**NAMA : RAIHAN EKA SAPUTRA**

**NIM : 2009076001**

**PRODI : TEKNIK ELEKTRO**

**UTS SISTEM OPERASI**

1. Jika diketahui 6 antrian proses (A, B, C, D, E, F) Dengan waktu kedatangan secara bersamaan yaitu: 0. Lama eksekusi tiap-tiap antrian proses secara berurutan 1,3,7,5,5,3. Hitunglah Turn Arround Time (TA) dengan menggunakan teknik penjadwalan proses:
2. **First In First Out (F I F O)**

Diket :

* Waktu kedatangan secara bersamaan = 0
* Lama eksekusi tiap antrian secara berurutan adalah 1, 3, 7, 5, 5, dan 3.

**Menghitung Finish Time:**

Pada Teknik FIFO waktu selesai antrian proses akan di perbarui setelah nantinya antrian proses sebelumnya selesai. Untuk mengetahui finish time nya maka dilakukan proses berikut:

Antrian A:

- Finish TimeA = Waktu kedatangan A + Lama eksekusi A = 0 + 1 = 1

Antrian B:

- Finish TimeB = Finish TimeA + Lama eksekusi B = 1 + 3 = 4

Antrian C:

- Finish TimeC = Finish TimeB + Lama eksekusi C = 4 + 7 = 11

Antrian D:

- Finish TimeD = Finish TimeC + Lama eksekusi D = 11 + 5 = 16

Antrian E:

- Finish TimeE = Finish TimeD + Lama eksekusi E = 16 + 5 = 21

Antrian F:

- Finish TimeF = Finish TimeE + Lama eksekusi F = 21 + 3 = 24

**Menghitung Turnaround Time (TA) untuk setiap antrian proses:**

TA bisa dihitung dengan cara mengurangi waktu kedatangan antrian proses dari waktu selesai antrian proses tersebut.

Antrian A:

- (TA) A = Waktu selesai A - Waktu kedatangan A = 1 - 0 = 1

Antrian B:

- (TA) B = Waktu selesai B - Waktu kedatangan B = 4 - 0 = 4

Antrian C:

- (TA) C = Waktu selesai C - Waktu kedatangan C = 11 - 0 = 11

Antrian D:

- (TA) D = Waktu selesai D - Waktu kedatangan D = 16 - 0 = 16

Antrian E:

- (TA) E = Waktu selesai E - Waktu kedatangan E = 21 - 0 = 21

Antrian F:

- (TA) F = Waktu selesai F - Waktu kedatangan F = 24 - 0 = 24

Jadi, Kesimpualan yang di dapat Turnaround Time (TA) untuk setiap antrian proses adalah:

**A = 1, B = 4, C = 11, D = 16, E = 21, F = 24.**

**B. Shortest Job First (S J F)**

**Diketahui :**

* Waktu kedatangan secara bersamaan adalah = 0
* Lama eksekusi tiap antrian proses secara berurutan adalah 1, 3, 7, 5, 5, dan 3.

Langkah proses menggunakan metode Shortest Job First (S J F):

**Mengurutkan antrian proses berdasarkan lama eksekusi dengan ascending:**

Antrian A= 1

Antrian B= 3

Antrian F= 3

Antrian D= 5

Antrian E= 5

Antrian C= 7

**Hitung waktu selesainya setiap antrian proses:**

Antrian A:

- Waktu selesai A = Waktu kedatangan A + Lama eksekusi A = 0 + 1 = 1

Antrian B:

- Waktu selesai B = Waktu selesai A + Lama eksekusi B = 1 + 3 = 4

Antrian F:

- Waktu selesai F = Waktu selesai B + Lama eksekusi F = 4 + 3 = 7

Antrian D:

- Waktu selesai D = Waktu selesai F + Lama eksekusi D = 7 + 5 = 12

Antrian E:

- Waktu selesai E = Waktu selesai D + Lama eksekusi E = 12 + 5 = 17

Antrian C:

- Waktu selesai C = Waktu selesai E + Lama eksekusi C = 17 + 7 = 24

**Hitung Turnaround Time (TA) untuk setiap antrian proses:**

Antrian A:

- (TA) A = Waktu selesai A - Waktu kedatangan A = 1 - 0 = 1

Antrian B:

- (TA) B = Waktu selesai B - Waktu kedatangan B = 4 - 0 = 4

Antrian F:

- (TA) F = Waktu selesai F - Waktu kedatangan F = 7 - 0 = 7

Antrian D:

- (TA) D = Waktu selesai D - Waktu kedatangan D = 12 - 0 = 12

Antrian E:

- (TA) E = Waktu selesai E - Waktu kedatangan E = 17 - 0 = 17

Antrian C:

- (TA) C = Waktu selesai C - Waktu kedatangan C = 24 - 0 = 24

Jadi, Kesimpulan yang di dapat yaitu Turnaround Time (TA) untuk setiap antrian proses dengan menggunakan teknik Shortest Job First (SJF) sebagai berikut:

**A = 1, B = 4, F = 7, D = 12, E = 17, C = 24.**

**C. Round Robin jika diketahui Quantum = 2**

**Diketahui :**

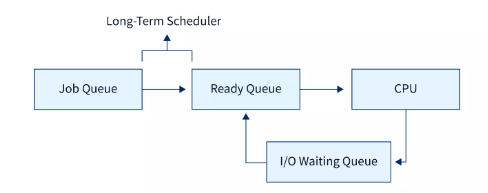
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | | B | C | D | | E | F | B | | C | D | E | F | | C | D | | E | C |
| 0 | | 1 | 3 | 5 | | 7 | 19 | 11 | | 12 | 14 | 16 | 18 | | 19 | 21 | | 22 | 23 24 |
|  | |  |  |  | |  |  |  | |  |  |  |  | |  |  | |  |  |
| Proses | | | | Waktu Tiba | | | | Lama Eksekusi | | Mulai Eksekusi | | | Selesai Eksekusi | | | TA | | |
| A | | | | 0 | | | | 1 | | 0 | | | 1 | | | 1 | | |
| B | | | | 0 | | | | 3 | | 1 | | | 12 | | | 12 | | |
| C | | | | 0 | | | | 7 | | 3 | | | 24 | | | 24 | | |
| D | | | | 0 | | | | 5 | | 5 | | | 22 | | | 22 | | |
| E | | | | 0 | | | | 5 | | 7 | | | 23 | | | 23 | | |
| F | | | | 0 | | | | 3 | | 9 | | | 19 | | | 19 | | |

Jadi, Kesimpulan yang di dapat yaitu Turnaround Time (TA) untuk setiap antrian proses dengan menggunakan teknik **Round Robin** sebagai berikut:

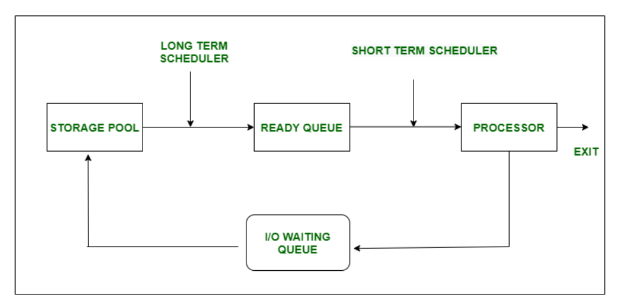
**A = 1, B = 12, C = 24, D = 22, E = 23, F = 19.**

1. Dalam Penjadwalan proses terdapat tiga macam tipe penjadwalan, sebutkan dan jelaskan disertai gambar!

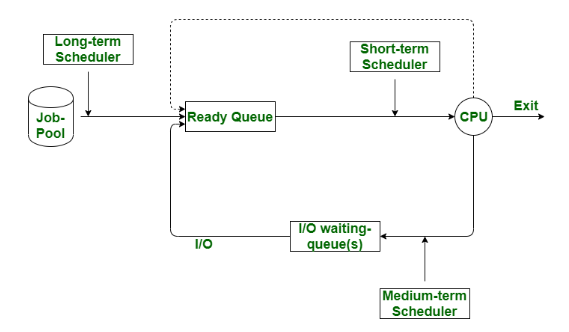
* Penjadwalan jangka panjang bertanggung jawab untuk memilih proses dari antrian pekerjaan dan memuatnya ke dalam memori utama. Tujuan penjadwalan jangka panjang adalah nantinya untuk menjaga keseimbangan antara jumlah proses dalam memori dan jumlah memori yang tersedia.



* Penjadwalan jangka pendek bertanggung jawab untuk memilih proses selanjutnya untuk dijalankan pada CPU. Tujuan penjadwalan jangka pendek adalah untuk memaksimalkan penggunaan CPU dan meminimalkan waktu respons proses.



* Penjadwalan jangka menengah bertanggung jawab untuk menukar proses antara memori utama dan penyimpanan sekunder. Tujuan penjadwalan jangka menengah adalah untuk meningkatkan kinerja sistem dengan memindahkan proses yang tidak aktif menggunakan CPU ke penyimpanan sekunder.



