

SISTEM OPERASI: KONSEP DASAR DAN IMPLEMENTASI MODERN



TIM PENULIS :

- Lely Priska D. Tampubolon
- Pastima Simanjuntak
- Johni S Pasaribu
- Aulia Nurul Iftitah
- Anas

SISTEM OPERASI:

Konsep Dasar dan Implementasi Modern

Lely Priska D. Tampubolon
Pastima Simanjuntak
Johni S Pasaribu
Aulia Nurul Iftitah
Anas



GET PRESS INDONESIA

SISTEM OPERASI: KONSEP DASAR DAN IMPLEMENTASI MODERN

Penulis :

Lely Priska D. Tampubolon
Pastima Simanjuntak
Johani S Pasaribu
Aulia Nurul Iftitah
Anas

ISBN : 978-634-255-749-5

Editor : Hendra Nusa Pratama, M.Kom.

Desain Sampul dan Tata Letak : Mila Sari, M.Si.

PENERBIT : GET PRESS INDONESIA

Anggota IKAPI No. 033/SBA/2022

Jl. Palarik RT 01 RW 06 Kelurahan Air Pacah
Kecamatan Koto Tangah Padang Sumatera Barat
website: www.getpress.co.id
email: adm.getpress@gmail.com

Cetakan pertama, September 2026

Hak cipta dilindungi undang-undang
Dilarang memperbanyak karya tulis ini dalam bentuk
dan dengan cara apapun tanpa izin tertulis dari penerbit.

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, yang telah memberikan rahmat dan karunia-Nya sehingga buku berjudul *Sistem Operasi: Konsep Dasar dan Implementasi Modern* ini dapat diselesaikan. Isi buku ini mencakup pembahasan tentang konsep dasar perangkat komputer dan arsitektur modern, konsep dasar dan evolusi sistem operasi, komponen sistem operasi pada sistem modern, penjadwalan cpu pada sistem modern dan *real-time*, manajemen input/ouput dan perangkat modern. Akhir kata, semoga buku ini dapat menjadi referensi dan dapat memberikan manfaat serta inspirasi bagi semua pembaca.

Padang, Agustus 2026

Penulis

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR	vii
DAFTAR TABEL.....	viii
BAB 1 KONSEP DASAR PERANGKAT KOMPUTER DAN ARSITEKTUR MODERN	1
1.1 Sistem Komputer Modern	4
1.1.1 Pengertian Sistem Komputer.....	4
1.1.2 Komponen Sistem Komputer.....	6
1.1.3 Siklus Pengolahan Data.....	8
1.1.4 Karakteristik Sistem Komputer Modern.....	9
1.1.5 Hubungan Sistem Komputer dengan Sistem Operasi	11
1.2 Arsitektur Dasar Komputer	11
1.2.1 Arsitektur Von Neumann	12
1.2.2 Arsitektur Harvard	12
1.3 Unit Pemrosesan Pusat (CPU).....	13
1.4 Evolusi Prosesor Modern	13
1.4.1 <i>Single-Core Processor</i>	13
1.4.2 <i>Multicore Processor</i>	14
1.5 Hierarki Memori pada Sistem Modern.....	14
1.6 Sistem Bus dan Interkoneksi.....	14
1.7 Akselerator Komputasi Modern.....	15
1.8 Arsitektur Komputer Modern.....	15
1.9 Hubungan Arsitektur Komputer dengan Sistem Operasi	16
1.10 Studi Kasus Arsitektur Intel Core Ultra dan AI PC.....	17
DAFTAR PUSTAKA	19

BAB 2 KONSEP DASAR DAN EVOLUSI SISTEM OPERASI	21
2.1 Definisi dan Fungsi Sistem Operasi	22
2.1.1 Definisi Sistem Operasi	22
2.1.2 Fungsi Utama Sistem Operasi.....	23
2.2 Arsitektur Sistem Operasi	25
2.2.1 Struktur Monolitik	25
2.2.2 Arsitektur Mikrokernel.....	26
2.2.3 Arsitektur Hybrid.....	26
2.2.4 Arsitektur Exokernel dan Unikernel.....	27
2.3 Evolusi Sistem Operasi	27
2.3.1 Era Pra-Sistem Operasi (1940-an – awal 1950-an).....	27
2.3.2 Sistem Batch (1950-an – 1960-an)	28
2.3.3 <i>Multiprogramming</i> dan <i>Time-Sharing</i> (1960-an)	29
2.3.4 Era UNIX dan Perkembangannya (1969 – 1980-an).....	29
2.3.5 Era Personal Computer (1980-an).....	30
2.3.6 Era Sistem Operasi Modern (1990-an).....	31
2.3.7 Era Virtualisasi dan Cloud Computing (2000-an – 2010-an)	32
2.3.8 Era Mobile dan Sistem Tertanam (2007 – sekarang)	33
2.3.9 Era Container dan Serverless (2010-an – sekarang)	34
2.3.10 Sistem Operasi di Era AI dan Edge Computing	35
2.4 Perbandingan Sistem Operasi Utama	35
2.5 Tantangan dan Tren Sistem Operasi Masa Depan	36
2.5.1 Keamanan dan Privasi.....	36
2.5.2 Heterogenitas dan Akselerator	37
2.5.3 Sistem Operasi untuk Komputasi Kuantum.....	37
DAFTAR PUSTAKA	39

BAB 3 KOMPONEN SISTEM OPERASI PADA SISTEM MODERN.....	43
3.1 Pengantar Sistem Operasi	43
3.2 Antarmuka Pengguna dan Sistem Operasi	45
3.3 Posisi Sistem Operasi dalam Sistem Komputer	46
3.4 Kompleksitas Sistem Operasi Modern	47
3.5 Pengertian Sistem Operasi	47
3.5.1 Sistem Operasi sebagai Mesin yang Diperluas (<i>Extended Machine</i>)	48
3.5.2 Sistem Operasi sebagai Pengelola Sumber Daya (<i>Resource Manager</i>)	49
3.6 Komponen Sistem Operasi Pada Sistem Modern	49
3.6.1 Manajemen Proses (<i>Process Management</i>)	50
3.6.2 Manajemen Memori (<i>Memory Management</i>)	51
3.6.3 Manajemen Penyimpanan Sekunder (<i>Secondary Storage Management</i>)	52
3.6.4 Sistem Input/Output (<i>I/O System</i>)	53
3.6.5 Manajemen Berkas (<i>File Management</i>)	54
3.6.6 Sistem Proteksi (<i>Protection System</i>)	55
3.6.7 Jaringan (<i>Networking</i>)	55
3.6.8 Sistem Interpreter Perintah (<i>Command Interpreter System</i>)	56
DAFTAR PUSTAKA	58
BAB 4 PENJADWALAN CPU PADA SISTEM MODERN DAN REAL-TIME	59
4.1 Bagaimana CPU Mengatur Waktu?	59
4.1.1 Siklus Kerja CPU dan I/O	60
4.1.2 Preemptive vs Non-Preemptive	60
4.1.3 Kriteria Penjadwalan yang Baik	61
4.2 Algoritma Penjadwalan Klasik	61

4.2.1	Siapa Cepat Dia Dapat (FCFS) & Tugas Terpendek Didahulukan (SJF).....	61
4.2.2	Contoh Simulasi Algoritma FCFS (<i>First Come First Served</i>).....	62
4.2.3	Contoh Simulasi Algoritma SJF (<i>Shortest Job First</i>)	65
4.2.4	Sistem Antrean Bergiliran (<i>Round Robin</i>)	67
4.2.5	Mekanisme Tata Kerja Round Robin	67
4.2.6	Penjadwalan Berbasis Prioritas	70
4.2.7	Mekanisme Tata Kerja <i>Priorty Scheduling</i>	70
4.2.8	Antrean Bertingkat (<i>Multilevel Feedback Queue</i>)	71
4.3	Penjadwalan pada Komputer Modern (<i>Multicore</i> dan <i>Multiprocessor</i>).....	72
4.3.1	Tantangan Banyak Inti (<i>Multicore</i>)	72
4.3.2	Keseimbangan Beban (<i>Load Balancing</i>).....	72
4.3.3	Kedekatan Memori (<i>NUMA Awareness</i>)	72
4.4	Penjadwalan di Dunia Virtual (Virtualisasi, Cloud, dan Kontainer).....	73
4.4.1	Berbagi CPU di Cloud	73
4.4.2	Penjadwalan Virtual Machine vs Kontainer.....	73
4.5	Mengenal Penjadwalan <i>Real-Time</i> (Sistem Waktu-Nyata)....	74
4.5.1	Apa itu Sistem <i>Real-Time</i> ?.....	74
4.5.2	Sistem Saklek vs Sistem Toleran	75
4.5.3	Mengatur Jadwal Berdasarkan Waktu (RM dan EDF)	75
4.6	Hemat Energi pada Embedded System dan IoT	76
4.6.1	Penjadwalan Hemat Daya (DVFS)	76
4.6.2	Sistem Berbasis Waktu vs Berbasis Event	76
4.6.3	Mengintip RTOS (FreeRTOS & Zephyr).....	77
4.7	Studi Kasus: Cara Kerja OS yang Kita Gunakan Hari Ini	77
4.7.1	Penjadwal di Linux (<i>Completely Fair Scheduler</i> - CFS) ...	77
4.7.2	Penjadwal di Windows dan Android	78

4.7.3 Tantangan CPU Heterogen (big.LITTLE)	78
DAFTAR PUSTAKA	80
BAB 5 MANAJEMEN INPUT/OUTPUT DAN PERANGKAT MODERN.....	83
5.1 Arsitektur dan Karakteristik Perangkat I/O	84
5.1.1 Klasifikasi Perangkat I/O	84
5.1.2 Perangkat Penyimpanan Modern	86
5.2 Teknik Penanganan I/O	88
5.2.1 Programmed I/O.....	88
5.2.2 Interrupt-Driven I/O.....	88
5.2.3 <i>Direct Memory Access</i> (DMA)	89
5.3 Struktur Software Sistem I/O	90
5.3.1 Layered I/O Software	90
5.3.2 <i>Buffering, Caching, dan Spooling</i>	92
5.4 Perangkat I/O Modern dan Teknologi Terkini.....	93
5.4.1 Universal Serial Bus (USB) dan Generasi Terbarunya....	93
5.4.2 Thunderbolt dan Thunderbolt 4	94
5.4.3 Perangkat Jaringan Modern dan I/O Virtualisasi	95
5.4.4 Cloud Storage dan I/O Terdistribusi	96
5.5 Manajemen Kinerja dan Optimasi I/O.....	96
5.5.1 Penjadwalan Disk.....	96
5.5.2 RAID dan Redundansi Data	97
5.5.3 Error Handling dan Recovery.....	98
DAFTAR PUSTAKA	100
INDEKS	102
GLOSARIUM	107
BIODATA PENULIS	102

DAFTAR GAMBAR

Gambar 3. 1 Di mana posisi sistem operasi.....	45
Gambar 3. 2 Sistem Operasi Mengubah Perangkat Keras yang Buruk Menjadi Abstraksi yang Indah.....	48

DAFTAR TABEL

Tabel 1. 1 Komponen Sistem Komputer Modern.....	7
Tabel 1. 2 Perbandingan Sistem Komputer Konvensional dan Modern	10
Tabel 2. 1 Perbandingan Sistem Operasi Utama	36
Tabel 4. 1 Data Proses FCFS.....	62
Tabel 4. 2 Data Proses FCFS (<i>Start Time</i> dan <i>Finish Time</i>)	63
Tabel 4. 3 Hasil Start Time dan <i>Finish Time</i> FCFS	64
Tabel 4. 4 Hasil Perhitungan FCFS	65
Tabel 4. 5 Data Proses SJF	66
Tabel 4. 6 Hasil Perhitungan SJF	67
Tabel 4. 7 Data Proses RR.....	68
Tabel 4. 8 Tata Kerja RR.....	69
Tabel 4. 9 Hasil Perhitungan RR	69
Tabel 4. 10 Data Proses PS	70
Tabel 4. 11 Hasil Perhitungan PS.....	71
Tabel 5. 1 Klasifikasi Perangkat I/O Berdasarkan Kecepatan dan Metode Akses	85
Tabel 5. 2 Perbandingan Karakteristik Perangkat Penyimpanan Modern	87
Tabel 5. 3 Perbandingan Teknik Penanganan I/O	90
Tabel 5. 4 Evolusi Standar USB.....	94

BAB 1

KONSEP DASAR PERANGKAT KOMPUTER DAN ARSITEKTUR MODERN

Oleh Lely Priska D. Tampubolon

Perkembangan teknologi informasi dan komunikasi telah mengubah hampir seluruh aspek kehidupan manusia, mulai dari aktivitas pendidikan, bisnis, pemerintahan, kesehatan, hingga industri manufaktur. Transformasi digital yang berlangsung secara masif didukung oleh kemajuan perangkat komputer yang semakin cerdas, cepat, dan efisien. Komputer tidak lagi berfungsi sebagai alat pengolah data semata, tetapi telah berkembang menjadi platform utama dalam menjalankan berbagai layanan digital seperti komputasi awan (*cloud computing*), kecerdasan buatan (*artificial intelligence*), Internet of Things (IoT), analitik data besar (*big data analytics*), komputasi berkinerja tinggi (*high-performance computing*), serta sistem siber-fisik (*cyber-physical systems*). Perkembangan tersebut menunjukkan bahwa pemahaman mengenai perangkat komputer dan arsitekturnya menjadi landasan penting dalam mempelajari sistem operasi modern (Tanenbaum and Bos, 2015).

Pada era digital saat ini, kebutuhan terhadap kemampuan komputasi meningkat secara eksponensial. Organisasi, industri, dan institusi pendidikan membutuhkan sistem komputer yang mampu memproses data dalam jumlah

besar secara cepat, aman, dan efisien. Kondisi tersebut mendorong produsen perangkat keras untuk terus mengembangkan arsitektur komputer melalui peningkatan jumlah inti prosesor (*multicore*), penggunaan akselerator komputasi seperti *Graphics Processing Unit* (GPU) dan *Neural Processing Unit* (NPU), serta pengembangan teknologi memori dan interkoneksi berkecepatan tinggi. Inovasi tersebut bertujuan untuk meningkatkan kinerja sistem sekaligus mengurangi konsumsi energi sehingga mendukung konsep *green computing* dan komputasi berkelanjutan (Hennessy, Patterson and Kozyrakis, 2019).

Sistem operasi merupakan perangkat lunak sistem yang berperan sebagai penghubung antara pengguna dengan perangkat keras komputer. Sistem operasi bertanggung jawab mengelola sumber daya perangkat keras seperti prosesor, memori, media penyimpanan, dan perangkat masukan-keluaran agar dapat digunakan secara efisien oleh berbagai aplikasi. Oleh karena itu, pemahaman terhadap karakteristik perangkat keras menjadi prasyarat penting untuk memahami bagaimana sistem operasi melakukan penjadwalan proses, manajemen memori, pengelolaan perangkat I/O, sistem berkas, hingga virtualisasi. Semakin kompleks arsitektur perangkat keras yang digunakan, semakin kompleks pula mekanisme pengelolaan yang harus dilakukan oleh sistem operasi . (Silberschatz, Galvin and Gagne, 2018).

Perkembangan arsitektur komputer juga mengalami perubahan paradigma. Jika pada masa awal komputer peningkatan kinerja dilakukan dengan meningkatkan frekuensi prosesor, saat ini peningkatan performa lebih banyak dicapai melalui pemrosesan paralel, komputasi heterogen (*heterogeneous computing*), serta integrasi berbagai jenis prosesor dalam satu sistem. Sebagai contoh, prosesor modern tidak hanya terdiri atas *Central Processing Unit* (CPU), tetapi juga mengintegrasikan GPU, NPU, dan

berbagai akselerator lain dalam satu *System-on-Chip* (SoC). Pendekatan ini memungkinkan distribusi beban kerja sesuai karakteristik masing-masing unit pemrosesan sehingga menghasilkan efisiensi yang lebih tinggi, khususnya untuk aplikasi berbasis kecerdasan buatan dan analitik data (Hennessy, Patterson and Kozyrakis, 2019).

Selain perkembangan pada prosesor, teknologi memori dan media penyimpanan juga mengalami inovasi yang signifikan. Penggunaan *Solid State Drive* (SSD) berbasis NVMe, memori DDR5, serta teknologi interkoneksi seperti *Peripheral Component Interconnect Express* (PCIe) Gen5 dan *Compute Express Link* (CXL) telah meningkatkan kecepatan akses data secara signifikan dibandingkan generasi sebelumnya. Perubahan tersebut menuntut sistem operasi untuk mampu mengoptimalkan pemanfaatan sumber daya melalui mekanisme manajemen memori, penjadwalan akses perangkat, dan pengelolaan komunikasi antarperangkat yang lebih adaptif.

Di sisi lain, perkembangan komputasi awan (*cloud computing*) dan komputasi tepi (*edge computing*) telah mengubah cara sumber daya komputasi dikelola. Infrastruktur komputasi modern tidak lagi terbatas pada satu komputer fisik, tetapi terdiri atas ribuan server yang saling terhubung melalui jaringan berkecepatan tinggi. Sistem operasi modern harus mampu mendukung virtualisasi, kontainerisasi, serta orkestrasi sumber daya agar aplikasi dapat dijalankan secara fleksibel, elastis, dan memiliki tingkat ketersediaan yang tinggi. Oleh karena itu, konsep arsitektur komputer modern tidak dapat dipisahkan dari perkembangan sistem operasi dan teknologi komputasi terkini (H. and C., 2023).

Bab ini bertujuan memberikan pemahaman mendasar mengenai konsep perangkat komputer dan arsitektur modern sebagai landasan untuk mempelajari sistem operasi. Pembahasan dimulai dari pengertian sistem komputer,

komponen utama perangkat keras, prinsip kerja arsitektur komputer, evolusi prosesor, hierarki memori, teknologi interkoneksi, hingga perkembangan arsitektur modern seperti *System-on-Chip*, komputasi heterogen, dan akselerator kecerdasan buatan. Selain itu, bab ini juga menyajikan beberapa studi kasus mengenai implementasi arsitektur komputer pada prosesor modern sehingga pembaca memperoleh gambaran mengenai hubungan antara perkembangan perangkat keras dengan desain sistem operasi masa kini.

Pada akhir pembelajaran, pembaca diharapkan mampu menjelaskan konsep dasar sistem komputer, mengidentifikasi fungsi setiap komponen perangkat keras, memahami prinsip kerja berbagai arsitektur komputer, menganalisis perkembangan teknologi prosesor dan memori, serta menjelaskan implikasi perkembangan arsitektur modern terhadap desain dan implementasi sistem operasi. Pemahaman tersebut akan menjadi fondasi bagi pembahasan pada bab-bab selanjutnya mengenai evolusi sistem operasi, manajemen proses, manajemen memori, sistem berkas, virtualisasi, dan teknologi kontainer.

1.1 Sistem Komputer Modern

1.1.1 Pengertian Sistem Komputer

Sistem komputer merupakan suatu kesatuan yang terdiri atas perangkat keras (*hardware*), perangkat lunak (*software*), dan pengguna (*brainware*) yang saling berinteraksi untuk menerima masukan (*input*), memproses data (*processing*), menghasilkan keluaran (*output*), serta menyimpan informasi (*storage*). Ketiga komponen tersebut bekerja secara terpadu sehingga mampu menyelesaikan berbagai tugas komputasi sesuai kebutuhan pengguna. Tanpa adanya salah satu

komponen tersebut, sistem komputer tidak dapat berfungsi secara optimal (Silberschatz *et al.*, 2018).

Pada awal perkembangannya, komputer dirancang untuk melakukan perhitungan numerik dalam skala besar. Seiring dengan kemajuan teknologi semikonduktor dan mikroprosesor, fungsi komputer berkembang menjadi platform multifungsi yang mampu menjalankan aplikasi perkantoran, multimedia, simulasi ilmiah, analisis data, kecerdasan buatan, hingga pengendalian perangkat Internet of Things (IoT). Perubahan tersebut menyebabkan sistem komputer modern tidak lagi dipandang sebagai sekumpulan perangkat elektronik, melainkan sebagai suatu ekosistem komputasi yang mengintegrasikan perangkat keras, perangkat lunak, jaringan komunikasi, dan layanan digital (Hennessy, Patterson and Kozyrakis, 2019).

Dalam perspektif sistem operasi, sistem komputer dipandang sebagai sekumpulan sumber daya (*resources*) yang harus dikelola secara efisien. Sumber daya tersebut meliputi prosesor, memori, media penyimpanan, perangkat masukan-keluaran, serta jaringan komputer. Sistem operasi bertugas mengalokasikan sumber daya tersebut kepada berbagai aplikasi sehingga seluruh proses dapat berjalan secara bersamaan tanpa saling mengganggu. Oleh karena itu, pemahaman mengenai struktur sistem komputer merupakan landasan utama dalam mempelajari desain dan implementasi sistem operasi.

Perkembangan teknologi digital juga telah mengubah paradigma sistem komputer dari sistem yang bersifat terpusat (*stand-alone computing*) menjadi sistem yang saling terhubung (*networked computing*). Komputer modern mampu berkomunikasi dengan perangkat lain melalui jaringan lokal maupun internet

sehingga memungkinkan implementasi komputasi awan (*cloud computing*), komputasi tepi (*edge computing*), serta komputasi terdistribusi (*distributed computing*). Karakteristik tersebut menjadikan sistem komputer modern memiliki tingkat fleksibilitas, skalabilitas, dan interoperabilitas yang jauh lebih tinggi dibandingkan generasi sebelumnya (H. and C., 2023).

1.1.2 Komponen Sistem Komputer

Sistem komputer modern terdiri atas tiga komponen utama, yaitu perangkat keras (*hardware*), perangkat lunak (*software*), dan brainware. Ketiga komponen tersebut saling melengkapi dalam membentuk suatu sistem komputasi yang utuh.

a. Perangkat Keras (*Hardware*)

Perangkat keras merupakan seluruh komponen fisik yang menyusun sistem komputer dan dapat diamati secara langsung. Fungsi utama perangkat keras adalah menyediakan fasilitas komputasi yang diperlukan untuk menjalankan instruksi program. Berdasarkan fungsinya, perangkat keras dapat dikelompokkan menjadi beberapa kategori.

- 1) Unit Pemrosesan (CPU) berfungsi sebagai pusat pengendali sekaligus pelaksana seluruh instruksi program.
- 2) Memori utama (RAM) digunakan untuk menyimpan data dan instruksi yang sedang diproses.
- 3) Media penyimpanan (SSD/HDD) berfungsi untuk menyimpan data secara permanen.
- 4) Perangkat input digunakan untuk memasukkan data ke dalam sistem komputer.

