Pengantar Sistem Operasi: Manajemen Proses dan Penjadwalan FCFS

Pendahuluan

Bagian 1: Pengantar Sistem Operasi (Introduction to Operating Systems)

1.1 Apa Itu Sistem Operasi dan Mengapa Penting?

- Definisi Dasar
  Sistem Operasi, atau sering disingkat OS (Operating System), merupakan perangkat lunak (software) yang paling mendasar dan penting dalam sebuah sistem komputer.1 Ia berperan sebagai penghubung atau jembatan antara pengguna (user), perangkat keras (hardware) komputer seperti CPU, memori, dan hard disk, serta perangkat lunak aplikasi lainnya seperti browser web atau pengolah kata.2 Tanpa adanya sistem operasi, komputer hanyalah sekumpulan komponen elektronik yang tidak dapat menjalankan fungsinya atau program apapun, kecuali program booting awal.3 Sistem operasi adalah perangkat lunak pertama yang dimuat ke dalam memori ketika komputer dinyalakan (proses booting) dan selanjutnya mengendalikan seluruh aktivitas yang terjadi di dalam komputer.1
- Analogi Sederhana
  Untuk memahami peran OS dengan lebih mudah, kita bisa menggunakan beberapa analogi:
  - **Manajer Kantor/Pabrik** [2]**:** Bayangkan OS sebagai seorang manajer umum dalam sebuah kantor. Manajer ini bertugas mengatur semua sumber daya yang ada – mulai dari pegawai (diibaratkan sebagai aplikasi) hingga peralatan kantor (diibaratkan sebagai hardware) – agar semuanya dapat bekerja sama secara harmonis dan efisien. Manajer juga menjadi penghubung utama antara pemilik perusahaan (pengguna) dengan seluruh kegiatan operasional di kantor (komputer). Tanpa manajer, aktivitas kantor akan kacau dan tidak terarah.
  - **Pemerintah dalam Negara** [6]**:** OS dapat diibaratkan seperti pemerintah dalam suatu negara. Pemerintah bertugas mengatur dan mengawasi penggunaan berbagai sumber daya negara (seperti sumber daya alam, infrastruktur - diibaratkan sebagai hardware) oleh berbagai pihak (masyarakat, perusahaan - diibaratkan sebagai program dan pengguna). Pemerintah memastikan bahwa negara (komputer) berjalan sesuai aturan dan dalam kondisi yang baik.

    - **Lantai Rumah** [2]**:** Analogi lain adalah lantai di dalam sebuah rumah. Lantai merupakan dasar tempat kita meletakkan semua perabotan (seperti meja, kursi - diibaratkan sebagai aplikasi) dan tempat kita beraktivitas. Tanpa lantai yang kokoh, perabotan tidak dapat diletakkan dan digunakan dengan baik. Demikian pula, aplikasi komputer tidak dapat berjalan dan berfungsi tanpa adanya sistem operasi sebagai dasarnya. Aplikasi-aplikasi tersebut seolah "diletakkan" dan dijalankan di atas "lantai" sistem operasi.
- Pentingnya OS
  Keberadaan sistem operasi sangatlah krusial. OS memungkinkan kita untuk menjalankan berbagai program aplikasi, menggunakan perangkat keras seperti mouse, keyboard, monitor, dan printer dengan mudah, serta berinteraksi dengan komputer secara keseluruhan.1 Salah satu peran terpenting OS adalah menyembunyikan kerumitan detail teknis perangkat keras dari pengguna dan aplikasi. Ia menyediakan sebuah antarmuka (interface) yang lebih sederhana dan ramah pengguna, baik berupa teks maupun grafis.1 Selain itu, OS juga bertanggung jawab untuk memastikan bahwa komputer berjalan secara efisien dengan mengelola sumber daya secara optimal dan menjaga keamanan sistem dari akses yang tidak sah.1
  Fungsi-fungsi OS yang beragam ini secara kolektif menciptakan sebuah **lapisan abstraksi** yang esensial. Lapisan ini menyederhanakan interaksi dengan kompleksitas hardware, memungkinkan pengembangan dan penggunaan aplikasi yang lebih mudah dan efisien. Tanpa abstraksi ini, penggunaan komputer seperti yang kita kenal saat ini tidak akan praktis. Ini menunjukkan bahwa OS bukan hanya sekumpulan fungsi terpisah, melainkan sebuah desain fundamental yang membuat sistem komputer dapat digunakan secara efektif.[1]
- Contoh OS Populer
  Ada banyak sistem operasi yang digunakan saat ini, baik untuk komputer desktop, laptop, maupun perangkat mobile. Beberapa contoh yang paling dikenal antara lain: Microsoft Windows, Apple macOS (untuk komputer Apple), Linux (bersifat open-source dan memiliki banyak varian seperti Ubuntu), Google Android (untuk smartphone dan tablet), dan Apple iOS (untuk iPhone dan iPad).1

1.2 Fungsi Utama Sistem Operasi

Sistem operasi memiliki berbagai fungsi penting yang memungkinkannya mengelola sistem komputer secara efektif. Berikut adalah beberapa fungsi utamanya:

- **Menjalankan Operasi Dasar:** Ini adalah fungsi paling fundamental. OS menyediakan lingkungan dasar agar perangkat lunak lain dapat berjalan. Sebelum aplikasi bisa berfungsi, OS-lah yang memungkinkannya untuk dieksekusi dan

hasilnya ditampilkan kepada pengguna.[2] OS adalah komponen vital yang mendasari kerja semua software lain.

- **Manajemen Sumber Daya (Resource Manager):** OS bertindak sebagai pengelola sumber daya utama komputer. Ia mengontrol dan mengalokasikan penggunaan sumber daya penting seperti waktu pemrosesan CPU (Central Processing Unit), alokasi memori utama (RAM), ruang penyimpanan pada hard disk atau SSD, serta akses ke perangkat Input/Output (I/O) seperti keyboard, mouse, printer, dan jaringan.[1] Manajemen ini bertujuan agar sumber daya dapat digunakan secara efisien dan adil oleh berbagai program dan pengguna yang membutuhkannya.
- **Mengatur Kerja Hardware dan Software:** OS menjadi jembatan komunikasi antara perangkat keras dan perangkat lunak. Ia memastikan bahwa instruksi dari perangkat lunak dapat diterjemahkan menjadi aksi oleh perangkat keras, dan sebaliknya, status dari perangkat keras dapat diketahui oleh perangkat lunak.[2]
- **Menyediakan Antarmuka (Interface):** OS menyediakan cara bagi pengguna untuk berinteraksi dengan komputer. Antarmuka ini bisa berupa *Command Line Interface* (CLI), di mana pengguna mengetikkan perintah teks (contohnya pada sistem operasi DOS [3]), atau *Graphical User Interface* (GUI), yang menggunakan elemen visual seperti ikon, jendela, dan menu (contohnya pada Windows, macOS, dan Linux modern).[1] GUI memudahkan pengguna awam untuk mengoperasikan komputer.
- **Wadah Program/Aplikasi:** Meskipun aplikasi disimpan di media penyimpanan, secara konseptual aplikasi tersebut "berjalan di dalam" atau "melekat pada" sistem operasi.[2] OS menyediakan lingkungan dan layanan yang diperlukan agar aplikasi dapat dieksekusi.
- **Mengkoordinasi Kerja Perangkat:** OS tidak hanya mengatur hubungan antara hardware dan software, tetapi juga mengkoordinasikan aktivitas di dalam komputer. Ia dapat menyusun eksekusi program yang kompleks menjadi langkah-langkah yang lebih sederhana dan berurutan, sehingga aplikasi dapat bekerja lebih efisien.[4]
- **Mengoptimalkan Fungsi Perangkat:** OS berusaha mengoptimalkan kinerja keseluruhan sistem. Contohnya termasuk mengatur penjadwalan penggunaan CPU, mengelola cara data dipanggil dari memori atau hard disk, dan mengatur waktu koneksi jaringan agar penggunaan sumber daya menjadi seefisien mungkin.[4]
- **Keamanan dan Proteksi (Gate Keeper):** OS memiliki peran penting dalam menjaga keamanan sistem. Ia mengontrol siapa saja (pengguna) yang berhak mengakses komputer atau program tertentu, seringkali melalui mekanisme seperti akun pengguna dan kata sandi. OS juga melindungi file dan data dari akses atau

modifikasi yang tidak sah, serta mengawasi aktivitas pengguna di dalam sistem.[1]

Bagian 2: Memahami Konsep Proses (Understanding the Process Concept)

Setelah memahami peran sistem operasi secara umum, kita akan masuk ke salah satu konsep paling fundamental dalam OS, yaitu **proses**.

