, J, & Wang, H (2020). Research on virtualization security in cloud computing. IOP conference series: materials …, iopscience.iop.org, https://doi.org/10.1088/1757-899X/806/1/012027

Chen, L, Zhang, Y, Tian, B, Ai, Y, Cao, D, & ... (2022). Parallel driving OS: A ubiquitous operating system for autonomous driving in CPSS. IEEE Transactions on …, ieeexplore.ieee.org, https://ieeexplore.ieee.org/abstract/document/9956903/

Compastié, M, Badonnel, R, Festor, O, & He, R (2020). From virtualization security issues to cloud protection opportunities: An in-depth analysis of system virtualization models. Computers & Security, Elsevier, https://www.sciencedirect.com/science/article/pii/S0167404820301814

Demigha, O, & Larguet, R (2021). Hardware-based solutions for trusted cloud computing. Computers & Security, Elsevier, https://www.sciencedirect.com/science/article/pii/S0167404820303904

Erblat, B, Oz, TB, Tal, H, & Herzog, S (2021). Discovery of resources associated with cloud operating system. US Patent 11,095,506, Google Patents, https://patents.google.com/patent/US11095506B1/en

Essien, NP, & Umo, MC (2024). Conceptual Analysis of Optimization Strategies in Operating System (Windows and Unix) in the Current IT Trend from 2015 to 2023. International Journal of Contemporary …, journals.iapaar.com, https://journals.iapaar.com/index.php/ijcarn/article/view/199

Feng, E, Lu, X, Du, D, Yang, B, Jiang, X, Xia, Y, & ... (2021). Scalable memory protection in the {PENGLAI} enclave. … on Operating Systems …, usenix.org, https://www.usenix.org/conference/osdi21/presentation/feng

Gala, G, Fohler, G, Tummeltshammer, P, & ... (2021). RT-cloud: Virtualization technologies and cloud computing for railway use-case. … Computing

(ISORC), ieeexplore.ieee.org, https://ieeexplore.ieee.org/abstract/document/9469907/

Gala, G, Fohler, G, Tummeltshammer, P, & ... (2021). RT-cloud: Virtualization technologies and *cloud computin*g for railway use-case. … Computing (ISORC), ieeexplore.ieee.org, https://ieeexplore.ieee.org/abstract/document/9469907/

George, SS, & Pramila, RS (2021). A review of different techniques in cloud computing. *Materials Today: Proce*edings, Elsevier, https://www.sciencedirect.com/science/article/pii/S2214785321019015

Gervais, C, Reed, ST, Goodwin, NS, Baker, JD, & ... (2020). System and method *for clo*ud-based operating system event and data access monitoring. US Patent …, Google Patents, https://patents.google.com/patent/US1079*1134B2/en*

Gill, BS, Gupta, K, & Cui, M (2020). Architecture for managing I/O and storage for a virtualization environment using executable containers and virtual machines. US Patent 10,721,290, Google Patents, https://patents.google.com/patent/US10721290B2/en

Haris, RM, Khan, KM, & Nhlabatsi, A (2022). Live migration of virtual machine memory content in networked systems. Computer Net*works, Elsevier, https://www.sciencedirect.*com/science/article/pii/S1389128622000962

Helali, L, & Omri, MN (2021). A survey of data center consolidation in cloud computing systems. Computer Science R*eview, Elsevier,* https://www.sciencedirect.com/science/article/pii/S157401372100006X

Hof, A Van't, & Nieh, J (2022). {BlackBox}: a container security monitor for protecting containers on untrusted operating systems. 16th USENIX Symposium on Operating Systems …, usenix.org, https://www.usenix.org/conference/osdi22/presentation/vant-hof

Holovnia, OS, & Oleksiuk, V (2022). Selecting cloud computing software for a virtual online lab*oratory supportin*g the Operating Systems course. CTE 2021: 9th Workshop on Cloud …, lib.iitta.gov.ua, https://lib.iitta.gov.ua/id/eprint/733355/

Hosseinzadeh, S, Sequeiros, B, Inácio, PRM, & ... (2020). Recent trends in applying TPM to cloud computing. *Security and …, Wiley Online Library, ht*tps://doi.org/10.1002/spy2.93

Kafhali, S El, Mir, I El, & Hanini, M (2022). Security threats, defense mechanisms, challenges, and future directions in cloud computing.