- 5) Perangkat output menampilkan hasil pengolahan data.
- 6) Perangkat komunikasi memungkinkan komputer terhubung ke jaringan.

Pada komputer modern, perangkat keras berkembang menuju arsitektur heterogen yang mengintegrasikan CPU, GPU, dan NPU dalam satu sistem untuk meningkatkan performa komputasi, khususnya pada aplikasi berbasis kecerdasan buatan.

b. Perangkat Lunak (*Software*)

Perangkat lunak merupakan sekumpulan instruksi yang mengendalikan operasi perangkat keras. Berdasarkan fungsinya, perangkat lunak dibedakan menjadi beberapa kelompok.

- 1) Sistem Operasi (*Operating System*)
- 2) Perangkat Lunak Aplikasi
- 3) Utility Software
- 4) Middleware
- 5) Firmware

Sistem operasi memiliki peran sentral karena mengelola seluruh sumber daya perangkat keras sehingga aplikasi dapat dijalankan secara efisien.

Tabel 1. 1 Komponen Sistem Komputer Modern

Komponen	Fungsi Utama	Contoh
Hardware	Menyediakan sumber daya komputasi	CPU, RAM, SSD, GPU
Software	Mengendalikan perangkat keras	Windows, Linux, Android

Komponen	Fungsi Utama	Contoh
Brainware	Mengoperasikan dan mengelola sistem	Administrator, Programmer, End User

Sumber: Diadaptasi dari (Tanenbaum and Bos, 2015)

c. Brainware

Brainware merupakan manusia yang menggunakan maupun mengelola sistem komputer. Dalam lingkungan organisasi modern, brainware tidak hanya berperan sebagai pengguna, tetapi juga sebagai pengembang, administrator, analis data, administrator jaringan, serta pengelola keamanan siber.

Perkembangan teknologi kecerdasan buatan menyebabkan peran brainware mengalami perubahan. Pengguna tidak lagi hanya berinteraksi dengan komputer melalui antarmuka grafis, tetapi juga melalui agen cerdas (*intelligent agents*) dan sistem berbasis bahasa alami (*natural language interfaces*).

1.1.3 Siklus Pengolahan Data

Sistem komputer bekerja melalui suatu siklus pengolahan data yang terdiri atas empat tahapan utama, yaitu input, processing, output, dan storage.

Pada tahap input, data dimasukkan ke dalam sistem menggunakan perangkat masukan seperti keyboard, mouse, kamera, sensor, maupun perangkat IoT. Data tersebut kemudian disimpan sementara di memori utama untuk diproses oleh CPU.

Tahap processing merupakan inti dari seluruh aktivitas komputasi. Pada tahap ini, prosesor mengeksekusi instruksi program menggunakan mekanisme fetch-decode-execute. Data yang diproses dapat berupa operasi aritmetika, logika, manipulasi basis data, analisis citra, hingga pelatihan model kecerdasan buatan.

Setelah proses selesai, sistem menghasilkan output berupa informasi yang ditampilkan melalui monitor, printer, speaker, atau perangkat lainnya. Hasil pengolahan selanjutnya dapat disimpan secara permanen pada media penyimpanan sehingga dapat digunakan kembali pada waktu mendatang.

Pada sistem komputer modern, siklus tersebut berlangsung sangat cepat dan dapat dijalankan secara paralel pada banyak inti prosesor (multicore) maupun akselerator komputasi seperti GPU. Hal ini memungkinkan komputer memproses jutaan hingga miliaran instruksi setiap detik.

1.1.4 Karakteristik Sistem Komputer Modern

1. Berbeda dengan komputer generasi sebelumnya, sistem komputer modern memiliki sejumlah karakteristik yang mendukung kebutuhan komputasi masa kini.
2. Pemrosesan Paralel (Parallel Processing)
Komputer modern menggunakan prosesor multicore dan manycore sehingga mampu menjalankan banyak proses secara bersamaan.
3. Komputasi Heterogen (Heterogeneous Computing)
Integrasi CPU, GPU, dan NPU memungkinkan distribusi beban kerja sesuai karakteristik masing-masing unit pemrosesan.
4. Virtualisasi
Teknologi virtualisasi memungkinkan satu

perangkat keras menjalankan beberapa sistem operasi secara bersamaan melalui mesin virtual.

5. **Konektivitas** Tinggi
Komputer modern selalu terhubung dengan jaringan lokal maupun internet sehingga mendukung layanan berbasis cloud.
6. **Efisiensi Energi**
Desain prosesor dan sistem operasi modern mengoptimalkan konsumsi daya melalui mekanisme *dynamic voltage and frequency scaling* (DVFS) dan penjadwalan beban kerja.
7. **Keamanan Terintegrasi**
Sistem komputer modern telah mengintegrasikan fitur keamanan berbasis perangkat keras, seperti *Trusted Platform Module* (TPM), *Secure Boot*, dan enkripsi perangkat keras.

Tabel 1. 2 Perbandingan Sistem Komputer Konvensional dan Modern

Aspek	Sistem Konvensional	Sistem Modern
Prosesor	Single-core	Multicore, Manycore
Pemrosesan	Sequential	Parallel dan Heterogeneous
Penyimpanan	HDD	SSD NVMe, Cloud Storage
Komputasi	Lokal	Cloud dan Edge Computing
AI	Tidak tersedia	GPU, TPU, NPU
Virtualisasi	Terbatas	Mendukung Virtual Machine dan Container

Sumber: Diadaptasi dari (Hennessy, Patterson and Kozyrakis, 2019)

1.1.5 Hubungan Sistem Komputer dengan Sistem Operasi

Sistem operasi merupakan perangkat lunak sistem yang bertugas mengelola seluruh sumber daya perangkat keras agar dapat dimanfaatkan secara optimal oleh pengguna dan aplikasi. Dalam konteks ini, sistem operasi bertindak sebagai *resource manager* sekaligus *extended machine*. Sebagai *resource manager*, sistem operasi mengalokasikan waktu prosesor, ruang memori, media penyimpanan, dan perangkat masukan-keluaran kepada setiap proses secara adil dan efisien. Sebagai *extended machine*, sistem operasi menyederhanakan kompleksitas perangkat keras melalui antarmuka pemrograman dan layanan sistem sehingga pengembang aplikasi tidak perlu berinteraksi langsung dengan perangkat keras.

Seiring berkembangnya arsitektur komputer modern, tanggung jawab sistem operasi juga semakin kompleks. Sistem operasi tidak hanya mengelola CPU dan memori, tetapi juga harus mampu menjadwalkan beban kerja pada GPU dan NPU, mendukung virtualisasi, mengelola kontainer, serta mengoptimalkan penggunaan sumber daya pada lingkungan cloud dan edge computing. Oleh karena itu, pemahaman mengenai sistem komputer modern menjadi dasar yang sangat penting sebelum mempelajari konsep-konsep lanjutan mengenai manajemen proses, memori, sistem berkas, dan virtualisasi yang akan dibahas pada bab-bab berikutnya.

1.2 Arsitektur Dasar Komputer

Arsitektur komputer menjelaskan struktur dan organisasi sistem komputer yang menentukan bagaimana perangkat keras menjalankan instruksi program. Model

arsitektur yang paling banyak digunakan hingga saat ini adalah arsitektur Von Neumann.

1.2.1 Arsitektur Von Neumann

Arsitektur Von Neumann diperkenalkan oleh John Von Neumann pada tahun 1945 dan menjadi dasar hampir seluruh sistem komputer modern.

Komponen utamanya meliputi:

1. Central Processing Unit (CPU)
2. Memory Unit
3. Input Unit
4. Output Unit
5. System Bus

Keunggulan utama arsitektur ini adalah kesederhanaan desain dan fleksibilitas pemrograman. Namun demikian, arsitektur ini memiliki keterbatasan yang dikenal sebagai *Von Neumann Bottleneck*, yaitu keterbatasan transfer data antara CPU dan memori.

1.2.2 Arsitektur Harvard

Arsitektur Harvard menggunakan memori terpisah untuk instruksi dan data.

Keunggulannya:

- a. Transfer data lebih cepat
- b. Mendukung pemrosesan paralel
- c. Banyak digunakan pada mikrokontroler dan sistem tertanam

1.3 Unit Pemrosesan Pusat (CPU)

CPU merupakan komponen utama yang bertugas mengeksekusi instruksi program.

CPU modern terdiri atas beberapa bagian utama:

- a. *Arithmetic Logic Unit* (ALU). ALU bertanggung jawab melakukan operasi:
 1. Penjumlahan
 2. Pengurangan
 3. Perkalian
 4. Pembagian
 5. Operasi logika
- b. *Control Unit* (CU). Control Unit mengatur aliran data dan koordinasi seluruh komponen komputer selama proses eksekusi instruksi.
- c. Register. Register merupakan memori tercepat yang berada di dalam CPU dan digunakan untuk menyimpan data yang sedang diproses.
- d. *Cache Memory*.

1.4 Evolusi Prosesor Modern

Perkembangan prosesor telah mengalami transformasi signifikan dari prosesor inti tunggal (*single-core*) menuju prosesor banyak inti (*multicore processor*).

1.4.1 Single-Core Processor

Pada generasi awal, satu prosesor hanya memiliki satu unit pemrosesan sehingga seluruh instruksi dijalankan secara berurutan.

1.4.2 Multicore Processor

Multicore processor mengintegrasikan beberapa inti pemrosesan dalam satu chip.

Keuntungan Multicore

- a. Meningkatkan performa
- b. Mengurangi konsumsi daya
- c. Mendukung komputasi paralel
- d. Mendukung multitasking

1.5 Hierarki Memori pada Sistem Modern

Sistem komputer modern menggunakan hierarki memori untuk menyeimbangkan kecepatan, kapasitas, dan biaya.

- a. Register. Memiliki kecepatan akses tertinggi, namun kapasitas sangat kecil,
- b. Cache digunakan untuk menyimpan data yang sering diakses CPU. Jenis cache: L1 Cache, L2 Cache, dan L3 Cache,
- c. Main memory. RAM berfungsi menyimpan program yang sedang dijalankan,
- d. Secondary memory. Media penyimpanan permanen meliputi: SSD, HDD, dan NVMe Storage.

1.6 Sistem Bus dan Interkoneksi

Komponen komputer saling berkomunikasi melalui sistem bus. Jenis bus utama:

- a. Data Bus
- b. Address Bus
- c. Control Bus

Perkembangan terbaru menggunakan teknologi:

- a. PCI Express (PCIe)
- b. NVLink
- c. Compute Express Link (CXL)

Teknologi CXL menjadi standar penting dalam pusat data modern karena memungkinkan CPU dan akselerator berbagi memori secara efisien.

1.7 Akselerator Komputasi Modern

Kebutuhan komputasi modern tidak lagi dapat dipenuhi hanya oleh CPU.

- a. *Graphics Processing Unit* (GPU)
 - 1. GPU awalnya dirancang untuk pengolahan grafis, namun saat ini digunakan secara luas untuk:
 - 2. Deep Learning
 - 3. Machine Learning
 - 4. *High Performance Computing* (HPC)
- b. *Tensor Processing Unit* (TPU) dikembangkan khusus untuk mempercepat komputasi kecerdasan buatan.
- c. *Neural Processing Unit* (NPU) digunakan pada perangkat bergerak dan komputer generasi terbaru untuk menjalankan model AI secara lokal.

1.8 Arsitektur Komputer Modern

Perkembangan kebutuhan komputasi telah melahirkan berbagai inovasi arsitektur.

- a. *Heterogeneous Computing*

Menggabungkan beberapa jenis prosesor: CPU, GPU, NPU, dan FPGA untuk meningkatkan efisiensi dan performa.

b. *Edge Computing Architecture*

Komputasi dilakukan lebih dekat dengan sumber data sehingga mengurangi latensi. Contoh implementasi: Smart City, IoT dan kendaraan otonom.

c. *Cloud Computing Architecture*

Cloud computing menyediakan sumber daya komputasi secara elastis melalui internet.

Karakteristik utama:

1. *On-demand resource*
2. *Scalability*
3. *Virtualization*
4. *Resource pooling*

1.9 Hubungan Arsitektur Komputer dengan Sistem Operasi

Sistem operasi harus memahami karakteristik perangkat keras untuk mengelola sumber daya secara optimal.

Hubungan tersebut meliputi:

- a. Manajemen CPU
- b. Manajemen memori
- c. Manajemen penyimpanan
- d. Manajemen perangkat I/O
- e. Virtualisasi

f. Keamanan sistem

Perkembangan multicore, GPU, cloud computing, dan AI accelerator menyebabkan sistem operasi modern harus mampu mendukung komputasi paralel, virtualisasi, serta orkestrasi sumber daya secara dinamis.

1.10 Studi Kasus Arsitektur Intel Core Ultra dan AI PC

Perkembangan kecerdasan buatan (*Artificial Intelligence/AI*) telah mendorong produsen prosesor untuk mengintegrasikan akselerator AI secara langsung ke dalam prosesor. Salah satu contoh implementasi terbaru adalah Intel Core Ultra yang diperkenalkan sebagai fondasi komputer generasi AI PC (Intel Corporation, 2024), (Liu *et al.*, 2025).

Intel Core Ultra menggabungkan tiga komponen pemrosesan utama dalam satu chip, yaitu:

1. *Central Processing Unit* (CPU)
2. *Graphics Processing Unit* (GPU)
3. *Neural Processing Unit* (NPU)
4. Pendekatan ini dikenal sebagai *heterogeneous computing architecture* karena memanfaatkan berbagai jenis prosesor untuk menangani beban kerja yang berbeda.

CPU digunakan untuk menjalankan aplikasi umum dan sistem operasi. GPU digunakan untuk komputasi paralel dan pemrosesan grafis. Sementara itu, NPU bertugas menjalankan inferensi AI dengan konsumsi energi yang lebih rendah

dibandingkan CPU atau GPU (Dalmia, Kumar and Sinclair, 2024).

Dampak terhadap Sistem Operasi. Sistem operasi modern harus mampu:

- a. Mengelola beban kerja AI pada NPU.
- b. Menentukan distribusi proses ke CPU atau GPU.
- c. Mengoptimalkan konsumsi daya.
- d. Mengelola memori bersama antar unit pemrosesan.

Windows 11, Linux Kernel 6.x, dan macOS terbaru telah mulai mengintegrasikan dukungan terhadap akselerator AI melalui framework khusus seperti DirectML, OpenVINO, dan Core ML.

Sistem komputer modern terdiri atas perangkat keras, perangkat lunak, dan pengguna yang bekerja secara terpadu. Arsitektur komputer berkembang dari model Von Neumann menuju sistem multicore dan heterogen yang menggabungkan CPU, GPU, dan akselerator AI. Hierarki memori, sistem interkoneksi berkecepatan tinggi, serta teknologi cloud dan edge computing menjadi fondasi utama komputasi modern.

Pemahaman terhadap perangkat komputer dan arsitektur modern sangat penting karena menjadi dasar bagi pembahasan sistem operasi pada bab-bab berikutnya, terutama terkait manajemen proses, memori, penyimpanan, dan virtualisasi.

DAFTAR PUSTAKA

- Chauhan, Naresh. (2014) *Principles of operating systems*. Oxford University Press.
- Dalmia, P., Kumar, R.S. and Sinclair, M.D., 2024, November. CPElide: Efficient multi-chiplet GPU implicit synchronization. In *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 700-717). IEEE.
- H., R. and C., A. (2023) *Operating Systems: Three Easy Pieces*.
- Hennessy, J.L., Patterson, D.A. and Kozyrakis, C. (2019) *Computer Architecture: A Quantitative Approach*. Seventh Edition. Morgan Kaufmann Publisher.
- Intel Corporation (2024) *Intel® Core™ Ultra Processor Architecture, Santa Clara, CA: Intel Corporation*. Available at: <https://www.intel.com/content/www/us/en/products/details/processors/core-ultra.html> (Accessed: June 29, 2026).
- Liu, Y. *et al.* (2025) "Integration of Federated Learning and AI-Generated Content: A Survey of Overview, Opportunities, Challenges, and Solutions," *IEEE Communications Surveys & Tutorials*, 27(5), pp. 3308–3338. Available at: <https://doi.org/10.1109/COMST.2024.3523350>.
- Silberschatz, A., Galvin, P.B. and Gagne, G. (2018) *Operating System Concepts*. Tenth edition. Wiley.
- Stallings, W. (2016) *Computer Organization and Architecture: Designing for Performance*. Tenth Edition. Pearson.
- Tanenbaum, A.S. and Bos, H. (2015) *Modern Operating Systems*. Pearson.

BAB 2

KONSEP DASAR DAN EVOLUSI SISTEM OPERASI

Oleh Pastima Simanjuntak

Sistem operasi merupakan perangkat lunak fundamental yang menjadi jantung dari setiap sistem komputasi modern. Tanpa sistem operasi, perangkat keras komputer hanyalah sekumpulan komponen elektronik yang tidak dapat digunakan secara efektif oleh pengguna maupun aplikasi. Sistem operasi bertindak sebagai perantara antara perangkat keras dan perangkat lunak aplikasi, mengelola sumber daya komputasi, dan menyediakan antarmuka yang memungkinkan interaksi manusia-mesin yang efisien (Silberschatz *et al.*, 2021).

Dalam era komputasi modern yang ditandai oleh proliferasi perangkat pintar, komputasi awan, dan kecerdasan buatan, pemahaman mendalam tentang konsep dasar sistem operasi menjadi semakin krusial. Sistem operasi tidak lagi hanya beroperasi pada komputer desktop atau server konvensional, melainkan telah berevolusi untuk mendukung berbagai paradigma komputasi baru termasuk virtualisasi, containerisasi, dan komputasi tepi (edge computing) (Tanenbaum & Bos, 2023).

Bab ini membahas secara komprehensif konsep dasar dan perjalanan evolusi sistem operasi, mulai dari sistem batch sederhana pada era 1950-an hingga sistem operasi terdistribusi dan berbasis cloud yang digunakan saat ini.

Pemahaman tentang sejarah dan evolusi ini penting karena banyak konsep fundamental yang diterapkan pada sistem operasi modern merupakan penyempurnaan dari ide-ide yang pertama kali dikembangkan beberapa dekade lalu (Arpaci-Dusseau & Arpaci-Dusseau, 2023).

2.1 Definisi dan Fungsi Sistem Operasi

2.1.1 Definisi Sistem Operasi

Sistem operasi dapat didefinisikan dari berbagai perspektif bergantung pada sudut pandang yang digunakan. Dari perspektif pengguna, sistem operasi adalah perangkat lunak yang menyediakan antarmuka nyaman untuk menggunakan perangkat keras komputer. Dari perspektif sistem, sistem operasi adalah manajer sumber daya yang mengalokasikan dan mengelola memori, waktu prosesor, perangkat I/O, dan sumber daya lainnya di antara berbagai program dan pengguna yang bersaing (Silberschatz *et al.*, 2021)

Secara teknis, sistem operasi terdiri atas kernel yang selalu berjalan pada komputer serta sekumpulan perangkat lunak sistem yang membuat lingkungan komputasi menjadi berguna dan mudah dioperasikan. Kernel merupakan inti dari sistem operasi yang menangani interaksi langsung dengan perangkat keras dan menyediakan layanan dasar bagi semua komponen lain dari sistem (Kerrisk, 2021).

Pemahaman modern tentang sistem operasi juga mencakup konsep abstraksi yang kuat. Sistem operasi menyembunyikan kompleksitas perangkat keras di balik antarmuka yang sederhana dan konsisten, memungkinkan programmer untuk menulis aplikasi tanpa perlu mengetahui detail spesifik perangkat keras yang digunakan. Abstraksi ini merupakan salah satu

kontribusi terpenting sistem operasi terhadap kemajuan industri perangkat lunak (Anderson & Dahlin, 2022).

2.1.2 Fungsi Utama Sistem Operasi

Sistem operasi menjalankan beberapa fungsi utama yang dapat dikategorikan sebagai berikut:

a. Manajemen Proses

Proses adalah unit dasar eksekusi dalam sistem operasi. Manajemen proses mencakup pembuatan dan penghapusan proses pengguna maupun sistem, penjadwalan proses dan thread pada CPU, mekanisme sinkronisasi dan komunikasi antarproses, serta penanganan deadlock (Love, 2020).

Dalam lingkungan multicore dan multiprocessor yang umum saat ini, manajemen proses menjadi semakin kompleks. Sistem operasi harus mampu mendistribusikan beban kerja secara efisien di antara beberapa core atau prosesor, mengelola koherensi cache, dan meminimalkan overhead komunikasi antarprosesor (Corbet, Rubini, & Kroah-Hartman, 2020).

b. Manajemen Memori

Memori utama (RAM) adalah sumber daya yang sangat berharga dalam sistem komputer. Sistem operasi bertanggung jawab untuk melacak bagian memori yang sedang digunakan dan oleh siapa, memutuskan proses mana yang akan dimuat ke memori ketika ada ruang, dan mengalokasikan serta membebaskan ruang memori sesuai kebutuhan (Gorman, 2021).

Teknik manajemen memori virtual memungkinkan proses untuk menggunakan lebih banyak memori daripada yang tersedia secara fisik, dengan memanfaatkan ruang pada media penyimpanan sekunder sebagai ekstensi RAM. Inovasi dalam manajemen memori terus berkembang, termasuk teknik huge pages, memory-mapped files, dan NUMA-aware memory allocation untuk sistem dengan arsitektur memori non-uniform (Gorman, 2021).

c. Manajemen Penyimpanan

Sistem operasi menyediakan tampilan yang seragam dari berbagai media penyimpanan melalui abstraksi sistem berkas (file system). Fungsi ini mencakup pembuatan dan penghapusan berkas dan direktori, manipulasi berkas dan direktori, pemetaan berkas ke penyimpanan sekunder, dan pencadangan berkas ke media penyimpanan stabil (Bhatt & Patel, 2022).

Sistem berkas modern seperti ext4, NTFS, Btrfs, dan ZFS menyediakan fitur canggih seperti journaling untuk ketahanan terhadap kegagalan, kompresi data transparan, deduplikasi, dan snapshot untuk pencadangan efisien. Dengan pertumbuhan kapasitas penyimpanan yang pesat, manajemen penyimpanan semakin menjadi aspek kritis dalam desain sistem operasi modern (Bhatt & Patel, 2022).

d. Manajemen Perangkat I/O

Sistem operasi menyembunyikan keunikan perangkat keras I/O dari pengguna melalui subsistem I/O yang terdiri dari komponen manajemen memori termasuk buffering, caching, dan spooling; antarmuka driver perangkat yang

umum; driver untuk perangkat keras tertentu; serta mekanisme penanganan interrupt dan DMA (Corbet *et al.*, 2020).

e. Keamanan dan Proteksi

Dengan meningkatnya ancaman siber, fungsi keamanan sistem operasi semakin mendapat perhatian. Sistem operasi modern menyediakan mekanisme autentikasi pengguna, kontrol akses berbasis peran dan atribut, enkripsi sistem berkas, sandboxing aplikasi, dan deteksi intrusi tingkat kernel (Ahmad *et al.*, 2021). Pendekatan keamanan berbasis prinsip least privilege dan defense in depth kini menjadi standar dalam desain sistem operasi modern.