2.1 Dari Program Menjadi Proses

Seringkali istilah "program" dan "proses" digunakan secara bergantian, namun dalam konteks sistem operasi, keduanya memiliki makna yang berbeda:

- **Program:** Sebuah program adalah kumpulan instruksi yang ditulis dalam bahasa pemrograman dan disimpan dalam sebuah file di media penyimpanan (seperti hard disk atau SSD). File ini sering disebut sebagai *executable file*. Program dalam bentuk file ini bersifat **pasif**; ia hanyalah sekumpulan kode yang belum melakukan apa-apa.[14]
- **Proses:** Sebuah proses adalah program yang sedang **dieksekusi** atau dijalankan oleh sistem komputer. Ketika Anda mengklik ikon sebuah aplikasi atau menjalankan perintah di terminal, sistem operasi akan memuat program tersebut dari disk ke memori utama (RAM) dan mulai menjalankannya. Saat itulah program tersebut menjadi sebuah **proses** yang **aktif**.[6]
- **Analogi:** Kita bisa kembali menggunakan analogi resep masakan. **Resep** yang tertulis di buku adalah **program** (pasif). Ketika seorang **koki mulai aktif mengikuti langkah-langkah** dalam resep tersebut – mengukur bahan, mencampur, memasak di atas kompor – maka aktivitas memasak yang sedang berlangsung itulah yang disebut **proses** (aktif).

Sistem operasi bertanggung jawab penuh atas transformasi program menjadi proses, termasuk memuat kode program ke memori dan mengalokasikan sumber daya awal yang diperlukan.[13]

2.2 Definisi Proses: Lebih dari Sekadar Program Berjalan

Mendefinisikan proses hanya sebagai "program yang sedang dieksekusi" sebenarnya belum lengkap. Konsep proses dalam sistem operasi mencakup lebih banyak hal:

- **Unit Kerja Dasar:** Proses dianggap sebagai unit kerja dasar atau terkecil yang dikelola dan dijadwalkan oleh sistem operasi untuk dieksekusi oleh CPU.[13] Setiap aktivitas yang dilakukan komputer, mulai dari menjalankan aplikasi pengolah kata hingga tugas latar belakang sistem, direpresentasikan sebagai satu atau lebih proses.

- **Kebutuhan Sumber Daya:** Setiap proses membutuhkan sumber daya untuk dapat berjalan dan menyelesaikan tugasnya. Sumber daya ini meliputi waktu penggunaan CPU (*CPU time*), alokasi memori (RAM), akses ke file-file di disk, serta penggunaan perangkat Input/Output (I/O) seperti printer atau kartu jaringan.[6] Sistem operasilah yang bertugas mengelola dan mengalokasikan sumber daya ini kepada proses-proses yang membutuhkannya.[12]
- **Konteks Eksekusi Lengkap:** Sebuah proses tidak hanya terdiri dari kode program itu sendiri, tetapi juga mencakup seluruh **konteks eksekusi** yang terkait dengannya. Konteks ini menyimpan semua informasi yang diperlukan agar proses dapat berjalan dengan benar dan dapat dihentikan serta dilanjutkan kembali oleh OS. Komponen utama dari konteks eksekusi meliputi [13]:
    - **Kode Program (Text Section):** Kumpulan instruksi mesin yang akan dieksekusi oleh CPU.
    - **Program Counter (PC):** Sebuah register khusus yang menyimpan alamat memori dari instruksi *berikutnya* yang akan dieksekusi. PC sangat penting untuk melacak alur eksekusi proses.
    - **Register CPU:** Nilai-nilai yang tersimpan di dalam register-register CPU pada saat proses sedang berjalan. Ini termasuk register serbaguna, register indeks, *stack pointer*, dll. Nilai-nilai ini mencerminkan status komputasi proses saat itu.
    - **Stack:** Sebuah area memori yang digunakan untuk menyimpan data sementara selama eksekusi fungsi atau prosedur. Ini biasanya berisi parameter fungsi, variabel lokal, dan alamat kembali (alamat instruksi yang harus dijalankan setelah fungsi selesai).
    - **Data Section:** Area memori yang menyimpan variabel global dan statis yang digunakan oleh program.
    - **Heap:** Area memori yang digunakan untuk alokasi memori dinamis, yaitu memori yang diminta oleh program saat sedang berjalan (runtime).
    - **Informasi Sumber Daya OS:** Catatan tentang sumber daya sistem operasi yang sedang digunakan oleh proses, seperti file-file yang sedang dibuka atau koneksi jaringan yang aktif.

2.3 Status-Status Proses (Siklus Hidup Proses)

Selama masa hidupnya, sebuah proses akan berpindah melalui serangkaian **status** atau **keadaan (state)** yang berbeda. Status ini mencerminkan aktivitas atau kondisi proses pada saat itu. Sistem operasi melacak status setiap proses dan mengelolanya. Meskipun nama dan jumlah status bisa sedikit bervariasi antar sistem operasi, lima status dasar berikut ini umum dijumpai [12]:

1. **New (Baru):** Ini adalah status awal ketika sebuah proses baru saja **dibuat** oleh sistem operasi. OS sedang melakukan inisialisasi dan mengalokasikan struktur data yang diperlukan, tetapi proses belum siap untuk dieksekusi.[13]
2. **Ready (Siap):** Setelah proses selesai dibuat dan sumber daya awal dialokasikan, ia masuk ke status Ready. Dalam status ini, proses sudah berada di memori utama dan **siap untuk dieksekusi** oleh CPU, namun sedang **menunggu giliran** karena CPU mungkin sedang menjalankan proses lain.[12] Biasanya, terdapat sebuah antrian (disebut *Ready Queue*) yang berisi semua proses yang berada dalam status Ready.[14]
3. **Running (Berjalan):** Ketika penjadwal CPU (CPU scheduler) memilih sebuah proses dari Ready Queue untuk dieksekusi, proses tersebut berpindah ke status Running. Ini berarti instruksi-instruksi dari proses tersebut sedang **aktif dijalankan oleh CPU**.[12] Penting untuk diingat bahwa pada satu inti prosesor (CPU core), hanya **satu** proses yang dapat berada dalam status Running pada satu waktu tertentu.[14]
4. **Waiting (Menunggu) / Blocked:** Sebuah proses akan beralih ke status Waiting (atau sering juga disebut Blocked) jika ia **tidak dapat melanjutkan eksekusi** karena harus **menunggu suatu kejadian (event)** tertentu terjadi.[12] Kejadian ini bisa bermacam-macam, contohnya:
    - Menunggu selesainya operasi Input/Output (I/O), seperti membaca data dari hard disk atau menerima data dari jaringan.
    - Menunggu input dari pengguna melalui keyboard.
    - Menunggu alokasi memori tambahan.
    - Menunggu sinyal atau data dari proses lain. Proses dalam status Waiting tidak menggunakan CPU, meskipun ia masih berada di memori. Setelah kejadian yang ditunggu selesai, proses tersebut biasanya akan kembali ke status Ready, siap untuk bersaing mendapatkan CPU lagi.
5. **Terminated (Selesai):** Status ini menandakan bahwa proses telah **menyelesaikan eksekusinya** secara normal, atau dihentikan secara paksa oleh sistem operasi (misalnya karena terjadi error atau atas perintah pengguna).[13] Ketika sebuah proses terminated, sistem operasi akan membebaskan semua sumber daya yang sebelumnya dialokasikan untuk proses tersebut (seperti memori dan file yang terbuka) dan menghapus informasinya dari sistem.

