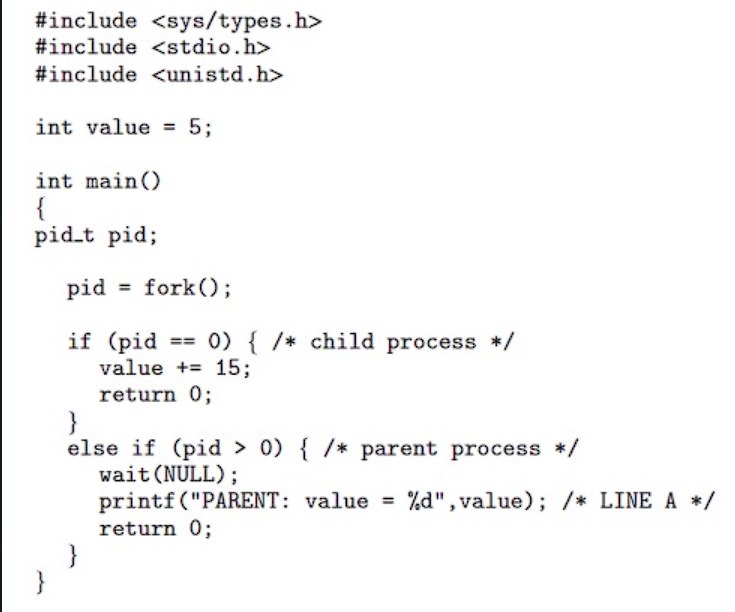
Nama : M Fathur Rahman   
NIM : 2009076039  
Matakuliah : Sistem Operasi

Prodi : Teknik Elektro   
  
Ujian Tengah Semester 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:
   1. First In First Out (F I F O)
   2. Shortest Job First (S J F)
   3. Round Robin jika diketahui Quantum = 2
2. Dalam Penjadwalan proses terdapat tiga macam tipe penjadwalan, sebutkan dan jelaskan disertai gambar!
3. Sumber daya apa yang digunakan saat thread dibuat? Bagaimana mereka berbeda dari yang digunakan ketika suatu proses dibuat?
4. Output apa yang akan ditampilkan pada LINE A? Jelaskan!



Penyelesaian:

1. Diketahui :

* Antrian : A, B, C, D, E, F
* Waktu tiba : 0
* Lama Eksekusi : 1, 3, 7, 5, 5, 3

Ditanya : Turn Around (TA) dengan teknik FIFO, Shortest Job First (SJF), & Round Robin (Quantum = 2)

Penyelesaian :

* FIFO
* Yang pertama kita tentukan TA menggunakan rumus sehingga

∑TA = 1 + 4 + 11 + 16 + 21 + 24 = 77 milidetik

Rata – rata TA = 77/6 = 12,834 milidetik

* Kedua, kita masukkan informasi-informasi yang diketahui ke dalam tabel seperti di bawah ini:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nama Proses | Waktu Tiba | Lama Eksekusi | Mulai Eksekusi | Selesai Eksekusi | TA (Turn Around Time) |
| A | 0 | 1 | 0 | 1 | 1 |
| B | 0 | 3 | 1 | 4 | 4 |
| C | 0 | 7 | 4 | 11 | 11 |
| D | 0 | 5 | 11 | 16 | 16 |
| E | 0 | 5 | 16 | 21 | 21 |
| F | 0 | 3 | 21 | 24 | 24 |

* Ketiga, menampilkan ke dalam gantt chart

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| 0 | 1 | 4 | 11 | 16 | 21 24 |

* Shortest Job First
* Yang pertama menentukan TA dengan rumus sehingga didapat:

∑TA = 1 + 4 + 7 + 12 + 17 + 24 = 65 milidetik

Rata – rata TA = 65/6 = 10,84 milidetik

* Menuliskan informasi mulai eksekusi, selesai eksekusi, dan TA ke dalam tabel seperti di bawah ini:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nama Proses | Waktu Tiba | Lama Eksekusi | Mulai Eksekusi | Selesai Eksekusi | TA (Turn Around Time) |
| A | 0 | 1 | 0 | 1 | 1 |
| B | 0 | 3 | 1 | 14 | 14 |
| C | 0 | 7 | 4 | 26 | 26 |
| D | 0 | 5 | 7 | 24 | 24 |
| E | 0 | 5 | 12 | 25 | 25 |
| F | 0 | 3 | 17 | 21 | 21 |

* Bentuk Gantt Chartnya yaitu:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| 0 | 1 | 4 | 7 | 12 | 17 24 |

* Round Robin Quantum 2
* Yang Pertama menentukan TA sehingga didapat:

