

LAPORAN STUDI KASUS PENJADWALAN PROSES FCFS (FIRST COME FIRST SERVED)

Nama : Muhammad Fiqri
NIM : 2212000062
Kelas : IF Siang A
Mata Kuliah : Sistem Operasi
Dosen : Elida Tuti Siregar , M.Kom

PENDAHULUAN

FCFS (First-Come, First-Served) adalah algoritma penjadwalan yang paling sederhana dan paling umum digunakan dalam sistem operasi serta manajemen antrean. Prinsip dasar dari FCFS adalah melayani proses atau tugas berdasarkan urutan kedatangan; proses yang datang lebih awal akan dilayani terlebih dahulu tanpa mempertimbangkan durasi eksekusi atau prioritas. Meskipun metode ini mudah dipahami dan diimplementasikan, penggunaannya dapat menghasilkan efek yang kurang efisien dalam situasi tertentu, seperti keterlambatan bagi proses yang memiliki waktu eksekusi yang lebih singkat.

FCFS banyak diterapkan dalam berbagai konteks, mulai dari sistem antrian di layanan pelanggan hingga pemrosesan tugas dalam sistem komputasi. Dalam laporan ini, akan dibahas lebih lanjut mengenai cara kerja FCFS, aplikasi praktisnya, serta kelebihan dan kekurangan yang perlu diperhatikan. Dengan memahami karakteristik dan penerapan FCFS, diharapkan pembaca dapat mengevaluasi metode ini dalam konteks sistem penjadwalan yang lebih kompleks dan menentukan kapan penggunaannya paling sesuai.

PENGERTIAN FCFS (FIRST COME FIRST SERVED)

FCFS (First-Come, First-Served) adalah algoritma penjadwalan yang digunakan untuk mengatur antrian dalam sistem komputer, di mana proses atau tugas yang pertama kali datang akan dilayani terlebih dahulu. Algoritma ini paling sederhana dan mudah dipahami, serta sering digunakan dalam berbagai aplikasi, termasuk dalam manajemen sumber daya komputer dan sistem antrean. Dalam konteks sistem operasi, FCFS adalah salah satu metode dasar untuk menjadwalkan proses dalam CPU.

FCFS bekerja berdasarkan prinsip sederhana: proses yang datang pertama kali akan mendapatkan akses ke CPU terlebih dahulu. Dalam sistem ini, proses ditempatkan dalam antrian dan diproses satu per satu. Ketika sebuah proses selesai, sistem akan melanjutkan ke proses berikutnya dalam antrean. Meskipun

metode ini mudah diterapkan, waktu tunggu dapat meningkat jika proses yang lebih panjang datang terlebih dahulu, sehingga dapat mempengaruhi efisiensi keseluruhan sistem.

FCFS banyak digunakan dalam berbagai bidang, termasuk dalam sistem operasi komputer, layanan pelanggan, dan manajemen antrian. Misalnya, dalam antrian bank, nasabah yang tiba lebih awal akan dilayani terlebih dahulu. Dalam konteks komputasi, FCFS digunakan untuk mengelola proses dalam CPU, memastikan bahwa tidak ada proses yang diblokir terlalu lama. Hal ini menjadikannya penting untuk memahami algoritma ini sebagai dasar dalam sistem penjadwalan yang lebih kompleks.

Salah satu kelebihan utama dari FCFS adalah kesederhanaannya. Implementasi algoritma ini tidak memerlukan kompleksitas tambahan, sehingga mudah dipahami dan diterapkan. Namun, terdapat juga kekurangan, seperti potensi inefisiensi waktu tunggu. Jika proses panjang datang terlebih dahulu, proses yang lebih pendek harus menunggu lebih lama, yang dikenal sebagai fenomena "convoy effect". Hal ini dapat mengakibatkan penggunaan sumber daya yang kurang optimal dalam jangka panjang.

FCFS adalah algoritma penjadwalan yang sederhana namun efektif, dengan aplikasi yang luas dalam berbagai bidang. Memahami cara kerjanya dan implikasinya sangat penting, terutama dalam konteks sistem yang lebih kompleks. Meskipun memiliki kelemahan, FCFS tetap menjadi dasar yang penting dalam mempelajari algoritma penjadwalan lainnya dan bagaimana proses dikelola dalam sistem komputasi.