2.2 Arsitektur Sistem Operasi

2.2.1 Struktur Monolitik

Arsitektur monolitik merupakan pendekatan tertua dalam desain sistem operasi, di mana seluruh komponen sistem operasi berjalan dalam satu ruang alamat kernel yang sama. Dalam pendekatan ini, semua layanan sistem operasi seperti manajemen proses, manajemen memori, sistem berkas, dan driver perangkat diimplementasikan sebagai satu program besar yang berjalan dalam mode kernel (Tanenbaum & Bos, 2023).

Keunggulan utama arsitektur monolitik adalah performansi yang tinggi karena komunikasi antar komponen dilakukan melalui pemanggilan fungsi langsung tanpa overhead tambahan. Sistem operasi Linux dan sistem BSD adalah contoh sistem operasi dengan arsitektur monolitik yang masih banyak digunakan hingga saat ini. Meskipun disebut monolitik,

Linux modern sebenarnya mendukung modul kernel yang dapat dimuat secara dinamis, memberikan fleksibilitas tanpa mengorbankan performansi (Love, 2020).

2.2.2 Arsitektur Mikrokernel

Pendekatan mikrokernel berusaha meminimalkan fungsi yang dijalankan dalam mode kernel, hanya menyimpan komponen paling fundamental seperti komunikasi antar proses, manajemen memori dasar, dan penjadwalan CPU. Layanan lain seperti driver perangkat, sistem berkas, dan protokol jaringan diimplementasikan sebagai server yang berjalan dalam mode pengguna (Liedtke, 2020).

Keunggulan arsitektur mikrokernel meliputi modularitas yang tinggi, kemudahan pengujian dan verifikasi formal, serta ketahanan yang lebih baik karena kegagalan satu komponen tidak langsung menghancurkan keseluruhan sistem. Sistem operasi QNX, seL4, dan Minix 3 adalah contoh implementasi mikrokernel yang digunakan dalam sistem tertanam kritis dan penelitian keamanan sistem operasi (Heiser & Leslie, 2023).

2.2.3 Arsitektur Hybrid

Banyak sistem operasi modern mengadopsi pendekatan hybrid yang menggabungkan aspek arsitektur monolitik dan mikrokernel. Windows NT dan derivatifnya menggunakan kernel hybrid yang menempatkan beberapa subsistem yang biasanya berada di mode pengguna (seperti graphics subsystem) ke dalam kernel untuk alasan performansi, sementara tetap mempertahankan beberapa karakteristik mikrokernel (Russinovich *et al.*, 2022).

MacOS dan iOS dari Apple juga menggunakan arsitektur hybrid berbasis XNU yang menggabungkan komponen dari Mach (mikrokernel) dengan komponen BSD monolitik. Pendekatan ini memungkinkan Apple untuk mendapatkan manfaat dari kedua arsitektur: fleksibilitas dan portabilitas dari Mach, serta performansi dan kematangan dari komponen BSD (Levin, 2021).

2.2.4 Arsitektur Exokernel dan Unikernel

Exokernel merupakan pendekatan ekstrem yang meminimalkan abstraksi yang disediakan kernel, memberikan kontrol langsung atas sumber daya perangkat keras kepada aplikasi. Filosofinya adalah bahwa abstraksi tingkat rendah yang disediakan secara tradisional oleh kernel sebenarnya membatasi fleksibilitas dan performansi aplikasi (Engler *et al.*, 2020).

Unikernel adalah pendekatan modern yang semakin populer dalam komputasi cloud dan IoT, di mana aplikasi dikompilasi bersama dengan hanya komponen sistem operasi yang dibutuhkan ke dalam satu image yang dapat berjalan langsung di atas hypervisor atau perangkat keras. Pendekatan ini menghasilkan gambar yang sangat kecil, cepat di-boot, dan memiliki attack surface yang minimal (Madhavapeddy & Scott, 2022).

2.3 Evolusi Sistem Operasi

2.3.1 Era Pra-Sistem Operasi (1940-an – awal 1950-an)

Komputer generasi pertama seperti ENIAC dan UNIVAC tidak memiliki sistem operasi sama sekali. Pemrogram harus berinteraksi langsung dengan

perangkat keras melalui panel kontrol, dan setiap program harus memasukkan semua instruksi yang diperlukan untuk mengoperasikan perangkat keras. Komputer digunakan secara serial, artinya satu pengguna menggunakan seluruh mesin untuk satu pekerjaan pada satu waktu (Silberschatz *et al.*, 2021).

Ketiadaan sistem operasi membuat penggunaan komputer sangat tidak efisien. Sebagian besar waktu komputer terbuang sia-sia karena operator harus secara manual memuat program, menunggu eksekusi selesai, memeriksa hasil, dan kemudian mempersiapkan program berikutnya. Inefisiensi inilah yang mendorong pengembangan sistem operasi pertama (Anderson & Dahlin, 2022).

2.3.2 Sistem Batch (1950-an – 1960-an)

Generasi pertama sistem operasi adalah sistem batch, yang dikembangkan untuk mengatasi inefisiensi dalam penggunaan komputer. Dalam sistem ini, pekerjaan (*jobs*) dikumpulkan dalam kelompok (*batch*) dan dieksekusi secara berurutan tanpa intervensi manusia. Sistem operasi sederhana akan secara otomatis memuat dan menjalankan pekerjaan berikutnya setelah pekerjaan sebelumnya selesai (Tanenbaum & Bos, 2023).

IBM 701 dan IBM 704 adalah di antara komputer pertama yang menggunakan sistem batch primitif. Kemudian, General Motors Research Laboratories mengembangkan sistem operasi batch pertama untuk IBM 701 pada tahun 1956. IBSYS untuk IBM 7090/7094 merupakan penyempurnaan signifikan yang memperkenalkan konsep job control language (JCL) untuk menspesifikasikan persyaratan pekerjaan kepada sistem operasi.

2.3.3 Multiprogramming dan Time-Sharing (1960-an)

Multiprogramming merupakan inovasi revolusioner yang memungkinkan beberapa program berada di memori secara bersamaan. Ketika satu program sedang menunggu operasi I/O, CPU dapat mengeksekusi program lain, sehingga utilisasi CPU meningkat secara dramatis. Sistem operasi OS/360 dari IBM, yang diperkenalkan pada tahun 1964, merupakan salah satu implementasi multiprogramming yang paling berpengaruh (Love, 2020).

Time-sharing adalah ekstensi logis dari multiprogramming yang memberikan ilusi kepada setiap pengguna bahwa mereka memiliki komputer sendiri. Sistem CTSS (*Compatible Time-Sharing System*) yang dikembangkan di MIT pada awal 1960-an merupakan pelopor konsep ini. Time-sharing memungkinkan banyak pengguna berinteraksi secara bersamaan dengan sistem komputer melalui terminal, yang menjadi dasar bagi sistem interaktif yang kita kenal saat ini (Anderson & Dahlin, 2022).

Proyek Multics (*Multiplexed Information and Computing Service*), yang dimulai pada 1965 sebagai kolaborasi antara MIT, Bell Labs, dan General Electric, memperkenalkan banyak konsep penting yang masih relevan hingga saat ini, termasuk konsep sistem berkas hierarkis, segmentasi memori, dan keamanan berbasis ring privilege. Meskipun Multics sendiri tidak terlalu sukses secara komersial, ide-idenya sangat berpengaruh dalam pengembangan UNIX (Tanenbaum & Bos, 2023).

2.3.4 Era UNIX dan Perkembangannya (1969 - 1980-an)

UNIX, yang dikembangkan oleh Ken Thompson dan Dennis Ritchie di Bell Labs pada tahun 1969-1970, merupakan tonggak sejarah dalam perkembangan

sistem operasi. UNIX dirancang dengan filosofi sederhana namun kuat: tulis program yang melakukan satu hal dengan baik, tulis program yang bekerja sama, dan tulis program yang menangani aliran teks karena itu adalah antarmuka universal (Kernighan, 2020).

Penulisan ulang UNIX dalam bahasa C pada tahun 1973 merupakan terobosan revolusioner yang memungkinkan sistem operasi diportasi ke berbagai platform perangkat keras. Sebelumnya, sistem operasi ditulis dalam bahasa assembly yang spesifik untuk arsitektur tertentu. Portabilitas UNIX memungkinkan penyebarannya yang luas di universitas dan lembaga penelitian, yang pada gilirannya memicu inovasi dan pengembangan lebih lanjut (Love, 2020).

BSD (*Berkeley Software Distribution*) UNIX yang dikembangkan di University of California, Berkeley mulai pertengahan 1970-an, memperkenalkan banyak fitur penting seperti virtual memory, TCP/IP networking stack, dan sistem berkas yang lebih canggih. FreeBSD, NetBSD, OpenBSD, dan macOS semuanya merupakan turunan langsung atau tidak langsung dari BSD UNIX (Kerrisk, 2021).

2.3.5 Era Personal Computer (1980-an)

Munculnya komputer pribadi pada akhir 1970-an dan awal 1980-an membawa sistem operasi ke tangan pengguna umum yang tidak memiliki latar belakang teknis. CP/M (*Control Program for Microcomputers*) yang dikembangkan oleh Gary Kildall merupakan salah satu sistem operasi PC pertama yang sukses, digunakan secara luas pada komputer berbasis Intel 8080 dan Zilog Z80.

MS-DOS (*Microsoft Disk Operating System*), yang dikembangkan oleh Microsoft dan diadopsi oleh IBM untuk PC IBM yang diluncurkan pada 1981, mendominasi pasar PC selama satu dekade. Meskipun sederhana dan berbasis teks, MS-DOS meletakkan fondasi bagi ekosistem perangkat lunak PC yang kemudian berkembang pesat. Keterbatasan MS-DOS, terutama ketiadaan antarmuka grafis dan multitasking, mendorong pengembangan generasi sistem operasi berikutnya (Rusinovitch *et al.*, 2022).

Apple Macintosh yang diluncurkan pada tahun 1984 memperkenalkan antarmuka pengguna grafis (GUI) ke pasar massal, yang terinspirasi dari penelitian di Xerox PARC. Microsoft kemudian merespons dengan Windows, yang awalnya hanya merupakan shell grafis di atas MS-DOS namun kemudian berkembang menjadi sistem operasi penuh. Persaingan antara Apple dan Microsoft dalam memperebutkan pasar PC mendorong inovasi GUI yang pesat (Levin, 2021).

2.3.6 Era Sistem Operasi Modern (1990-an)

Tahun 1991 menandai lahirnya proyek Linux oleh Linus Torvalds, seorang mahasiswa dari Finlandia. Linux, yang dikembangkan sebagai reimplementasi bebas dari UNIX untuk arsitektur Intel x86, dengan cepat mendapat kontribusi dari komunitas pengembang di seluruh dunia. Model pengembangan open source yang diadopsi Linux terbukti sangat sukses dan menghasilkan sistem operasi yang kaya fitur dan sangat stabil (Torvalds & Diamond, 2021).

Windows NT yang diluncurkan Microsoft pada tahun 1993 merupakan terobosan besar karena memperkenalkan arsitektur 32-bit sejati, memori terproteksi, dan dukungan multiprocessor yang sesungguhnya. NT menjadi dasar bagi seluruh lini

Windows modern. Pada era yang sama, Sun Microsystems mengembangkan Solaris yang menjadi standar untuk workstation dan server berbasis UNIX di lingkungan enterprise (Rusinovich *et al.*, 2022).

Perkembangan internet pada pertengahan 1990-an secara fundamental mengubah ekspektasi dari sistem operasi. Kapabilitas jaringan yang kuat, keamanan, dan kemampuan menjalankan layanan web menjadi persyaratan penting. Sistem operasi harus mampu menangani ribuan koneksi simultan dengan overhead minimal, mendorong inovasi dalam subsistem jaringan dan I/O (Kerrisk, 2021).

2.3.7 Era Virtualisasi dan Cloud Computing (2000-an – 2010-an)

Virtualisasi, yang memungkinkan satu perangkat keras fisik menjalankan beberapa sistem operasi secara bersamaan, menjadi teknologi transformatif pada dekade pertama abad ke-21. VMware, yang didirikan pada 1998, mempopulerkan virtualisasi penuh (full virtualization) untuk arsitektur x86. Teknologi ini memungkinkan konsolidasi server yang dramatis, meningkatkan utilisasi perangkat keras, dan menyederhanakan manajemen infrastruktur TI (Barham *et al.*, 2023).

Paravirtualisasi yang diperkenalkan oleh Xen pada tahun 2003 menawarkan alternatif yang lebih efisien dengan memodifikasi guest OS agar sadar akan lingkungan virtual. Kemudian, Intel dan AMD memperkenalkan dukungan virtualisasi langsung pada hardware (VT-x dan AMD-V), yang memungkinkan virtualisasi penuh dengan overhead minimal dan mendorong adopsi teknologi ini secara masif (Barham *et al.*, 2023).

Cloud computing mengubah cara sistem operasi digunakan dan dikelola. Amazon Web Services, yang diluncurkan pada 2006, mempopulerkan konsep *Infrastructure as a Service* (IaaS) yang memungkinkan pengguna untuk menyewa kapasitas komputasi secara elastis sesuai kebutuhan. Dalam paradigma ini, sistem operasi tidak hanya perlu efisien dalam mengelola sumber daya fisik, tetapi juga harus mampu berjalan secara efektif sebagai virtual machine dalam lingkungan cloud (Liu & Deng, 2021).

2.3.8 Era Mobile dan Sistem Tertanam (2007 – sekarang)

Peluncuran iPhone pada tahun 2007 menandai dimulainya era baru komputasi mobile. iOS dan Android, dua sistem operasi mobile yang mendominasi pasar, didasarkan pada fondasi UNIX/Linux tetapi dioptimalkan secara ekstensif untuk perangkat dengan sumber daya terbatas, layar sentuh, konektivitas nirkabel, dan persyaratan daya baterai yang ketat (Kim & Lee, 2022).

Android, yang berbasis kernel Linux, mengadopsi model eksekusi berbasis mesin virtual (Dalvik, kemudian digantikan oleh Android *Runtime*/ART) untuk memungkinkan portabilitas aplikasi di antara berbagai perangkat keras yang berbeda. iOS, yang berbasis XNU (kernel macOS), menggunakan kompilasi native untuk performansi yang optimal. Persaingan antara kedua platform ini mendorong inovasi yang pesat dalam optimisasi performansi, efisiensi energi, dan keamanan mobile (Kim & Lee, 2022).

Internet of Things (IoT) membawa tantangan baru bagi sistem operasi. Miliaran perangkat tertanam dengan sumber daya sangat terbatas memerlukan sistem operasi yang ultra-ringan namun tetap aman dan dapat diperbarui dari jarak jauh. FreeRTOS, Zephyr, dan

RIOT adalah contoh sistem operasi yang dirancang khusus untuk lingkungan IoT (Yaqoob *et al.*, 2022).

2.3.9 Era Container dan Serverless (2010-an – sekarang)

Containerisasi, yang dipopulerkan oleh Docker pada tahun 2013, merupakan pendekatan revolusioner dalam deployment dan isolasi aplikasi. Container menggunakan fitur kernel Linux seperti namespaces dan cgroups untuk menyediakan isolasi ringan tanpa overhead penuh dari virtualisasi tradisional. Kubernetes, yang dikembangkan oleh Google dan dirilis sebagai open source pada 2014, menjadi platform orkestrasi container standar de facto (Bernstein, 2022).

Teknologi container mendorong sistem operasi untuk semakin mengekspos primitif keamanan dan isolasi tingkat kernel dengan cara yang lebih granular dan efisien. Fitur-fitur seperti user namespaces, seccomp filters, dan Linux Security Modules menjadi semakin penting dalam konteks keamanan container (Corbet *et al.*, 2020).

Komputasi serverless membawa abstraksi lebih jauh lagi, di mana pengembang tidak perlu memikirkan server atau sistem operasi sama sekali. Platform seperti AWS Lambda, Google Cloud Functions, dan Azure Functions mengelola infrastruktur di balik layar, termasuk sistem operasi dan runtime. Meskipun pengguna tidak berinteraksi langsung dengan sistem operasi, sistem operasi yang efisien dan cepat di-boot menjadi kunci untuk minimizing cold start latency dalam lingkungan serverless (Liu & Deng, 2021).

2.3.10 Sistem Operasi di Era AI dan Edge Computing

Perkembangan kecerdasan buatan dan pembelajaran mesin telah menciptakan kebutuhan komputasi baru yang mempengaruhi desain sistem operasi. Sistem operasi modern perlu menyediakan dukungan yang efisien untuk akselerator AI seperti GPU, TPU, dan NPU, termasuk manajemen memori khusus untuk array multidimensi besar dan primitif sinkronisasi yang efisien untuk komputasi paralel masif (Wang *et al.*, 2022).

Edge computing, yang memindahkan komputasi lebih dekat ke sumber data, menciptakan ekosistem komputasi terdistribusi yang heterogen. Sistem operasi untuk edge nodes harus mampu mengelola sumber daya terbatas, beradaptasi dengan kondisi jaringan yang bervariasi, dan berkoordinasi dengan sistem cloud secara efisien. Real-time operating systems (RTOS) yang dikombinasikan dengan kapabilitas machine learning di perangkat (*on-device ML*) menjadi area riset yang aktif (Yaqoob *et al.*, 2022).

2.4 Perbandingan Sistem Operasi Utama

Ekosistem sistem operasi saat ini didominasi oleh beberapa keluarga besar yang masing-masing memiliki karakteristik, kekuatan, dan domain aplikasi yang berbeda. Pemahaman tentang perbedaan dan persamaan antara sistem operasi-sistem operasi ini penting untuk memilih platform yang tepat untuk berbagai kebutuhan komputasi.

Tabel 2. 1 Perbandingan Sistem Operasi Utama

Aspek	Linux	Windows	macOS	Android/ iOS
Kernel	Monolitik modular	Hybrid (NT)	Hybrid (XNU)	Linux/ XNU
Lisensi	GPL (open source)	Proprietary	Proprietary	Apache/ Proprietary
Keamanan	SELinux/ AppArmor	Windows Defender + BitLocker	Gatekeeper + SIP	Sandbox ketat
Multiuser	Ya, penuh	Ya, penuh	Ya, penuh	Terbatas
Virtualisasi	KVM/ LXC native	Hyper-V	Hypervisor. framework	Terbatas
Penggunaan Cloud	Dominan (>90%)	Signifikan	Minimal	Minimal

2.5 Tantangan dan Tren Sistem Operasi Masa Depan

2.5.1 Keamanan dan Privasi

Keamanan sistem operasi menghadapi tantangan yang semakin kompleks seiring meningkatnya serangan siber yang canggih. Serangan tingkat hardware seperti Spectre dan Meltdown yang ditemukan pada tahun 2018 mengungkap kerentanan fundamental dalam cara prosesor modern bekerja dan memaksa perubahan signifikan dalam desain sistem operasi (Ahmad *et al.*, 2021). Mitigasi untuk serangan-serangan ini, meskipun diperlukan, menimbulkan dampak performansi yang signifikan dan memperlihatkan trade-off mendasar antara keamanan dan performansi dalam desain sistem operasi.

Sistem operasi masa depan perlu mengintegrasikan keamanan lebih dalam pada level arsitektur, bukan hanya sebagai lapisan tambahan. Teknologi seperti ARM TrustZone, Intel SGX, dan AMD SEV menyediakan enklave aman berbasis hardware yang dapat digunakan untuk melindungi data dan kode sensitif bahkan dari sistem operasi itu sendiri. Pengembangan sistem operasi yang mampu memanfaatkan fitur-fitur hardware ini secara efektif merupakan tantangan penelitian yang aktif (Heiser & Leslie, 2023).

2.5.2 Heterogenitas dan Akselerator

Sistem komputasi modern semakin heterogen, dengan CPU multicore general-purpose yang dikombinasikan dengan GPU, FPGA, NPU, dan akselerator khusus lainnya. Sistem operasi perlu menyediakan abstraksi yang efisien untuk mengelola ekosistem hardware yang beragam ini, memungkinkan aplikasi untuk memanfaatkan akselerator yang paling sesuai untuk tugas tertentu (Wang *et al.*, 2022).

Tantangan ini semakin kompleks dengan munculnya arsitektur memori heterogen di mana memori CPU dan memori akselerator mungkin memiliki karakteristik yang sangat berbeda (kapasitas, latensi, bandwidth). Sistem operasi harus mampu mengelola memori secara cerdas di antara tipe-tipe memori yang berbeda ini, dengan mempertimbangkan pola akses aplikasi dan karakteristik masing-masing tipe memori (Gorman, 2021).

2.5.3 Sistem Operasi untuk Komputasi Kuantum

Komputer kuantum yang sedang berkembang pesat memerlukan paradigma sistem operasi yang sama sekali baru. Tidak seperti komputer klasik yang beroperasi pada bit biner, komputer kuantum menggunakan qubit

yang dapat berada dalam superposisi keadaan. Manajemen qubit, koreksi kesalahan kuantum, dan penjadwalan operasi kuantum memerlukan abstraksi dan algoritma yang berbeda fundamental dari sistem operasi konvensional (Wang *et al.*, 2022) .

Meskipun komputer kuantum skala besar yang sepenuhnya fault-tolerant masih beberapa dekade lagi, penelitian tentang sistem operasi kuantum telah dimulai. Sistem seperti QOS (Quantum Operating System) sedang dikembangkan untuk mengelola sistem kuantum-klasik hybrid, di mana komputer klasik digunakan untuk mengontrol dan mengkoordinasikan operasi kuantum (Liu & Deng, 2021).

