Nama : Zia Meidita Felia

Nim : 2009076050

UTS Sistem Operasi

Soal

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:

a. First In First Out (F I F O)

b. Shortest Job First (S J F)

c. Round Robin jika diketahui Quantum = 2

1. Dalam Penjadwalan proses terdapat tiga macam tipe penjadwalan, sebutkan dan jelaskan disertai gambar!
2. Sumber daya apa yang digunakan saat thread dibuat? Bagaimana mereka berbeda dari yang digunakan ketika suatu proses dibuat?
3. Output apa yang akan ditampilkan pada LINE A? Jelaskan!



Jawab

1. a. FIFO

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nama Proses | Waktu Tiba | Lama Eksekusi | Mulai Eksekusi | Waktu Tunggu | Selesai Eksekusi | TA |
| A | 0 | 1 | 0 | 0 | 1 | 1 |
| B | 0 | 3 | 1 | 1 | 4 | 4 |
| C | 0 | 7 | 4 | 4 | 11 | 11 |
| D | 0 | 5 | 11 | 11 | 16 | 16 |
| E | 0 | 5 | 16 | 16 | 19 | 21 |
| F | 0 | 3 | 19 | 19 | 22 | 22 |

Untuk menghitung waktu tunggu gunakan rumus “ WT = mulai eksekusi – waktu tiba”

Untuk menghitung nilai TA gunakan rumus “TA = waktu tunggu + lama eksekusi”

Untuk menghitung nilai rerata TA gunakan rumus “”

Jadi, untuk nilai milidetik dan nilai Rerata TA = 12,5 milidetik

b. SJF

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nama Proses | Waktu Tiba | Lama Eksekusi | Mulai Eksekusi | Waktu Tunggu | Selesai Eksekusi | TA |
| A | 0 | 1 | 0 | 0 | 1 | 1 |
| B | 0 | 3 | 1 | 1 | 4 | 4 |
| C | 0 | 3 | 4 | 4 | 7 | 7 |
| D | 0 | 5 | 7 | 7 | 12 | 12 |
| E | 0 | 5 | 12 | 12 | 17 | 17 |
| F | 0 | 7 | 17 | 17 | 24 | 24 |

Untuk menghitung waktu tunggu gunakan rumus “ WT = mulai eksekusi – waktu tiba”

Untuk menghitung nilai gunakan rumus “ = waktu tunggu + lama eksekusi”

Untuk menghitung nilai rerata TA gunakan rumus “”

Jadi, untuk nilai milidetik dan nilai Rerata TA = 10,833 milidetik

c. Round Robin dengan Q = 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nama Proses | Waktu Tiba | Lama Eksekusi | Mulai Eksekusi | Waktu Tunggu | Selesai Eksekusi | TA |
| A | 0 | 1 | 0 | 0 | 1 | 1 |
| B | 0 | 3 | 1 | 1 | 12 | 12 |
| C | 0 | 7 | 3 | 3 | 24 | 24 |
| D | 0 | 5 | 5 | 5 | 22 | 22 |
| E | 0 | 5 | 7 | 7 | 23 | 23 |
| F | 0 | 3 | 9 | 9 | 19 | 19 |

Untuk menghitung waktu tunggu gunakan rumus “ WT = mulai eksekusi – waktu tiba”

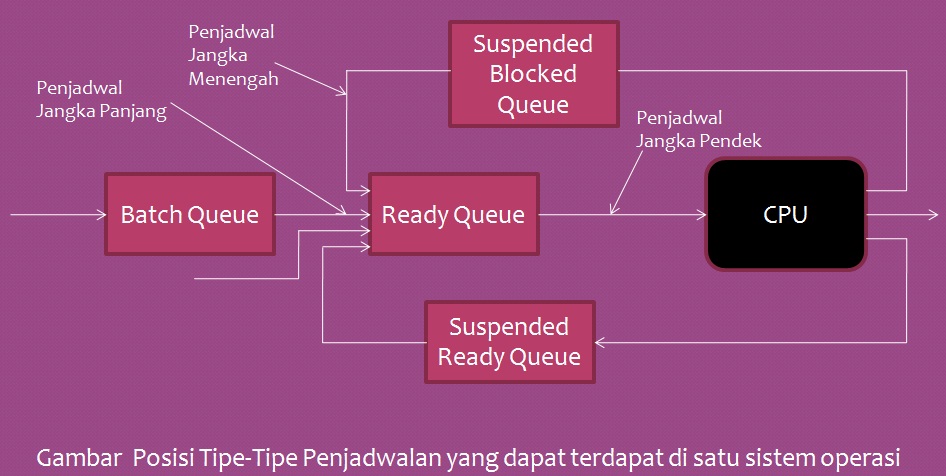
Untuk menghitung nilai gunakan rumus “ = waktu tunggu + lama eksekusi”

