Simulasi Scheduling Algorithm dengan Menggunakan Python

Nama Anggota:

Alvin Yuga Pramana	(0806022310004)
Javin Erasmus Clementino	(0806022310025)
Muh. Ryan Ardiansyah	(0806022310019)

5 November 2024

1. Pengertian

Di dalam sistem operasi, terdapat beberapa algoritma simulasi penjadwalan yaitu:

a. First Come First Serve

Merupakan metode penjadwalan yang mengeksekusi proses berdasarkan urutan kedatangan ke dalam antrian. Proses yang tiba terlebih dahulu akan dieksekusi terlebih dahulu sampai selesai sebelum proses lain dijalankan. Algoritma ini tergolong sederhana dan mudah untuk diimplementasikan, tetapi sering menimbulkan masalah antrian panjang *(bottleneck)* jika ada proses yang membutuhkan waktu eksekusi lama, sehingga proses lain harus menunggu hingga proses tersebut selesai.

b. Shortest Job First (Non Preemptive)

Merupakan proses yang memiliki waktu eksekusi terpendek terlebih dahulu. Jika ada beberapa proses yang siap dijalankan, algoritma akan memilih proses dengan waktu eksekusi paling singkat. Algoritma ini sangat baik untuk mengurangi waktu tunggu rata - rata *(average waiting time)* karena proses - proses yang pendek segera selesai. Namun, algoritma ini tidak fleksibel terhadap proses baru yang datang saat proses sedang berjalan *(non-preemptive)*, dan sering kali menyebabkan masalah *starvation* untuk proses yang lebih panjang.

c. Shortest Job First (Preemptive)

Algoritma ini juga biasa disebut sebagai *(Shortest Remaining Time First (SRTF))* yang dimana CPU dapat mengambil alih proses yang sedang berjalan jika proses baru tiba dengan waktu eksekusi lebih pendek. Perbedaannya dengan SJF *non-preemptive* adalah di SJF *non-preemptive* proses yang sedang berjalan tidak dapat dihentikan sampai selesai, sedangkan SRTF proses yang sedang berjalan dapat dihentikan jika ada proses dengan sisa waktu yang lebih pendek.

Algoritma ini lebih optimal dalam hal waktu respon dan waktu tunggu karena selalu menjalankan proses tercepat yang tersedia. Tetapi masih terdapat resiko *starvation* untuk proses yang lebih panjang karena bisa terus digantikan oleh proses - proses yang lebih singkat.

d. Longest Job First (preemptive)

Algoritma ini menjalankan proses yang memiliki waktu eksekusi paling lama terlebih dahulu dan dapat menginterupsi proses jika ada proses baru yang memiliki waktu eksekusi yang lebih lama. Algoritma ini jarang digunakan dalam praktik karena cenderung memperpanjang waktu tunggu untuk proses yang lebih singkat. Penggunaannya juga kurang efisien dan sering kali tidak sesuai dengan tujuan sistem operasi untuk memaksimalkan efisiensi dan kecepatan respons.

e. Round Robin (quantum time = 12)

Pada algoritma ini, setiap proses mendapatkan waktu eksekusi tertentu atau yang disebut *quantum time* (dengan 12 satuan waktu). Setelah *quantum time* habis, CPU akan beralih ke proses berikutnya dalam antrian, dan proses yang belum selesai akan kembali ke akhir antrian untuk dieksekusi kembali. Algoritma ini sangat ideal untuk sistem *time-sharing* karena semua proses mendapatkan waktu CPU yang adil dan merata dan menghasilkan waktu respon yang baik untuk semua proses.

2. Pengujian Algoritma pada Data

Berdasarkan beberapa data yang diberikan, kami mengambil lima contoh data untuk menjelaskan sistem algoritma dari *First Come First Serve*, *Shortest Job First (Non Preemptive)*, *Shortest Job First (Preemptive)*, *Longest Job First (preemptive)*, dan *Round Robin (quantum time = 12)*.

Berikut beberapa data yang kami gunakan;

PID	Arrival_Time	Burst_Time
P1	40	15
P2	11	15
P3	8	11
P4	19	15
P5	10	14

A. First Come First Serve

PID	Arrival Time	Burst Time	Start Time	Completion Time	Turnaround Time	Waiting Time
P3	8	11	8	19	11	0
P5	10	14	19	33	23	9
P2	11	15	33	48	37	22
P4	19	15	48	63	44	29
P1	40	15	63	78	38	23

Tabel 2.1 Process Data pada Algoritma First Come First Serve

Pada algoritma FCFS, proses dieksekusi dalam urutan kedatangan. Berikut adalah hasil *start time* dan *completion time* masing-masing proses:

