Praktikum 9

Penjadwalan CPU 1

POKOK BAHASAN:

- ✓ Membuat program simuliasi Pendawalan CPU

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami cara Penjadwalan CPU.
- ✓ Membuat Aplikasi simulasi Algoritma penjadwalan first come first serve
- ✓ Membuat Aplikasi simulasi Algoritma penjadwalan shortest job first
- ✓ Membuat Aplikasi simulasi Algoritma penjadwalan round robin

DASAR TEORI:

1 Penjadwalan CPU

Penjadual CPU adalah basis dari multi programming sistem operasi. Dengan men-switch CPU diantara proses. Akibatnya sistem operasi bisa membuat komputer produktif.

Algoritma penjadual CPU yang berbeda mempunyai property yang berbeda. Dalam memilih algoritma yang digunakan untuk situasi tertentu, kita harus memikirkan properti yang berbeda untuk algoritma yang berbeda. Banyak kriteria yang dianjurkan utnuk membandingkan penjadual CPU algoritma. Kritria yang biasanya digunakan dalam memilih adalah:

1. CPU utilization: kita ingin menjaga CPU sesibuk mungkin. CPU utilization akan mempunyai range dari 0 ke 100 persen. Di sistem yang sebenarnya seharusnya ia mempunyai range dari 40 persen samapi 90 persen.
2. Throughput: jika CPU sibuk mengeksekusi proses, jika begitu kerja telah dilaksanakan. Salah satu ukuran kerja adalah banyak proses yang diselesaikan per unit waktu, disebut througput. Untuk proses yang lama mungkin 1 proses per jam; untuk proses yang sebentar mungkin 10 proses perdetik.
3. Turnaround time: dari sudur pandang proses tertentu, kriteria yang penting adalah berapa lama untuk mengeksekusi proses tersebut. Interval dari waktu yang diizinkan dengan waktu yang dibutuhkan untuk menyelesaikan sebuah prose disebut turn-around time. Trun around time adalah jumlah periode untuk menunggu untuk bisa ke memori, menunggu di ready queue, eksekusi di CPU, dan melakukan I/O.
4. Waiting time: algoritma penjadual CPU tidak mempengaruhi waktu untuk melaksanakan proses tersebut atau I/O; itu hanya mempengaruhi jumlah waktu yang dibutuhkan proses di antrian ready. Waiting time adalah jumlah periode menghabiskan di antrian ready.
5. Response time: di sistem yang interaktif, turnaround time mungkin bukan waktu yang terbaik untuk kriteria. Sering sebuah proses bisa memproduksi output diawal, dan bisa meneruskan hasil yang baru sementara hasil yang sebelumnya telah diberikan ke user. Ukuran yang lain adalah waktu dari pengiriamn permintaan sampai respon yang pertama di berikan. Ini disebut response time, yaitu waktu untuk memulai memberikan respon, tetapi bukan waktu yang dipakai output untu respon tersebut.

Biasanya yang dilakukan adalah memaksimalkan CPU utilization dan throughput, dan minimalkan turnaround time, waiting time, dan response time

2 Algoritma Penjadual First Come, First Served

Penjadual CPU berurusan dengan permasalahan memutuskan proses mana yang akan dillaksanakan, oleh karena itu banyak bermacam algoritma penjadual, di seksi ini kita akan mendiskripsikan beberapa algoritma.

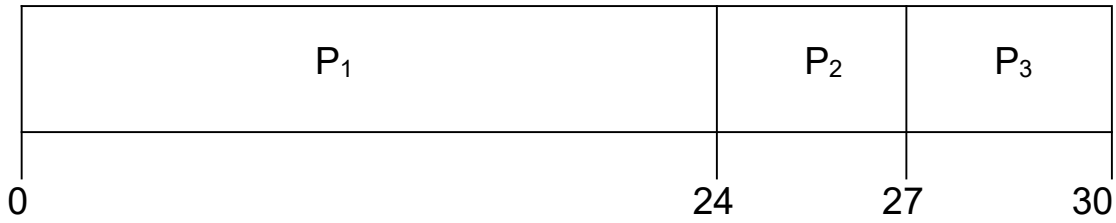
Ini merupakan algoritma yang paling sederhana, dengan skema proses yang meminta CPU mendapat prioritas. Implementasi dari FCFS mudah diatasi dengan FIFO queue.