Untuk menghitung nilai rerata TA gunakan rumus “”

Jadi, untuk nilai milidetik dan nilai Rerata TA = 16,84 milidetik

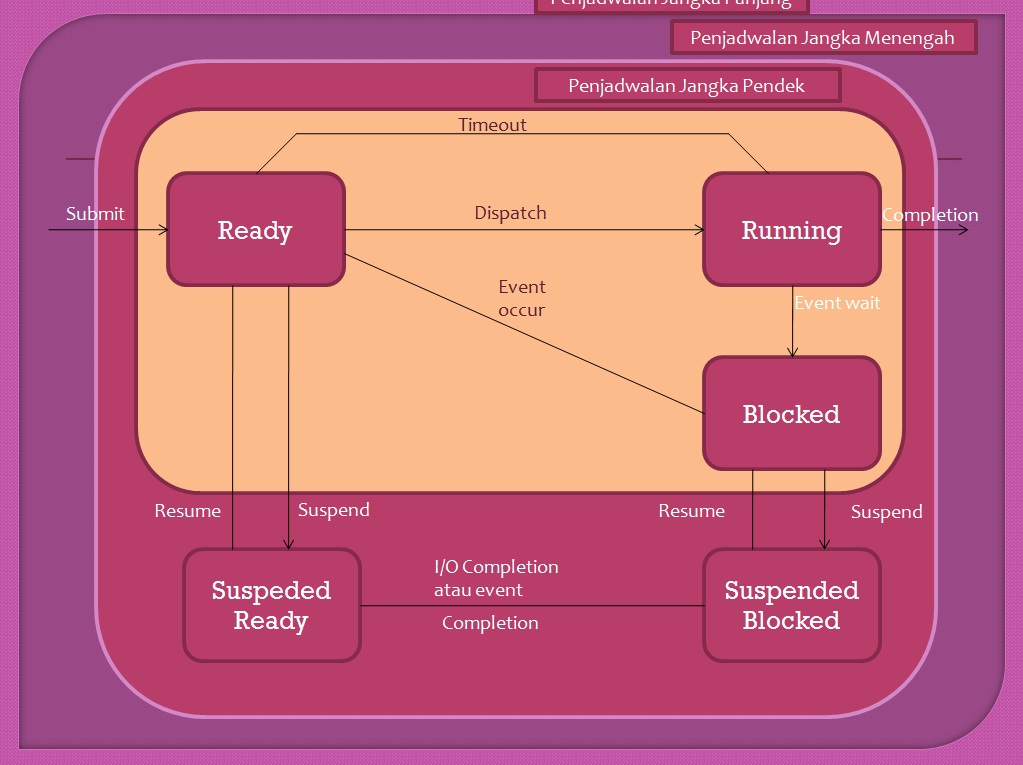
1. Terdapat tiga tipe penjadwalan secara bersama-sama pada sistem operasi, yaitu:
2. Penjadwalan jangka pendek (short-term scheduller)

Penjadwalan ini bertugas untuk menjadwalakan alokasi pemroses diantara proses-proses yang ready pada memori utama. Tujuan utama penjadwalan ini berguna untuk memaksimumkan kinerja agar memenuhi satu kumpulan kriteria yang diharapkan. Penjadwalan ini dijalankan setiap terjadinya pengalihan proses untuk memilih proses berikutnya yang harus dijalankan.