∑TA = 1 + 14 + 26 + 24 + 25 + 21 = 111 milidetik

Rata – rata TA = 111/6 = 18,5 milidetik

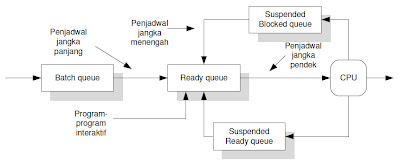
* Menuliskan informasi mulai eksekusi, selesai eksekusi, dan TA ke dalam tabel seperti di bawah ini:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nama Proses | Waktu Tiba | Lama Eksekusi | Mulai Eksekusi | Selesai Eksekusi | TA (Turn Around Time) |
| A | 0 | 1 | 0 | 1 | 1 |
| B | 0 | 3 | 1 | 4 | 4 |
| C | 0 | 7 | 3 | 7 |  |
| D | 0 | 5 | 7 | 12 | 12 |
| E | 0 | 5 | 9 | 17 | 17 |
| F | 0 | 3 | 11 | 24 | 24 |

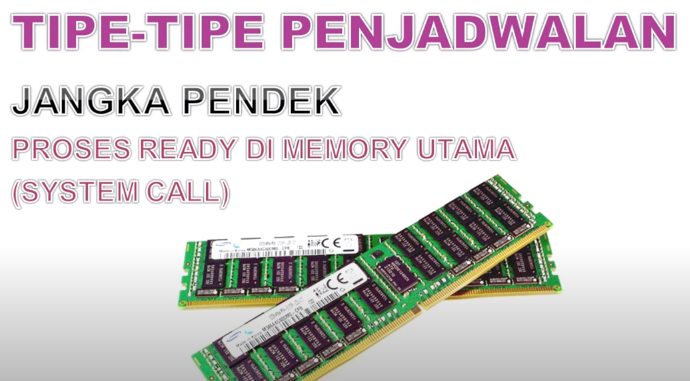
* Bentuk Gantt Chartnya :

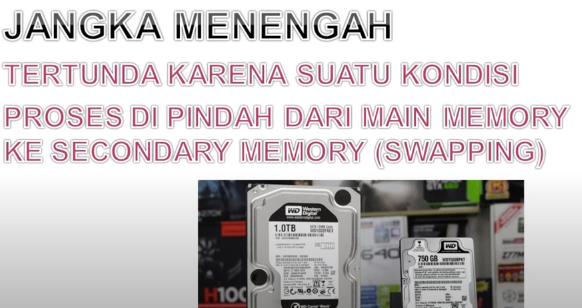
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | B | C | D | E | F | C | D | E | C |
| 0 | 1 | 3 | 7 | 9 | 11 | 13 | 14 | 16 | 18 | 20 | 21 | 23 | 24 | 25 26 |

2. Berikut ini adalah tiga macam tipe penjadwalan, yaitu:



1. **Penjadwalan jangka pendek (short-termscheduller)** adalah penjadwal yang bertugas untuk menjadwalkan alokasi processor di antara proses-proses yang ready di memory utama. Proses-proses ini biasanya berupa perintah-perintah yang sudah disediakan oleh sistem operasi atau yang kita sebut dengan system call. Sasaran utama dari penjadwal ini adalah memaksimalkan kinerja untuk memenuhi suatu kumpulan kriteria yang diharapkan dari perintah2 yang diberikan. Karena proses-proses tersebut diantrikan/dijadwalkan pada memory utama, semakin besar kapasitas memory maka semakin banyak pula proses2 yang dapat diantrikan/dijadwalkan berada di memory utama tersebut, sehingga semakin banyak dan semakin cepat pula perintah2 yang dapat dieksekusi dan diselesaikan. Jadi kapasitas memory sangat berpengaruh pada kecepatan dan kinerja komputer yang digunakan.