STUDI KASUS : SISTEM MANAJEMEN ANTRIAN PADA SERVER WEB

Sebuah perusahaan e-commerce mengoperasikan server web untuk menangani permintaan pengguna yang datang dari berbagai sumber, termasuk aplikasi mobile dan situs web. Setiap permintaan yang masuk harus diproses untuk menampilkan halaman produk, memproses pembayaran, dan melakukan berbagai fungsi lainnya.

Perusahaan menghadapi tantangan dalam menangani lonjakan trafik saat berlangsungnya promosi besar-besaran. Dengan banyaknya permintaan yang datang secara bersamaan, server mengalami kesulitan dalam memproses setiap permintaan, yang mengakibatkan waktu tunggu yang lama bagi pengguna.

Perusahaan memutuskan untuk menerapkan algoritma FCFS (First-Come, First-Served) guna mengelola permintaan yang masuk. Dengan FCFS, setiap permintaan akan diproses sesuai dengan urutan kedatangan. Ini berarti bahwa permintaan pertama yang diterima akan menjadi yang pertama diproses, tanpa memperhatikan prioritas atau jenis permintaan.

Data permintaan dalam studi kasus ini mencakup beberapa atribut penting yang berfungsi untuk menjelaskan waktu kedatangan dan durasi proses setiap permintaan. Berikut rincian masing-masing atribut dalam data permintaan :

Permintaan	Waktu Kedatangan (ms)	Waktu Proses (ms)
A	0	5
B	1	3
C	2	8
D	3	6
E	4	4

1. Permintaan : Setiap permintaan diberi label (A, B, C, D, E) untuk mempermudah identifikasi dan pelacakan urutan permintaan yang masuk.
2. Waktu Kedatangan (ms) : Menunjukkan waktu (dalam milidetik) ketika setiap permintaan masuk ke dalam sistem. Nilai waktu kedatangan ini penting karena algoritma FCFS memprioritaskan permintaan yang datang lebih awal. Misalnya, permintaan A datang pertama pada waktu 0 ms, diikuti oleh permintaan B pada 1 ms, C pada 2 ms, D pada 3 ms, dan E pada 4 ms.
3. Waktu Proses (ms) : Menunjukkan berapa lama waktu yang dibutuhkan untuk menyelesaikan setiap permintaan (juga dalam milidetik). Misalnya, permintaan A memerlukan 5 ms untuk selesai, sementara permintaan C membutuhkan waktu paling lama, yaitu 8 ms.

Setelah menerapkan algoritma FCFS, perusahaan berhasil mengurangi waktu tunggu rata-rata bagi pengguna. Meskipun FCFS mudah diimplementasikan dan menghasilkan urutan pemrosesan yang adil, perusahaan juga menyadari bahwa dalam beberapa situasi, hal ini dapat menyebabkan "convoy effect," di mana permintaan yang lebih lama memperlambat semua permintaan berikutnya.

Setelah beberapa waktu menggunakan FCFS, perusahaan melakukan evaluasi terhadap sistem mereka. Mereka menemukan bahwa meskipun FCFS membantu menyeimbangkan beban, mereka perlu mempertimbangkan algoritma penjadwalan lain, seperti Shortest Job First (SJF) atau Round Robin, terutama untuk situasi di mana waktu respons lebih kritis.

Studi kasus ini menunjukkan penerapan algoritma FCFS dalam manajemen proses pada server web, menggarisbawahi pentingnya pemilihan algoritma penjadwalan yang tepat dalam sistem operasi untuk meningkatkan efisiensi dan pengalaman pengguna.