- Diagram Transisi Status
  Perpindahan proses antar status dapat divisualisasikan dengan diagram transisi:
  Code snippet
  ```
  graph LR
    A(New) --> B(Ready);
    B --> C(Running);
  ```

```
    C -- Event Wait / I/O Request --> D(Waiting);
    C -- Interrupt / Time Slice Expired --> B;
    C -- Exit --> E(Terminated);
    D -- Event Occurs / I/O Complete --> B;

    style A fill:#f9f,stroke:#333,stroke-width:2px
    style B fill:#ccf,stroke:#333,stroke-width:2px
    style C fill:#cfc,stroke:#333,stroke-width:2px
    style D fill:#ff9,stroke:#333,stroke-width:2px
    style E fill:#ccc,stroke:#333,stroke-width:2px
```

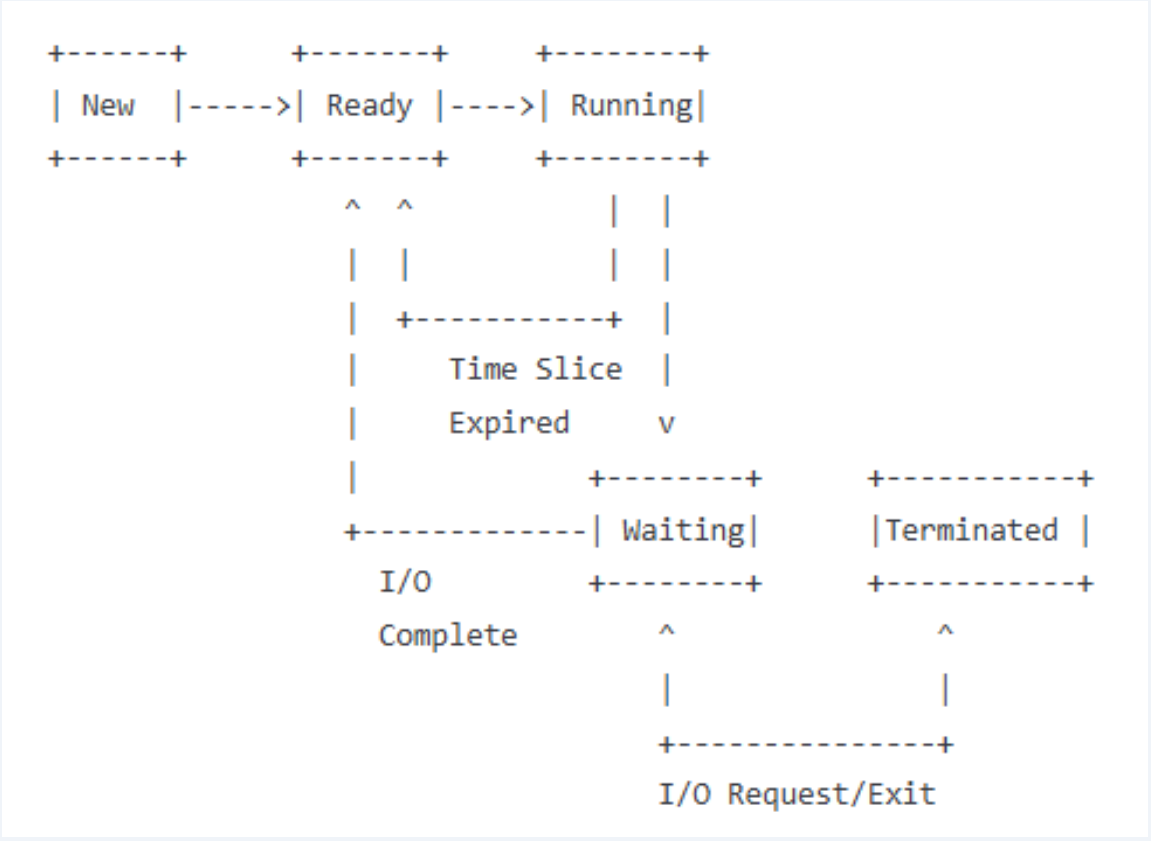

Diagram ini menunjukkan alur umum siklus hidup proses. Proses dimulai dari New, menjadi Ready, lalu dipilih untuk Running. Dari Running, ia bisa selesai (Terminated), menunggu event (Waiting), atau kembali ke Ready (karena waktu habis atau interupsi). Dari Waiting, setelah event selesai, ia kembali ke Ready.

2.4 Pengantar Process Control Block (PCB): Kartu Identitas Proses

Bagaimana sistem operasi dapat melacak semua informasi tentang setiap proses yang berjalan, termasuk statusnya, program counter, register, alokasi memori, dan lainnya? Jawabannya adalah melalui sebuah struktur data fundamental yang disebut **Process Control Block (PCB)**.

- **Representasi Proses di OS:** Di dalam sistem operasi, setiap proses yang aktif direpresentasikan oleh PCB-nya masing-masing. PCB bisa dianggap sebagai "kartu identitas" atau "paspor" untuk sebuah proses.[14] Ketika sebuah proses baru dibuat, sistem operasi juga akan membuatkan PCB baru untuknya.[27] PCB ini akan terus diperbarui selama proses berjalan.
- **Informasi Penting dalam PCB:** PCB menyimpan semua informasi penting yang dibutuhkan oleh sistem operasi untuk mengelola dan mengontrol sebuah proses. Isi dari PCB bisa bervariasi antar sistem operasi, tetapi umumnya mencakup [13]:
    - **Process State:** Status proses saat ini (misalnya, Ready, Running, Waiting).
    - **Process ID (PID):** Sebuah nomor unik yang mengidentifikasi proses tersebut di dalam sistem.
    - **Program Counter (PC):** Menyimpan alamat memori dari instruksi selanjutnya yang harus dieksekusi ketika proses ini mendapatkan giliran CPU lagi.
    - **CPU Registers:** Salinan dari isi semua register CPU (seperti akumulator, register indeks, *stack pointer*, register serbaguna) pada saat proses terakhir kali berhenti berjalan. Informasi ini krusial untuk melanjutkan eksekusi proses dari titik yang sama.
    - **CPU Scheduling Information:** Informasi yang relevan untuk penjadwalan, seperti prioritas proses, pointer ke antrian penjadwalan di mana proses ini berada (misalnya, Ready Queue atau Waiting Queue).
    - **Memory Management Information:** Detail tentang bagaimana memori dialokasikan untuk proses ini. Ini bisa berupa nilai *base* dan *limit registers*, atau penunjuk ke tabel halaman (*page tables*) atau tabel segmen (*segment tables*) yang digunakan oleh proses.
    - **Accounting Information:** Informasi untuk keperluan akuntansi dan pemantauan, seperti jumlah waktu CPU yang telah digunakan oleh proses, batas waktu eksekusi, ID pengguna pemilik proses, dll.
    - **I/O Status Information:** Informasi tentang perangkat I/O yang sedang digunakan atau dialokasikan untuk proses ini, serta daftar file yang sedang dibuka oleh proses.
- **Pentingnya PCB:** PCB memegang peranan sentral dalam manajemen proses, terutama dalam lingkungan *multitasking* di mana banyak proses berjalan seolah-olah secara bersamaan.[28] Fungsi utama PCB adalah:
    - **Menyimpan Konteks Proses:** PCB menyimpan seluruh konteks eksekusi

proses ketika proses tersebut tidak sedang berjalan.

- **Memungkinkan Context Switching:** Ketika sistem operasi memutuskan untuk menghentikan satu proses (misalnya P_A) dan mulai menjalankan proses lain (misalnya P_B), terjadi proses yang disebut **Context Switching**.[15] Selama *context switch*, OS melakukan langkah berikut:
  1. Menyimpan status P_A saat ini (nilai PC, register CPU, dll.) ke dalam PCB milik P_A.
  2. Memuat status P_B dari PCB milik P_B ke dalam register-register CPU.
  3. Melanjutkan eksekusi P_B dari instruksi yang ditunjuk oleh PC yang baru dimuat. Dengan menyimpan dan memulihkan konteks melalui PCB, OS dapat menghentikan dan melanjutkan proses tanpa kehilangan informasi.
- **Dasar Penjadwalan:** Penjadwal CPU menggunakan informasi dalam PCB (seperti status dan prioritas) untuk memutuskan proses mana yang akan dijalankan berikutnya.[26]
- **Manajemen Sumber Daya:** OS menggunakan informasi di PCB untuk melacak sumber daya (memori, file, perangkat I/O) yang dialokasikan ke setiap proses.