Contoh:

Proses	Burst Time
P1	24
P2	3
P3	3

Gambar 9-1. Kedatangan Proses

misal urutan kedatangan adalah P1, P2, P3 Gantt Chart untuk ini adalah:



Gambar 9-2. Gannt Chart Kedatangan Proses I.

- **Waiting time untuk *P1* = 0; *P2* = 24; *P3* = 27**
- **Average waiting time: (0 + 24 + 27)/3 = 17**

misal proses dibalik sehingga urutan kedatangan adalah P3, P2, P1.

Gantt chartnya adalah:

Gambar 9-3. Gannt Chart Kedatangan Proses III

- **Waiting time untuk *P1* = 6; *P2* = 0; *P3* = 3**
- **Average waiting time: (6 + 0 + 3)/3 = 3**

Dari dua contoh diatas bahwa kasus kedua lebih baik dari kasus pertama, karena pengaruh kedatangan disamping itu FCFS mempunyai kelemahan yaitu convoy effect dimana seandainya ada sebuah proses yang kecil tetapi dia mengantri dengan proses yang membutuhkan waktu yang lama mengakibatkan proses tersebut akan lama dieksekusi.

Penjadual FCFS algoritma adalah nonpremptive. Ketika CPU telah dialokasikan untuk sebuah proses, proses tetap menahan CPU sampai selssai. FCFS algortima jelas merupakan masalah bagi sistem time-sharing, dimana sangat penting untuk user mendapatkan pembagian CPU pada regular interval. Itu akan menjadi bencana untuk megizinkan satu proses pada CPU untuk waktu yang tidak terbatas

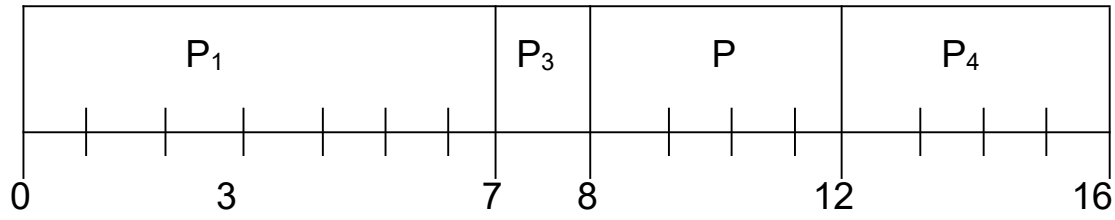
3 Penjadwalan Shortest Job First

Salah satu algoritma yang lain adalah Shortest Job First. Algoritma ini berkaitan dengan waktu setiap proses. Ketika CPU bebas proses yang mempunyai waktu terpendek untuk menyelesaikannya mendapat prioritas. Seandainya dua proses atau lebih mempunyai waktu yang sama maka FCFS algoritma digunakan untuk menyelesaikan masalah tersebut.

Contoh:

Arrival Time	Burst Time	proses
0.0	7	*P1*
2.0	4	*P2*
4.0	1	*P3*
5.0	4	*P4*

Gambar 9-4. Kedatangan Proses.



Gambar 9-5. Gannt Chart SJF Non-Preemtive.

Rata-rata Menunggu = (0 + 6 + 3 + 7)/4 = 4

TUGAS PENDAHULUAN:

1. Lihat Tabel dibawah,,Gambarkan 4 diagram Chart yang mengilustrasikan eksekusi dari proses-proses tersebut menggunakan FCFS, SJF nonpreemptive, round robin.