1. **Penjadwalan jangka menengah (medium termscheduller)** adalah pada saat eksekusi selama waktu tertentu, proses yang dijadwalkan kemungkinan akan mengalami penundaan karena suatu kondisi, nah proses-proses yang tertunda ini tidak akan tereksekusi sehingga kondisi-kondisi yang menyebabkan tertundanya dihilangkan. Namun, jika proses tersebut masih berada di main memory maka akan menjadi mubazir karena prosesnya tidak dieksekusi namun menggunakan ruang yang ada. Agar ruang yang ada di main memory tersebut dapat dimanfaatkan oleh proses-proses yang lain, maka proses yang tertunda ini dipindahkan ke secondary memory atau disebut swapping. Nah penjadwalan jangka menengah ini yang menangani proses-proses swapping ini. Proses2 yang mempunyai kepentingan yang kecil pada saat itu dianggap sebagai proses yang tertunda, tetapi begitu kondisi yang membuatnya tertunda sudah hilang atau misalnya proses-proses yang memiliki kepentingan yang sudah selesai dieksekusi semua, maka proses-proses yang di swab tadi akan dimasukkan atau dikembalikan ke main memory. Tipe penjadwalan jangka menengah inilah yang mengendalikan transisi dari status suspend kembali menjadi status ready atau yang disebut dengan proses-proses swapping, baik itu swap dari main memory ke secondary memory atau sebaliknya dari secondary memory ke main memory untuk segera di eksekusi.
2. **Penjadwalan jangka panjang (long-termscheduller)** adalah penjadwalan yang diterapkan ketika adanya penambahan proses-proses baru ke dalam kelompok proses yang akan dieksekusi oleh komputer di mana proses-proses yang baru tersebut masih berlokasi di secondary memory. Proses-proses yang akan ditambahkan inilah yang disebut dengan batch atau kumpulan-kumpulan dari proses yang nantinya akan dieksekusi. Pada penjadwalan jangka panjang ini apabila semakin banyak proses yang diciptakan maka kualitas tiap layanan untuk setiap proses akan berkurang, karena itu untuk memastikan pelayanan dari tiap prosesnya tidak berkurang maka perlu diperhatikan kapasitas main memory yang dimiliki oleh komputer kita. Untuk memaksimalkan kinerja komputer ada baiknya kita mengaplikasikan atau menggunakan memory dengan kapasitas yang besar agar setiap proses dapat dilayani sesuai dengan kebutuhan-kebutuhannya tanpa mengurangi kualitas dari pelayanan2 yang dibutuhkan oleh proses-proses yang sedang berjalan. Jika kapasitas memory yang terdapat pada komputer ataupun gadget kita tidak memadai maka ini akan menyebabkan komputer atau gadget kita bekerja tidak maksimal, sehingga terlihat jeda pada setiap proses yang diberikan. Ini berkaitan dengan respons time. Yaitu misalnya kita memberikan perintah pada gadget atau komputer kita namun karena kapasitas memory untuk mengeksekusi proses-proses yang ada tidak cukup maka terjadi respon time yang panjang atau biasa disebut ngelag. Jadi pastikan komputer atau gadget kita memiliki kapasitas memory yang cukup untuk mengeksekusi proses-proses yang akan kita jalankan melalui program/aplikasi/software yang akan kita gunakan.



1. Thread, atau kadang-kadang disebut proses ringan (lightweight), adalah unit dasar dari utilisasi CPU. Di dalamnya terdapat ID thread, program counter, register, dan stack. Dan saling berbagi dengan thread lain dalam proses yang sama. Jadi Sumber daya apa yang digunakan saat thread dibuat? Bagaimana mereka berbeda dari yang digunakan ketika suatu proses dibuat?

* Thread adalah bagian dari proses dan disebut proses ringan.
* Sebuah proses harus memiliki setidaknya satu Thread. Oleh karena itu, membuat proses berarti membuat proses dan membuat Thread.
* Karena Thread adalah bagian dari proses, tidak ada sumber daya tambahan yang digunakan saat Thread dibuat, melainkan berbagi ruang memori dari proses tempat utas khusus ini dibuat.
* Pembuatan Thread itu murah. Oleh karena itu, ini disebut proses ringan.
* Sedangkan pembuatan proses cukup mahal karena harus menyiapkan ruang memori virtual yang benar-benar baru untuk proses dengan ruang alamatnya sendiri.
* Pembuatan proses membutuhkan banyak waktu CPU dan disebut proses kelas berat.
* Oleh karena itu, tidak ada sumber daya yang digunakan saat Thread dibuat sedangkan ruang memori baru dibuat saat membuat proses.

4. Output yang akan ditampilkan Line A ialah teks PARENT : value = 5.

Alasan kenapa line a hanya menampilkan value bernilai 5, itu disebabkan adanya function fork().

fork()adalah bagaimana Anda membuat proses baru di Unix. Saat Anda menelepon fork, Anda membuat salinan proses Anda sendiri yang memiliki ruang alamatnya sendiri. Hal ini memungkinkan banyak tugas berjalan secara independen satu sama lain seolah-olah masing-masing memiliki memori penuh mesin untuk diri mereka sendiri. fork() adalah cara Unix membuat proses baru. Pada saat Anda memanggil fork(), proses Anda dikloning, dan dua proses berbeda melanjutkan eksekusi dari sana. Salah satunya, child, akan memiliki fork() return 0. Yang lainnya, parent, akan memiliki fork() mengembalikan PID (ID proses) anak.

Karena pada if line a berisikan fork bernilai positif maka dia termasuk parent proses. Jadi kenapa dilakukan penduplikasinan berguna untuk pada saat child proses mengalami error, maka tidak mengganggu parent proses. Tapi jika parent proses yang mengalami gangguan atau error, maka child proses akan terikut kena dampaknya.