Karena berisi informasi yang sangat penting dan sensitif, PCB biasanya disimpan di area memori kernel yang terlindungi dari akses langsung oleh program pengguna biasa.[28]Secara esensial, PCB adalah **manifestasi data konkret** dari konsep abstrak "proses". Ia mengubah ide tentang program yang berjalan menjadi sebuah entitas yang dapat dikelola secara efisien oleh sistem operasi dengan menyimpan semua atribut dan status pentingnya di satu lokasi terstruktur. Tanpa PCB, manajemen banyak proses secara bersamaan (multitasking) dalam sistem operasi modern tidak mungkin dilakukan secara efektif. PCB menjadi jembatan krusial antara konsep teoritis proses dan implementasi praktisnya di dalam OS.[13]

Bagian 3: Pengantar Penjadwalan Proses oleh CPU (Introduction to CPU Process Scheduling)

Kita telah memahami apa itu proses dan bagaimana OS melacaknya menggunakan PCB. Sekarang, kita akan membahas bagaimana OS memutuskan proses mana yang harus dijalankan oleh CPU pada suatu waktu tertentu. Inilah inti dari **penjadwalan CPU**.

3.1 Kenapa CPU Perlu Menjadwalkan Proses?

- **Kebutuhan Berbagi Sumber Daya:** Di hampir semua sistem komputer modern, terdapat banyak program atau aplikasi yang ingin dijalankan secara bersamaan. Ini berarti ada banyak proses yang berada dalam status Ready (siap dieksekusi) pada saat yang sama.[9] Namun, jumlah CPU (atau inti CPU) biasanya terbatas,

seringkali hanya satu atau beberapa saja. CPU adalah sumber daya komputasi utama yang sangat berharga dan harus dibagi secara efisien di antara semua proses yang membutuhkannya.

- **Tujuan Multitasking:** Tujuan utama dari sistem operasi modern adalah mendukung *multitasking* atau *multiprogramming*. Ini berarti sistem dapat menjalankan beberapa proses seolah-olah secara bersamaan. Caranya adalah dengan membuat CPU beralih dari satu proses ke proses lain dengan sangat cepat, sebuah teknik yang disebut *time-sharing* atau *interleaving*.[11] Meskipun pada satu inti CPU hanya satu proses yang benar-benar berjalan pada satu instan waktu, perpindahan yang cepat ini memberikan ilusi kepada pengguna bahwa banyak tugas berjalan secara paralel. Hal ini penting untuk meningkatkan utilisasi (pemanfaatan) CPU – agar CPU tidak menganggur saat satu proses menunggu I/O – dan juga untuk membuat sistem terasa lebih responsif.[11]
- **Peran Penjadwal (Scheduler):** Untuk mengatur pembagian waktu CPU ini, sistem operasi memiliki komponen khusus yang disebut **Penjadwal CPU (CPU Scheduler)**. Penjadwal CPU ini kadang disebut juga *short-term scheduler*.[35] Tugas utamanya adalah **memilih** salah satu proses dari sekumpulan proses yang berada di *Ready Queue* (antrian proses siap) untuk dialokasikan ke CPU dan mulai dijalankan.[13] Keputusan penjadwalan ini perlu diambil oleh OS setiap kali terjadi perubahan status proses yang signifikan, misalnya ketika:
    - Sebuah proses beralih dari status Running ke Waiting (misalnya, meminta I/O).
    - Sebuah proses beralih dari status Running ke Ready (misalnya, karena *time slice*-nya habis atau ada interupsi).
    - Sebuah proses beralih dari status Waiting ke Ready (misalnya, operasi I/O selesai).
    - Sebuah proses selesai (Terminated).[36]

3.2 Tujuan Utama dan Kriteria Evaluasi Penjadwalan

Pemilihan proses mana yang akan dijalankan berikutnya bukanlah keputusan acak. Penjadwal CPU bekerja berdasarkan **algoritma penjadwalan** tertentu, dan setiap algoritma dirancang untuk mencapai tujuan-tujuan spesifik.

- **Tujuan Umum Penjadwalan:** Sasaran utama dari setiap algoritma penjadwalan adalah untuk **mengoptimalkan kinerja sistem komputer secara keseluruhan** berdasarkan kriteria-kriteria tertentu.[35] Tujuan ini bisa sedikit berbeda tergantung pada jenis sistem operasi:
    - **Sistem Batch:** Fokus utamanya biasanya adalah memaksimalkan *throughput* (jumlah pekerjaan yang selesai) dan utilisasi CPU, serta meminimalkan *turnaround time*.[36]

    - **Sistem Interaktif:** Fokus utamanya adalah meminimalkan *response time* (waktu tanggap) agar pengguna merasa sistem cepat merespons perintah, dan menjaga proporsionalitas (ekspektasi pengguna terpenuhi).[36]
    - **Sistem Real-time:** Fokus utamanya adalah memenuhi *deadline* (batas waktu) yang ketat untuk tugas-tugas kritis dan menjaga prediktabilitas kinerja.[36]

  Secara umum, tujuan penjadwalan mencakup [35]:
    - **Memaksimalkan CPU Utilization:** Menjaga CPU sesibuk mungkin mengerjakan tugas-tugas yang berguna.
    - **Memaksimalkan Throughput:** Menyelesaikan sebanyak mungkin proses dalam satu satuan waktu.
    - **Meminimalkan Turnaround Time:** Mengurangi waktu total yang dibutuhkan setiap proses dari awal hingga akhir.
    - **Meminimalkan Waiting Time:** Mengurangi waktu tunggu proses di Ready Queue.
    - **Meminimalkan Response Time:** Memberikan tanggapan awal secepat mungkin kepada pengguna.
    - **Memastikan Fairness (Keadilan):** Memberikan kesempatan yang adil bagi setiap proses untuk mendapatkan waktu CPU.
- **Kriteria Evaluasi Algoritma Penjadwalan:** Untuk membandingkan seberapa baik kinerja berbagai algoritma penjadwalan, kita menggunakan kriteria-kriteria kuantitatif berikut [35]:
    - **CPU Utilization:** Dinyatakan dalam persentase (%), mengukur seberapa besar bagian waktu CPU yang digunakan untuk menjalankan proses (bukan menganggur). Nilai yang diinginkan adalah setinggi mungkin, idealnya mendekati 100%, meskipun dalam praktiknya sulit dicapai karena adanya waktu untuk I/O dan overhead sistem.
    - **Throughput:** Jumlah proses yang berhasil diselesaikan per satuan waktu (misalnya, proses per detik atau proses per jam). Nilai yang diinginkan adalah setinggi mungkin.
    - **Turnaround Time (TA Time / Waktu Penyelesaian):** Total waktu yang dihabiskan oleh sebuah proses sejak ia masuk ke sistem (tiba di Ready Queue) hingga ia benar-benar selesai dieksekusi. Ini adalah jumlah dari waktu eksekusi (termasuk waktu CPU dan I/O) dan waktu tunggu. TAT=WaktuSelesai−WaktuKedatangan atau TAT=BurstTime+WaitingTime. Nilai yang diinginkan adalah serendah mungkin.
    - **Waiting Time (Waktu Tunggu):** Total waktu yang dihabiskan oleh sebuah proses menunggu di dalam Ready Queue sebelum mendapatkan giliran dieksekusi oleh CPU. WT=WaktuMulaiEksekusi−WaktuKedatangan. Nilai yang diinginkan adalah serendah mungkin.

    - **Response Time (Waktu Tanggap):** Waktu yang dibutuhkan sejak sebuah permintaan atau perintah dimasukkan hingga respons pertama dihasilkan oleh sistem. Ini sangat penting untuk sistem interaktif. Nilai yang diinginkan adalah serendah mungkin.
    - **Fairness (Keadilan):** Mengukur seberapa adil pembagian waktu CPU di antara proses-proses yang bersaing. Tujuannya adalah untuk memastikan tidak ada proses yang "kelaparan" (starvation), yaitu menunggu terlalu lama atau tidak pernah mendapatkan giliran CPU sama sekali.