DAFTAR PUSTAKA

- Anderson, T., & Dahlin, M. (2022). Operating systems: Principles and practice (3rd ed.). Recursive Books.
- Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2023). Operating systems: Three easy pieces (Version 1.10). Arpaci-Dusseau Books. Diakses dari <https://pages.cs.wisc.edu/~remzi/OSTEP/>
- Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., ... Warfield, A. (2023). Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5), 164–177. <https://doi.org/10.1145/1165389.945462>
- Bernstein, D. (2022). Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, 1(3), 81–84. <https://doi.org/10.1109/MCC.2014.51>
- Bhatt, P., & Patel, R. (2022). Modern file system design: Journaling, copy-on-write, and deduplication techniques in Linux storage stacks. *Journal of Systems and Software*, 185, Article 111169. <https://doi.org/10.1016/j.jss.2021.111169>
- Cerdeira, D., Santos, N., Fonseca, P., & Pinto, S. (2020). SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems. *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 1416–1432. <https://doi.org/10.1109/SP40000.2020.00061>
- Corbet, J., Rubini, A., & Kroah-Hartman, G. (2020). *Linux device drivers* (4th ed.). O'Reilly Media.
- Gorman, M. (2021). *Understanding the Linux virtual memory manager* (2nd ed.). Prentice Hall.
- Heiser, G. (2020). *The seL4 microkernel: An introduction*. The seL4 Foundation. Diakses dari <https://sel4.systems/>

- Kernighan, B. W. (2020). UNIX: A history and a memoir. Kindle Direct Publishing.
- Kerrisk, M. (2021). The Linux programming interface: A Linux and UNIX system programming handbook (2nd ed.). No Starch Press.
- Kim, S., & Lee, H. (2022). A comparative analysis of Android and iOS security architectures and their evolution in the 5G era. *IEEE Transactions on Mobile Computing*, 21(8), 2710–2724. <https://doi.org/10.1109/TMC.2021.3065023>
- Klein, G., Andronick, J., Fernandez, M., Kuz, I., Murray, T., & Heiser, G. (2021). Formally verified software in the real world. *Communications of the ACM*, 61(10), 68–77. <https://doi.org/10.1145/3230627>
- Levin, J. (2021). *OS internals: Volume I – User mode (MacOS and iOS internals) (3rd ed.). TechnoGeeks Press.
- Liu, J., & Deng, Q. (2021). Cloud-native operating system design: Challenges and opportunities in serverless and container-based architectures. *Journal of Cloud Computing: Advances, Systems and Applications*, 10(1), 1–18. <https://doi.org/10.1186/s13677-021-00253-7>
- Love, R. (2020). Linux kernel development (3rd ed.). Addison-Wesley Professional.
- Madhavapeddy, A., & Scott, D. J. (2022). Unikernels: Rise of the virtual library operating system. *Communications of the ACM*, 57(1), 61–69. <https://doi.org/10.1145/2541883.2541895>

- Meneghello, F., Calore, M., Zennaro, D., Corubolo, F., & Sisinni, E. (2020). IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices. *IEEE Internet of Things Journal*, 6(5), 8182–8201. <https://doi.org/10.1109/JIOT.2019.2935189>
- Russinovich, M., Solomon, D. A., & Ionescu, A. (2022). *Windows internals, part 1: System architecture, processes, threads, memory management, and more* (8th ed.). Microsoft Press.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2021). *Operating system concepts* (10th ed.). John Wiley & Sons.
- Tanenbaum, A. S., & Bos, H. (2023). *Modern operating systems* (5th ed.). Pearson.
- Torvalds, L., & Diamond, D. (2021). *Just for fun: The story of an accidental revolutionary* (Revised ed.). HarperBusiness.
- Wang, H., Zhang, Y., & Chen, X. (2022). Operating system support for heterogeneous computing: GPU scheduling, memory management, and AI accelerator integration. *ACM Computing Surveys*, 54(9), 1–38. <https://doi.org/10.1145/3468886>
- Yaqoob, I., Ahmed, E., Hashem, I. A. T., Khan, A. I., Gani, A., Imran, M., & Guizani, M. (2022). Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. *IEEE Wireless Communications*, 24(3), 10–16. <https://doi.org/10.1109/MWC.2017.1600421>
- Zhang, Y., Luo, X., & Cai, H. (2020). Energy-efficient task scheduling for delay-sensitive applications in heterogeneous mobile edge computing. *IEEE Transactions on Mobile Computing*, 20(9), 2861–2874. <https://doi.org/10.1109/TMC.2020.2990219>

BAB 3

KOMPONEN SISTEM OPERASI PADA SISTEM MODERN

Oleh Johani S Pasaribu

3.1 Pengantar Sistem Operasi

Komputer modern terdiri atas satu atau lebih prosesor, memori utama, media penyimpanan seperti hard disk maupun Flash drive, printer, keyboard, mouse, monitor, antarmuka jaringan, serta berbagai perangkat masukan dan keluaran lainnya. Keseluruhan komponen tersebut membentuk sebuah sistem yang sangat kompleks.

Apabila setiap pengembang aplikasi harus memahami secara mendalam cara kerja seluruh perangkat keras tersebut, pengembangan perangkat lunak akan menjadi sangat sulit dan memerlukan waktu yang sangat lama. Selain itu, pengelolaan seluruh sumber daya komputer agar dapat dimanfaatkan secara optimal merupakan pekerjaan yang tidak sederhana. Oleh karena itu, komputer modern dilengkapi dengan perangkat lunak khusus yang disebut **sistem operasi (*operating system*)**.

Sistem operasi berfungsi menyediakan model komputer yang lebih sederhana, terstruktur, dan mudah digunakan oleh pengguna maupun aplikasi. Di samping itu, sistem operasi bertugas mengelola seluruh sumber daya perangkat keras sehingga dapat dimanfaatkan secara efisien dan aman. Dengan kata lain, sistem operasi menjadi penghubung antara

perangkat keras komputer dengan perangkat lunak aplikasi yang digunakan oleh pengguna (Abraham Silberschatz, Greg Gagne, 2018).

Perlu dipahami bahwa perangkat bergerak seperti smartphone dan tablet pada dasarnya juga merupakan komputer, hanya saja dikemas dalam bentuk yang lebih kecil dan menggunakan layar sentuh sebagai media interaksi utama. Oleh sebab itu, perangkat-perangkat tersebut juga memerlukan sistem operasi.

Sistem operasi iOS yang digunakan pada perangkat mobile Apple memiliki banyak kesamaan dengan macOS yang berjalan pada komputer desktop dan laptop Apple. Perbedaan ukuran perangkat dan penggunaan layar sentuh tidak mengubah secara mendasar fungsi utama sistem operasi. Demikian pula, perangkat Android menggunakan Linux sebagai sistem operasi inti yang berinteraksi langsung dengan perangkat keras, sedangkan antarmuka yang dikenal pengguna sebagai Android sebenarnya merupakan lapisan perangkat lunak yang berjalan di atas Linux.

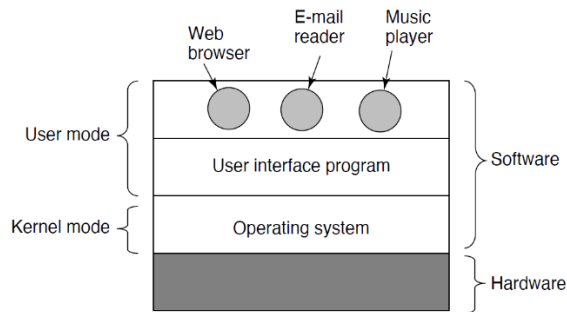
Karena macOS dan iOS berasal dari keluarga UNIX, sementara Linux dikembangkan dengan konsep yang serupa, keluarga sistem operasi UNIX dan turunannya menjadi salah satu sistem operasi yang paling banyak digunakan di dunia. Oleh sebab itu, pemahaman mengenai konsep UNIX sangat penting dalam mempelajari sistem operasi modern (Andrew S. Tanenbaum, 2023).

Sebagian besar pengguna mungkin telah menggunakan sistem operasi seperti Microsoft Windows, Linux, FreeBSD, atau macOS. Namun demikian, apa yang terlihat oleh pengguna sehari-hari belum tentu menggambarkan keseluruhan cara kerja sistem operasi yang sebenarnya.

3.2 Antarmuka Pengguna dan Sistem Operasi

Program yang berinteraksi langsung dengan pengguna sering disebut sebagai **shell** apabila berbasis teks, atau **Graphical User Interface (GUI)** apabila menggunakan ikon, jendela, dan elemen visual lainnya. Meskipun demikian, shell maupun GUI sebenarnya bukan bagian inti dari sistem operasi. Keduanya hanyalah program yang memanfaatkan layanan sistem operasi untuk menjalankan berbagai tugas.

Pada struktur komputer modern, perangkat keras berada pada lapisan paling bawah. Perangkat keras tersebut mencakup prosesor, memori, media penyimpanan, monitor, keyboard, dan berbagai komponen fisik lainnya. Di atas perangkat keras terdapat lapisan perangkat lunak, termasuk sistem operasi dan aplikasi pengguna.



Gambar 3. 1 Di mana posisi sistem operasi

(Andrew S. Tanenbaum, 2023)

Sebagian besar komputer modern memiliki dua mode operasi utama, yaitu **kernel mode** dan **user mode**. Sistem operasi menjalankan fungsi-fungsi intinya pada kernel mode atau supervisor mode. Pada mode ini, sistem operasi memiliki akses penuh terhadap seluruh perangkat keras dan dapat mengeksekusi seluruh instruksi yang tersedia pada mesin.

Sebaliknya, aplikasi pengguna dijalankan pada user mode, di mana hanya sebagian instruksi yang diizinkan untuk digunakan. Instruksi yang berkaitan dengan pengendalian perangkat keras, operasi input/output, maupun mekanisme keamanan sistem tidak dapat dijalankan secara langsung oleh aplikasi pengguna. Pembatasan ini bertujuan untuk menjaga stabilitas dan keamanan sistem secara keseluruhan.

Perbedaan antara kernel mode dan user mode merupakan konsep fundamental dalam sistem operasi modern dan menjadi dasar bagi berbagai mekanisme proteksi serta pengelolaan sumber daya.

3.3 Posisi Sistem Operasi dalam Sistem Komputer

Shell atau GUI merupakan antarmuka yang digunakan pengguna untuk menjalankan berbagai aplikasi, seperti peramban web, email, dan pemutar musik. Meskipun demikian, seluruh aplikasi tersebut bergantung pada layanan yang disediakan oleh sistem operasi yang berjalan langsung di atas perangkat keras.

Berbeda dengan aplikasi biasa, sistem operasi memiliki kendali penuh terhadap sumber daya komputer, seperti CPU, memori, dan perangkat input/output. Oleh karena itu, komponen inti sistem operasi dilindungi oleh mekanisme keamanan sehingga tidak dapat diubah secara bebas oleh pengguna.

Namun, batas antara sistem operasi dan perangkat lunak lain tidak selalu tegas. Pada sistem tertanam atau sistem berbasis interpreter, beberapa fungsi yang umumnya berada di kernel dapat dijalankan pada user mode. Selain itu, terdapat pula program pendukung, seperti pengelola kata sandi, yang meskipun bukan bagian inti sistem operasi, tetap memerlukan perlindungan khusus.

3.4 Kompleksitas Sistem Operasi Modern

Sistem operasi modern merupakan perangkat lunak yang sangat besar dan kompleks, terdiri atas jutaan baris kode yang terus berkembang. Kompleksitas tersebut membuat tidak ada satu orang pun yang memahami keseluruhan sistem secara menyeluruh.

Karena pengembangannya membutuhkan waktu, biaya, dan sumber daya yang besar, sistem operasi umumnya berevolusi secara bertahap daripada dibangun kembali dari awal. Hal ini terlihat pada perkembangan keluarga Windows, mulai dari Windows 95 hingga Windows 11, yang mempertahankan kemudahan penggunaan bagi pengguna meskipun mengalami perubahan besar pada struktur internalnya.

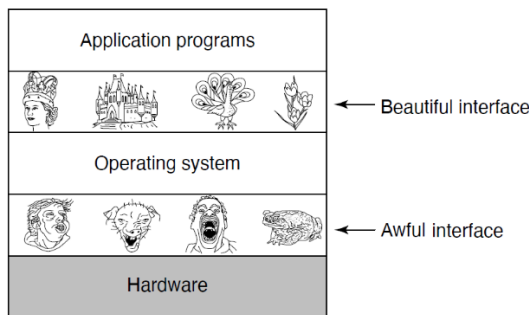
Perkembangan tersebut menunjukkan bahwa sistem operasi harus mampu menyeimbangkan inovasi teknologi dengan kebutuhan pengguna akan konsistensi dan kenyamanan dalam berinteraksi dengan komputer.

3.5 Pengertian Sistem Operasi

Sistem operasi adalah perangkat lunak yang menghubungkan pengguna dan aplikasi dengan perangkat keras komputer. Secara umum, sistem operasi memiliki dua fungsi utama, yaitu menyediakan abstraksi yang memudahkan pengembangan aplikasi serta mengelola sumber daya perangkat keras secara efisien. Oleh karena itu, sistem operasi dapat dipahami sebagai **mesin yang diperluas (*extended machine*)** dan **pengelola sumber daya (*resource manager*)**.

3.5.1 Sistem Operasi sebagai Mesin yang Diperluas (*Extended Machine*)

Perangkat keras komputer memiliki struktur dan mekanisme kerja yang kompleks sehingga sulit digunakan secara langsung oleh programmer. Untuk mengatasi hal tersebut, sistem operasi menyediakan berbagai abstraksi, seperti **driver perangkat** dan **berkas (file)**, yang menyederhanakan interaksi dengan perangkat keras seperti terlihat dalam Gambar 1.2.



Gambar 3.2 Sistem Operasi Mengubah Perangkat Keras yang Buruk Menjadi Abstraksi yang Indah

(Andrew S. Tanenbaum, 2023)

Melalui abstraksi tersebut, pengguna dan aplikasi dapat mengakses data tanpa harus memahami detail teknis media penyimpanan atau perangkat input/output. Dengan demikian, sistem operasi mengubah kompleksitas perangkat keras menjadi lingkungan kerja yang lebih sederhana, konsisten, dan mudah digunakan. Fokus utama sistem operasi adalah menyediakan layanan dan abstraksi bagi program aplikasi, sedangkan antarmuka pengguna seperti shell atau GUI hanya menjadi sarana interaksi bagi pengguna akhir.

3.5.2 Sistem Operasi sebagai Pengelola Sumber Daya (*Resource Manager*)

Selain menyediakan abstraksi, sistem operasi juga berfungsi mengelola sumber daya komputer, seperti CPU, memori, media penyimpanan, perangkat jaringan, dan perangkat input/output lainnya. Sistem operasi mengatur penggunaan sumber daya tersebut agar dapat digunakan secara bersamaan oleh banyak program tanpa menimbulkan konflik. Misalnya, pada penggunaan printer bersama, sistem operasi mengatur antrian pencetakan sehingga hasil keluaran tetap teratur.

Dalam lingkungan multi-pengguna, sistem operasi juga bertugas melindungi pengguna dan data dari akses yang tidak sah. Secara umum, tanggung jawab sistem operasi meliputi:

- Melacak penggunaan sumber daya komputer.
- Mengalokasikan sumber daya kepada program yang membutuhkan
- Mengendalikan akses dan keamanan sumber daya.
- Menyelesaikan konflik penggunaan antarprogram maupun pengguna.

3.6 Komponen Sistem Operasi Pada Sistem Modern

Sistem operasi modern terdiri atas sejumlah komponen utama yang bekerja secara terpadu untuk menyediakan lingkungan eksekusi bagi program dan pengguna. Agar sistem yang kompleks ini dapat dibangun dan dipelihara dengan baik, sistem operasi dibagi menjadi modul-modul kecil yang memiliki fungsi serta antarmuka yang jelas.

Perancangan sistem operasi merupakan pekerjaan yang sangat besar dan menuntut penetapan tujuan sistem sejak awal. Jenis sistem yang akan dibangun menjadi dasar dalam menentukan algoritma, strategi, dan mekanisme yang akan digunakan. Walaupun setiap sistem operasi memiliki struktur yang berbeda, sebagian besar sistem modern memiliki komponen-komponen utama yang serupa, yaitu (Abraham Silberschatz, Greg Gagne, 2018; Andrew S. Tanenbaum, 2023; GeeksforGeeks, 2025):

1. Manajemen Proses (*Process Management*)
2. Manajemen Memori (*Memory Management*)
3. Manajemen Penyimpanan Sekunder (*Secondary Storage Management*)
4. Manajemen Berkas (*File Management*)
5. Sistem Input/Output (*I/O System*)
6. Jaringan (*Networking*)
7. Sistem Interpreter Perintah (*Command Interpreter System*)
8. Sistem Proteksi (*Protection System*)

3.6.1 Manajemen Proses (*Process Management*)

CPU menjalankan berbagai program secara bersamaan. Selain mengeksekusi program pengguna, CPU juga digunakan untuk menjalankan aktivitas sistem lainnya yang disebut proses. Proses dapat dipahami sebagai program yang sedang dijalankan.

Sebuah pekerjaan batch, program pengguna pada sistem berbagi waktu, maupun tugas sistem seperti spooling merupakan contoh proses. Namun demikian, konsep proses sebenarnya lebih luas daripada sekadar program yang aktif. Untuk menjalankan tugasnya, suatu proses memerlukan berbagai sumber daya, seperti waktu CPU, memori, berkas, serta perangkat input/output. Sumber daya tersebut diberikan ketika proses dibuat. Selain itu, proses juga dapat menerima

data awal sebagai masukan. Sebagai contoh, proses yang bertugas menampilkan informasi suatu berkas akan menerima nama berkas tersebut sebagai input sebelum melakukan pemrosesan.

Perlu dipahami bahwa program dan proses adalah dua hal yang berbeda. Program bersifat pasif karena hanya berupa kumpulan instruksi, sedangkan proses bersifat aktif karena instruksi tersebut sedang dieksekusi. Dua proses yang menjalankan program yang sama tetap dianggap sebagai dua entitas yang berbeda karena masing-masing memiliki alur eksekusi sendiri.

Dalam sistem komputer, proses merupakan unit kerja utama. Sebagian proses menjalankan kode sistem operasi, sedangkan sebagian lainnya menjalankan kode pengguna. Seluruh proses tersebut dapat berjalan secara bersamaan.

Sistem operasi bertanggung jawab terhadap berbagai aktivitas berikut:

- Membuat dan menghapus proses pengguna maupun proses sistem.
- Menangguhkan dan melanjutkan eksekusi proses.
- Menyediakan mekanisme sinkronisasi antarproses.
- Menyediakan mekanisme penanganan deadlock.

3.6.2 Manajemen Memori (*Memory Management*)

Memori merupakan komponen yang sangat penting dalam sistem komputer modern. Memori dapat dipandang sebagai kumpulan lokasi penyimpanan yang masing-masing memiliki alamat tersendiri. CPU melakukan interaksi dengan memori melalui operasi pembacaan dan penulisan pada alamat tertentu.

Agar sebuah program dapat dijalankan, program tersebut harus dipetakan ke alamat memori yang sebenarnya dan dimuat ke dalam memori utama. Selama proses eksekusi berlangsung, CPU mengambil instruksi dan data dari lokasi memori tersebut.

Untuk meningkatkan efisiensi penggunaan CPU dan mempercepat respons sistem, beberapa proses biasanya ditempatkan di dalam memori secara bersamaan. Berbagai algoritma manajemen memori dapat diterapkan, dan pemilihannya sangat bergantung pada kondisi sistem serta rancangan perangkat keras yang digunakan. Setiap algoritma umumnya memerlukan dukungan perangkat keras tertentu.

Tanggung jawab sistem operasi dalam manajemen memori meliputi:

- Melacak bagian memori yang sedang digunakan beserta proses yang menggunakannya.
- Menentukan proses mana yang akan dimuat ke memori ketika ruang tersedia.
- Melakukan alokasi dan dealokasi memori sesuai kebutuhan.

3.6.3 Manajemen Penyimpanan Sekunder (*Secondary Storage Management*)

Tujuan utama sistem komputer adalah menjalankan program. Program dan data yang sedang diproses harus berada di memori utama selama eksekusi berlangsung. Karena kapasitas memori utama terbatas, diperlukan media penyimpanan sekunder untuk menyimpan data dan program secara permanen.

Sebagian besar sistem komputer modern menggunakan hard disk atau media penyimpanan serupa sebagai tempat penyimpanan utama bagi program dan data. Program seperti compiler, assembler,

editor, dan utilitas lainnya biasanya disimpan di media tersebut sebelum dimuat ke memori utama.

Pengelolaan media penyimpanan menjadi salah satu fungsi yang sangat penting dalam sistem operasi. Meskipun terdapat alternatif seperti pita magnetik, media tersebut umumnya lebih lambat dan hanya mendukung akses berurutan sehingga lebih sesuai untuk penyimpanan arsip yang jarang digunakan.

Dalam pengelolaan penyimpanan sekunder, sistem operasi bertugas untuk:

- Mengelola ruang kosong pada media penyimpanan.
- Melakukan alokasi ruang penyimpanan.
- Menentukan jadwal akses disk (disk scheduling).

3.6.4 Sistem Input/Output (I/O System)

Salah satu fungsi sistem operasi adalah menyembunyikan kerumitan perangkat keras dari pengguna maupun aplikasi. Pengguna tidak perlu memahami detail teknis setiap perangkat input dan output yang digunakan.

Sistem I/O biasanya mencakup beberapa komponen berikut:

- Sistem buffering dan caching.
- Kode driver perangkat umum.
- Driver khusus untuk perangkat keras tertentu.

Driver perangkat merupakan bagian yang memahami karakteristik unik dari suatu perangkat. Dengan adanya driver, sistem operasi dapat memberikan antarmuka yang seragam bagi aplikasi meskipun perangkat keras yang digunakan berbeda.

3.6.5 Manajemen Berkas (*File Management*)

Manajemen berkas merupakan salah satu layanan sistem operasi yang paling sering digunakan pengguna. Data dapat disimpan dalam berbagai media fisik, seperti pita magnetik, hard disk, maupun media penyimpanan lainnya yang masing-masing memiliki karakteristik tersendiri.

Untuk memudahkan penggunaan, sistem operasi menyediakan pandangan logis yang seragam terhadap penyimpanan data melalui konsep berkas (file). Berkas merupakan unit penyimpanan logis yang dipetakan oleh sistem operasi ke media fisik.

Berkas adalah kumpulan informasi yang saling berkaitan dan dibuat oleh pengguna. Berkas dapat berupa program, data numerik, teks, maupun data dengan format tertentu. Secara umum, berkas dapat dipandang sebagai urutan bit, byte, baris, atau rekaman yang maknanya ditentukan oleh pembuatnya.

Sistem operasi juga mengorganisasi berkas ke dalam direktori agar lebih mudah dikelola. Pada lingkungan multi-pengguna, mekanisme pengendalian hak akses diperlukan untuk menentukan siapa saja yang dapat menggunakan suatu berkas.