PENJELASAN CODE PROGRAM

Berikut adalah penjelasan lengkap kode program algoritma First-Come, First-Served (FCFS) untuk kasus *Sistem Manajemen Antrian pada Server Web*, dimana server web sebuah perusahaan e-commerce menangani permintaan pengguna. Server ini memproses halaman produk dan pembayaran berdasarkan urutan kedatangan permintaan. FCFS memastikan bahwa permintaan yang masuk pertama diproses lebih dulu, tanpa memperhatikan prioritas. Program ini bertujuan untuk menghitung waktu penyelesaian (completion time), waktu turnaround, dan waktu tunggu (waiting time) dari setiap permintaan yang diterima.

1. Kelas Request

Pertama, kita membuat kelas Request untuk merepresentasikan setiap permintaan pengguna.

```
class Request:
    def __init__(self, name, arrival_time, processing_time):
        self.name = name
        self.arrival_time = arrival_time
        self.processing_time = processing_time
        self.completion_time = 0
        self.turnaround_time = 0
        self.waiting_time = 0
```

Kelas ini memiliki atribut berikut :

- *name* : Nama permintaan, seperti "A", "B", dll.
- *arrival_time* : Waktu kedatangan permintaan dalam milidetik (ms).
- *processing_time* : Waktu yang dibutuhkan untuk memproses permintaan dalam ms.
- *completion_time* : Waktu saat permintaan selesai diproses. Diinisialisasi dengan 0 karena nilainya baru dihitung dalam fungsi *calculate_fcfs*.
- *turnaround_time* : Waktu total yang dihabiskan permintaan dalam sistem, dihitung sebagai waktu penyelesaian dikurangi waktu kedatangan.
- *waiting_time* : Waktu tunggu sebelum permintaan mulai diproses, dihitung sebagai waktu turnaround dikurangi waktu proses.

Pada studi kasus ini, setiap permintaan pengguna yang diterima server e-commerce akan diwakili oleh objek Request.

2. Fungsi *calculate_fcfs*

Fungsi ini menghitung waktu penyelesaian, turnaround, dan waktu tunggu dengan menerapkan algoritma FCFS.

```
def calculate_fcfs(requests):
    current_time = 0

    for request in requests:
        # Update waktu saat ini jika lebih kecil dari waktu kedatangan permintaan
        if current_time < request.arrival_time:
            current_time = request.arrival_time

        # Hitung waktu penyelesaian
        current_time += request.processing_time
        request.completion_time = current_time

        # Hitung waktu turnaround
        request.turnaround_time = request.completion_time - request.arrival_time

        # Hitung waktu tunggu
        request.waiting_time = request.turnaround_time - request.processing_time
```

Fungsi `calculate_fcfs` melakukan perhitungan sebagai berikut :

- *current_time* : Variabel untuk melacak waktu berjalan yang akan diperbarui sesuai dengan setiap permintaan yang masuk.
- *Loop for* : Setiap permintaan diproses berdasarkan urutan kedatangan.
 - Jika waktu saat ini lebih kecil dari waktu kedatangan, server idle sampai waktu kedatangan permintaan berikutnya.
 - *completion_time* : Dihitung sebagai waktu saat ini (*current_time*) ditambah waktu proses dari permintaan.
 - *turnaround_time* : Selisih waktu antara waktu penyelesaian dan waktu kedatangan, mewakili total waktu yang dihabiskan dalam sistem.
 - *waiting_time* : Selisih waktu antara waktu turnaround dan waktu proses, menunjukkan waktu tunggu sebelum permintaan diproses.

Pada kasus ini, ketika server menerima banyak permintaan secara bersamaan, algoritma ini menjaga urutan yang adil dengan memproses permintaan pertama terlebih dahulu, sesuai dengan waktu kedatangan.

3. Fungsi `display_results`

Fungsi ini digunakan untuk menampilkan hasil perhitungan waktu penyelesaian, turnaround, dan waktu tunggu dari setiap permintaan dalam bentuk tabel yang rapi.