Archives of Computational Methods in …, Springer, https://doi.org/10.1007/s11831-021-09573-y

Katal, A, Dahiya, S, & Choudhury, T (2023). Energy efficiency in cloud computing data centers: a survey on software technologies. Cluster Computing, Springer, https://doi.org/10.1007/s10586-022-03713-0

Krishna, DS, Srinivas, G, & ... (2023). Novel private cloud architecture: A three tier approach to deploy private cloud using virtual machine manager. Intelligent Decision …, journals.sagepub.com, https://doi.org/10.3233/IDT-229035

Lee, S, Yu, Y, Tang, Y, Khandelwal, A, Zhong, L, & ... (2021). Mind: In-network memory management for disaggregated data centers. … on Operating Systems …, dl.acm.org, https://doi.org/10.1145/3477132.3483561

Ma, Y, Shen, Z, & Shen, J (2024). Cloud Computing and Hyperscale Data Centers: A Comparative Study of Usage Patterns. Journal of Theory and Practice of …, centuryscipub.com, https://centuryscipub.com/index.php/jtpes/article/view/614

Malallah, HS, Zeebaree, SRM, & ... (2021). A comprehensive study of kernel (issues and concepts) in different operating systems. … in Computer …, journal.251news.co.in, http://journal.251news.co.in/id/eprint/128/

Mangalampalli, S, Sree, PK, Swain, SK, & ... (2023). Cloud computing and virtualization. … of Cloud with AI for …, Wiley Online Library, https://doi.org/10.1002/9781119905233.ch2

Margaritov, A, Ustiugov, D, Shahab, A, & ... (2021). Ptemagnet: Fine-grained physical memory reservation for faster page walks in public clouds. … and Operating Systems, dl.acm.org, https://doi.org/10.1145/3445814.3446704

Marinescu, DC (2022). Cloud computing: theory and practice., books.google.com, https://books.google.com/books?hl = en&lr=&id =XOBWE AAAQ BAJ&oi=fnd&pg=PP1&dq=operating+system+virtualization+memory+management+security+cloud+comp uting&ots=F1mgv9qypb&sig=TxmUHJR7l_YzY7buFw NxZCEWd-s

Mavridis, I, & Karatza, H (2023). Orchestrated sandboxed containers, unikernels, and virtual machines for isolation-enhanced multitenant workloads and serverless computing in cloud. Concurrency and Computation: Practice …, Wiley Online Library, https://doi.org/10.1002/cpe.6365

*Midha, S, T*ripathi, K, & Sharma, MK (2021). Practical Implications of Using Dockers on Virtualized SDN.. Webology, pdfs.semanticscholar.org, https://pdfs.semanticscholar.org/0619/a4a7d9d7dac8ece0ea9026f8929e798cf311.pdf

Moustafa, N, K*eshky, M, Debiez, E, &* ... (2020). Federated TON_IoT Windows datasets for evaluating AI-based security applications. … privacy in computing …, ieeexplore.ieee.org, h*ttps://ieeexplore.*ieee.org/abstract/document/9343133/

Oliveira, F, Araujo, J, Matos, R, Lins, L, & ... (2020). Experimental evaluation of software aging effects in a container-based virtualization platfo*rm. … on Systems, Man …, ieeexplore.ieee.org, https:*//ieeexplore.ieee.org/abstract/document/9283358/

Palavalli, A, & Gaurav, K (2021). Methods and systems to optimize operating system license costs in a virtual data center. US Patent 11,182,713, Google Patents, https://pa*tents.google.com/patent*/US11182713B2/en

Pei, Z, Ping, L, & Shengmei, L (2020). Cloud computing technology and its applications. ZTE Communications, zte.magtechjournal.com, http://zte.magtechjournal.com/CN/article/*downloadArticleFile.*do?attachType=PDF&id=393

Saeed, A, Garraghan, P, & ... (2020). Cross-VM network channel attacks and countermeasures within cloud computing environments. … and Secure Computing, ieeexplore.ieee.org, h*ttps://ieeexplore.ieee.o*rg/abstract/document/9253612/

Shukur, H, Zeebaree, S, Zebari, R, Zeebaree, D, & ... (2020). Cloud computing virtualization of resources allocation for distributed systems. Journal of Applied …, *jastt.org, https*://www.jastt.org/index.php/jasttpath/article/view/31

Shukur, H, Zeebaree, S, Zebari, R, Zeebaree, D, & ... (2020). Cloud computing virtualization of resources allocation for distributed systems. Journal of Appl*ied …, jastt.org, https://www*.jastt.org/index.php/jasttpath/article/view/31

Sierra-Arriaga, F, Branco, R, & Lee, B (2020). Security issues and challenges for virtualization technologies. ACM Comput*ing Surveys (CSUR)*, dl.acm.org, https://doi.org/10.1145/3382190

Tabrizchi, H, & Rafsanjani, M Kuchaki (2020). A survey on security challenges in cloud computing: issues, threats, and solutio*ns. The*

*journal of super*computing, Springer, https://doi.org/10.1007/s11227-020-03213-1

Uddin, M, Khalique, A, Jumani, AK, Ullah, SS, & Hussain, S (2021). Next-generation blockchain-enabled virtualized cloud security solutions: review an*d open challenges. Ele*ctronics, mdpi.com, https://www.mdpi.com/2079-9292/10/20/2493

Vegesna, VV (2021). Analysis of Data Confidentiality Methods in Cloud Computing for Attaining Enhanced Security in Cloud Storage. Middle East Jo*urnal of Applied Scienc*e & …, papers.ssrn.com, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4418127