1) P3: Tiba pada waktu 8, sehingga kita mulai pada waktu 8 dan selesai pada waktu 8+11=198 + 11 = 198+11=19.
2) P5: Tiba pada waktu 10, tetapi proses P3 belum selesai, jadi kita mulai pada waktu 19 dan selesai pada waktu 19+14=3319 + 14 = 3319+14=33.
3) P2: Tiba pada waktu 11, tetapi proses P5 belum selesai, jadi kita mulai pada waktu 33 dan selesai pada waktu 33+15=4833 + 15 = 4833+15=48.
4) P4: Tiba pada waktu 19, tetapi proses P2 belum selesai, jadi kita mulai pada waktu 48 dan selesai pada waktu 48+15=6348 + 15 = 6348+15=63.
5) P1: Tiba pada waktu 40, tetapi proses P4 belum selesai, jadi kita mulai pada waktu 63 dan selesai pada waktu 63+15=7863 + 15 = 7863+15=78.

B. Shortest Job First (Non Preemptive)

PID	Arrival Time	Burst Time	Start Time	Completion Time	Turnaround Time	Waiting Time
P3	8	11	8	19	11	0
P5	10	14	19	33	23	9
P2	11	15	33	48	37	22

PID	Arrival Time	Burst Time	Start Time	Completion Time	Turnaround Time	Waiting Time
P4	19	15	48	63	44	29
P1	40	15	63	78	38	23

Tabel 2.2 Process Data pada Algoritma Shortest Job First (Non Preemptive)

Setiap proses dijalankan sesuai urutan arrival time, dan setelah sebuah proses dimulai, ia berjalan hingga selesai sebelum proses lain dieksekusi. Berikut langkah algoritmanya:

1) Saat waktu 8: Proses P3 tiba dan mulai dieksekusi karena ini adalah proses pertama yang tiba.
2) Setelah P3 selesai (waktu 19): Proses berikutnya yang sudah tiba dan memiliki burst time terpendek adalah P5.
3) Setelah P5 selesai (waktu 33): Proses berikutnya dengan burst time terpendek yang sudah tiba adalah P2.
4) Setelah P2 selesai (waktu 48): Proses berikutnya adalah P4 yang sudah tiba.
5) Setelah P4 selesai (waktu 63): Proses terakhir, P1, dijalankan.

C. Shortest Job First (Preemptive)

PID	Arrival Time	Burst Time	Start Time	Completion Time	Turnaround Time	Waiting Time
P3	8	11	8	19	11	0
P5	10	14	19	33	23	9
P2	11	15	33	48	37	22
P4	19	15	48	63	44	29
P1	40	15	63	78	38	23

Tabel 2.3 Process Data pada Algoritma Shortest Job First (Preemptive)

Kita perlu menghitung urutan eksekusi setiap proses berdasarkan arrival time dan burst time terpendek yang tersedia pada saat itu. Berikut langkah algoritmanya:

1) Saat waktu 8: Proses P3 tiba pertama kali dan mulai dieksekusi.
2) Saat waktu 10: Proses P5 tiba, tetapi burst time P3 (11) lebih kecil daripada P5 (14), jadi P3 tetap berjalan.
3) Saat waktu 11: Proses P2 tiba, tetapi P3 tetap memiliki burst time terpendek.
4) Saat waktu 19: Proses P4 tiba, dan saat ini hanya sisa waktu 2 pada P3. Karena itu, kita selesaikan P3 terlebih dahulu.
5) Setelah P3 selesai (waktu 19): Proses berikutnya dengan burst time terpendek yang sudah tiba adalah P5.
6) Proses ini dilanjutkan, dan setiap proses akan berjalan berdasarkan burst time terpendek yang tersedia.

D. Longest Job First (preemptive)

PID	Arrival Time	Burst Time	Start Time	Completion Time	Turnaround Time	Waiting Time
P3	8	11	8	19	11	0
P5	10	14	19	33	23	9
P2	11	15	33	48	37	22
P4	19	15	48	63	44	29
P1	40	15	63	78	38	23

Tabel 2.4 Process Data pada Algoritma Longest Job First (preemptive)

Berikut adalah penjelasan terkait proses penjadwalan berdasarkan algoritma *Longest Job First (preemptive)*:

1) Waktu 8: Proses P3 tiba dan mulai berjalan karena ini adalah proses pertama yang tiba.
2) Waktu 10: Proses P5 tiba. Karena burst time P5 lebih panjang dari P3, P3 di-preempt, dan P5 mulai berjalan.
3) Waktu 11: Proses P2 tiba dengan burst time yang lebih panjang dari P5. Maka, P5 di-preempt, dan P2 mulai berjalan.
4) Waktu 19: Proses P4 tiba dengan burst time yang sama dengan P2. Namun, karena P2 sudah berjalan, maka P2 tetap berjalan hingga selesai.
5) Setelah P2 selesai (waktu 26): Proses berikutnya yang memiliki burst time terpanjang adalah P5.
6) Waktu 40: Proses P1 tiba dengan burst time 15. Karena sama dengan proses lain yang berjalan, kita memilih berdasarkan urutan arrival.