Tugas sistem operasi dalam manajemen berkas meliputi:

- Membuat dan menghapus berkas.
- Membuat dan menghapus direktori.
- Menyediakan operasi untuk memanipulasi berkas dan direktori.
- Memetakan berkas ke media penyimpanan fisik.
- Melakukan pencadangan data pada media yang bersifat permanen.

3.6.6 Sistem Proteksi (*Protection System*)

Setiap proses dalam sistem operasi harus dilindungi dari gangguan proses lainnya. Oleh karena itu, diperlukan mekanisme yang memastikan bahwa memori, CPU, berkas, dan sumber daya lainnya hanya dapat digunakan oleh proses yang memiliki hak akses yang sah.

Sebagai contoh, mekanisme pengalamatan memori mencegah proses mengakses ruang memori milik proses lain. Timer sistem memastikan bahwa tidak ada proses yang menguasai CPU tanpa batas waktu. Selain itu, operasi input/output langsung oleh proses pengguna juga dibatasi demi menjaga integritas perangkat keras.

Proteksi merupakan mekanisme pengendalian akses terhadap sumber daya sistem. Melalui proteksi, aturan penggunaan sumber daya dapat diterapkan dan ditegakkan secara konsisten.

Penerapan proteksi juga meningkatkan keandalan sistem karena kesalahan pada suatu subsistem dapat dideteksi lebih awal sebelum memengaruhi subsistem lainnya. Sumber daya yang tidak dilindungi rentan terhadap penyalahgunaan maupun akses oleh pihak yang tidak berwenang.

3.6.7 Jaringan (*Networking*)

Sistem terdistribusi terdiri atas sejumlah prosesor yang tidak berbagi memori maupun clock yang sama. Masing-masing prosesor memiliki memori lokal sendiri dan saling berkomunikasi melalui jaringan komunikasi.

Jaringan tersebut dapat dibangun menggunakan berbagai media, seperti bus berkecepatan tinggi atau saluran telekomunikasi. Topologi jaringan dapat berupa hubungan penuh maupun sebagian. Dalam

perancangannya, sistem jaringan harus mempertimbangkan strategi routing, metode koneksi, serta aspek keamanan komunikasi.

Keberadaan sistem terdistribusi memungkinkan pengguna mengakses sumber daya bersama sehingga memberikan berbagai keuntungan, antara lain:

- Peningkatan kecepatan pemrosesan.
- Ketersediaan data yang lebih baik.
- Keandalan sistem yang lebih tinggi.

3.6.8 Sistem Interpreter Perintah (*Command Interpreter System*)

Interpreter perintah merupakan salah satu komponen terpenting dalam sistem operasi karena berfungsi sebagai penghubung utama antara pengguna dan sistem.

Perintah kepada sistem operasi diberikan melalui pernyataan kontrol. Pada sistem batch maupun sistem berbagi waktu, program interpreter dijalankan secara otomatis ketika pekerjaan baru dimulai atau ketika pengguna melakukan login.

Komponen ini dikenal dengan berbagai nama, seperti:

- Control Card Interpreter.
- Command Line Interpreter.
- Shell pada sistem Unix.

Fungsi utamanya adalah membaca perintah dari pengguna, menafsirkannya, dan kemudian menjalankan perintah tersebut. Perintah-perintah yang diproses oleh interpreter mencakup berbagai aspek sistem operasi, seperti:

- Manajemen proses.
- Penanganan input/output.
- Pengelolaan penyimpanan sekunder.

- Manajemen memori utama.
- Akses sistem berkas.
- Mekanisme proteksi.
- Komunikasi jaringan.

Dengan demikian, command interpreter menjadi antarmuka utama yang memungkinkan pengguna memanfaatkan seluruh layanan yang disediakan oleh sistem operasi modern.

DAFTAR PUSTAKA

- Abraham Silberschatz, Greg Gagne, P. B. G. (2018) *Operating System Concepts*. New York: Wiley.
- Andrew S. Tanenbaum, H. B. (2023) *Modern Operating Systems*. New Jersey: Pearson Education, Inc.
- GeeksforGeeks (2025) *Components of Operating System*, *GeeksforGeeks*. Available at:
<https://www.geeksforgeeks.org/operating-systems/components-of-operating-system/>.

BAB 4

PENJADWALAN CPU PADA SISTEM MODERN DAN *REAL-TIME*

Oleh Aulia Nurul Iftitah

4.1 Bagaimana CPU Mengatur Waktu?

Bayangkan Anda adalah seorang montir komputer yang sangat andal di sebuah bengkel, tetapi pelanggan yang datang membawa laptop rusak ada puluhan orang. Anda tidak bisa memperbaiki semuanya sekaligus dalam satu detik yang sama, bukan? Anda harus memilih siapa yang dilayani duluan, berapa lama waktu untuk tiap laptop, dan kapan harus beralih ke laptop berikutnya.

Seperti itulah kerja CPU (otak komputer). CPU itu sangat cepat, tetapi pada satu waktu, satu inti (*core*) CPU sebenarnya hanya bisa mengerjakan satu instruksi. Sementara itu, ada ratusan aplikasi yang antre minta dilayani—mulai dari browser, pemutar musik, hingga sistem operasi itu sendiri. Penjadwalan CPU adalah cara sistem operasi membagi-bagi waktu CPU agar semua aplikasi bisa berjalan bergantian dengan sangat cepat, sampai kita merasa semuanya berjalan bersamaan.

4.1.1 Siklus Kerja CPU dan I/O

Kalau kita perhatikan, aktivitas sebuah program itu selalu berulang dalam dua fase:

1. CPU Burst: Saat program sedang sibuk menghitung, mengolah data, atau menjalankan logika.
2. I/O Burst: Saat program berhenti sejenak untuk menunggu sesuatu, misalnya menunggu pengguna mengetik keyboard, membaca data dari harddisk, atau mendownload file dari internet.

Penjadwal CPU memanfaatkan momen I/O Burst ini. Ketika sebuah aplikasi sedang sibuk menunggu data dari luar, CPU tidak boleh menganggur. Sistem operasi akan langsung mengambil CPU dan memberikannya kepada aplikasi lain yang sudah siap dihitung.

4.1.2 Preemptive vs Non-Preemptive

Dalam membagi waktu, ada dua sifat utama sistem operasi:

- Non-Preemptive (Suka Rela): Sekali sebuah aplikasi memakai CPU, aplikasi itu akan memegangnya terus sampai selesai atau sampai ia sendiri yang sukarela berhenti karena ingin menunggu I/O. Sistem operasi tidak bisa memaksa aplikasi itu berhenti di tengah jalan. Sistem lama menggunakan cara ini, tapi kelemahannya, jika ada satu program yang *hang* atau *error*, seluruh komputer ikut macet.
- Preemptive (Paksaan): Ini yang dipakai komputer modern sekarang. Sistem operasi punya kuasa penuh. Jika sebuah aplikasi sudah memakai CPU terlalu lama, sistem operasi bisa langsung menghentikannya secara paksa (melakukan *context switch*), menyimpan posisinya, dan memberikan

CPU ke aplikasi lain yang lebih darurat atau sudah mengantre lama.

4.1.3 Kriteria Penjadwalan yang Baik

Bagaimana kita tahu sebuah sistem operasi punya aturan penjadwalan yang bagus? Ada beberapa tolok ukurnya:

- Adil (*Fairness*): Semua program kebagian jatah CPU, tidak ada yang dicuekin sampai *freeze*.
- Throughput Maksimal: Jumlah tugas yang berhasil diselesaikan dalam satu satuan waktu sebanyak mungkin.
- Waktu Respons Cepat (*Response Time*): Begitu kita klik mouse atau ketik keyboard, komputer langsung merespons tanpa jeda yang terasa.

4.2 Algoritma Penjadwalan Klasik

Untuk mengatur antrean aplikasi, sistem operasi zaman dulu menggunakan beberapa rumus atau algoritma dasar. Konsep-konsep ini menjadi fondasi bagi sistem operasi modern sekarang.

4.2.1 Siapa Cepat Dia Dapat (FCFS) & Tugas Terpendek Didahulukan (SJF)

- *First-Come, First-Served* (FCFS): Konsepnya mirip seperti antrean di kasir supermarket. Siapa yang datang duluan, dia yang dilayani sampai selesai. Masalahnya, kalau di depan ada tugas yang sangat berat dan lama, tugas-tugas kecil di belakangnya harus menunggu lama sekali. Ini disebut efek konvoi (*convoy effect*).

- *Shortest Job First (SJF)*: Di sini, sistem operasi akan melihat durasi tugas. Yang paling cepat selesai akan ditaruh di depan antrean. Ini bagus untuk memangkas waktu tunggu, tapi masalahnya: dari mana sistem operasi bisa tahu pasti berapa lama sebuah program akan berjalan sebelum program itu selesai dijalankan? Jadi, algoritma ini lebih sering jadi teori atau prediksi saja.

4.2.2 Contoh Simulasi Algoritma FCFS (*First Come First Served*)

Misalkan terdapat empat proses yang masuk ke CPU secara bersamaan (Arrival Time = 0). Data proses dapat dilihat pada Tabel 7.1.

Tabel 4. 1 Data Proses FCFS

Proses	Arrival Time (AT)	Burst Time (BT)	Start Time (ST)	Finish Time (FT)	Waiting Time (WT)	Turnaround Time (TAT)
P1	0	8				
P2	0	4				
P3	0	9				
P4	0	5				

Karena menggunakan algoritma First Come First Served (FCFS), semua proses datang pada waktu 0, sehingga akan dieksekusi sesuai urutan kedatangannya, yaitu P1, P2, P3, kemudian P4.

Langkah 1. Menentukan *Start Time* (ST)

Proses pertama selalu dimulai pada waktu 0.

P1 mulai pada waktu 0 $\rightarrow$ ST = 0

Proses berikutnya baru dapat dimulai setelah proses sebelumnya selesai.

Langkah 2. Menentukan *Finish Time* (FT)

Rumusnya:

Finish Time = Start Time + Burst Time

Maka:

P1 $\rightarrow$ 0 + 8 = 8

Sehingga tabelnya menjadi:

Tabel 4. 2 Data Proses FCFS (*Start Time* dan *Finish Time*)

Proses	Arrival Time (AT)	Burst Time (BT)	Start Time (ST)	Finish Time (FT)
P1	0	8	0	8
P2	0	4		
P3	0	9		
P4	0	5		

Langkah 3. Menentukan *Start Time* (ST) Pada P2

P2 mulai setelah P1 selesai $\rightarrow$ ST = 8

Langkah 4. Menentukan *Finish Time* (FT) Pada P2

P2 $\rightarrow$ 8 + 4 = 12

Sehingga tabel untuk P1-P4:

Tabel 4. 3 Hasil Start Time dan *Finish Time* FCFS

Proses	Arrival Time (AT)	Burst Time (BT)	Start Time (ST)	Finish Time (FT)
P1	0	8	0	8
P2	0	4	8	12
P3	0	9	12	21
P4	0	5	21	26

Langkah 5. Menentukan Waiting Time (WT)

Waiting Time (WT) = Start Time – Arrival Time.

Langkah 6. Menentukan Turnaround Time (TAT)

Turnaround Time (TAT) = Finish Time – Arrival Time.

Sehingga hasil tabelnya:

Tabel 4. 4 Hasil Perhitungan FCFS

Proses	Arrival Time (AT)	Burst Time (BT)	Start Time (ST)	Finish Time (FT)	Waiting Time (WT)	Turnaround Time (TAT)
P1	0	8	0	8	0	8
P2	0	4	8	12	8	12
P3	0	9	12	21	12	21
P4	0	5	21	26	21	26

Pada algoritma FCFS, proses yang datang lebih dahulu akan memperoleh giliran lebih dulu tanpa memperhatikan lamanya waktu eksekusi. Akibatnya, jika proses pertama membutuhkan waktu yang cukup lama, maka proses berikutnya harus menunggu hingga proses tersebut selesai. Kondisi ini menyebabkan rata-rata waktu tunggu menjadi lebih besar.

4.2.3 Contoh Simulasi Algoritma SJF (*Shortest Job First*)

Data proses yang digunakan masih sama seperti pada simulasi sebelumnya.

Tabel 4. 5 Data Proses SJF

Proses	Arrival Time (AT)	Burst Time (BT)		Proses	Arrival Time (AT)	Burst Time (BT)
P1	0	8		P2	0	4
P2	0	4		P4	0	5
P3	0	9		P1	0	8
P4	0	5		P3	0	9

Pada algoritma Shortest Job First (SJF), proses dengan Burst Time paling kecil akan dijalankan terlebih dahulu. Oleh karena itu urutan proses menjadi P2, P4, P1, dan P3.

Setelah proses diurutkan berdasarkan nilai Burst Time (BT) dari yang terkecil hingga terbesar, langkah perhitungan selanjutnya dilakukan dengan cara yang sama seperti pada algoritma FCFS. Nilai **Start Time** (ST) ditentukan berdasarkan waktu selesai proses sebelumnya, kemudian **Finish Time** (FT) diperoleh dari penjumlahan antara *Start Time* (ST) dan *Burst Time* (BT). Selanjutnya, *Waiting Time* (WT) dihitung dari selisih *Start Time* (ST) dengan *Arrival Time* (AT), sedangkan *Turnaround Time* (TAT) diperoleh dari selisih *Finish Time* (FT) dengan *Arrival Time* (AT). Sehingga tabelnya sebagai berikut:

Tabel 4. 6 Hasil Perhitungan SJF

Proses	Arrival Time (AT)	Burst Time (BT)	Start Time (ST)	Finish Time (FT)	Waiting Time (WT)	Turnaround Time (TAT)
P2	0	4	0	4	0	4
P4	0	5	4	9	4	9
P1	0	8	9	17	9	17
P3	0	9	17	26	17	26

4.2.4 Sistem Antrean Bergiliran (*Round Robin*)

Ini adalah algoritma yang paling adil untuk sistem interaktif. Konsepnya menggunakan batas waktu kecil yang disebut Time Quantum (misalnya 10 milidetik).

Setiap aplikasi dalam antrean diberi jatah waktu 10 milidetik untuk memakai CPU. Jika waktunya habis dan tugasnya belum kelar, aplikasi tersebut dipotong, dipindahkan ke urutan paling belakang, dan CPU diberikan ke aplikasi berikutnya. Karena perputaran ini terjadi dalam hitungan milidetik, mata manusia melihatnya seolah-olah semua aplikasi berjalan bersamaan tanpa putus.

4.2.5 Mekanisme Tata Kerja Round Robin

Berdasarkan Tabel 7.6, sistem memiliki tiga proses utama yaitu P1, P2, dan P3 dengan alokasi waktu kebutuhan CPU (*Burst Time*) yang berbeda. Nilai Quantum ditetapkan sebesar 3. Aturan mutlak dalam skenario ini menentukan bahwa CPU hanya diizinkan mengerjakan suatu proses secara terus-menerus selama

maksimal 3 unit waktu, sebelum wajib berpindah ke proses berikutnya dalam antrean

Tabel 4. 7 Data Proses RR

Proses	Burst
P1	9
P2	5
P3	2

Proses pelaksanaan penghitungan secara bertahap digambarkan melalui siklus perputaran (*putaran*) berikut:

Langkah 1. Putaran Pertama

CPU mulai mengeksekusi proses P1. Karena Burst Time P1 adalah 9, CPU memprosesnya selama 3 unit waktu (batas maksimal quantum). Sisa waktu untuk P1 kini berkurang menjadi 6 unit waktu ($9 - 3 = 6$), lalu P1 dimasukkan kembali ke ujung antrean.

CPU beralih ke proses P2. Proses P2 dieksekusi selama 3 unit waktu, menyisakan 3 unit waktu ($5 - 3 = 2$).

CPU selanjutnya mengambil proses P3. P3 hanya membutuhkan 3 unit waktu untuk selesai. Setelah diproses sesuai batas quantum ($2 - 3 = 0$), proses P3 dinyatakan selesai sepenuhnya.

Langkah 2. Putaran Kedua

CPU kembali melayani proses yang masih tersisa dalam antrean. Proses P1 diambil kembali dan diberikan waktu 3 unit waktu lagi. Sisa waktunya kini berkurang menjadi 3 unit waktu ($6 - 3 = 3$).

Giliran beralih ke proses P2. Karena sisa waktu P2 hanya tinggal 3 unit waktu, operasi pemrosesan kali ini berhasil mengurangi nilainya hingga nol ($2 - 3 = 0$). Dengan demikian, proses P2 selesai sepenuhnya.

Langkah 3. Putaran Ketiga

Pada tahap ini, hanya proses P1 yang masih aktif di dalam sistem. CPU mengeksekusi sisa 3 unit waktu terakhir dari proses P1 ($3 - 3 = 0$). Dengan selesainya pelaksanaan P1, seluruh siklus penjadwalan untuk ketiga proses tersebut selesai dijalankan secara adil tanpa ada satu proses pun yang memonopoli CPU secara mutlak.

Tabel 4. 8 Tata Kerja RR

Putaran 1		Putaran 2		Putaran 3	
Proses	Burst	Proses	Burst	Proses	Burst
P1	$9 - 3 = 6$	P1	$6 - 3 = 3$	P1	$3 - 3 = 0$
P2	$5 - 3 = 2$	P2	$2 - 3 = 0$	P2	0
P3	$2 - 3 = 0$	P3	0	P3	0

Sehingga memiliki hasil perhitungan seperti Tabel 7.8.

Tabel 4. 9 Hasil Perhitungan RR

Putaran	P1	P2	P3
Awalan	9	5	2
1	6	2	0
2	3	0	0
3	0	0	0

4.2.6 Penjadwalan Berbasis Prioritas

Tidak semua aplikasi punya derajat yang sama. Menggerakkan kursor mouse tentu lebih penting daripada proses cek update software di latar belakang. Pada sistem berbasis prioritas, setiap tugas diberi angka prioritas. CPU akan selalu memilih tugas dengan prioritas tertinggi.

Tantangannya adalah risiko Starvation (kelaparan), di mana tugas berprioritas rendah tidak pernah kebagian CPU karena kalah terus dengan tugas baru berprioritas tinggi. Solusinya adalah *aging* (penuaan)—semakin lama sebuah tugas mengantre, prioritasnya akan dinaikkan pelan-pelan oleh sistem agar suatu saat ia bisa selesai.

4.2.7 Mekanisme Tata Kerja *Priority Scheduling*

Dalam skema pelaksanaan algoritma ini, sistem operasi dapat menggunakan pendekatan hierarki angka terkecil ke terbesar. Pada kasus ini, angka terkecil mewakili tingkat keutamaan yang paling tinggi (Prioritas Utama/Tertinggi = 1).

Tabel 4. 10 Data Proses PS

Proses	Priority
P1	3
P2	1
P3	4
P4	2

Melalui Tabel 4.10, operasi pemrosesan CPU akan dijalankan menurut kronologi jadwal berikut:

- P2 dieksekusi terlebih dahulu karena memegang status prioritas tertinggi (Nilai = 1).

- Setelah P2 rampung, CPU bertukar arah untuk memproses P4 yang berada pada posisi keutamaan kedua (Nilai = 2).
- Selanjutnya, giliran diserahkan kepada proses P1 (Nilai = 3).
- Terakhir, proses P3 akan diproses paling akhir karena memiliki nilai prioritas terbesar (Nilai = 4), yang berarti membawa tingkat kepentingan paling rendah dalam ekosistem sistem operasi pada saat itu.

Sehingga hasil yang didapatkan serpti pada Tabel 4.11.

Tabel 4. 11 Hasil Perhitungan PS

Proses	Priority		Proses	Priority
P1	3		P2	1
P2	1		P4	2
P3	4		P1	3
P4	2		P3	4

4.2.8 Antrean Bertingkat (*Multilevel Feedback Queue*)

Daripada hanya pakai satu jenis antrean, sistem modern menggabungkan beberapa antrean sekaligus. Tugas-tugas dikelompokkan: ada antrean khusus untuk aplikasi interaktif (yang butuh respon cepat) dan antrean untuk tugas latar belakang (seperti download file).

Hebatnya, algoritma ini punya *feedback* (umpan balik). Jika ada aplikasi di antrean interaktif tapi ternyata dia memakai CPU terlalu lama tanpa henti, sistem akan otomatis menurunkannya ke antrean kelas bawah agar tidak mengganggu aplikasi lain.

4.3 Penjadwalan pada Komputer Modern (*Multicore* dan *Multiprocessor*)

Komputer sekarang jarang yang hanya punya satu otak. Biasanya sudah *Multicore* (Dual-core, Quad-core, dan seterusnya). Mengatur satu CPU saja sudah pusing, apalagi jika harus mengatur banyak inti CPU sekaligus.

4.3.1 Tantangan Banyak Inti (*Multicore*)

Ketika ada banyak inti CPU, muncul masalah koordinasi. Jangan sampai ada dua inti CPU yang tidak sengaja mengambil dan mengerjakan tugas yang sama persis di waktu bersamaan, karena itu akan merusak data dan membuang-buang daya.

4.3.2 Keseimbangan Beban (*Load Balancing*)

Sistem operasi harus menjadi mandor yang adil melalui *load balancing*. Ada dua caranya:

- *Push Migration*: Ada satu proses pengawas yang memantau beban kerja. Jika dia melihat ada inti CPU yang senggang sementara inti yang lain sangat sibuk, dia akan "mendorong" tugas dari inti yang sibuk ke inti yang kosong.
- *Pull Migration*: Inti CPU yang sedang mengganggu berinisiatif untuk "menarik" sendiri tugas yang sedang mengantre di inti CPU lain.

4.3.3 Kedekatan Memori (*NUMA Awareness*)

Pada komputer server besar, memori (RAM) dibagi-bagi dan diletakkan dekat dengan inti CPU tertentu. Mengakses RAM yang jarak fisiknya dekat dengan inti CPU itu jauh lebih cepat daripada mengakses RAM yang jalurnya jauh di ujung *motherboard*.

Sistem operasi modern harus pintar (*NUMA-aware*): sebisa mungkin, ia akan menjadwalkan suatu tugas di inti CPU yang letaknya bersebelahan dengan lokasi memori tempat data tugas tersebut disimpan (disebut *Processor Affinity*).

4.4 Penjadwalan di Dunia Virtual (Virtualisasi, Cloud, dan Kontainer)

Sekarang banyak perusahaan tidak lagi membeli komputer fisik untuk setiap aplikasi, melainkan menyewa *Cloud* (seperti AWS atau Google Cloud) atau menggunakan teknologi Virtual Machine (VM) dan Kontainer (Docker).

