**TUGAS**

**Penjadwalan Proses**

**dan**

**Cara Menghitung Penjadwalan dengan Algoritma**

Disusun untuk memenuhi tugas mata kuliah Sistem Operasi

Dosen:

Sy. Yuliani, S.Kom.,M.T.



Disusun Oleh :

Arif Krismunandar (116705033)

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SUNAN GUNUNG DJATI**

**BANDUNG**

**2017**

*FCFS (First Come First Served)*

Algoritma ini merupakan algoritma penjadwalan yang paling sederhana yang digunakan CPU. Dengan menggunakan algoritma ini setiap proses yang berada pada status *ready* dimasukkan kedalam *FIFO queue* atau antrian dengan prinsip *first in first out*, sesuai dengan waktu kedatangannya. Proses yang tiba terlebih dahulu yang akan dieksekusi.

Contoh

Ada tiga buah proses yang datang secara bersamaan yaitu pada 0 ms, P1 memiliki burst time 24 ms, P2 memiliki burst time 3 ms, dan P3 memiliki burst time 3 ms. Hitunglah *waiting time* rata-rata dan*turnaround time*( *burst time + waiting time*) dari ketiga proses tersebut dengan menggunakan algoritma FCFS. *Waiting time* untuk P1 adalah 0 ms (P1 tidak perlu menunggu), sedangkan untuk P2 adalah sebesar 24 ms (menunggu P1 selesai), dan untuk P3 sebesar 27 ms (menunggu P1 dan P2 selesai).

**Gambar 14.1. Gantt Chart Kedatangan Proses**

|  |
| --- |
| Gantt Chart Kedatangan Proses |

Urutan kedatangan adalah P1, P2 , P3; *gantt chart* untuk urutan ini adalah:

*Waiting time* rata-ratanya adalah sebesar(0+24+27)/3 = 17ms. *Turnaround time* untuk P1 sebesar 24 ms, sedangkan untuk P2 sebesar 27 ms (dihitung dari awal kedatangan P2 hingga selesai dieksekusi), untuk P3 sebesar 30 ms. *Turnaround time* rata-rata untuk ketiga proses tersebut adalah (24+27+30)/3 = 27 ms.

Kelemahan dari algoritma ini:

1. *Waiting time* rata-ratanya cukup lama.
2. Terjadinya *convoy effect,* yaitu proses-proses menunggu lama untuk menunggu 1 proses besar yang sedang dieksekusi oleh CPU. Algoritma ini juga menerapkan konsep non-preemptive, yaitu setiap proses yang sedang dieksekusi oleh CPU tidak dapat di-interrupt oleh proses yang lain.

Misalkan proses dibalik sehingga urutan kedatangan adalah P3, P2, P1. *Waiting time* adalah P1=6; P2=3; P3=0. *Average waiting time*: (6+3+0)/3=3.

**Gambar 14.2. Gantt Chart Kedatangan Proses Sesudah Urutan Kedatangan Dibalik**

|  |
| --- |
| Gantt Chart Kedatangan Proses Sesudah Urutan Kedatangan Dibalik |

## Round Robin(RR)

Algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar time quantum. Jika time quantum-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 quantum.

Algoritma ini sepenuhnya bergantung besarnya time quantum. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma first come first served. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang.

Permasalahan utama pada Round Robin adalah menentukan besarnya time quantum. Jika time quantum yang ditentukan terlalu kecil, maka sebagian besar proses tidak akan selesai dalam 1 quantum. Hal ini tidak baik karena akan terjadi banyak switch, padahal CPU memerlukan waktu untuk beralih dari suatu proses ke proses lain (disebut dengan context switches time). Sebaliknya, jika time quantum terlalu besar, algoritma Round Robin akan berjalan seperti algoritma first come first served. Time quantum yang ideal adalah jika 80% dari total proses memiliki CPU burst time yang lebih kecil dari 1 time quantum.

**Gambar 14.4. Urutan Kejadian Algoritma Round Robin**

|  |
| --- |
| Urutan Kejadian Algoritma Round Robin |

**Gambar 14.5. Penggunaan Waktu Quantum**

|  |
| --- |
| Penggunaan Waktu Quantum |

*SRF (Shortest Remaining First)*

Penjadwalan ini merupakan:

Penjadwalan preemptive

Penjadwalan berprioritas dinamis

Penjadwalan SRF merupakan perbaikan dari SJF. SJF merupakan penjadwalan nonpreemptive sedang SRF adalah preemptive yang dapat digunakan untuk sistem timesharing.

Ketentuan:

Pada SRT, proses dengan sisa waktu jalan diestimasi terendah dijalankan, termasuk proses-proses yang baru tiba.

Perbedaan SRF dengan SJF

Pada SJF, begitu proses dieksekusi, proses dijalankan sampai selesai.