1. Sumber daya apa yang digunakan saat thread dibuat? Bagaimana mereka berbeda dari yang digunakan ketika suatu proses dibuat?

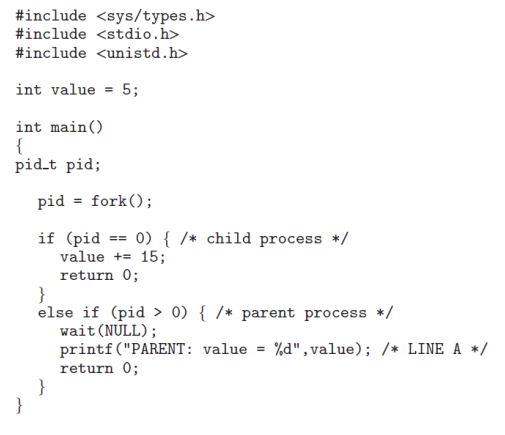
**Ketika sebuah thread dibuat, sumber daya yang digunakan meliputi:**

* 1. Stack: Setiap thread memiliki stack sendiri yang digunakan untuk menyimpan variabel lokal, parameter, dan pemanggilan fungsi. Stack thread bersifat privat dan berbeda dari stack proses utama.
  2. Register: Thread memiliki register sendiri, seperti register program counter (PC) yang menyimpan alamat instruksi yang sedang dieksekusi, serta register lainnya seperti register umum (misalnya, register EAX, EBX, ECX, dll.) yang digunakan untuk operasi komputasi.
  3. Variabel Thread-Local: Setiap thread memiliki variabel-variabel yang hanya dapat diakses oleh thread tersebut. Variabel thread-local ini berguna untuk menyimpan data yang bersifat lokal atau bersifat thread-spesifik.
  4. Data Struktur Kontrol Thread: Setiap thread memerlukan data struktur kontrol (thread control block atau TCB) yang menyimpan informasi tentang status thread, seperti identitas thread, prioritas, state, dan lain-lain.
  5. Waktu Prosesor: Thread membutuhkan alokasi waktu prosesor (CPU) untuk dieksekusi. CPU dialokasikan secara bergantian antara thread-thread yang aktif dalam sistem.

**Perbedaan dengan pembuatan proses adalah sebagai berikut:**

* 1. Ruang Alamat: Setiap proses memiliki ruang alamat virtual yang terpisah. Proses-proses berjalan dalam lingkungan memori yang terisolasi, sehingga masing-masing proses tidak dapat mengakses atau memodifikasi memori dari proses lain secara langsung.
  2. Sumber Daya Sistem: Saat membuat proses, sumber daya sistem yang signifikan dibutuhkan, seperti ruang memori, deskriptor file, dan lain-lain. Setiap proses memiliki sumber daya yang independen, yang perlu dialokasikan dan dikelola oleh sistem operasi.
  3. Perlindungan: Proses memiliki perlindungan yang lebih kuat terhadap interferensi antara satu proses dengan proses lainnya. Mekanisme perlindungan, seperti mekanisme hak akses, digunakan untuk mencegah akses tidak sah atau perubahan terhadap sumber daya proses lain.
  4. Isolasi: Proses bekerja dalam lingkungan yang terisolasi, sehingga kesalahan atau kegagalan pada satu proses tidak akan mempengaruhi proses-proses lainnya. Proses-proses berjalan secara independen dan dapat berinteraksi melalui mekanisme komunikasi yang ditentukan.

1. Output apa yang akan ditampilkan pada LINE A? Jelaskan!



Output yang akan ditampilkan adalah "PARENT: value = 5".

Alasannya output tersebut adalah karena **fork()** akan menghasilkan dua proses terpisah: proses parent dan proses child. Setelah pemanggilan **fork()**, proses child akan memiliki salinan variabel **value** yang bernilai 5. Namun, karena setiap proses memiliki ruang memori yang terpisah, perubahan nilai **value** dalam proses child tidak akan mempengaruhi nilai **value** dalam proses parent. Ketika proses child selesai dieksekusi dan proses parent menunggu dengan menggunakan **wait(NULL)**, proses parent akan melanjutkan eksekusinya. nilai **value** dalam proses parent tetap 5, karena perubahan yang dilakukan dalam proses child tidak mempengaruhinya. Sehingga, setiap iterasi program akan mencetak "PARENT: value = 5" pada ("LINE A").