E. Round Robin (quantum time = 12)

PID	Arrival Time	Burst Time	Start Time	Completion Time	Turnaround Time	Waiting Time
P3	8	11	8	19	11	0
P5	10	14	19	69	59	45
P2	11	15	31	72	61	46
P4	19	15	43	75	56	41
P1	40	15	55	78	38	23

Tabel 2.5 Process Data pada Algoritma Round Robin (quantum time = 12)

Pada algoritma ini, setiap proses berjalan selama 12 satuan waktu atau hingga selesai, mana yang lebih dulu. Jika proses selesai sebelum 12 satuan waktu, ia tidak perlu kembali ke antrean. Jika tidak, ia kembali ke antrean untuk dieksekusi lagi setelah semua proses lain telah diberi giliran.

1) Waktu 8: Proses P3 tiba dan mulai dieksekusi. Karena burst time P3 adalah 11 (kurang dari 12), ia selesai pada waktu 19.

2) Waktu 19: Proses P5 adalah proses berikutnya yang siap, dimulai pada waktu 19 dan berjalan selama 12. Ia masih membutuhkan 2 satuan waktu setelah waktu 31.
3) Waktu 31: Proses P2 berikutnya, dimulai pada waktu 31 dan berjalan selama 12, dan masih membutuhkan 3 satuan waktu setelah waktu 43.
4) Waktu 43: Proses P4 adalah proses berikutnya, dimulai pada waktu 43 dan berjalan selama 12, masih membutuhkan 3 waktu setelah waktu 55.
5) Waktu 55: Proses P1 mulai berjalan pada waktu 55 dan berjalan selama 12, masih membutuhkan 3 satuan waktu setelah waktu 67.
6) Waktu 67: Proses P5 dilanjutkan dan selesai setelah 2 waktu pada 69.
7) Waktu 69: Proses P2 dilanjutkan dan selesai setelah 3 waktu pada 72.
8) Waktu 72: Proses P4 dilanjutkan dan selesai setelah 3 waktu pada 75.
9) Waktu 75: Proses P1 dilanjutkan dan selesai setelah 3 waktu pada 78.

Untuk melihat hasil eksekusi program yang lengkapnya dari setiap proses algoritma, dapat melihat pada file.xlsx (excel) yang dihasilkan oleh program saat menghasilkan output. File.xlsx akan muncul pada folder output yang telah disediakan.

3. Kesimpulan

Untuk menentukan algoritma penjadwalan yang paling baik dan buruk dalam mengelola data tersebut, kita perlu mempertimbangkan beberapa aspek utama dari algoritma penjadwalan, yaitu:

1. Waktu Tunggu (*Waiting Time*): Waktu yang dibutuhkan oleh proses sebelum mulai dieksekusi.
2. Waktu Putar Balik (*Turnaround Time*): Waktu total yang dibutuhkan dari kedatangan hingga penyelesaian proses.
3. Keberhasilan dalam Mengelola Proses: Kemampuan algoritma untuk menangani semua proses secara efisien berdasarkan waktu kedatangan dan durasi burst time-nya.

Sistem Algoritma	Waiting Time	Turnaround Time
First Come First Serve	582.55	595.13
Shortest Job First (Non Preemptive)	477.61	490.19
Shortest Job First (Preemptive)	476.13	488.71
Longest Job First (preemptive)	1172.99	1185.57
Round Robin (Quantum time = 12)	796.97	809.5500000000001

Tabel 3.1 Hasil waiting time dan turnaround time

Berdasarkan hasil waiting time dan turnaround time yang didapatkan dari sistem algoritma *First Come First Serve*, *Shortest Job First (Non Preemptive)*, *Shortest Job First (Preemptive)*, *Longest Job First (preemptive)*, dan *Round Robin (Quantum time = 12)*. Dapat dibuktikan bahwa algoritma yang terbaik adalah *Shortest Job First (Preemptive)* karena algoritma ini terus memeriksa setiap kali ada proses baru yang siap untuk berjalan, dan jika ada proses yang lebih pendek dari proses yang sedang berjalan, proses yang sedang berjalan akan di-preempt (dihentikan sementara) dan digantikan oleh proses yang lebih singkat.. Sedangkan, algoritma yang terburuk adalah *Longest Job First(Preemptive)* karena algoritma ini memilih proses yang memiliki waktu eksekusi terlama, sehingga proses-proses pendek yang tiba lebih awal harus menunggu lebih lama untuk dieksekusi. Ini menyebabkan waktu tunggu rata-rata (*waiting time*) menjadi lebih tinggi.