Wailly, A, & Legouge, P (2020). Method for monitoring the security of a virtual *machine in a cloud computing* architecture. US Patent 10,540,499, Google Patents, https://patents.google.com/patent/US10540499B2/en

Wang, X, Du, J, & Liu, H (2022). Performance and isolation analy*sis of RunC, gVisor* and Kata Containers runtimes. Cluster Computing, Springer, https://doi.org/10.1007/s10586-021-03517-8

Zolfaghari, R, Sahafi, A, Rahmani, AM, & ... (2021). Application of virtual machine consolidation in cloud c*omputing systems. Sustainable Computing …*, Elsevier, https://www.sciencedirect.com/science/article/pii/S2210537921000172

.

BIOGRAFI PENULIS

Dody Herdiana

lahir di Sumedang, pendidikan yang ditempuh setelah lulus SMA: S-1 Teknik Informatika UNIKOM Bandung, S-2 Teknik Informatika STTI Benarif Jakarta, dan S-3 ICT Asia E University Malaysia. Dody Herdiana aktif meneliti pada area penelitian: Logika Infrmatika, Kecerdasan Buatan, Sosial Komputer, dan Sistem Terapan. Paper-paper penelitian yang dipublikasikan sudah bisa diakses serta terindeks di beberapa publisher dan jurnal bereputasi.

Esa Firmansyah

Lahir di Bandung 1979, setelah menyelesaikan SMA Negeri 24 Bandung, melanjutkan kuliah S1 di STMIK PMBI Bandung, dan S2 di STTI Benarif Jakarta, Indonesia. Saat ini sedang melanjutkan S3 ICT Asia E University Malaysia, Malaysia. Penelitiannya berfokus pada Teknologi Informasi, Sistem Informasi & Smart City.

Muhammad Agreindra Helmiawan

Biasa dipanggil "Agre" lahir di Jakarta tahun 1986. Pendidikan setelah lulus SMK: S1-Teknik Informatika STMIK Sumedang, S2-Magister Teknik Informatika Universitas Langlangbuana Bandung, dan saat ini sedang melanjutkan studi S3 ICT Asia E University Malaysia. Agre aktif meneliti pada area penelitian: Keamanan Sistem, Keamanan Jaringan Komputer, Jaringan Komputer, Sistem Operasi. Dari area penelitian yang dilakukan telah diterbitkan pada beberapa jurnal bereputasi yang terindeks Nasional dan Internasional.

Yopi Hidayatul Akbar

Lahir di Tasikmalaya, pendidikan yang ditempuh setelah lulus SMA: S-1 Sistem Informasi STMIK Sumedang, S-2 Teknik Informatika Universitas Langlangbuana Bandung. Yopi Hidayatul Akbar mulai menulis sejak masih muda, Pengalamannya di bidang teknologi memberinya pendekatan yang segar dalam literatur, terutama ketika mengeksplorasi dampak teknologi pada kehidupan sosial, budaya, dan psikologis masyarakat. Dengan ketekunan dan dedikasi, ia menerbitkan beberapa buku dan artikel.

SISTEM OPERASI

Teori dan Konsep Dasar

Sistem operasi populer seperti Windows, Linux, macOS, Android, dan iOS tidak hanya memimpin pasar dalam kategori mereka masing-masing, tetapi juga mempengaruhi bagaimana perangkat keras dan perangkat lunak berkembang. Windows, misalnya, telah menjadi standar de facto untuk komputer pribadi, menawarkan antarmuka grafis yang user-friendly dan dukungan ekosistem aplikasi yang luas. Linux, dengan sifat open-source dan fleksibilitas tinggi, telah menjadi fondasi dari sebagian besar server, superkomputer, dan perangkat IoT. macOS menawarkan integrasi yang mulus dengan produk Apple lainnya dan memberikan pengalaman pengguna premium yang dioptimalkan untuk performa dan keamanan. Di dunia mobile, Android memberikan kebebasan kustomisasi kepada produsen dan pengguna, sementara iOS dikenal dengan keamanan yang ketat dan kontrol kualitas aplikasi yang tinggi.

Setiap sistem operasi ini telah berkembang dan beradaptasi dengan perubahan teknologi serta kebutuhan pasar yang dinamis. Kemajuan dalam virtualisasi, cloud computing, serta Internet of Things (IoT) juga mendorong sistem operasi untuk terus berinovasi. Melalui studi kasus ini, kita dapat mempelajari bagaimana sistem operasi beradaptasi terhadap tren dan tantangan baru, serta bagaimana desain dan fitur mereka terus berevolusi untuk memenuhi ekspektasi pengguna yang terus berubah.

NAFAL Publishing

Nafal Global Nusantara

nafalglobalnusantara@gmail.com
Nafal Global Nusantara
nafalpublishing
+6281284872750

pindai di sini

Pendidikan +17

ISBN 978-634-7025-52-4