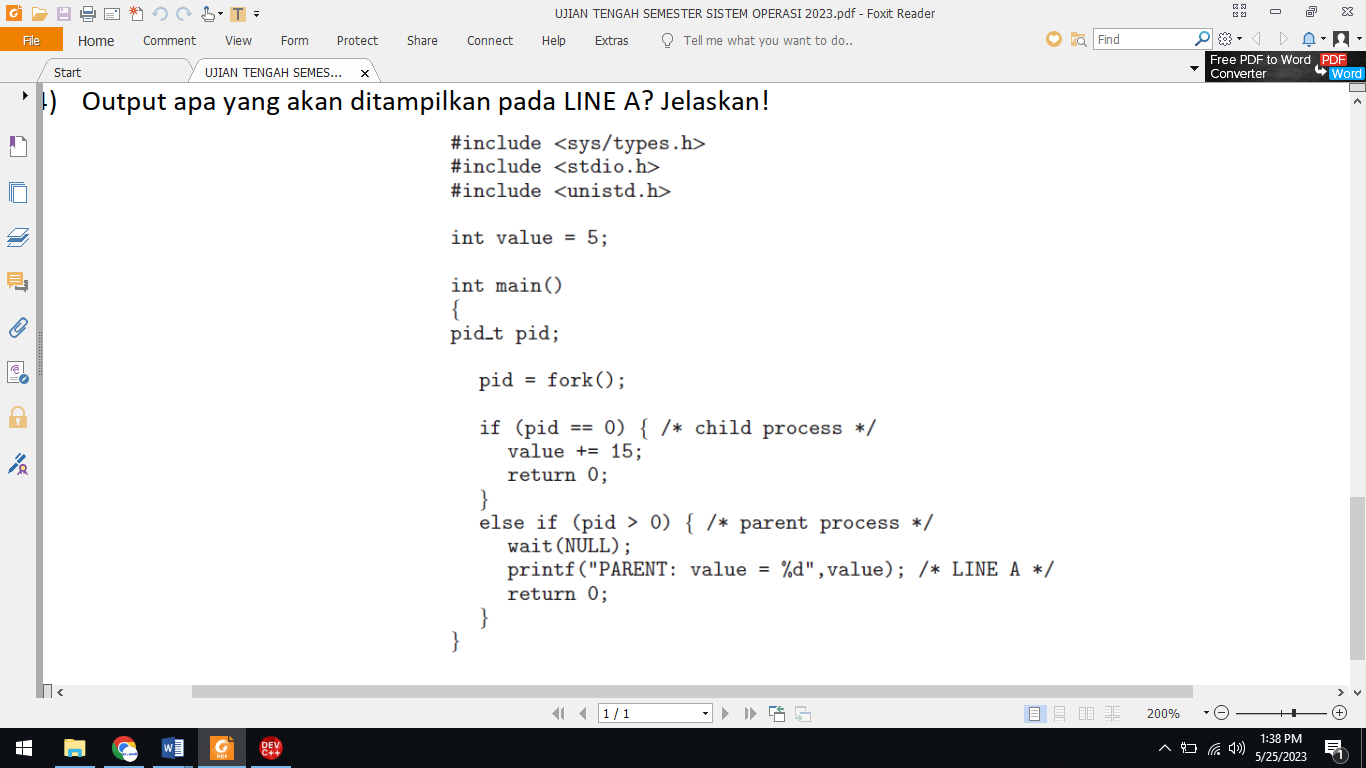
Nama : Andi Rachman Zikry

NIM : 2009076013

Ket : Ujian Tengah Semester\_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 Trun Arround Time (TA) dengan menggunakan teknik penjadwalan proses:
   1. *Frist In Frist Out* (FIFO)
   2. *Shortest Job first* (SJF)
   3. *Round Robin* jika diketahui Quantum = 2
2. Dalam penjadwalan proses terdapat tiga macam 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 !