Penting untuk dipahami bahwa kriteria-kriteria ini seringkali saling bertentangan atau melibatkan **trade-off**. Misalnya, usaha untuk meminimalkan *Response Time* dengan sering melakukan *context switch* (memberi giliran cepat ke banyak proses) dapat meningkatkan overhead sistem dan justru menurunkan *Throughput* secara keseluruhan. Sebaliknya, algoritma yang memaksimalkan *Throughput* dengan menjalankan proses panjang hingga selesai mungkin akan menghasilkan *Response Time* yang buruk bagi proses-proses lain. Tidak ada satu algoritma penjadwalan yang sempurna untuk semua kondisi. Pemilihan algoritma yang tepat sangat bergantung pada tujuan utama sistem (apakah lebih mementingkan throughput, response time, atau fairness) dan karakteristik beban kerja (apakah banyak proses pendek, panjang, CPU-bound, atau I/O-bound).[41] Inilah alasan mengapa terdapat berbagai macam algoritma penjadwalan yang akan kita pelajari.

Bagian 4: Algoritma Penjadwalan First-Come, First-Served (FCFS)

Kita akan memulai pembahasan algoritma penjadwalan dengan yang paling dasar dan intuitif, yaitu First-Come, First-Served (FCFS).

4.1 Prinsip Dasar FCFS: Siapa Cepat Dia Dapat

- **Definisi:** Algoritma FCFS adalah metode penjadwalan CPU yang paling sederhana.[48] Prinsip kerjanya sangat lugas: proses yang **meminta akses ke CPU terlebih dahulu** (yaitu, proses yang **tiba di Ready Queue paling awal**) akan **dilayani atau dieksekusi pertama kali** oleh CPU.[34]
- **Struktur Antrian:** Implementasi FCFS menggunakan struktur data antrian (queue) yang mengikuti prinsip **FIFO (First-In, First-Out)**.[51] Ketika sebuah proses siap untuk dieksekusi, ia akan dimasukkan ke bagian belakang (ekor) antrian Ready. Ketika CPU menjadi tersedia, proses yang berada di bagian paling depan (kepala) antrian akan dipilih untuk dijalankan.
- **Analogi:** Analogi yang paling pas adalah antrian di dunia nyata, seperti antrian di loket bank, kasir supermarket, atau loket tiket.[51] Siapa pun yang datang dan masuk ke antrian lebih dulu akan dilayani terlebih dahulu.
- **Sifat Non-Preemptive:** FCFS merupakan algoritma penjadwalan **non-preemptive**.[35] Ini berarti bahwa sekali sebuah proses telah mendapatkan

alokasi CPU dan mulai berjalan, ia akan terus menggunakan CPU tersebut **sampai ia selesai** sepenuhnya atau sampai ia **secara sukarela melepaskan CPU** (misalnya, karena perlu melakukan operasi I/O sehingga masuk ke status Waiting). Tidak ada proses lain, meskipun mungkin memiliki prioritas lebih tinggi atau waktu eksekusi (burst time) yang lebih pendek, yang dapat merebut atau menyela (preempt) CPU dari proses yang sedang berjalan.[35]

4.2 Langkah Kerja Algoritma FCFS

Cara kerja algoritma FCFS dapat dirinci sebagai berikut:

1. Ketika sebuah proses baru menjadi siap untuk dieksekusi (masuk ke status Ready), Process Control Block (PCB) dari proses tersebut ditambahkan ke **bagian belakang (ekor)** dari Ready Queue (yang berstruktur FIFO).
2. Ketika CPU saat ini tidak sedang menjalankan proses apa pun (menjadi *idle*), penjadwal CPU akan memeriksa Ready Queue.
3. Jika Ready Queue tidak kosong, penjadwal akan memilih proses yang berada di **bagian paling depan (kepala)** antrian.
4. PCB proses yang terpilih tersebut kemudian dihapus dari Ready Queue.
5. Proses tersebut dialokasikan ke CPU dan mulai berjalan (masuk ke status Running).
6. Proses ini akan terus berjalan tanpa gangguan (karena sifat non-preemptive) hingga salah satu dari dua kondisi terpenuhi:
    - Proses menyelesaikan seluruh tugasnya (mencapai akhir programnya). Proses kemudian masuk ke status Terminated.
    - Proses meminta layanan sistem operasi yang mengharuskannya menunggu, seperti operasi I/O. Proses kemudian masuk ke status Waiting.
7. Setelah proses melepaskan CPU (karena Terminated atau Waiting), CPU kembali menjadi *idle*, dan penjadwal akan kembali ke langkah 2 untuk memilih proses berikutnya dari Ready Queue (jika ada).

4.3 Contoh Penerapan FCFS

Mari kita terapkan algoritma FCFS pada contoh kasus berikut untuk memahami cara kerjanya dan menghitung metrik kinerjanya.

- **Kasus:** Terdapat 4 proses (P1, P2, P3, P4) yang tiba di sistem pada waktu yang berbeda dengan kebutuhan waktu eksekusi (Burst Time - BT) sebagai berikut:

Proses	Waktu Kedatangan (Arrival Time - AT) (ms)	Waktu Eksekusi (Burst Time - BT) (ms)

P1	0	8
P2	2	4
P3	3	9
P4	7	5

- **Membuat Gantt Chart:** Gantt chart membantu memvisualisasikan urutan dan waktu eksekusi proses di CPU. Berdasarkan prinsip FCFS (urut berdasarkan AT):
  1. **P1** tiba pertama kali (AT=0). Ia langsung mendapatkan CPU dan berjalan selama 8 ms. Selesai pada waktu **8**.
  2. **P2** tiba pada AT=2. Namun, ia harus menunggu P1 selesai. P2 baru bisa mulai berjalan pada waktu **8**. Ia berjalan selama 4 ms. Selesai pada waktu 8 + 4 = **12**.
  3. **P3** tiba pada AT=3. Ia harus menunggu P1 dan P2 selesai. P3 baru bisa mulai berjalan pada waktu **12**. Ia berjalan selama 9 ms. Selesai pada waktu 12 + 9 = **21**.
  4. **P4** tiba pada AT=7. Ia harus menunggu P1, P2, dan P3 selesai. P4 baru bisa mulai berjalan pada waktu **21**. Ia berjalan selama 5 ms. Selesai pada waktu 21 + 5 = **26**.

Visualisasi dalam Gantt Chart [53]:`+----+----+----+----+`

```
| P1 | P2 | P3 | P4 |
+----+----+----+----+
0 8 12 21 26 (Waktu dalam ms)
```

- Menghitung Waktu Tunggu (Waiting Time - WT) dan AWT:
  Waktu tunggu adalah lamanya proses menunggu di Ready Queue sebelum dieksekusi.
  Rumus: WT=WaktuMulaiEksekusi−WaktuKedatangan 35
  - WT(P1) = 0 - 0 = 0 ms
  - WT(P2) = 8 - 2 = 6 ms
  - WT(P3) = 12 - 3 = 9 ms
  - WT(P4) = 21 - 7 = 14 ms

  Total Waktu Tunggu = 0 + 6 + 9 + 14 = 29 msAWT (Average Waiting Time) = Total WT / Jumlah Proses = 29 / 4 = 7.25 ms 34
- Menghitung Waktu Penyelesaian (Turnaround Time - TAT) dan ATT:
  Turnaround time adalah total waktu sejak proses tiba hingga proses selesai.

Rumus: TAT=WaktuSelesaiEksekusi−WaktuKedatangan 35
(Alternatif: TAT=BurstTime+WaitingTime 35)
Waktu Selesai (Completion Time - CT) dari Gantt Chart: CT(P1)=8, CT(P2)=12, CT(P3)=21, CT(P4)=26.