4.4.1 Berbagi CPU di Cloud

Di pusat data (*data center*), satu komputer server fisik yang sangat besar bisa dibagi-bagi untuk disewakan kepada puluhan pengguna berbeda (*multi-tenant*). Tantangan penjadwalan di sini adalah Fair-Share. Sistem operasi server harus memastikan bahwa pengguna A yang membayar paket murah tidak mengambil jatah daya CPU milik pengguna B yang membayar paket premium.

4.4.2 Penjadwalan Virtual Machine vs Kontainer

Banyak penelitian ilmiah membuktikan bahwa teknologi Kontainer (seperti Docker) umumnya memiliki beban kerja (*overhead*) yang jauh lebih ringan dan cepat saat meminta jatah ke CPU dibandingkan Virtual Machine tradisional seperti KVM atau VMware (Yadav, 2018). Bahkan untuk urusan sistem yang butuh waktu tanggap instan (*real-time workloads*), sistem berbasis kontainer mencatat waktu tunda (*latency*) yang sangat rendah (Abeni, 2024).

Namun, keunggulan kontainer ini tidak mutlak. Untuk beberapa jenis tugas khusus yang sangat padat aktivitas baca-tulis datanya (*I/O & storage heavy*), Virtual Machine (VM) bisa mengimbangi atau bahkan memberikan performa yang sedikit lebih baik daripada kontainer (Cui, 2020).

- Pada Virtual Machine (VM): Ada lapisan tambahan yang disebut *Hypervisor*. Penjadwalan terjadi dua kali (dua lapis). Lapisan pertama, OS di dalam VM menjadwalkan aplikasinya sendiri. Lapisan kedua, *Hypervisor* menjadwalkan VM tersebut pada CPU fisik asli. Ini membuat performanya sedikit lebih berat.
- Pada Kontainer (*Docker*): Kontainer tidak membawa OS sendiri, mereka menumpang langsung pada OS komputer induk. Jadi, penjadwalannya jauh lebih ringkas dan cepat karena langsung diatur oleh satu penjadwal milik OS induk.

4.5 Mengetahui Penjadwalan *Real-Time* (Sistem Waktu-Nyata)

Komputer biasa (seperti laptop kita) mengutamakan kenyamanan pengguna. Jika aplikasi agak lambat sedikit, kita paling hanya kesal. Namun, pada Sistem *Real-Time*, keterlambatan adalah sebuah kegagalan fatal.

4.5.1 Apa itu Sistem *Real-Time*?

Sistem *Real-Time* adalah sistem yang kebenaran hasilnya tidak hanya tergantung pada "apakah hitungannya benar", tapi juga pada "kapan hasil itu keluar". Mereka dikejar oleh waktu mati yang disebut Deadline.

4.5.2 Sistem Saklek vs Sistem Toleran

- *Hard Real-Time* (Saklek): Tugas harus selesai sebelum *deadline*. Jika terlambat satu milidetik saja, akibatnya bisa fatal atau bencana. Contohnya: sistem *airbag* mobil, sistem kendali rem pesawat, atau robot medis operasi jantung.
- *Soft Real-Time* (Toleran): Sistem mengutamakan tugas penting ini agar selesai cepat, tapi kalau sesekali terlambat, dampaknya tidak merusak materi atau nyawa, hanya penurunan kualitas saja. Contohnya: *live streaming* video (kalau telat, videonya hanya patah-patah sedikit).

4.5.3 Mengatur Jadwal Berdasarkan Waktu (RM dan EDF)

Bagaimana menjamin tugas real-time tidak melewati *deadline*? Ada dua pendekatan populer:

- *Rate Monotonic* (RM): Tugas yang paling sering muncul (periodenya paling pendek) otomatis dianggap paling darurat dan diberi prioritas tertinggi. Aturan ini sifatnya kaku dan permanen sejak awal.
- *Earliest Deadline First* (EDF): Ini lebih dinamis. Sistem melihat tugas mana yang waktu tenggat jalannya paling mepet dengan jam sekarang, dialah yang langsung diprioritaskan. Prioritas bisa berubah-ubah di tengah jalan tergantung sisa waktu *deadline*.

4.6 Hemat Energi pada Embedded System dan IoT

Perangkat kecil seperti jam tangan pintar (*smartwatch*), sensor banjir, atau perangkat pintar IoT lainnya tidak colok ke listrik dinding; mereka hidup dari baterai. Oleh karena itu, cara CPU menjadwalkan tugas harus ramah energi.

4.6.1 Penjadwalan Hemat Daya (DVFS)

DVFS singkatan dari *Dynamic Voltage and Frequency Scaling*. Intinya: buat apa CPU berlari kencang kalau tugasnya hanya membaca sensor suhu ruangan? Penjadwal yang pintar akan menurunkan kecepatan (*frekuensi*) dan tegangan listrik CPU seminimal mungkin agar baterai awet. Begitu ada tugas berat datang, barulah kecepatan CPU dinaikkan kembali ke performa maksimal.

Para peneliti sudah membuat berbagai rumus nyata untuk hal ini. Sebagai contoh, pada sistem operasi FreeRTOS, penambahan algoritma DVFS dinamis berhasil menghemat konsumsi energi baterai sebesar 5% hingga 10% tanpa mengurangi kecepatan performa aplikasinya (Ramegowda, 2022). Bahkan untuk sistem dengan tingkat kedaruratan campuran (*mixed-criticality*), modifikasi algoritma berbasis DVFS sanggup memotong konsumsi daya rata-rata hingga 24,55% (Zhang, 2023).

4.6.2 Sistem Berbasis Waktu vs Berbasis Event

- *Time-Triggered*: Perangkat dinyalakan dan bekerja berdasarkan alarm waktu yang kaku. Misalnya: bangun setiap 1 jam sekali, baca data, kirim, lalu tidur lagi.
- *Event-Driven*: Perangkat dalam kondisi tidur total (*deep sleep*) demi menghemat baterai, dan baru akan bangun seketika jika ada kejadian (*event*)

khusus. Riset pada sistem operasi Zephyr RTOS menunjukkan bahwa membiarkan CPU menyelesaikan tugasnya dengan sangat cepat lalu langsung menjatuhkannya ke mode tidur lelap (*deep sleep*) jauh lebih efektif menghemat baterai (Torres, 2025).

4.6.3 Mengintip RTOS (FreeRTOS & Zephyr)

Pada perangkat IoT, kita tidak memakai Windows atau Linux karena terlalu besar dan berat. Kita menggunakan RTOS (*Real-Time Operating System*) berukuran mikro seperti FreeRTOS atau Zephyr. Penjadwal pada RTOS dibuat sangat sederhana, andal, dan prediktif, sehingga mereka bisa berjalan stabil bertahun-tahun pada cip komputer yang kapasitas memorinya sangat kecil.

4.7 Studi Kasus: Cara Kerja OS yang Kita Gunakan Hari Ini

Mari kita lihat bagaimana teori-teori di atas diterapkan pada sistem operasi yang kita pakai sehari-hari.

4.7.1 Penjadwal di Linux (*Completely Fair Scheduler* - CFS)

Linux menggunakan penjadwal bernama CFS. Sesuai namanya, tujuan utamanya adalah "keadilan yang sempurna". CFS mencatat waktu pemakaian CPU dari setiap program dalam variabel bernama *vruntime* (*virtual runtime*). Program yang paling jarang memakai CPU otomatis punya angka *vruntime* terkecil, dan dialah yang selalu dipilih Linux untuk berjalan berikutnya.

Konsep keadilan ini terus dikembangkan. Untuk kebutuhan komputasi awan modern (*serverless cloud*), penjadwal CFS dimodifikasi agar bisa memprioritaskan

fungsi-fungsi aplikasi yang paling ringan secara dinamis, berhasil meningkatkan ketepatan waktu respons sistem sebesar 5% hingga 30% (Isstaif, 2023).

4.7.2 Penjadwal di Windows dan Android

- Windows: Menggunakan sistem prioritas dinamis yang sangat berorientasi pada kenyamanan pengguna. Jika Anda sedang membuka sebuah aplikasi (misalnya Word) dan jendelanya sedang aktif di depan layar (*foreground*), Windows akan otomatis mendongkrak prioritas aplikasi tersebut agar terasa sangat lancar dan responsif saat Anda mengetik.
- Android: Karena berjalan di perangkat layar sentuh yang butuh respon instan, Android memodifikasi cara kerja CFS bawaan Linux. Peneliti menemukan bahwa jika sistem tidak diatur, aplikasi latar belakang bisa mengganggu kehalusan layar HP. Oleh karena itu, Android menggunakan sistem khusus yang mengenali rantai aktivitas pengguna secara langsung; begitu layar disentuh, sistem otomatis memotong tugas lain dan memberikan prioritas penuh pada visual layar, yang terbukti memangkas waktu tunda (*response time*) interaksi hingga 77,35% (Huh, 2015).

4.7.3 Tantangan CPU Heterogen (big.LITTLE)

Hampir semua prosesor HP zaman sekarang (seperti Snapdragon atau MediaTek) menggunakan arsitektur big.LITTLE. Di dalam satu cip, ada dua jenis inti CPU yang berbeda:

- Inti "LITTLE" (Hemat Daya): Kecil, lambat, tapi sangat irit baterai.

- Inti "big" (Performa Tinggi): Besar, sangat cepat, tapi boros baterai.

Menyeimbangkan keadilan (*fairness*) dan keawetan baterai pada arsitektur ini sangatlah menantang. Oleh karena itu, sistem operasi modern saat ini dilengkapi dengan kerangka kerja cerdas seperti Heterogeneous Fairness-Aware Energy Efficient (HFEE) yang mengatur kapan teknologi DVFS harus diturunkan dan ke inti mana sebuah tugas harus dilemparkan (Salami, 2021). Jika Anda hanya berkiriman pesan, tugas dilempar ke inti LITTLE dengan daya rendah. Namun, begitu game berat dibuka, penjadwal langsung memindahkan tugas tersebut ke inti big demi menjaga performa (Salami, 2021)

Tugas berat sistem operasi saat ini adalah mendeteksi jenis aplikasi Anda secara instan. Jika Anda hanya membalas pesan WhatsApp atau mendengarkan musik, sistem operasi akan melempar tugas itu ke inti LITTLE. Namun, begitu Anda membuka game 3D yang berat, sistem operasi akan langsung memindahkan tugas tersebut ke inti *big* agar tidak patah-patah.

DAFTAR PUSTAKA

- Abeni, Luca. 2024. Virtualized Real-Time Workloads in Containers and Virtual Machines. *Journal of System Architecture*, 154, 1–9. <https://www.sciencedirect.com/science/article/pii/S1383762124001759?via%3Dihub>.
- Huh, Sungju, Yoo, Jonghun, dan Hong, Seongsoo. 2015. Cross-Layer Resource and Scheduling for Improving Interactivity in Android. *Software: Practice and Experience*, 45(11), 1549–1570. <https://onlinelibrary.wiley.com/doi/10.1002/spe.2285>.
- Isstaif, Al Amjad Tawfiq, dan Mortier, Richard. 2023. Towards Latency-Aware Linux Scheduling for Serverless Workloads. *Proceedings of the 1st Workshop on Serverless Systems, Applications and Methodologies*, 19–26. <https://dl.acm.org/doi/10.1145/3592533.3592807>.
- Ramegowda, Deepak, dan Lin, Man. 2022. Energy Efficient Mixed Task Handling on Real-Time Embedded System Using FreeRTOS. *Journal of Systems Architecture*, 131. <https://www.sciencedirect.com/science/article/abs/pii/S1383762122002016?via%3Dihub>.
- Salami, B., Noori, Hamid, dan Naghibzadeh, Mahmoud. 2021. Fairness-Aware Energy Efficient Scheduling on Heterogeneous Multi-Core Processors. *IEEE Transactions on Computers*, 70(01), 72–82. <https://ieeexplore.ieee.org/document/9055136>.

- Torres, Gustavo M., Souza, Joao V. L. De, Aguiar, Joao G. F., dan Sousa, Janisley O. De. 2025. Kemampuan Performance and Power Cosumption of Real-Time Task Scheduling for Embedded Systems Devices. *IEEE Latin America Transactions*), 1–6.
<https://ieeexplore.ieee.org/document/10929994/metrics>.
- Yadav, R., Sousa, E., Callou, G. 2018. Performance Comparison Between Virtual Machines and Docker Containers. *IEEE Latin America Transactions*, 16(08), 2282–2288.
<https://ieeexplore.ieee.org/document/8528247>
- Zhang, Yi Wen. 2023. DVFS-Based Energy-Aware Scheduling of Imprecise Mixed-Criticality Real-Time Tasks. *Journal of Systems Architecture*, 137.
<https://www.sciencedirect.com/science/article/abs/pii/S1383762123000280?via%3Dihub>.

BAB 5

MANAJEMEN INPUT/OUTPUT DAN PERANGKAT MODERN

Oleh Anas

Sistem operasi modern menangani berbagai macam perangkat keras yang memiliki karakteristik, kecepatan, dan metode akses yang sangat beragam. Manajemen input/output (I/O) merupakan salah satu fungsi inti sistem operasi yang bertanggung jawab atas koordinasi, pengendalian, dan komunikasi antara perangkat keras dengan sistem komputer secara efisien. Tanpa mekanisme manajemen I/O yang baik, kinerja sistem secara keseluruhan akan menurun drastis karena perangkat I/O umumnya memiliki kecepatan yang jauh lebih lambat dibandingkan dengan unit pemrosesan utama (CPU) dan memori utama (Silberschatz, dkk., 2018).

Perangkat I/O mencakup berbagai komponen seperti keyboard, mouse, monitor, printer, disk drive, jaringan, dan perangkat multimedia. Masing-masing perangkat memiliki protokol komunikasi, format data, dan kecepatan transfer yang berbeda-beda. Sistem operasi harus mampu menyediakan antarmuka yang seragam (*uniform interface*) sehingga aplikasi pengguna dapat mengakses berbagai perangkat tanpa perlu mengetahui detail teknis spesifik dari setiap perangkat (Tanenbaum, 2014).

Perkembangan teknologi perangkat keras modern seperti *Solid-State Drive (SSD)*, *Non-Volatile Memory Express (NVMe)*, Universal Serial Bus (USB) generasi terbaru, dan perangkat berbasis cloud computing telah mengubah paradigma manajemen I/O. Kecepatan transfer data yang semakin tinggi dan latensi yang semakin rendah menuntut sistem operasi untuk mengadopsi teknik-teknik baru dalam penanganan I/O agar dapat memaksimalkan potensi perangkat-perangkat modern tersebut (Stallings, 2018).

Bab ini akan membahas secara komprehensif arsitektur manajemen I/O, teknik-teknik penanganan I/O, perangkat-perangkat modern, serta strategi optimisasi kinerja I/O dalam sistem operasi. Pemahaman yang mendalam tentang manajemen I/O sangat penting bagi mahasiswa, praktisi, dan pengembang sistem dalam merancang dan mengimplementasikan sistem komputer yang responsif dan efisien.

5.1 Arsitektur dan Karakteristik Perangkat I/O

5.1.1 Klasifikasi Perangkat I/O

Perangkat input/output dapat diklasifikasikan berdasarkan beberapa kriteria, antara lain berdasarkan arah aliran data, kecepatan transfer, dan metode akses. Berdasarkan arah aliran data, perangkat I/O dapat dibedakan menjadi perangkat input (membawa data ke sistem), perangkat output (mengirim data keluar dari sistem), dan perangkat input/output dua arah (bidirectional) seperti disk dan jaringan (Silberschatz, dkk., 2018).

Berdasarkan metode akses, perangkat I/O dapat dikelompokkan menjadi perangkat akses blok (block-access devices) dan perangkat akses karakter (character-access devices). Perangkat akses blok, seperti

hard disk dan SSD, mengorganisasi data dalam unit-unit berukuran tetap yang disebut blok. Setiap blok memiliki alamat unik sehingga memungkinkan akses acak (*random access*) ke blok mana pun. Sebaliknya, perangkat akses karakter, seperti keyboard, mouse, dan printer, mengalirkan data sebagai urutan byte tanpa struktur blok dan umumnya hanya mendukung akses sekuensial (Tanenbaum, 2014).

Berdasarkan kecepatan, perangkat I/O dapat dibedakan menjadi perangkat berkecepatan rendah (*low-speed*), menengah (*medium-speed*), dan tinggi (*high-speed*). Perangkat berkecepatan rendah meliputi keyboard dan mouse dengan kecepatan transfer beberapa byte per detik. Perangkat kecepatan menengah mencakup printer dan scanner dengan kecepatan ratusan hingga ribuan byte per detik. Perangkat kecepatan tinggi mencakup disk, jaringan gigabit, dan perangkat grafis dengan kecepatan transfer jutaan hingga miliaran byte per detik (Stallings, 2018). Tabel 5.1 menunjukkan klasifikasi perangkat I/O berdasarkan kecepatan transfer dan metode akses.

Tabel 5.1 Klasifikasi Perangkat I/O Berdasarkan Kecepatan dan Metode Akses

Klasifikasi	Perangkat Contoh	Metode Akses	Kecepatan Transfer	Karakteristik
Perangkat Input	Keyboard, Mouse, Scanner	Karakter	Rendah (1-100 Bps)	Interaktif, buffer kecil
Perangkat Output	Monitor, Printer, Speaker	Karakter	Rendah-Menengah (100-50 KBps)	Buffering diperlukan
Perangkat Penyimpanan	Hard Disk, SSD, Optical Drive	Blok	Tinggi (100 MBps-7 GBps)	Akses acak, caching penting

Klasifikasi	Perangkat Contoh	Metode Akses	Kecepatan Transfer	Karakteristik
Perangkat Jaringan	Ethernet, Wi-Fi Adapter	Blok/Karakter	Tinggi (1 Gbps–10 Gbps)	Protokol kompleks, buffer besar
Perangkat Multimedia	Webcam, Microphone, GPU	Blok/Karakter	Sangat Tinggi (GBps–TBps)	Real-time, latency-sensitive

Sumber: Diadaptasi dari Silberschatz, Galvin, & Gagne (2018) dan Stallings (2018)

5.1.2 Perangkat Penyimpanan Modern

Perangkat penyimpanan merupakan komponen vital dalam sistem I/O yang menentukan kecepatan dan kapasitas sistem secara keseluruhan. *Hard Disk Drive* (HDD) tradisional menggunakan media magnetik berputar dengan kecepatan 5.400 hingga 15.000 RPM. Meskipun menawarkan kapasitas besar dengan biaya per gigabyte rendah, HDD memiliki keterbatasan dalam hal kecepatan akses akibat komponen mekanis (Silberschatz, dkk., 2018).

Solid-State Drive (SSD) merupakan perangkat penyimpanan modern yang menggunakan memori flash NAND sebagai media penyimpanannya. Tidak adanya komponen bergerak membuat SSD memiliki waktu akses yang jauh lebih cepat, konsumsi daya lebih rendah, dan daya tahan fisik yang lebih baik dibandingkan HDD. SSD menggunakan antarmuka SATA (Serial ATA) generasi III dengan kecepatan teoritis hingga 600 MBps atau antarmuka NVMe (*Non-Volatile Memory Express*) melalui slot PCIe dengan kecepatan hingga 7 GBps (Stallings, 2018).

NVMe adalah protokol komunikasi dan antarmuka perangkat keras yang dirancang khusus untuk media penyimpanan non-volatile seperti SSD. NVMe mengoptimalkan subsistem memori dengan mengurangi overhead, mendukung antrian perangkat keras yang lebih dalam (hingga 64.000 antrian), dan memungkinkan paralelisasi operasi I/O yang lebih efisien dibandingkan dengan protokol AHCI tradisional (Tanenbaum, 2014). Tabel 5.2 membandingkan karakteristik antara HDD, SSD SATA, dan SSD NVMe.

Tabel 5. 2 Perbandingan Karakteristik Perangkat Penyimpanan Modern

Parameter	HDD (SATA)	SSD (SATA III)	SSD (NVMe PCIe 4.0)
Kecepatan Baca Sekuensial	100–200 MBps	450–550 MBps	3.500–7.000 MBps
Kecepatan Tulis Sekuensial	80–160 MBps	400–520 MBps	3.000–5.000 MBps
IOPS (Random Read)	50–200	75.000–100.000	500.000–1.000.000
Latensi Akses	5–10 ms	0.1–0.2 ms	0.02–0.1 ms
Daya Tahan Guncangan	Rendah	Tinggi	Tinggi
Konsumsi Daya	6–15 Watt	2–4 Watt	3–8 Watt
Kapasitas Umum	1–20 TB	128 GB–4 TB	256 GB–4 TB

Sumber: Diadaptasi dari Stallings (2018) dan dokumentasi teknis industri (2023)

5.2 Teknik Penanganan I/O

Sistem operasi menggunakan beberapa teknik untuk menangani operasi I/O, masing-masing dengan kompleksitas, overhead, dan efisiensi yang berbeda. Pemilihan teknik yang tepat sangat bergantung pada karakteristik perangkat, persyaratan kinerja, dan kebutuhan aplikasi (Stallings, 2018).

5.2.1 Programmed I/O

Programmed I/O, juga dikenal sebagai polling I/O, merupakan teknik penanganan I/O yang paling sederhana. Dalam metode ini, CPU secara aktif memantau status perangkat I/O melalui register status yang disediakan oleh perangkat. CPU terus-menerus memeriksa (poll) apakah perangkat siap untuk transfer data. Jika perangkat siap, CPU melakukan transfer data. Jika belum siap, CPU menunggu dengan terus memeriksa status perangkat (Tanenbaum, 2014).

Kelemahan utama dari programmed I/O adalah penggunaan CPU yang tidak efisien. CPU terikat (*busy-waiting*) selama perangkat I/O menyiapkan data, yang pada umumnya jauh lebih lambat dari siklus CPU. Waktu CPU yang berharga terbuang untuk menunggu perangkat I/O, sehingga teknik ini hanya cocok untuk sistem sederhana atau perangkat yang sangat cepat (Silberschatz, dkk., 2018).

5.2.2 Interrupt-Driven I/O

Untuk mengatasi kelemahan busy-waiting pada programmed I/O, dikembangkan teknik interrupt-driven I/O. Dalam metode ini, CPU mengirimkan perintah I/O ke perangkat dan kemudian melanjutkan eksekusi tugas lainnya. Ketika perangkat I/O selesai menyiapkan data atau operasi, perangkat mengirimkan

sinyal interupsi (*interrupt*) ke CPU untuk memberitahukan bahwa data telah siap (Stallings, 2018).