Tabel Proses

Proses	Burst Time	Prioritas
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Keterangan:

- Misal diberikan beberapa proses dibawah ini dengan panjang CPU burst (dalam milidetik)
- Semua proses diasumsikan datang pada saat t=0 dan quantum time= 3

PERCOBAAN

1. Login ke sistem GNU/Linux kemudian buka terminal.
2. Lakukan percobaan terhadap coding-coding FCFS berikut.

```
#include<stdio.h>
#include<string.h>
main()
{
  // n= banyak proses, AT= Arrival Time, WT= Waiting Time, TAT=
TurnAround Time
  // b= burst time, TotWT= Total Waiting Time, TotTA= Total
TurnAround time
  // name= nama proses, AvWT= Average Waiting Time, AvTA= Average
TurnAround time

  int n, AT[100], b[100], i, j, tmp, WT[100], TAT[100], time[100];
  int TotWT=0, TotTA=0;
  float AvWT, AvTA;
  char name[20][20], tmpName[20];

  printf("\t Algoritma CPU Scheduling FCFS \n");
  puts("");
  printf("Jumlah Proses\t= "); scanf("%d", &n);
  puts("");

  // Masukkan data yang diproses
  for(i=0; i<n; i++){
    fflush(stdin);

    printf(Nama Proses\t= "); scanf("%s", &name[i]);
    printf("Arrival time\t= "); scanf("%d", &AT[i]);
    printf("Burst time\t= "); scanf("%d", &b[i]);
    puts("");
  }

  // Urutkan Data
  for(i=0; i<n; i++){
    for(j=i+1; j<n; j++)
      if(AT[i]>AT[j]){
        //tukar nama
        strcpy(tmpName, name[i]);
        strcpy(name[i], name[j]);
        strcpy(name[j], tmpName);

        //tukar arrival time
        tmp=AT[i];
        AT[i]=AT[j];
        AT[j]=tmp;

        //tukar burst
        tmp=b[i];
        b[i]=b[j];
        b[j]=tmp;
      }
    }

    time[0]=AT[0];

    puts("\n\t Tabel Proses ");
    puts("=========================================");
    printf("| no | proses\t | time arrival\t | burst |\n");
    puts("————————————");

    for (i=0; i<n; i++){
      printf("| %2d | %s\t |   \t%d\t | %d\t |\n", i+1, name[i],
```

```
    AT[i], b[i]);
 62       time[i+1]=time[i]+b[i]; //menghitung time pada gant chart
 63       WT[i]=time[i]-AT[i];
 64       TAT[i]=time[i+1]-AT[i];
 65       TotWT+=WT[i];
 66       TotTA+=TAT[i];
 67     }
 68
 69     puts("=========================================");
 70     printf("\tTotal waiting time\t= %d \n", TotWT);
 71     printf("\tTurn around time\t= %d \n", TotTA);
 72     puts("\n\t Tabel Waktu Proses ");
 73     puts("=================================================");
 74      printf("| no | proses\t | waiting time\t | turn arround\t
    |\n");
 75     puts("———————————————-");
 76
 77     for(i=0; i<n; i++){
 78       printf("| %2d | %s\t |  \t%d\t | \t%d\t |\n", i+1, name[i],
    WT[i], TAT[i]);
 79   }
 80
 81   puts("=================================================");
 82
 83   //untuk Gant Chart
 84   puts("\n");
 85   puts("\t Gant-Chart \n");
 86   for(i=0; i<n; i++){
 87     printf(" %s\t ", name[i]);
 88   }
 89
 90   puts("");
 91   for(i=0; i<n; i++){
 92     printf("|=========");
 93   }
 94
 95   printf("|\n");
 96   for(i=0; i<=n; i++){
 97     printf(" %d\t ", time[i]);
 98   }
 99   printf("//diperoleh dari penjumlahan Burst");
100
101   puts("\n");
102   AvWT=(float)TotWT/n;
103   AvTA=(float)TotTA/n;
104   printf("\tAverage Waiting Time : %f\n", AvWT);
105   printf("\tAverage Turn Around TIme : %f\n", AvTA);
106 }
```

3. Compile menggunakan **`gcc –o fcfs fcfs.c`** lalu run menggunakan **`./fcfs`**
4. Masukkan variabel berikut dengan jumlah proses sebanyak 3:

Nama Proses	Arrival Time	Burst Time
p1	0	3
p2	1	2
p3	2	1

LAPORAN RESMI:
1. Analisa hasil percobaan yang Anda lakukan.
2. Buatlah Program di atas.
3. Berikan kesimpulan dari praktikum ini.