Pada SRT proses sedang berjalan (Running) dapat diambil alih oleh proses baru dengan sisa waktu jalan yang diestimasi lebih rendah.

Kelemahan:

SRT mempunyai overhead yang lebih besar dibandingkan SJF. SRT memerlukan penyimpanan waktu layanan yang telah dihabiskan proses dan kadang-kadang harus menangani peralihan.

Tibanya proses-proses kecil akan segera dijalankan.

Proses-proses lebih lama berarti dengan lama dan variasi waktu tunggu lebih lama dibandingkan pada SJF.

Secara teoritis SRT memberi waktu tunggu minimum tapi karena adanya overhead peralihan maka pada situasi tertentu SJF bisa memberi kinerja yang lebih baik dibanding SRT.

*HRN (Highest Ratio Next)*

Penjadwalan HRN merupakan:

Penjadwalan non-preemptive

Penjadwalan berprioritas dinamis

Penjadwalan ini juga untuk mengkoreksi kelemahan SJF. HRN adalah strategi penjadwalan nonpreemptive dengan prioritas proses tidak hanya merupakan fungsi dari waktu layanan tapi juga jumlah waktu tunggu proses.

Prioritas dinamis HRN dihitung berdasarkan rumus:

Prioritas = (Waktu tunggu + waktu layanan) / waktu layanan

Karena waktu layanan muncul sebagai pembagi maka proses yang lebih pendek mempunyai prioritas yang lebih baik. Karena waktu tunggu sebagai pembilang maka proses yang telah menunggu lebih lama juga mempunya kesempatan lebih bagus untuk memperoleh layanan pemrosesan.

Disebut HRN (High Response Next) karena (waktu tanggap adalah waktu tunggu + waktu layanan). Ketentuan HRN berarti agar memperoleh waktu tanggap tertinggi yang harus dilayani.

*Shortest Remaining Time (SRT)*

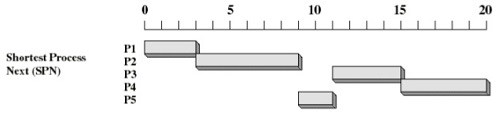
Atau disebut juga ***Sisa Waktu Terpendek Diutamakan***, mempunyai ciri- ciri :

* Penjadwalan preemptive
* Penjadwalan dengan prioritas
* Prioritas didasarkan pada pendeknya sisa proses, makin pendek sisa proses makin tinggi prioritasnya.

Langkah-langkah dalam melaksanakan penjadwalan ini :

* Harus selalu memperhatikan saat proses tiba atau proses selesai
* Dihitung lama sisa proses dari semua proses yang ada pada saat itu, kalau ada proses dengan sisa proses yang lebih pendek dari sisa proses pada proses yang dikerjakan, maka atas dasar preemptive, proses dihentikan diganti oleh proses dengan sisa proses yang terpendek

Shortest Process Next (SPN)

[](https://1.bp.blogspot.com/-jJzXQLwN8wA/VrZF3j3gGMI/AAAAAAAAIhY/O_TWE58qBTM/s1600/spn.jpg)

Eksekusi proses diatur berdasarkan perkiraan ukuran proses terkecil. Sehingga proses yang datang belakangan akan ditaruh didepan dan dieksekusi terlebih dahulu jika ukuran proses tersebut paling kecil diantara proses yang lain.

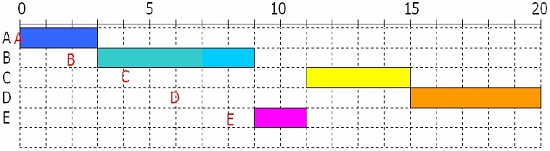
**Kelebihan :**

* Dapat mencegah kerugian proses kecil seperti yang dialami FCFS
* Throughput tinggi
* Proses kecil mempunyai response time kecil.

**Kekurangan :**

* Scheduler harus mengetahui atau memperkirakan ukuran setiap proses yang akan dieksekusi.
* Proses besar dapat mengalami starvation
* Overhead bisa tinggi

**Contoh:**  
Seperti kasus pada FCFS yang diselesaikan dengan metode SPN.  
  
**Gantt Chart**

[](https://2.bp.blogspot.com/-nbov6hYNy4I/VrZGaHAhV9I/AAAAAAAAIhc/Yym1JiUkrp0/s1600/gantt%2Bchart.jpg)

**Tabel Proses**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | A | B | C | D | E | Mean |
| Finsih Time | 3 | 9 | 15 | 20 | 11 |  |
| Arival Time | 0 | 2 | 4 | 6 | 8 |  |
| TAT | 3 | 7 | 11 | 14 | 3 | 7.60 |
| Service Time | 3 | 6 | 4 | 5 | 2 |  |
| NTAT | 1.00 | 1.17 | 2.75 | 2.80 | 1.50 | 1.84 |