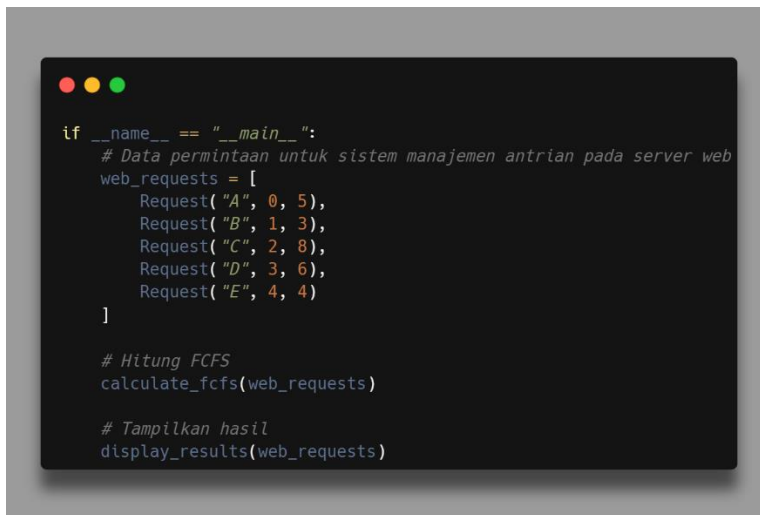
```
def display_results(requests):
    print(f"{'Permintaan':<15}{'Kedatangan':<15}{'Proses':<10}{'Penyelesaian':<15}{'Turnaround':<15}{'Tunggu':<10}")
    for request in requests:
        print(f"{request.name:<15}{request.arrival_time:<15}{request.processing_time:<10}{request.completion_time:<15}{request.turnaround_time:<15}{request.waiting_time:<10}")
```

- `print(f"{'Permintaan':<15}...")` : Bagian ini mencetak header tabel yang menampilkan nama setiap kolom.
 - *Permintaan* : Kolom ini menunjukkan nama permintaan (misalnya A, B, C).
 - *Kedatangan* : Waktu kedatangan setiap permintaan.
 - *Proses* : Waktu yang dibutuhkan server untuk memproses permintaan.
 - *Penyelesaian* : Waktu penyelesaian permintaan (waktu di mana pemrosesan selesai).
 - *Turnaround* : Waktu total yang dihabiskan permintaan dalam sistem, dari kedatangan hingga penyelesaian.
 - *Tunggu* : Waktu yang dihabiskan permintaan dalam antrian sebelum diproses.
- *Format {:<15}* : Format ini menyesuaikan lebar setiap kolom agar konsisten dan rapi. Angka 15 menentukan lebar kolom dalam karakter. Misalnya, `{:<15}` berarti kolom tersebut lebar 15 karakter dan rata kiri.

- *Loop for request in requests* : Loop ini mengulang setiap permintaan yang ada di daftar requests, sehingga data masing-masing permintaan dapat dicetak dalam format tabel yang sama.
 - *request.name* : Nama dari setiap permintaan.
 - *request.arrival_time* : Waktu kedatangan dari setiap permintaan.
 - *request.processing_time* : Waktu yang diperlukan untuk memproses setiap permintaan.
 - *request.completion_time* : Waktu penyelesaian setiap permintaan.
 - *request.turnaround_time* : Waktu turnaround (total waktu dalam sistem).
 - *request.waiting_time* : Waktu tunggu dari setiap permintaan.

4. Eksekusi Program

Pada bagian ini, kita menjalankan seluruh program, dari memasukkan data permintaan hingga menghitung dan menampilkan hasilnya.



```

if __name__ == "__main__":
    # Data permintaan untuk sistem manajemen antrian pada server web
    web_requests = [
        Request("A", 0, 5),
        Request("B", 1, 3),
        Request("C", 2, 8),
        Request("D", 3, 6),
        Request("E", 4, 4)
    ]

    # Hitung FCFS
    calculate_fcfs(web_requests)

    # Tampilkan hasil
    display_results(web_requests)
  