- TAT(P1) = 8 - 0 = 8 ms (atau 8 + 0 = 8)
- TAT(P2) = 12 - 2 = 10 ms (atau 4 + 6 = 10)
- TAT(P3) = 21 - 3 = 18 ms (atau 9 + 9 = 18)
- TAT(P4) = 26 - 7 = 19 ms (atau 5 + 14 = 19)

Total Turnaround Time = 8 + 10 + 18 + 19 = 55 msATT (Average Turnaround Time) = Total TAT / Jumlah Proses = 55 / 4 = 13.75 ms 34

- Tabel Perhitungan FCFS:
  Tabel berikut merangkum hasil perhitungan untuk mempermudah pemahaman:

Proses	**Arrival Time (AT)**	**Burst Time (BT)**	**Start Time**	**Completion Time (CT)**	**Turnaround Time (TAT = CT - AT)**	**Waiting Time (WT = Start Time - AT)**
P1	0	8	0	8	8	0
P2	2	4	8	12	10	6
P3	3	9	12	21	18	9
P4	7	5	21	26	19	14
Total	-	-	-	-	**55**	**29**
Rata-rata (Average)	-	-	-	-	**13.75**	**7.25**

Tabel ini sangat berguna bagi pemula karena secara visual mengorganisir semua metrik kunci (AT, BT, Waktu Mulai, CT, TAT, WT) untuk setiap proses. Ini memungkinkan mahasiswa melihat hubungan antar metrik (misalnya, bagaimana CT dihitung dari Waktu Mulai + BT, bagaimana TAT dihitung dari CT - AT, bagaimana WT dihitung dari Waktu Mulai - AT) dan melacak perhitungan langkah demi langkah. Ini mengubah

konsep abstrak AWT dan ATT menjadi hasil perhitungan yang konkret dan dapat diverifikasi, memberikan cara sistematis untuk menyelesaikan masalah FCFS dan memperkuat pemahaman tentang definisi setiap metrik.

4.4 Kelebihan FCFS

Algoritma FCFS memiliki beberapa kelebihan, terutama karena kesederhanaannya:

- **Sederhana:** Ini adalah algoritma penjadwalan yang paling mudah untuk dipahami dan diimplementasikan dalam sistem operasi. Logika "siapa datang duluan, dilayani duluan" sangat intuitif.[36]
- **Adil (Berdasarkan Urutan Kedatangan):** Algoritma ini dianggap adil dalam artian bahwa ia tidak membeda-bedakan proses berdasarkan panjangnya atau prioritasnya. Setiap proses akan mendapatkan giliran sesuai dengan urutan kedatangannya.[38]

4.5 Kekurangan FCFS dan Efek Konvoi (Convoy Effect)

Meskipun sederhana, FCFS memiliki beberapa kekurangan signifikan yang membuatnya seringkali tidak efisien dalam praktiknya:

- **Waktu Tunggu Rata-rata Bisa Tinggi:** Kinerja FCFS, terutama dalam hal Average Waiting Time (AWT), sangat bergantung pada urutan kedatangan proses dan burst time-nya. Jika kebetulan proses dengan burst time yang sangat panjang tiba lebih awal, maka proses-proses lain yang mungkin jauh lebih pendek harus menunggu lama, sehingga AWT bisa menjadi sangat tinggi.[36]
- **Tidak Optimal:** Dibandingkan dengan algoritma penjadwalan lain (yang akan dibahas nanti), FCFS jarang menghasilkan AWT atau ATT yang paling minimal.[64] Kinerjanya bisa jauh dari optimal.
- **Convoy Effect (Efek Konvoi):** Ini adalah masalah paling terkenal dan kelemahan utama dari algoritma FCFS.[48]
    - **Penjelasan:** Efek Konvoi terjadi ketika sebuah proses dengan **waktu eksekusi (burst time) yang sangat panjang** (disebut proses "berat" atau *CPU-bound*) kebetulan berada di depan antrian Ready Queue. Karena FCFS bersifat non-preemptive, proses panjang ini akan **memonopoli penggunaan CPU** sampai ia selesai. Akibatnya, **semua proses lain** yang berada di belakangnya, termasuk proses-proses yang mungkin memiliki **burst time sangat pendek** (proses "ringan" atau *I/O-bound*), terpaksa **menunggu dalam antrian untuk waktu yang sangat lama**, meskipun CPU sebenarnya bisa menyelesaikan proses-proses pendek tersebut dengan cepat jika diberi kesempatan.[48]
    - **Analogi:** Bayangkan sebuah truk kontainer besar yang berjalan sangat lambat

di jalan raya satu jalur. Di belakangnya terbentuk antrian panjang mobil-mobil kecil yang sebenarnya bisa melaju jauh lebih cepat, tetapi mereka semua terhambat oleh truk lambat di depan.[61] Truk tersebut adalah proses panjang, dan mobil-mobil kecil adalah proses pendek yang terjebak dalam "konvoi".

- **Dampak Negatif Convoy Effect:**
  1. **Peningkatan Drastis AWT dan ATT:** Proses-proses pendek yang terjebak dalam konvoi akan mengalami waktu tunggu yang sangat lama, yang secara signifikan meningkatkan Average Waiting Time dan Average Turnaround Time sistem secara keseluruhan.[61]
  2. **Penurunan Utilisasi Sumber Daya:** Efek konvoi menyebabkan penggunaan sumber daya yang tidak efisien. Ketika proses panjang sedang menggunakan CPU, perangkat I/O mungkin menganggur karena proses-proses pendek (yang mungkin I/O-bound) sedang menunggu giliran CPU. Sebaliknya, ketika proses panjang tersebut akhirnya melakukan operasi I/O (melepaskan CPU), CPU bisa jadi menganggur karena proses-proses pendek di belakangnya mungkin sudah selesai menunggu I/O atau sudah selesai dieksekusi dengan cepat jika diberi kesempatan sebelumnya.[48] Terjadi pemborosan waktu pada CPU dan perangkat I/O.
  3. **Sistem Tidak Responsif:** Bagi pengguna sistem interaktif, efek konvoi membuat sistem terasa lambat dan tidak responsif karena tugas-tugas singkat harus menunggu tugas panjang selesai.[48]
- **Contoh Ilustrasi Convoy Effect:** Mari kita modifikasi contoh sebelumnya. Misalkan urutan kedatangan proses adalah P3 (BT=9, AT=0), P1 (BT=8, AT=1), P2 (BT=4, AT=2), P4 (BT=5, AT=3).
  - Gantt Chart:
    ```
    +----+----+----+----+
    | P3 | P1 | P2 | P4 |
    +----+----+----+----+
    0 9 17 21 26 (Waktu dalam ms)
    ```
    * Perhitungan WT baru:
    * WT(P3) = 0 - 0 = 0
    * WT(P1) = 9 - 1 = 8
    * WT(P2) = 17 - 2 = 15
    * WT(P4) = 21 - 3 = 18
    * AWT baru = (0 + 8 + 15 + 18) / 4 = 41 / 4 = 10.25 ms
    * Perhitungan TAT baru:

* TAT(P3) = 9 - 0 = 9
* TAT(P1) = 17 - 1 = 16
* TAT(P2) = 21 - 2 = 19
* TAT(P4) = 26 - 3 = 23
* ATT baru = (9 + 16 + 19 + 23) / 4 = 67 / 4 = 16.75 ms
* Perbandingan: Dengan hanya mengubah urutan kedatangan sehingga proses terpanjang (P3) datang pertama, AWT melonjak dari 7.25 ms menjadi 10.25 ms, dan ATT naik dari 13.75 ms menjadi 16.75 ms. Ini menunjukkan betapa sensitifnya FCFS terhadap urutan kedatangan dan bagaimana proses panjang di depan dapat memperburuk kinerja secara signifikan – inilah Convoy Effect.