Mekanisme interupsi memungkinkan CPU untuk melakukan multitasking secara lebih efektif. CPU tidak perlu menunggu perangkat I/O dan dapat mengerjakan proses lain hingga interupsi diterima. Namun, teknik ini masih memerlukan intervensi CPU untuk setiap unit data yang ditransfer. Untuk transfer data dalam jumlah besar, overhead interupsi yang terus-menerus dapat membebani CPU (Tanenbaum, 2014).

5.2.3 *Direct Memory Access (DMA)*

Direct Memory Access (DMA) adalah teknik canggih yang memungkinkan transfer data langsung antara perangkat I/O dan memori utama tanpa melibatkan CPU dalam setiap transfer data. DMA Controller (DMAC) adalah perangkat keras khusus yang mengambil alih tugas transfer data dari CPU. CPU hanya perlu memberikan instruksi awal kepada DMAC mengenai lokasi memori sumber, lokasi memori tujuan, dan jumlah data yang akan ditransfer (Silberschatz, dkk., 2018).

Setelah instruksi awal diberikan, DMAC mengelola seluruh proses transfer data secara mandiri. CPU hanya menerima satu interupsi ketika seluruh blok data telah ditransfer, bukan interupsi untuk setiap byte atau word. Teknik DMA sangat efisien untuk perangkat penyimpanan dengan transfer data besar seperti disk dan SSD, karena secara signifikan mengurangi beban CPU dan memungkinkan paralelisasi antara komputasi CPU dan operasi I/O (Stallings, 2018). Tabel 5.3 membandingkan karakteristik dari tiga teknik penanganan I/O.

Tabel 5. 3 Perbandingan Teknik Penanganan I/O

Parameter	Programmed I/O	Interrupt-Driven I/O	DMA
Keterlibatan CPU	Tinggi (busy-waiting)	Sedang (per byte/word)	Rendah (per blok)
Efisiensi CPU	Rendah	Sedang	Tinggi
Overhead Perangkat Keras	Minimal	Sedang	Memerlukan DMAC
Kecepatan Transfer	Rendah	Sedang	Sangat Tinggi
Cocok untuk Perangkat	Perangkat sederhana, cepat	Perangkat kecepatan menengah	Perangkat penyimpanan, jaringan
Kompleksitas Implementasi	Sederhana	Sedang	Kompleks

Sumber: Diadaptasi dari Tanenbaum (2014) dan Stallings (2018)

5.3 Struktur Software Sistem I/O

5.3.1 Layered I/O Software

Sistem operasi mengorganisasi software I/O dalam arsitektur berlapis (*layered architecture*) untuk menyediakan abstraksi yang bersih antara aplikasi pengguna dan perangkat keras. Struktur berlapis ini memudahkan pengembangan, pemeliharaan, dan portabilitas sistem operasi ke berbagai platform perangkat keras (Silberschatz, dkk., 2018).

Lapisan-lapisan software I/O dari atas ke bawah adalah sebagai berikut:

1. ***User-Level I/O Software:*** Berisi library dan modul I/O yang dijalankan dalam ruang pengguna (user space). Contohnya adalah pustaka standar bahasa pemrograman seperti `stdio.h` pada C yang menyediakan fungsi `printf()` dan `scanf()`. Lapisan ini memberikan antarmuka yang nyaman dan seragam bagi aplikasi pengguna (Tanenbaum, 2014).
2. ***Device-Independent Operating System Software:*** Merupakan inti dari subsistem I/O kernel yang menyediakan layanan I/O yang umum dan tidak bergantung pada perangkat spesifik. Fungsi-fungsi pada lapisan ini meliputi penamaan perangkat (naming), perlindungan (protection), buffering, caching, spooling, dan alokasi perangkat (Stallings, 2018).
3. ***Device Drivers:*** Merupakan modul perangkat lunak khusus yang dikembangkan untuk setiap jenis perangkat keras. Device driver mengenali detail teknis perangkat, mengendalikan register perangkat, dan menangani protokol komunikasi yang diperlukan. Device driver menerjemahkan permintaan I/O umum dari lapisan di atasnya menjadi perintah spesifik untuk perangkat keras (Silberschatz, dkk., 2018).
4. ***Interrupt Handlers:*** Menangani sinyal interupsi yang diterima dari perangkat keras. Interrupt handler berjalan dalam konteks kernel dan bertugas menyimpan konteks, menangani interupsi, dan mengembalikan konteks ke proses yang sedang berjalan (Tanenbaum, 2014).

5. **Hardware:** Merupakan lapisan perangkat keras I/O itu sendiri, termasuk perangkat fisik, bus, dan kontroler perangkat.

5.3.2 *Buffering, Caching, dan Spooling*

Buffering adalah teknik menyimpan data dalam area memori sementara (buffer) selama proses transfer antara perangkat dan aplikasi. Tujuan buffering adalah untuk mengatasi perbedaan kecepatan antara perangkat I/O dan CPU, serta untuk menangani perbedaan dalam ukuran unit data yang diproses. Terdapat tiga jenis buffering: single buffering, double buffering, dan circular buffering (Stallings, 2018).

Caching mirip dengan buffering namun memiliki tujuan yang berbeda. Cache menyimpan salinan data yang sering diakses dalam memori yang lebih cepat untuk mengurangi jumlah operasi I/O yang lambat. Ketika aplikasi meminta data, sistem pertama-tama memeriksa cache. Jika data ada di cache (cache hit), data dikembalikan tanpa perlu mengakses perangkat I/O. Jika tidak ada (cache miss), data dibaca dari perangkat dan salinannya disimpan di cache untuk akses mendatang (Tanenbaum, 2014).

Spooling (Simultaneous Peripheral Operations On-Line) adalah teknik khusus untuk perangkat output khusus seperti printer. Spooling menggunakan disk sebagai buffer besar untuk menyimpan data output dari banyak proses. Data output dari setiap proses ditulis ke file spool pada disk. Printer kemudian mengambil data dari file spool secara berurutan. Teknik ini memungkinkan proses pengguna untuk melanjutkan eksekusi tanpa menunggu printer selesai mencetak (Silberschatz, dkk., 2018).

5.4 Perangkat I/O Modern dan Teknologi Terkini

5.4.1 Universal Serial Bus (USB) dan Generasi Terbarunya

Universal Serial Bus (USB) merupakan standar antarmuka perangkat seri yang telah menjadi standar de facto untuk menghubungkan perangkat eksternal ke komputer. Sejak diperkenalkan pada tahun 1996, USB telah mengalami beberapa generasi evolusi dengan peningkatan kecepatan transfer yang signifikan (Stallings, 2018).

USB 2.0 yang dirilis pada tahun 2000 menawarkan kecepatan transfer hingga 480 Mbps (High Speed). USB 3.0 (SuperSpeed) yang dirilis pada tahun 2008 meningkatkan kecepatan hingga 5 Gbps. USB 3.1 Gen 2 mencapai 10 Gbps, sedangkan USB 3.2 dan USB4 yang merupakan generasi terbaru menawarkan kecepatan hingga 20 Gbps dan 40 Gbps secara berurutan (Silberschatz, dkk., 2018).

USB Type-C adalah konektor fisik yang dirancang untuk menggantikan berbagai jenis konektor lama seperti USB Type-A, Type-B, micro-USB, dan bahkan konektor daya. USB Type-C mendukung orientasi plug reversibel (tidak peduli mana sisi yang dimasukkan) dan dapat mengirimkan data, daya, serta sinyal video melalui satu kabel yang sama (Tanenbaum, 2014).

Tabel 5.4 menampilkan evolusi standar USB dari generasi ke generasi.

Tabel 5.4 Evolusi Standar USB

Generasi	Tahun Rilis	Kecepatan Maksimal	Konektor Umum	Karakteristik Utama
USB 1.0/1.1	1996/1998	1.5/12 Mbps	Type-A, Type-B	Plug-and-play dasar
USB 2.0	2000	480 Mbps	Type-A, Mini, Micro	High Speed, backward compatible
USB 3.0/3.1 Gen 1	2008/2013	5 Gbps	Type-A, Type-B, Micro	SuperSpeed, pins tambahan
USB 3.1 Gen 2	2013	10 Gbps	Type-A, Type-C	SuperSpeed+
USB 3.2	2017	20 Gbps	Type-C	Dual-lane operation
USB4	2019	40 Gbps	Type-C	Berbasis Thunderbolt 3, tunneling DisplayPort dan PCIe
USB4 Version 2.0	2022	80 Gbps	Type-C	Kecepatan hingga 120 Gbps untuk display

Sumber: Diadaptasi dari USB Implementers Forum (2022) dan Stallings (2018)

5.4.2 Thunderbolt dan Thunderbolt 4

Thunderbolt adalah teknologi antarmuka perangkat keras yang awalnya dikembangkan oleh Intel dalam kolaborasi dengan Apple. Thunderbolt mengintegrasikan transfer data PCIe dan DisplayPort ke dalam satu konektor, memungkinkan hubungan daisy-chain hingga enam perangkat dari satu port (Tanenbaum, 2014).

Thunderbolt 3 yang dirilis pada tahun 2015 menggunakan konektor USB Type-C dan menawarkan kecepatan transfer hingga 40 Gbps. Thunderbolt 4 yang diperkenalkan pada tahun 2020 mempertahankan kecepatan 40 Gbps namun dengan persyaratan minimum yang lebih ketat, termasuk dukungan untuk dua monitor 4K atau satu monitor 8K, pengisian daya hingga 100W, dan perlindungan keamanan DMA melalui Intel VT-d (Stallings, 2018).

Thunderbolt 5 yang diumumkan pada tahun 2023 menawarkan kecepatan transfer ganda menjadi 80 Gbps dengan mode Bandwidth Boost hingga 120 Gbps untuk display. Teknologi ini sangat relevan untuk perangkat penyimpanan eksternal berkecepatan tinggi, stasiun dok, dan aplikasi profesional yang memerlukan throughput data besar (Silberschatz, dkk., 2018).

5.4.3 Perangkat Jaringan Modern dan I/O Virtualisasi

Perkembangan teknologi jaringan modern seperti 5G, Wi-Fi 6 (802.11ax), dan Wi-Fi 7 (802.11be) telah membawa perubahan besar dalam paradigma I/O. Wi-Fi 6 menawarkan kecepatan teoritis hingga 9.6 Gbps dengan peningkatan efisiensi di lingkungan padat melalui teknologi OFDMA. Wi-Fi 7 yang mulai diadopsi pada tahun 2023 menawarkan kecepatan hingga 46 Gbps dengan pita lebar 320 MHz dan modulasi 4K-QAM (Stallings, 2018).

I/O Virtualisasi adalah teknologi yang memungkinkan berbagai sistem operasi atau aplikasi berbagi akses ke perangkat I/O fisik secara simultan. Single Root I/O Virtualization (SR-IOV) adalah spesifikasi perangkat keras yang memungkinkan satu perangkat fisik PCI Express muncul sebagai beberapa perangkat virtual (Virtual Functions). Teknologi ini sangat penting dalam lingkungan cloud computing dan

pusat data untuk mengurangi overhead virtualisasi dan meningkatkan kinerja I/O (Tanenbaum, 2014).

5.4.4 Cloud Storage dan I/O Terdistribusi

Cloud storage telah mengubah cara sistem operasi menangani penyimpanan data. Layanan penyimpanan cloud seperti Amazon S3, Google Cloud Storage, dan Microsoft Azure Blob Storage menyediakan penyimpanan objek terdistribusi dengan skalabilitas hampir tak terbatas. Sistem operasi modern dan aplikasi harus beradaptasi dengan model I/O di mana latensi jaringan menjadi faktor dominan dan operasi I/O bersifat asynchronous (Silberschatz, dkk., 2018).

Sistem file terdistribusi seperti Hadoop Distributed File System (HDFS), Ceph, dan GlusterFS dirancang untuk menangani I/O pada skala besar di across cluster server. Sistem operasi berbasis cloud menggunakan teknik seperti data locality (memindahkan komputasi ke data, bukan sebaliknya) untuk mengurangi lalu lintas jaringan dan meningkatkan kinerja I/O (Stallings, 2018).

5.5 Manajemen Kinerja dan Optimasi I/O

5.5.1 Penjadwalan Disk

Algoritma penjadwalan disk menentukan urutan pelayanan permintaan I/O untuk mengurangi waktu pencarian (seek time) dan rotasi (rotational latency). First-Come-First-Served (FCFS) adalah algoritma paling sederhana yang melayani permintaan sesuai urutan kedatangan. Algoritma Shortest Seek Time First (SSTF) memilih permintaan dengan jarak track terdekat dari posisi head saat ini, namun dapat menyebabkan starvation (Silberschatz, dkk., 2018).

SCAN (algoritma elevator) memindahkan disk head dari satu ujung ke ujung lain, melayani semua permintaan di jalurnya. C-SCAN (Circular SCAN) adalah varian yang hanya melayani permintaan dalam satu arah dan kembali ke ujung awal tanpa melayani permintaan. LOOK dan C-LOOK adalah optimasi yang hanya menggerakkan head sejauh permintaan terakhir dalam arah tersebut, bukan ke ujung disk (Tanenbaum, 2014).

Untuk SSD yang tidak memiliki komponen mekanis, algoritma penjadwalan tradisional menjadi kurang relevan. Namun, teknik pengelompokan (clustering), pemetaan logis ke fisik (Logical Block Addressing), dan algoritma wear leveling tetap penting untuk mengoptimalkan kinerja dan memperpanjang umur SSD (Stallings, 2018).

5.5.2 RAID dan Redundansi Data

Redundant Array of Independent Disks (RAID) adalah teknologi yang menggabungkan beberapa disk fisik menjadi satu unit logis untuk meningkatkan kinerja, kapasitas, atau keandalan. RAID 0 (striping) membagi data di berbagai disk untuk meningkatkan kecepatan namun tanpa redundansi. RAID 1 (mirroring) menduplikasi data di dua disk untuk keandalan tinggi. RAID 5 dan RAID 6 menggunakan striping dengan parity bit untuk keseimbangan antara kinerja, kapasitas, dan toleransi kesalahan (Tanenbaum, 2014).

RAID modern telah berkembang dengan adopsi SSD dalam konfigurasi RAID (sering disebut sebagai hybrid RAID). NVMe RAID menggunakan teknologi seperti Intel VROC (Virtual RAID on CPU) atau AMD RAID untuk menggabungkan beberapa SSD NVMe dengan overhead minimal. Sistem operasi harus mendukung dan mengoptimalkan driver untuk konfigurasi RAID modern ini (Stallings, 2018).

5.5.3 Error Handling dan Recovery

Perangkat I/O rentan terhadap berbagai jenis kesalahan (errors), mulai dari kesalahan transien seperti parity error atau checksum mismatch, hingga kesalahan permanen seperti kegagalan perangkat keras. Sistem operasi harus mampu mendeteksi, melaporkan, dan menangani kesalahan I/O dengan cara yang terstruktur (Silberschatz, dkk., 2018).

Teknik error handling meliputi penggunaan kode koreksi kesalahan (Error Correction Code/ECC), mekanisme retry (pengulangan operasi) untuk kesalahan transien, dan penggunaan log untuk melacak operasi I/O. Sistem operasi modern juga mengimplementasikan mekanisme hot-swapping dan hot-plugging yang memungkinkan perangkat diganti atau ditambahkan tanpa mematikan sistem (Tanenbaum, 2014).

Manajemen input/output merupakan aspek fundamental dalam sistem operasi yang memiliki dampak langsung terhadap kinerja, keandalan, dan kegunaan sistem komputer secara keseluruhan. Bab ini telah membahas secara komprehensif arsitektur perangkat I/O, klasifikasi perangkat berdasarkan karakteristik dan kecepatan, serta teknik-teknik penanganan I/O dari programmed I/O hingga DMA yang masing-masing memiliki kelebihan dan batasan sesuai konteks penggunaannya.

Perkembangan perangkat keras modern seperti SSD NVMe, USB4, Thunderbolt, dan teknologi jaringan generasi baru menuntut sistem operasi untuk terus beradaptasi. Teknik-teknik seperti buffering, caching, spooling, dan I/O virtualisasi menjadi semakin penting untuk menjembatani kesenjangan antara kecepatan CPU dan perangkat I/O, serta untuk mengelola kompleksitas perangkat modern.

Pemahaman yang mendalam tentang manajemen I/O sangat penting bagi praktisi dan pengembang sistem dalam merancang solusi komputasi yang efisien. Di era komputasi awan, Internet of Things (IoT), dan big data, kemampuan sistem untuk menangani I/O dengan throughput tinggi dan latensi rendah menjadi salah satu faktor penentu keberhasilan aplikasi modern. Oleh karena itu, penelitian dan inovasi dalam bidang manajemen I/O akan terus menjadi area yang strategis dalam perkembangan sistem operasi dan arsitektur komputer di masa depan.

DAFTAR PUSTAKA

- Arifin, Zainal. 2012. *Evaluasi Pembelajaran*. Jakarta: Direktorat Jenderal Pendidikan Islam Kementerian Agama RI.
- Arikunto, Suharsimi. 2001. *Dasar-Dasar Evaluasi Pendidikan*. Jakarta: PT Bumi Aksara.
- Asrul, dkk. 2014. *Evaluasi Pembelajaran*. Medan: Ciptapustaka Media.
- Danial, D., Nurjannah, N., & Mirna, M. 2019. "Evaluation of The Learning Program of Mathematics Study Program at Islamic Institute Of Muhammadiyah Sinjai." *Matematika Dan Pembelajaran*, 7(1), 65. <https://doi.org/10.33477/mp.v7i1.1046>.
- Fitriani. 2019. "Kemampuan Pemecahan Masalah Siswa SMP." *JTMT: Journal Tadris Matematika*, 01(01), 25–30. <http://journal.iaimsinjai.ac.id/index.php/Jtm/article/view/393>.
- Hamalik, Oemar. 2010. *Manajemen Pengembangan Kurikulum*. Bandung: PT Remaja Rosdakarya.
- Ratnawulan, Elis. 2014. *Evaluasi Pembelajaran*. Bandung: Penerbit Pustaka.
- Farida, Ida. 2017. *Evaluasi Pembelajaran*. Bandung: PT Remaja Rosdakarya.
- Afandi, Muhammad. 2013. *Evaluasi Pembelajaran Sekolah Dasar*. Semarang: UNISSULA Press.
- Sukardi. 2009. *Evaluasi Pendidikan*. Jakarta: PT Bumi Aksara.
- Silberschatz, A., Galvin, P. B., & Gagne, G. 2018. *Operating System Concepts* (10th ed.). Hoboken, NJ: John Wiley & Sons.

- Stallings, W. 2018. *Operating Systems: Internals and Design Principles* (9th ed.). Boston, MA: Pearson.
- Tanenbaum, A. S. 2014. *Modern Operating Systems* (4th ed.). Upper Saddle River, NJ: Pearson.
- USB Implementers Forum. 2022. *USB4 Specification Version 2.0*. Beaverton, OR: USB-IF.

INDEKS

A

Address Bus, 14
ALU, 13
AMD-V, 32
Arsitektur Harvard, ii, 12
Arsitektur Von Neumann, ii, 12
Artificial Intelligence (AI), 109

B

Brainware, 8
Buffering, vi, 85, 92
Bus, ii, vi, 14, 84, 93

C

Cache, 13, 14, 92
Cache Memory, 13
Caching, vi, 92
Cloud Computing, iii, 16, 32, 39, 40
Cloud Storage, vi, 10, 96
Container, iii, 10, 34
Containerisasi, 34
Control Bus, 14
Control Unit (CU), 13
CPU, ii, iv, v, 2, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 23, 26, 29, 37, 46, 49, 50, 51, 52, 55, 59, 60, 61, 62, 67, 68, 69, 70, 71, 72, 73, 74, 76, 77, 78, 83, 88, 89, 90, 92, 97, 98

D

Data Bus, 14
Deep Learning, 15
Device Driver, 91
DMA Controller, 89

E

Earliest Deadline First (EDF), 75
Edge Computing, iii, 10, 16, 35
Exokernel, iii, 27
Extended Machine, iv, 48

F

Fair-Share, 73
FCFS (First Come First Served), 62
File System, 96
Firmware, 7
FPGA, 16, 37

G

GPU, 2, 7, 9, 10, 11, 15, 16, 17, 18, 19, 35, 37, 41, 86

H

Hard Real-Time, 75
Hardware, 6, 7, 92
Heterogeneous Computing, 9, 15
Hierarki Memori, ii, 14
High Performance Computing (HPC), 15
Hypervisor, 36, 74

I

I/O Virtualization, 95
Interrupt, vi, 88, 90, 91
Interrupt Handler, 91
Interrupt-Driven I/O, vi, 88, 90

J

Journaling, 39

K

Kernel, 18, 22, 36

Kubernetes, 34, 39

L

Latency, 80

Load Balancing, v, 72

Logical Block Addressing, 97

M

Machine Learning, 15

Manycore, 10

Memory Management, iv, 50, 51

Multicore Processor, ii, 14

Multilevel Feedback Queue, v, 71

Multiprogramming, iii, 29

N

Neural Processing Unit, 2, 15, 17

Non-Volatile Memory, 84, 86

NPU, 2, 7, 9, 10, 11, 15, 16, 17, 18, 35, 37

NVMe, 3, 10, 14, 84, 86, 87, 97, 98

O

Operating System, 7, 19, 31, 38, 39, 58, 91, 100, 101

P

Paravirtualisasi, 32

PCI Express, 15, 95

PCIe, 3, 15, 86, 87, 94

Pemrosesan Paralel, 9

Process, iv, 50

Process Management, iv, 50

Processor Affinity, 73

Programmed I/O, vi, 88, 90

Protection, iv, 50, 55

R

RAID, vi, 97
RAM, 6, 7, 14, 23, 24, 72
Rate Monotonic (RM), 75
Register, 13, 14
Resource Manager, iv, 49
Round Robin, v, 67
RTOS (Real-Time Operating System), 77

S

Scalability, 16
Secure Boot, 10
Security, 34, 39
Seek Time, 96
Serverless, iii, 34, 80
Sistem Operasi, i, ii, iii, iv, vii, viii, 7, 11, 16, 18, 22, 23, 25, 27, 31, 35, 36, 37, 43, 45, 46, 47, 48, 49
SJF (Shortest Job First), v, 65
Soft Real-Time, 75
Software, vi, 7, 30, 39, 80, 90, 91, 109
Spooling, vi, 92
SR-IOV, 95
SSD, 3, 6, 7, 10, 14, 84, 85, 86, 87, 89, 97, 98
System Bus, 12
System-on-Chip (SoC), 3

T

Throughput, 61
Thunderbolt, vi, 94, 95, 98
Time-Sharing, iii, 29
TPU, 10, 15, 35

U

Unikernel, iii, 27
UNIX, iii, 29, 30, 31, 32, 33, 40, 44
USB, vi, viii, 84, 93, 94, 95, 101

V

Virtual Machine (VM), 73, 74

Virtualisasi, iii, v, vi, 9, 10, 16, 32, 36, 73, 95

Von Neumann Bottleneck, 12

VT-x, 32

GLOSARIUM

Access Control: Mekanisme yang menentukan siapa yang boleh mengakses sumber daya tertentu.