```

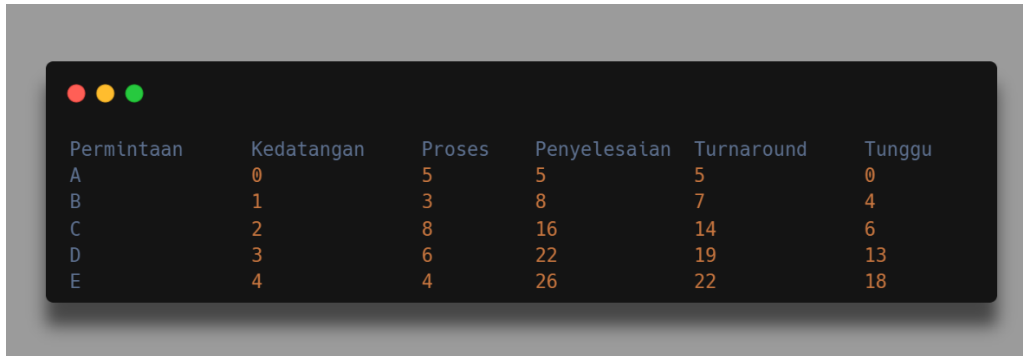
- *if __name__ == "__main__":* : Bagian ini memastikan bahwa kode di dalam blok ini hanya akan dieksekusi jika file ini dijalankan sebagai program utama. Jika file ini diimpor ke file lain, kode di dalam blok ini tidak akan dijalankan.
- *web_requests = [...]* : Bagian ini menginisialisasi daftar *web_requests* yang berisi lima objek *Request* untuk setiap permintaan, dengan atribut nama, waktu kedatangan, dan waktu proses yang diberikan sesuai studi kasus.
 - Permintaan "A" memiliki waktu kedatangan 0 ms dan waktu proses 5 ms.
 - Permintaan "B" memiliki waktu kedatangan 1 ms dan waktu proses 3 ms.
 - Permintaan "C" memiliki waktu kedatangan 2 ms dan waktu proses 8 ms.
 - Permintaan "D" memiliki waktu kedatangan 3 ms dan waktu proses 6 ms.
 - Permintaan "E" memiliki waktu kedatangan 4 ms dan waktu proses 4 ms.
- *calculate_fcfs(web_requests)* : Fungsi *calculate_fcfs* ini dipanggil dengan parameter *web_requests* untuk menghitung waktu penyelesaian, turnaround, dan waktu tunggu dari setiap

permintaan. Setelah pemanggilan fungsi ini, setiap objek Request dalam daftar `web_requests` akan memiliki atribut `completion_time`, `turnaround_time`, dan `waiting_time` yang sudah diisi.

- `display_results(web_requests)` : Fungsi `display_results` kemudian dipanggil untuk menampilkan hasil perhitungan waktu penyelesaian, turnaround, dan waktu tunggu dalam bentuk tabel.

5. Output Program

Output program akan menampilkan hasil perhitungan dalam bentuk tabel seperti berikut.



Permintaan	Kedatangan	Proses	Penyelesaian	Turnaround	Tunggu
A	0	5	5	5	0
B	1	3	8	7	4
C	2	8	16	14	6
D	3	6	22	19	13
E	4	4	26	22	18

Setelah eksekusi program, hasilnya ditampilkan dalam bentuk tabel yang berisi kolom-kolom utama seperti Permintaan, Kedatangan, Proses, Penyelesaian, Turnaround, dan Tunggu.

Permintaan A

- Kedatangan : Tiba pada 0 ms.
- Proses : Memerlukan waktu 5 ms untuk diselesaikan.
- Penyelesaian : Selesai diproses pada 5 ms. Karena permintaan ini tiba pertama kali, langsung diproses tanpa menunggu.
- Turnaround : Total waktu dalam sistem adalah 5 ms, yaitu sejak kedatangan hingga penyelesaian.
- Tunggu : Karena diproses segera setelah tiba, waktu tunggu adalah 0 ms.

Permintaan B

- Kedatangan : Tiba pada 1 ms.
- Proses : Memerlukan waktu 3 ms untuk diselesaikan.
- Penyelesaian : Selesai diproses pada 8 ms. Karena permintaan A sedang diproses saat kedatangan B, permintaan ini harus menunggu.
- Turnaround : Total waktu di dalam sistem adalah 7 ms (dari 1 ms hingga 8 ms).
- Tunggu : Harus menunggu 4 ms sebelum dapat diproses.