Kesederhanaan dan "keadilan" berdasarkan urutan kedatangan pada FCFS ternyata harus dibayar dengan potensi inefisiensi yang besar, terutama jika terdapat campuran proses dengan burst time yang bervariasi. Convoy Effect adalah manifestasi paling jelas dari trade-off ini. Fenomena ini menyoroti konflik antara definisi keadilan yang sederhana (siapa datang duluan) dengan tujuan efisiensi sistem (meminimalkan waktu tunggu, memaksimalkan utilisasi). Hal inilah yang mendorong pengembangan algoritma penjadwalan lain yang lebih kompleks, yang mencoba mencari keseimbangan lebih baik antara keadilan dan efisiensi, meskipun seringkali mengorbankan kesederhanaan FCFS.[38, 48, 50, 59, 61]

Bagian 5: Rangkuman (Summary)

- **Poin-Poin Kunci:**
  - **Sistem Operasi (OS)** adalah perangkat lunak inti yang bertindak sebagai perantara antara pengguna, hardware, dan software aplikasi. Fungsinya sangat vital, mencakup manajemen sumber daya, penyediaan antarmuka, eksekusi program, dan keamanan. OS menyederhanakan kompleksitas hardware melalui lapisan abstraksi.
  - **Proses** adalah sebuah program yang sedang aktif dieksekusi. Ia memiliki konteks eksekusi (kode, PC, register, stack, data, heap) dan siklus hidup dengan berbagai status (New, Ready, Running, Waiting, Terminated). Sistem operasi mengelola setiap proses menggunakan **Process Control Block (PCB)**, yang menyimpan semua informasi penting tentang proses tersebut.
  - **Penjadwalan CPU** adalah mekanisme OS untuk memilih proses mana dari Ready Queue yang akan dijalankan oleh CPU. Tujuannya adalah untuk mengoptimalkan kinerja sistem berdasarkan kriteria seperti utilisasi CPU, throughput, turnaround time, waiting time, response time, dan fairness.

Kriteria ini seringkali melibatkan trade-off.
    - **First-Come, First-Served (FCFS)** adalah algoritma penjadwalan paling sederhana yang bersifat non-preemptive. Ia melayani proses berdasarkan urutan kedatangan (prinsip FIFO).
    - **Kelebihan FCFS** adalah kesederhanaan dan kemudahan implementasinya. **Kekurangan utamanya** adalah potensi waktu tunggu rata-rata yang tinggi dan fenomena **Convoy Effect**, di mana proses dengan burst time panjang dapat menghambat proses-proses pendek di belakangnya, menyebabkan penurunan efisiensi dan utilisasi sumber daya sistem.
    - Kinerja FCFS dapat dianalisis dengan menghitung metrik seperti **Average Waiting Time (AWT)** dan **Average Turnaround Time (ATT)** menggunakan bantuan **Gantt Chart**.
- Penutup:
  Selamat! Anda telah menyelesaikan langkah pertama dalam memahami manajemen proses dan penjadwalan CPU dalam sistem operasi. Pemahaman tentang konsep dasar seperti proses, status proses, PCB, tujuan penjadwalan, serta cara kerja dan evaluasi algoritma sederhana seperti FCFS merupakan fondasi yang sangat penting. Anda kini memahami bagaimana OS mengelola eksekusi program dan tantangan yang muncul dalam membagi sumber daya CPU secara efisien. Di pertemuan atau modul berikutnya, kita akan melanjutkan perjalanan ini dengan mempelajari algoritma penjadwalan lain yang dirancang untuk mengatasi kelemahan FCFS dan menawarkan pendekatan yang berbeda dalam mengoptimalkan kinerja sistem. Teruslah bersemangat dalam belajar!

Works cited

1. Pengertian Sistem Operasi Perangkat Lunak serta Jenisnya! - Gramedia, accessed April 29, 2025, https://www.gramedia.com/literasi/pengertian-sistem-operasi/
2. Sistem Operasi: Pengertian, Contoh dan Fungsinya - ITBOX, accessed April 29, 2025, https://itbox.id/blog/sistem-operasi-pengertian-contoh-dan-fungsinya/
3. SISTEM OPERASI - GRACE HUSAIN - UNIVERSITAS NEGERI GORONTALO, accessed April 29, 2025, https://mahasiswa.ung.ac.id/311412099/home/2013/3/21/sistem_operasi.html
4. Mengenal Sistem Operasi | FTI - ARS University, accessed April 29, 2025, https://fti.ars.ac.id/blog/content/mengenal-sistem-operasi
5. sistem operasi | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/document/843067039/sistem-operasi
6. Buku Ajar SISTEM OPERASI - OSF, accessed April 29, 2025, https://osf.io/6yh4e/download
7. mekanisme sistem operasi - Universitas PGRI MADIUN, accessed April 29, 2025, http://eprint.unipma.ac.id/96/1/35.%20Mekanisme_sistem_operasi.pdf

8. Operating System: Definisi, Fungsi dan Jenisnya! - Herza.ID, accessed April 29, 2025, https://herza.id/blog/operating-system-definisi-fungsi-dan-jenisnya/
9. Sistem Operasi Adalah: Pengertian, Fungsi, Jenis dan Contoh - ITBOX, accessed April 29, 2025, https://itbox.id/blog/sistem-operasi-adalah/
10. Sistem Operasi: Arti, Fungsi, Jenis, dan Contoh - detikcom, accessed April 29, 2025, https://www.detik.com/jabar/jabar-gaskeun/d-6265048/sistem-operasi-arti-fungsi-jenis-dan-contoh
11. Sistem operasi - OnnoWiki - Onno Center, accessed April 29, 2025, http://onnocenter.or.id/wiki/index.php/Sistem_operasi
12. Mengenal Konsep Dasar Sistem Operasi - BBPPMPV Pertanian, accessed April 29, 2025, https://bbppmpvpertanian.kemdikbud.go.id/mengenal-konsep-dasar-sistem-operasi/
13. repository.bsi.ac.id, accessed April 29, 2025, https://repository.bsi.ac.id/repo/files/236871/download/Modul-Sistem-Operasi.pdf
14. PROCESS MANAGEMENT PROGRAM STUDI TEKNIK ..., accessed April 29, 2025, https://lms-paralel.esaunggul.ac.id/pluginfile.php?file=%2F75159%2Fmod_resource%2Fcontent%2F1%2FKelompok3_ProcessManagement%28Tommy_Reio_Dendy_Dodot_Junisai%29.pdf
15. arna.lecturer.pens.ac.id, accessed April 29, 2025, https://arna.lecturer.pens.ac.id/Diktat_SO/3.Proses%20Proses.pdf
16. Proses di Sistem Operasi | PPT - SlideShare, accessed April 29, 2025, https://www.slideshare.net/eDiagNostic/proses-di-sistem-operasi
17. SISTEM OPERASI - Repository USM, accessed April 29, 2025, https://repository.usm.ac.id/files/onlinepublications/G007/20170518061133-Sistem-Operasi.pdf
18. Deskripsi dan Kontrol Proses - WordPress.com, accessed April 29, 2025, https://komputersederhana.files.wordpress.com/2018/06/pert-03-ch03-deskripsi-dan-kontrol-proses-20100820.pdf
19. osf.io, accessed April 29, 2025, https://osf.io/pfgka/download
20. SISTEM OPERASI KOMPUTER - LMS-SPADA INDONESIA, accessed April 29, 2025, https://lmsspada.kemdikbud.go.id/pluginfile.php/183483/mod_resource/content/1/6%20-%20Sistem%20Operasi.pdf
21. Sistem Operasi Komputer: Pengertian, Fungsi, Jenis, Cara Kerja, dan Contohnya, accessed April 29, 2025, https://www.kompas.com/skola/read/2021/04/15/144350269/sistem-operasi-komputer-pengertian-fungsi-jenis-cara-kerja-dan-contohnya?page=all
22. Apa Yang Dimaksud Dengan Proses Dalam Sistem Operasi - Scribd, accessed April 29, 2025, https://id.scribd.com/document/784794020/Apa-yang-dimaksud-dengan-proses-dalam-sistem-operasi
23. Sistem operasi - Wikipedia bahasa Indonesia, ensiklopedia bebas, accessed April 29, 2025, https://id.wikipedia.org/wiki/Sistem_operasi
24. MANAJEMEN PROSES, accessed April 29, 2025,