Address Bus: Bus yang membawa alamat lokasi data atau instruksi.

Aging (Penuaan): Teknik meningkatkan prioritas suatu proses secara bertahap yang sudah menunggu terlalu lama di dalam antrian agar akhirnya bisa dieksekusi oleh CPU.

ALU (Arithmetic Logic Unit): Bagian/komponen CPU yang melakukan operasi aritmetika dan logika.

Arsitektur Harvard: Arsitektur yang menggunakan memori terpisah untuk instruksi dan data.

Arsitektur I/O: Struktur dan organisasi komponen perangkat keras serta perangkat lunak yang digunakan untuk melakukan operasi input/output dalam sistem komputer.

Arsitektur Von Neumann: Arsitektur komputer yang menggunakan memori bersama untuk menyimpan instruksi dan data, dengan CPU, input, output, dan system bus sebagai komponen utama.

Authentication: Proses memverifikasi identitas pengguna.

AWS (Amazon Web Services): Platform penyedia layanan infrastruktur cloud computing komersial milik perusahaan Amazon.

Batch System: Sistem yang mengumpulkan pekerjaan dalam kelompok dan menjalankannya secara berurutan.

Berikut adalah daftar glosarium yang telah digabung, diurutkan secara alfabetis (A–Z), dibersihkan dari duplikasi, dan disesuaikan formatnya:

Brainware: Manusia yang menggunakan, mengembangkan, mengelola, atau mengoperasikan sistem komputer.

Buffering: Teknik penyimpanan data sementara dalam memori untuk mengatasi perbedaan kecepatan atau menampung data selama proses I/O.

Cache Memory / Cache: Area/memori berkecepatan tinggi yang menyimpan salinan data yang sering diakses CPU untuk mempercepat akses data berikutnya.

Caching: Penyimpanan sementara data yang sering digunakan agar akses berikutnya lebih cepat.

Cgroups (Control Groups): Fitur Linux untuk mengatur dan membatasi penggunaan sumber daya oleh kelompok proses.

Cloud Computing / Cloud: Model/metode penyediaan sumber daya komputasi (seperti penyimpanan data, jaringan, dan kekuatan pemrosesan) melalui internet tanpa perlu mengelola fisik perangkat kerasnya sendiri.

Cloud Storage: Model penyimpanan data di mana data digital disimpan dalam server logis yang dikelola oleh penyedia layanan cloud dan dapat diakses melalui jaringan internet.

Completely Fair Scheduler (CFS): Algoritma penjadwalan bawaan pada kernel Linux modern yang membagi waktu pemrosesan CPU secara adil kepada setiap proses berdasarkan proporsi waktu virtual (virtual runtime).

Container: Lingkungan terisolasi untuk menjalankan aplikasi yang menggunakan kernel sistem operasi host.

Containerisasi: Pendekatan pengemasan dan isolasi aplikasi menggunakan container.

Context Switch: Proses yang dilakukan oleh Sistem Operasi untuk menyimpan status/kondisi dari suatu proses yang sedang berjalan, lalu beralih menjalankan proses lain yang berbeda pada CPU.

Control Bus: Bus yang membawa sinyal kendali untuk mengatur operasi sistem.

Control Unit (CU): Bagian CPU yang mengatur aliran data dan koordinasi eksekusi instruksi.

Core: Unit pemroses independen (otak fisik) di dalam sebuah chip CPU yang bertugas mengeksekusi instruksi program.

CPU (Central Processing Unit): Komponen perangkat keras utama komputer yang berfungsi sebagai otak untuk mengendalikan seluruh operasi dan mengeksekusi instruksi dari perangkat lunak.

CXL (Compute Express Link): Teknologi interkoneksi yang memungkinkan CPU dan akselerator berbagi sumber daya memori secara efisien.

Data Bus: Bus yang membawa data antar-komponen.

Data Center: Fasilitas fisik berupa gedung atau ruangan khusus yang digunakan untuk menempatkan ratusan hingga ribuan server komputer, sistem penyimpanan data, dan perangkat jaringan skala besar.

Defense in Depth: Strategi keamanan menggunakan beberapa lapisan perlindungan.

Device Driver: Program perangkat lunak khusus yang memungkinkan sistem operasi untuk berkomunikasi dan mengendalikan perangkat keras tertentu.

DMA (Direct Memory Access): Mekanisme perangkat keras yang memungkinkan perangkat I/O untuk mentransfer data langsung ke atau dari memori utama tanpa melibatkan CPU secara terus-menerus.

Docker: Platform perangkat lunak yang digunakan untuk mengemas, mendistribusikan, dan menjalankan aplikasi di dalam lingkungan terisolasi yang disebut container.

DVFS (Dynamic Voltage and Frequency Scaling): Teknik mengatur tegangan dan frekuensi prosesor secara dinamis untuk menghemat energi.

EDF (Earliest Deadline First): Algoritma yang memberikan prioritas kepada tugas dengan deadline paling dekat.

Edge Computing: Pemrosesan data yang dilakukan lebih dekat dengan sumber data untuk mengurangi latensi.

Embedded System (Sistem Tertanam): Sistem komputer dengan kombinasi perangkat keras dan lunak yang dirancang khusus untuk menjalankan satu fungsi spesifik di dalam perangkat lain (misalnya pada mesin cuci, kalkulator, atau sistem pengereman mobil).

Exokernel: Pendekatan kernel yang meminimalkan abstraksi dan memberikan aplikasi kontrol lebih langsung terhadap sumber daya perangkat keras.

Extended Machine: Peran sistem operasi dalam menyederhanakan kompleksitas perangkat keras melalui abstraksi dan layanan sistem.

- FCFS (First Come First Served):** Algoritma penjadwalan yang melayani proses berdasarkan urutan kedatangannya.
- File System:** Mekanisme yang mengatur penyimpanan, pengorganisasian, dan akses terhadap berkas dan direktori.
- FPGA:** Perangkat logika yang dapat diprogram untuk menjalankan fungsi komputasi tertentu.
- FreeRTOS:** Sistem Operasi Real-Time (RTOS) berskala kecil dan gratis (open-source) yang dirancang khusus untuk mikrokontroler dan perangkat tertanam (embedded system) berdaya rendah.
- Google Cloud:** Platform layanan cloud computing publik yang disediakan oleh Google untuk kebutuhan komputasi, penyimpanan data, dan pengembangan aplikasi.
- GPU (Graphics Processing Unit):** Prosesor khusus yang dirancang untuk pemrosesan grafis, komputasi paralel, dan akselerasi beban kerja AI.
- Hard Real-Time:** Sistem real-time yang kegagalan memenuhi batas waktu dapat dianggap sebagai kegagalan sistem.
- Hardware:** Komponen fisik komputer seperti CPU, RAM, SSD, GPU, dan perangkat input/output.
- HDD (Hard Disk Drive):** Media penyimpanan permanen berbasis piringan magnetik.
- Heterogeneous Computing:** Pendekatan yang menggabungkan CPU, GPU, NPU, FPGA, atau prosesor lain untuk menangani jenis beban kerja yang berbeda.
- Hybrid Kernel:** Arsitektur yang menggabungkan karakteristik monolitik dan mikrokernell.

Hypervisor: Lapisan perangkat lunak yang mengelola dan menjalankan mesin virtual.

I/O (Input/Output): Proses/mekanisme pertukaran data antara komponen internal komputer (seperti CPU/RAM) dengan perangkat luar/eksternal.

I/O Virtualization: Teknologi yang memungkinkan perangkat I/O fisik dibagi dan diakses secara virtual oleh beberapa sistem operasi atau aplikasi secara bersamaan.

Interrupt: Sinyal yang dikirim oleh perangkat keras atau perangkat lunak ke CPU untuk memberitahukan adanya peristiwa yang memerlukan perhatian segera.

Interrupt-Driven I/O: Metode I/O yang menggunakan interrupt untuk memberitahukan CPU ketika perangkat membutuhkan layanan.

IoT (Internet of Things): Jaringan/konsep objek fisik atau perangkat sehari-hari yang dilengkapi dengan sensor dan perangkat lunak sehingga bisa saling terhubung dan bertukar data melalui internet.

Journaling: Teknik sistem berkas untuk meningkatkan ketahanan terhadap kegagalan dengan mencatat perubahan sebelum diterapkan.

Kernel: Inti sistem operasi yang menangani interaksi langsung dengan perangkat keras dan menyediakan layanan dasar sistem.

Kubernetes: Platform orkestrasi container yang dibahas dalam konteks perkembangan containerisasi.

Latency: Waktu tunda antara permintaan operasi/kejadian dan munculnya respons pertama dari sistem atau perangkat.

Least Privilege: Prinsip memberikan hak akses minimum yang diperlukan kepada pengguna atau proses.

Load Balancing: Distribusi beban kerja secara merata di antara beberapa CPU atau core.

Manycore: Sistem prosesor dengan jumlah inti yang sangat banyak untuk mendukung pemrosesan paralel.

Memory Management: Pengelolaan penggunaan, alokasi, dan pembebasan memori oleh sistem operasi.

Microkernel: Arsitektur yang meminimalkan fungsi di dalam kernel dan menempatkan banyak layanan sebagai proses di ruang pengguna.

Monolithic Kernel: Arsitektur kernel yang menjalankan banyak layanan sistem operasi dalam ruang kernel yang sama.

Motherboard: Papan sirkuit cetak utama di dalam komputer yang berfungsi sebagai pusat koneksi untuk menghubungkan komponen seperti CPU, RAM, harddisk, dan kartu grafis.

Multicore / Multicore Processor: Chip prosesor tunggal yang memiliki dua atau lebih inti pemroses (core) fisik independen di dalamnya.

Multilevel Feedback Queue: Algoritma penjadwalan menggunakan beberapa antrian dengan tingkat prioritas yang dapat berubah.

Multiprocessor: Sistem komputer yang memiliki dua atau lebih CPU fisik terpisah yang bekerja bersama untuk mengeksekusi program.

Multiprogramming: Teknik memungkinkan beberapa program berada di memori sekaligus sehingga CPU

dapat berpindah ke program lain ketika suatu program menunggu I/O.

Multi-tenant: Arsitektur perangkat lunak atau infrastruktur di mana satu instans aplikasi atau server fisik melayani banyak pengguna (penyewa) yang berbeda, namun data antar-pengguna tetap terisolasi secara aman.

Namespaces: Fitur kernel Linux yang membantu memberikan isolasi lingkungan pada container.

Non-Preemptive Scheduling: Penjadwalan di mana proses yang sedang berjalan mempertahankan CPU sampai selesai atau secara sukarela melepaskannya.

NPU (Neural Processing Unit): Prosesor/unit pemrosesan khusus yang dirancang untuk komputasi kecerdasan buatan (AI) secara efisien.

NUMA (Non-Uniform Memory Access): Arsitektur memori di mana waktu akses memori bergantung pada lokasi memori relatif terhadap CPU.

NVMe (Non-Volatile Memory Express): Protokol antarmuka dan komunikasi berkecepatan tinggi yang dirancang khusus untuk mengakses media penyimpanan non-volatile melalui bus PCI Express dengan latensi rendah.

Open Source: Model pengembangan perangkat lunak dengan kode sumber yang tersedia untuk dipelajari dan dikembangkan.

OS (Operating System / Sistem Operasi): Perangkat lunak sistem yang bertugas mengelola seluruh sumber daya perangkat keras komputer dan menyediakan layanan dasar untuk aplikasi serta pengguna.

PCIe: Teknologi interkoneksi berkecepatan tinggi untuk menghubungkan perangkat seperti GPU dan SSD.

Pemrosesan Paralel: Pelaksanaan beberapa pekerjaan atau instruksi secara bersamaan menggunakan beberapa unit pemrosesan.

Penjadwalan CPU (CPU Scheduling): Proses menentukan proses atau thread yang memperoleh waktu penggunaan CPU.

Preemptive Scheduling: Penjadwalan yang memungkinkan sistem operasi mengambil CPU dari proses yang sedang berjalan.

Priority Scheduling: Penjadwalan berdasarkan tingkat prioritas proses.

Process Management: Pengelolaan pembuatan, penghapusan, penjadwalan, sinkronisasi, dan komunikasi proses.

Process: Unit dasar eksekusi dalam sistem operasi.

Processor Affinity: Preferensi agar suatu proses dijalankan pada CPU/core tertentu, terutama untuk memanfaatkan kedekatan data dan cache.

Programmed I/O: Metode I/O di mana CPU secara aktif mengendalikan proses transfer data.

Qubit: Unit informasi dasar pada komputasi kuantum yang dapat berada dalam superposisi keadaan.

RAID (Redundant Array of Independent Disks): Teknologi yang menggabungkan beberapa disk fisik menjadi satu unit logis untuk meningkatkan kinerja, kapasitas, atau keandalan penyimpanan data.

RAM (Random Access Memory): Memori utama komputer yang bersifat sementara (volatile), digunakan untuk

menyimpan data dari aplikasi yang sedang aktif berjalan.

Rate Monotonic (RM): Metode penjadwalan real-time berdasarkan prioritas yang berkaitan dengan periode tugas.

Real-Time System: Sistem yang harus memberikan respons dalam batas waktu tertentu.

Register: Memori berkecepatan sangat tinggi yang terdapat di dalam CPU untuk menyimpan data atau instruksi yang sedang diproses.

Resource Manager: Peran sistem operasi dalam mengalokasikan dan mengelola sumber daya komputer.

Round Robin (RR): Algoritma penjadwalan yang memberikan jatah waktu CPU secara bergiliran kepada proses.

RTOS (Real-Time Operating System): Sistem operasi yang dirancang untuk memenuhi persyaratan waktu respons yang dapat diprediksi.

Sandboxing: Teknik mengisolasi aplikasi agar aktivitasnya dibatasi pada lingkungan tertentu.

Secure Boot: Mekanisme keamanan yang memastikan komponen perangkat lunak yang dijalankan saat proses boot berasal dari sumber tepercaya.

Security / Proteksi: Mekanisme untuk melindungi sumber daya dan sistem dari akses atau tindakan yang tidak sah.

Serverless: Model komputasi di mana pengelolaan server dan infrastrukturnya dilakukan oleh penyedia platform.

SJF (Shortest Job First): Algoritma yang mendahulukan proses dengan waktu pengerjaan paling pendek.

Soft Real-Time: Sistem real-time yang masih dapat mentoleransi keterlambatan tertentu.

Software: Sekumpulan instruksi yang mengendalikan operasi perangkat keras.

Solid-State Drive (SSD): Perangkat penyimpanan data yang menggunakan memori flash dan tidak memiliki komponen bergerak, sehingga menawarkan kecepatan akses lebih tinggi dan konsumsi daya lebih rendah.

Spooling: Teknik manajemen I/O yang menggunakan disk sebagai buffer besar untuk mengelola/menampung permintaan output dari berbagai proses sebelum dikirim ke perangkat output.

SR-IOV (Single Root I/O Virtualization): Spesifikasi perangkat keras yang memungkinkan perangkat PCIe fisik muncul sebagai beberapa perangkat virtual independen untuk meningkatkan efisiensi dalam lingkungan virtualisasi.

Starvation (Kelaparan): Kondisi di mana suatu proses gagal dieksekusi oleh CPU dalam waktu yang sangat lama—bahkan selamanya—karena kalah bersaing dengan proses lain yang memiliki prioritas lebih tinggi.

System Bus: Jalur komunikasi yang menghubungkan berbagai komponen komputer.

System-on-Chip (SoC): Integrasi berbagai unit pemrosesan dan komponen sistem ke dalam satu chip.

Thread: Unit eksekusi yang lebih kecil di dalam suatu proses dan dapat dijadwalkan CPU.

Thunderbolt: Antarmuka perangkat keras berkecepatan tinggi yang menggabungkan transfer data PCI Express

dan video DisplayPort dalam satu konektor, dikembangkan oleh Intel.

Time-Sharing: Teknik yang memungkinkan banyak pengguna berinteraksi dengan sistem secara bersamaan dengan pembagian waktu CPU.

TPM (Trusted Platform Module): Komponen keamanan berbasis perangkat keras untuk membantu melindungi kunci dan informasi keamanan.

TPU (Tensor Processing Unit): Akselerator yang dirancang khusus untuk mempercepat komputasi AI.

Unikernel: Sistem yang menggabungkan aplikasi dengan komponen OS yang dibutuhkan menjadi satu image minimal.

UNIX: Sistem operasi yang menjadi tonggak penting dalam perkembangan sistem operasi modern.

USB (Universal Serial Bus): Standar industri untuk antarmuka komunikasi serial yang digunakan untuk menghubungkan perangkat eksternal ke komputer dengan kemampuan plug-and-play.

Virtual Machine (VM): Emulasi perangkat lunak dari sebuah komputer fisik yang berjalan di atas sistem komputer lain, lengkap dengan Sistem Operasi dan komponen virtualnya sendiri.

Virtual Memory: Mekanisme yang memungkinkan proses menggunakan ruang alamat lebih besar daripada memori fisik yang tersedia.

Virtualisasi: Teknologi yang memungkinkan satu perangkat keras fisik menjalankan beberapa lingkungan atau sistem operasi secara bersamaan.

Von Neumann Bottleneck: Keterbatasan transfer data antara CPU dan memori pada arsitektur Von Neumann.

Wear Leveling: Teknik yang digunakan dalam SSD untuk mendistribusikan operasi tulis secara merata di seluruh blok memori flash guna memperpanjang umur perangkat penyimpanan.

Zephyr: Sistem Operasi Real-Time (RTOS) berbasis open-source yang scalable dan aman, dirancang khusus untuk perangkat IoT dan embedded system yang memiliki keterbatasan memori.

BIODATA PENULIS



Lely Priska D. Tampubolon, S.Kom. MMSI

Dosen tetap Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Perbanas Institute sejak 1998.

Menyelesaikan S1 Prodi Teknik Komputer di STMIK Gunadarma, Pasca Sarjana (S2) di Universitas Gunadarma Prodi Sistem Informasi konsentrasi Sistem Informasi Bisnis. Peminatan di bidang sistem operasi, arsitektur komputer, fintech, dan blockchain. Email : lely.priska@perbanas.id.

BIODATA PENULIS



Pastima Simanjuntak, S.Kom., M.SI.

Dosen Program Studi Teknik Informatika

Fakultas Teknik dan Komputer Universitas Putera Batam

Memperoleh gelar Sarjana Komputer dari Universitas Katolik Santo Thomas Sumatera Utara pada tahun 2007 dan gelar Magister Sistem Informasi dari STMIK Putera Batam pada tahun 2012. Sejak tahun 2009 hingga saat ini, aktif sebagai dosen pada Fakultas Teknik dan Komputer Universitas Putera Batam. Selain menjalankan tugas pendidikan dan pengajaran, juga aktif dalam kegiatan penelitian, publikasi ilmiah pada tingkat nasional dan internasional, serta pengabdian kepada masyarakat sebagai bagian dari pelaksanaan Tridharma Perguruan Tinggi.

BIODATA PENULIS



Ir. Johni S. Pasaribu, MT.

Dosen Program Studi Sistem Informasi

Fakultas IT Politeknik Piksi Ganesha

Penulis aktif mengajar, melakukan riset dan menjadi dosen tetap di Politeknik Piksi Ganesha. Bidang keahlian mengajar dan penelitian penulis adalah Artificial Intelligence (AI), Decision Support System (DSS), Software Engineering dan Software Testing. Penulis juga aktif melakukan bimbingan mahasiswa S1 dan publikasi di jurnal-jurnal nasional maupun internasional.

BIODATA PENULIS



Aulia Nurul Iftitah, S.Pd., M.Pd.

Dosen Program Studi Bisnis Digital
STMIK KHARISMA Makassar

Penulis memulai kariernya sebagai dosen pada tahun 2025, setelah menyelesaikan seluruh rangkaian studi akademisnya di Universitas Negeri Makassar. Penulis menempuh pendidikan S1 di jurusan Pendidikan Teknik Informatika dan Komputer hingga lulus pada tahun 2021, lalu langsung melanjutkan studi S2 di bidang Pendidikan Teknologi Kejuruan dan berhasil meraih gelar Magister pada tahun 2024.

Kini, selain aktif mengajar di perguruan tinggi, Penulis menaruh minat besar pada pengembangan teknologi kejuruan dan metode pembelajaran interaktif. Buku *chapter* ini merupakan karya tulis ilmiah perdana yang diterbitkan oleh Penulis, sebagai langkah awal dalam mendokumentasikan pemikiran serta kontribusi nyata di bidang ilmu yang ditekuninya.

BIODATA PENULIS



Anas, S.Kom., M.Kom.

Penulis lahir di Pinrang, Sulawesi Selatan, pada 18 April 1989. Penulis menyelesaikan pendidikan Sarjana (S1) pada Program Studi Sistem Informasi di STMIK Dipanegara Makassar. Selanjutnya, penulis melanjutkan pendidikan Magister (S2) pada Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro Semarang, dan memperoleh gelar Magister Komputer (M.Kom.).

Karier profesional penulis dimulai sebagai guru di SMK Negeri 3 Pinrang dan SMK Negeri 4 Pinrang pada tahun 2012. Pengalaman tersebut menjadi fondasi dalam mengembangkan kompetensi di bidang pendidikan dan teknologi informasi. Pada tahun 2013, penulis beralih ke dunia pendidikan tinggi sebagai dosen di Universitas Ichsan Gorontalo, tempat penulis mengabdikan diri hingga tahun 2019. Selanjutnya, pada periode 2019–2023, penulis melanjutkan karier akademiknya sebagai dosen di Universitas Pohuwato.

Saat ini, penulis aktif sebagai dosen tetap Program Studi Sains Data di Institut Teknologi Bacharuddin Jusuf Habibie (ITH) Parepare. Dalam menjalankan Tri Dharma Perguruan Tinggi, penulis mengampu berbagai mata kuliah di bidang Data Science, Pengolahan Citra Digital, Basis Data, Pemrograman Python, Rekayasa Perangkat Lunak, Sistem Pendukung Keputusan, serta Wawasan Teknologi.

Selain mengajar, penulis juga aktif melaksanakan penelitian dan pengabdian kepada masyarakat dengan fokus pada bidang Data Science,

Melalui buku ini, penulis berharap dapat memberikan kontribusi nyata dalam pengembangan ilmu pengetahuan, khususnya di bidang teknologi informasi.