Permintaan C

- Kedatangan : Tiba pada 2 ms.
- Proses : Memerlukan waktu 8 ms untuk diselesaikan.
- Penyelesaian : Selesai diproses pada 16 ms. Permintaan C datang setelah A dan B, sehingga harus menunggu hingga B selesai.
- Turnaround : Total waktu di dalam sistem adalah 14 ms.

- Tunggu : Mengalami waktu tunggu 6 ms sebelum diproses.

Permintaan D

- Kedatangan : Tiba pada 3 ms.
- Proses : Memerlukan waktu 6 ms untuk diselesaikan.
- Penyelesaian : Selesai diproses pada 22 ms, setelah permintaan C selesai diproses.
- Turnaround : Total waktu di dalam sistem adalah 19 ms.
- Tunggu : Mengalami waktu tunggu yang lebih tinggi, yaitu 13 ms.

Permintaan E

- Kedatangan : Tiba pada 4 ms.
- Proses : Memerlukan waktu 4 ms untuk diselesaikan.
- Penyelesaian: Selesai diproses pada 26 ms.
- Turnaround : Total waktu di dalam sistem adalah 22 ms.
- Tunggu : Waktu tunggu meningkat menjadi 18 ms karena antrian semakin panjang.

Hasil output menunjukkan bahwa dalam sistem FCFS (First-Come, First-Served), setiap permintaan diproses sesuai urutan kedatangannya. Sistem ini sederhana, namun saat permintaan yang membutuhkan waktu proses lama datang lebih awal, seperti pada Permintaan C, permintaan-permintaan berikutnya harus menunggu lama. Situasi ini disebut convoy effect, di mana proses-proses yang lebih kecil tertahan di belakang proses yang lebih besar, sehingga waktu tunggu meningkat seiring jumlah permintaan.

PERHITUNGAN DAN ANALISIS HASIL FCFS

Langkah-langkah manual berikut ini menunjukkan cara memperoleh nilai Penyelesaian, Turnaround, dan Tunggu untuk setiap permintaan dalam sistem manajemen antrian server web menggunakan algoritma FCFS.

Langkah-langkah Manual

1. Tentukan Penyelesaian untuk setiap Permintaan

Penyelesaian menunjukkan kapan suatu permintaan selesai diproses. Rumus perhitungannya :

$$\text{Penyelesaian} = \text{Penyelesaian permintaan sebelumnya} + \text{Waktu Proses permintaan saat ini}$$

2. Hitung Turnaround untuk setiap Permintaan

Turnaround menunjukkan total waktu yang dihabiskan permintaan dalam sistem, dari kedatangan hingga penyelesaian. Rumus perhitungannya :

$$\text{Turnaround} = \text{Penyelesaian} - \text{Waktu Kedatangan}$$

3. Hitung Tunggu untuk setiap Permintaan

Tunggu adalah waktu yang dihabiskan permintaan dalam antrian sebelum mulai diproses. Rumus perhitungannya :

$$Tunggu = Turnaround - Waktu Proses$$

Tabel Hasil Perhitungan

Permintaan	Kedatangan	Proses	Penyelesaian	Turnaround	Tunggu
A	0	5	5	5	0
B	1	3	8	7	4
C	2	8	16	14	6
D	3	6	22	19	13
E	4	4	26	22	18

Langkah-langkah Perhitungan Manual

1. Permintaan A

- $Penyelesaian = 0 \text{ (kedatangan)} + 5 \text{ (proses)} = 5 \text{ ms}$
- $Turnaround = 5 \text{ (penyelesaian)} - 0 \text{ (kedatangan)} = 5 \text{ ms}$
- $Tunggu = 5 \text{ (turnaround)} - 5 \text{ (proses)} = 0 \text{ ms}$

2. Permintaan B

- $Penyelesaian = 5 \text{ (penyelesaian A)} + 3 \text{ (proses B)} = 8 \text{ ms}$
- $Turnaround = 8 \text{ (penyelesaian)} - 1 \text{ (kedatangan)} = 7 \text{ ms}$
- $Tunggu = 7 \text{ (turnaround)} - 3 \text{ (proses)} = 4 \text{ ms}$