https://repository.dinus.ac.id/docs/ajar/3-Manajemen_Proses.pptx
25. SistemOperasi Definisi | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/document/619509010/SistemOperasi-definisi
26. What is Process Control Block (PCB)? - Scaler Topics, accessed April 29, 2025, https://www.scaler.com/topics/operating-system/process-control-block-in-os/
27. What is a process control block PCB and its components. - Learn Steps, accessed April 29, 2025, https://www.learnsteps.com/what-is-a-process-control-block-pcb-and-its-components/
28. Process control block - Wikipedia, accessed April 29, 2025, https://en.wikipedia.org/wiki/Process_control_block
29. Process Table and Process Control Block (PCB) - GeeksforGeeks, accessed April 29, 2025, https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/
30. Process Control Block in OS | GeeksforGeeks, accessed April 29, 2025, https://www.geeksforgeeks.org/process-control-block-in-os/
31. What is Process Control Block (PCB) - Tutorialspoint, accessed April 29, 2025, https://www.tutorialspoint.com/what-is-process-control-block-pcb
32. Jelaskan secara singkat pengertian dan fungsi PCB - IPCB.net, accessed April 29, 2025, https://ipcb.net/id/10320.html
33. Dasar-Dasar Komputer: Memahami Macam-Macam Sistem Operasi - GCFGlobal, accessed April 29, 2025, https://edu.gcfglobal.org/en/tr_id-computer-basics/memahami-macammacam-sistem-operasi/1/
34. Analisis Perbandingan Algoritma Penjadwalan CPU First Come First Serve (FCFS) Dan Round Robin - ResearchGate, accessed April 29, 2025, https://www.researchgate.net/publication/358113599_Analisis_Perbandingan_Algoritma_Penjadwalan_CPU_First_Come_First_Serve_FCFS_Dan_Round_Robin
35. repository.dinus.ac.id, accessed April 29, 2025, https://repository.dinus.ac.id/docs/ajar/PenjadwalanProses_ver2.pdf
36. Penjadwalan / Scheduling – School of Computer Science, accessed April 29, 2025, https://socs.binus.ac.id/2020/11/16/penjadwalan-scheduling/
37. Penjadwalan Proses - Aristy's Blog, accessed April 29, 2025, https://aristi.dharmawacana.ac.id/sistem-operasi/penjadwalan-proses/
38. Penjadwalan Proses - SPADA UNS, accessed April 29, 2025, https://spada.uns.ac.id/mod/resource/view.php?id=11725
39. Penjadwalan Proses | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/doc/59260022/penjadwalan-proses
40. Slide 1, accessed April 29, 2025, https://repository.dinus.ac.id/docs/ajar/Algoritma_Penjadwalan_Proses_Bag_1.ppt
41. Makalah Penjadwalan Proses | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/document/691021097/MAKALAH-PENJADWALAN-PROSES
42. BAB I PENDAHULUAN 1.1 Latar Belakang Dengan berkembangnya ilmu teknologi komputer pada saat ini, semakin meningkat untuk memba - Teknokrat Repository, accessed April 29, 2025,

http://repository.teknokrat.ac.id/475/2/%2811%29Bablriadi.pdf
43. Algoritma penjadwalan proses | PPT - SlideShare, accessed April 29, 2025, https://www.slideshare.net/slideshow/algoritma-penjadwalan-proses-150977782/150977782
44. Analisis Perbandingan Algoritma Penjadwalan CPU First Come First Serve (FCFS) Dan Round Robin - OJS Seminar Indonesia Journal, accessed April 29, 2025, https://ejurnal.seminar-id.com/index.php/bits/article/download/1047/725/
45. Metode Evaluasi Penjadwalan | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/doc/60223029/metode-evaluasi-penjadwalan
46. Penjadwalan Proses | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/document/678080644/PENJADWALAN-PROSES
47. Penjadwalan-Proses.ppt - SlideShare, accessed April 29, 2025, https://www.slideshare.net/slideshow/penjadwalanprosesppt/263272211
48. Advantages and Disadvantages of various CPU scheduling ..., accessed April 29, 2025, https://www.geeksforgeeks.org/advantages-and-disadvantages-of-various-cpu-scheduling-algorithms/
49. 5 BAB II TINJAUAN PUSTAKA 2.1 Round Robin Round robin merupakan salah satu algoritma penjadwalan proses paling sederhana pada si - EPrints Repository UNTIRTA, accessed April 29, 2025, https://eprints.untirta.ac.id/20390/3/Hanif%20Anggit%20Wicaksono_3332150017_02.pdf
50. Diagram FCFS: Gita Chandika Putri D3 IF-34-02 613101046 | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/doc/50062463/pti
51. ANALISIS PENERAPAN ALGORITMA FIRST COME FIRST SERVED ..., accessed April 29, 2025, https://jim.teknokrat.ac.id/index.php/informatika/article/download/931/384
52. Algoritma First Come First Served | PDF - Scribd, accessed April 29, 2025, https://pt.scribd.com/doc/92948220/Algoritma-First-Come-First-Served
53. Menghitung Average Waiting Time Dalam Algoritma Penjadwalan ..., accessed April 29, 2025, https://www.scribd.com/document/332540969/Menghitung-Average-Waiting-Time-Dalam-Algoritma-Penjadwalan-Proses-First-Come-First-Served
54. Implementasi Metode FCFS(First Come First Served) Pada Aplikasi Pemesanan Makanan Menggunakan QR Code Berbasis Web service Stud, accessed April 29, 2025, http://repository.unmuhjember.ac.id/2694/9/ARTIKEL%20JURNAL.pdf
55. Analisis Performansi Algoritma Modified Round Robin Berdasarkan Prioritas dan Service Time untuk Penjadwalan Proses - Open Library Telkom University, accessed April 29, 2025, https://openlibrary.telkomuniversity.ac.id/pustaka/133708/analisis-performansi-algoritma-modified-round-robin-berdasarkan-prioritas-dan-service-time-untuk-penjadwalan-proses.html
56. Page 95 - C:\Users\RENO\Documents\MK Sistem Operasi\Folder Baru\, accessed April 29, 2025, https://online.flipbuilder.com/rhml/wlhc/files/basic-html/page95.html

57. First Come First Serve (FCFS) | CPU Scheduling Algorithm - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=TYzMMsyUGag&pp=0gcJCdgAo7VqN5tD
58. FCFS algorithm - an example - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=QHvjJS62yc4
59. Convoy Effect in FCFS Scheduling - Tutorialspoint, accessed April 29, 2025, https://www.tutorialspoint.com/convoy-effect-in-fcfs
60. 16 BAB 3 LANDASAN TEORI 3.1. Pengertian Penjadwalan Penjadwalan adalah aktivitas perencanaan untuk menentukan kapan dan di mana, accessed April 29, 2025, http://e-journal.uajy.ac.id/7239/4/3TI04609.pdf
61. Convoy Effect in Operating Systems | GeeksforGeeks, accessed April 29, 2025, https://www.geeksforgeeks.org/convoy-effect-operating-systems/
62. Solved Use the following data to draw a Gantt chart for FCFS | Chegg.com, accessed April 29, 2025, https://www.chegg.com/homework-help/questions-and-answers/use-following-data-draw-gantt-chart-fcfs-scheduling-show-wait-time-job-average-wait-time-p-q167887151
63. Solved 1. Consider the following processes with arrival time | Chegg.com, accessed April 29, 2025, https://www.chegg.com/homework-help/questions-and-answers/1-consider-following-processes-arrival-time-burst-time-draw-gantt-chart-fcfs-cpu-schedulin-q95681893
64. First Come First Served Scheduling (Solved Problem 1) - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=VSMAjMfJ6KQ
65. PRACTICE QUESTION ON FIRST COME FIRST SERVE (FCFS) SCHEDULING ALGORITHM - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=UQctxlnXk3k
66. First Come First Serve (FCFS) SCHEDULING ALGORITHM Example: Gantt Chart and Metrics Computation - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=_h1E5kOY4DQ
67. LATIHAN SOAL | PDF - Scribd, accessed April 29, 2025, https://id.scribd.com/document/847961731/LATIHAN-SOAL
68. First Come First Serve(FCFS) CPU Scheduling Algorithm with Example - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=9BuAbJXRdzU
69. Convoy Effect in FCFS || Lesson 17 || Operating Systems || Learning Monkey || - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=ra7OA9g98ks
70. 3.7 Convoy effect advantage disadvantage of FCFS scheduling algorithm first come first serve - YouTube, accessed April 29, 2025, https://www.youtube.com/watch?v=cc-9wfloMtg