Jawab :

1. Proses penjadwalan
   1. **Frist In Frist Out (FIFO)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nama | Waktu Tiba | Lama | Mulai | Selesai | Turn Arround |
| 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 |

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

Rata-Rata TA = 77 : 6 = 12,83 (milidetik)

**(Gannt Chart)**

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

* 1. **Shorten Job First (SJF)**

*(Proses Lama Ekseskusi diurutkan dari yang terkecil hingga yang terbesar)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nama | Waktu Tiba | Lama | Mulai | Selesai | Turn Arround |
| A | 0 | 1 | 0 | 1 | 1 |
| B | 0 | 3 | 1 | 4 | 4 |
| C | 0 | 3 | 4 | 7 | 7 |
| D | 0 | 5 | 7 | 12 | 12 |
| E | 0 | 5 | 12 | 17 | 17 |
| F | 0 | 7 | 17 | 24 | 24 |

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

Rata-Rata TA = 65 : 6 = 10,83 (milidetik)

**(Gannt Chart)**

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

* 1. **Round Robin**

**(Gannt Chart)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | B | C | D | E | F | C | D | E | C |
| 0 1 | 3 | 5 | 7 | 9 | 11 | 12 | 14 | 16 | 18 | 19 | 21 | 22 | 23 | 24 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nama | Waktu Tiba | Lama | Mulai | Selesai | Turn Arround |
| 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 |

∑TA = 1 + 12 + 24 + 22 + 23 + 19 = 101 (milidetik)

Rata-Rata TA = 101 : 6 = 16.83 (milidetik)

1. Terdapat tiga penjadwalan dalam sistem operasi yaitu

**Penjadwalan Jangka Pendek (Short Term Scheduller)**

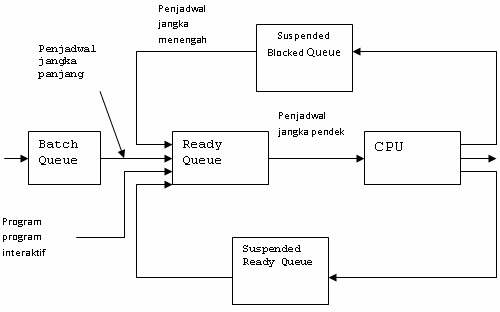
Tugasnya adalah merencanakan pengalokasian pemroses di antara proses yang siap dijalankan di memori utama. Setiap kali terjadi perpindahan proses, penjadwal akan memilih proses berikutnya yang akan dijalankan.

**Penjadwalan Jangka Menengah ( Medium Term Scheduller)**

Proses dipindahkan dari memori utama ke memori sekunder agar ada ruang untuk proses lain. Kapasitas memori utama terbatas untuk sejumlah proses aktif. Ketika proses tidak aktif, penjadwal akan menunda pemindahan dari memori utama ke memori sekunder, yang dikenal sebagai swapping.

**Penjadwalan Jangka Panjang (Long Term Scheduller)**

Penjadwal ini bekerja dengan antrian batch dan memilih batch berikutnya yang akan dieksekusi. Batch biasanya terdiri dari proses yang membutuhkan sumber daya intensif seperti pemrosesan waktu, memori, dan masukan/keluaran. Program-program ini memiliki prioritas yang rendah dan digunakan saat tidak ada aktivitas interaktif yang tinggi.



Gambar penjadwalan proses pada sistem operasi

1. Karena nilai thread lebih kecil dari proses, pada umumnya pembuatan thread menggunakan *resource* yang sedikit ketimbang pada saat pembuatan proses. Untuk membuat suatu proses, kita akan membutuhkan alokasi blok kontrol proses (PCB), dan struktur data yang lebih besar. PCB sendiri terdiri dari peta memori, kumpulan daftar file terbuka, dan juga variabel lingkungan. Kedua proses ini yaitu mengelola dan mengalokasikan peta memori merupakan aktivitas yang banyak memakan waktu. Menciptakan user atau kernel *threads* melibatkan pengalokasian struktur data kecil, yang digunakan untuk menampung set *register, stack* dan prioritas.
2. Outputnya akan berupa teks “PARENT : Value = 5

Segera setelah kita memanggil fork(), baik *child* dan *parent* berbagi data dan kode yang sama, dan ruang alamat *parent* disalin ke ruang alamat *child*. Ketika salah satu dari *child* atau *parent* mencoba untuk mengubah data (kode ditempatkan di *bagian read-only* dari file sehingga tidak dapat diubah), sebuah interupsi dibuat dan halaman ditulis untuk proses *child*, sebelum *parent* atau *child* dapat melakukan perubahan apa pun. Jadi, pada akhirnya, *parent* dan *child* memiliki variabel nilai mereka sendiri. Variabel yang dimiliki oleh *child* akan memiliki nilai 20 dan yang dimiliki oleh *parent* akan memiliki nilai 5.