3. Permintaan C

- $Penyelesaian = 8 \text{ (penyelesaian B)} + 8 \text{ (proses C)} = 16 \text{ ms}$
- $Turnaround = 16 \text{ (penyelesaian)} - 2 \text{ (kedatangan)} = 14 \text{ ms}$
- $Tunggu = 14 \text{ (turnaround)} - 8 \text{ (proses)} = 6 \text{ ms}$

4. Permintaan D

- $Penyelesaian = 16 \text{ (penyelesaian C)} + 6 \text{ (proses D)} = 22 \text{ ms}$
- $Turnaround = 22 \text{ (penyelesaian)} - 3 \text{ (kedatangan)} = 19 \text{ ms}$
- $Tunggu = 19 \text{ (turnaround)} - 6 \text{ (proses)} = 13 \text{ ms}$

5. Permintaan E

- $Penyelesaian = 22 \text{ (penyelesaian D)} + 4 \text{ (proses E)} = 26 \text{ ms}$
- $Turnaround = 26 \text{ (penyelesaian)} - 4 \text{ (kedatangan)} = 22 \text{ ms}$
- $Tunggu = 22 \text{ (turnaround)} - 4 \text{ (proses)} = 18 \text{ ms}$

Hasil perhitungan manual menunjukkan bahwa nilai yang diperoleh sesuai dengan output program, yang memastikan bahwa algoritma First-Come, First-Served (FCFS) diimplementasikan dengan benar. Setiap parameter yang dihitung—mulai dari waktu penyelesaian, waktu turnaround, hingga waktu tunggu—memberikan hasil yang konsisten dan akurat.

• Kesamaan Hasil

Setiap nilai dalam output program sama dengan hasil perhitungan manual. Misalnya, untuk permintaan A, waktu penyelesaian adalah 5 ms, waktu turnaround juga 5 ms, dan waktu tunggu adalah 0 ms. Hal ini menunjukkan bahwa permintaan A diproses segera setelah tiba tanpa ada

penundaan. Hal serupa berlaku untuk permintaan lainnya (B, C, D, dan E) yang juga memiliki nilai yang identik antara perhitungan manual dan output program.

- **Kesesuaian Waktu**

Waktu yang dihitung untuk setiap permintaan dalam kedua metode analisis tersebut sejalan. Untuk permintaan B, misalnya, waktu penyelesaian dihitung sebagai 8 ms, yang sama baik dalam perhitungan manual maupun output program. Proses ini berlanjut untuk permintaan C, D, dan E, di mana semua waktu yang dicatat sesuai dengan logika algoritma FCFS, di mana setiap permintaan diproses sesuai urutan kedatangannya.

- **Efisiensi Sistem**

Meskipun FCFS dikenal sebagai algoritma yang adil karena memproses permintaan berdasarkan urutan kedatangan, sistem ini juga memiliki potensi untuk meningkatkan waktu tunggu bagi permintaan yang lebih lama jika mereka datang lebih dulu. Misalnya, dalam studi kasus ini, permintaan C yang memerlukan waktu proses lebih lama (8 ms) dapat menyebabkan permintaan D dan E menunggu lebih lama. Hal ini menunjukkan adanya "convoy effect," di mana proses yang lebih lama dapat mengakibatkan penundaan untuk permintaan yang lebih cepat.

Kesimpulan

Analisis ini menunjukkan bahwa implementasi algoritma FCFS dalam program telah dilakukan dengan benar, seperti yang terkonfirmasi oleh perhitungan manual. Hasil yang konsisten antara kedua metode ini menunjukkan akurasi dalam proses penghitungan. Meskipun metode FCFS mudah diimplementasikan dan menawarkan keadilan dalam penanganan permintaan, perhatian perlu diberikan terhadap masalah efisiensi, terutama dalam konteks situasi nyata di mana waktu respons pengguna sangat penting. Pendekatan ini menunjukkan perlunya pemilihan algoritma penjadwalan yang tepat untuk mencapai keseimbangan antara keadilan dan efisiensi dalam sistem antrian.