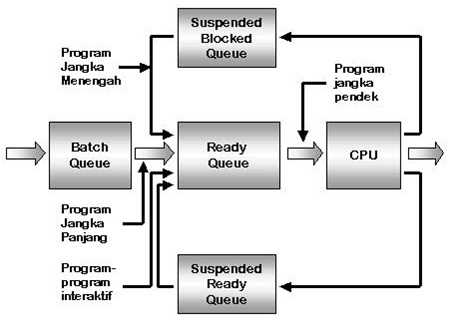
1. Penjadwalan jangka menengah (medium-term scheduller)

Setelah eksekusi selama suatu aktu, proses mungkin akan ditunda karena membuat permintaan layanan input output atau memanggil suatu sistem call. Agar ruang memori bermanfaat, maka proses akan dipindahkan dari memori utama ke memori sekunder agar tersedia ruang yang cukup untuk proses-proses lain. Aktivitas pemindahan proses yang ditunda dari memori utama ke memori sekunder disebut dengan swapping. Penjadwalan ini berguna untuk menangani proses-proses swapping dan mengendalikan transisi dari suspended-to-ready proses-proses swapping. Begitu penyebab tertunda hilang maka proses dimasukkan kembali ke memori utama.



1. Penjadwalan jangka panjang (long-term scheduller)

Penjadwalan jangka panjang bekerja terhadap antrian batch dan memilih batch berikutnya uang harus di eksekusi. Batch biasanya merupakan proses-proses dengan penggunaan sumber daya yang intensif (yaitu waktu pemroses, memori, perangkat masukan atau keluaran), program-program ini berprioritas rendah, digunakan sebagai pengisi (agar pemroses sibuk) selama periode aktifitas job-job interaktif rendah. Sasaran utama penjadwalan jangka panjang adalah memberi keseimbangan job-job campuran. Terdapat dua strategi penjadwalan, yaitu penjadwalan nonpreemptive (run-to-completion) dan penjadwalan preemptive. Pada penjadwalan nonpreemtive, begitu proses diberi jatah waktu pemroses maka pemroses tidak dapat diambil alih oleh proses lain sampai proses itu selesai. Pada strategi penjadwalan ini terdapat algoritma yang diterapkan diantaranya yaitu: FIFO (First In First Out), SJF (Shortest Job First), HRN (Highest-Ratio Next) dan MFQ (Multiple Feedback Queues). Pada penjadwalan preemptive, saat proses diberi jatah waktu memroses maka emroses dapat diambil alih proses lain sehingga proses disela sebelum selesai dan harus dilanjutkan menunggu jatah waktu pemroses tiba kembali ke proses itu. Pada strategi penjadwalan ini terdapat algoritma yang diterapkan diantaranya yaitu: RR (Round Robin), SRF (Shortest-Remaining-First), PS (Priority Schedulling) dan GS (Guaranteed Schedulling).



1. Pembuatan thread biasanya menggunakan sumber daya yang lebih sedikit dibanding pembuatan proses, dikarenakan thread lebih kecil daripada suatu proses. Untuk membuat suatu proses membutuhkan alokasi blok kontrol proses struktur data yang lebih besar. Alokasi ini mencakup peta memori, daftar file yang sudah terbuka dan juga variabel lingkungan. Untuk mengalokasikan dan mengelola peta memori biasanya merupakan sebuah aktivitas yang paling banyak memakan waktu. Untuk menciptakan user atau kernel thread biasanya melibatkan pengalokasian struktur data kecil untuk menampung set register, stack dan prioritas.
2. Output yang ditampilkan pada Line A adalah hasil value yang telah diinputkan. Pada Line A, program akan menampilkan nilai value setelah proses child selesai dieksekusi dengan menggunakan perintah ‘printf’. Kemudian, jika ‘fork()’ menghasilkan nilai positif maka nilai value akan berada didalam proses parent. Setelah berada di dalam proses parent, kita akan menunggu hingga proses child selesai di eksekusi. Setelah itu, nilai ‘value’ akan ditampilkan dengan menggunakan ‘printf’ di Line A.

