5

CPU Scheduling

Modul 5

CPU Scheduling

5.1. Tujuan Praktikum

Pelaksanaan praktikum ini bertujuan memberikan pengetahuan mengenai CPU scheduling dengan mengamati perilaku CPU saat menjalankan program yang diberikan.

5.2. Capaian Praktikum

1. Mampu menjelaskan konsep CPU Scheduling
2. Mampu mengenali karakteristik dan l

5.3. Dasar Teori

Pada sistem single-processor hanya ada satu proses yang berjalan pada satu waktu. Proses yang lain harus menunggu hingga CPU kosong dan bisa dijadwalkan kembali. Sedangkan pada system multiprogramming ada beberapa proses yang berjalan pada satu waktu dan untuk meningkatkan performansi dari CPU. Beberapa proses disimpan dalam memori pada waktu yang sama, ketika satu proses harus menunggu, sistem operasi mengambil CPU dari proses itu dan memberikannya ke proses yang lain.

5.3.1. Burst Cycle

Keberhasilan penjadwalan CPU tergantung pada eksekusi proses yang terdiri dari siklus eksekusi CPU dan I/O *wait*. Proses bergantian antara dua *state* itu. Eksekusi proses dimulai dengan CPU *burst*. Yang diikuti oleh I/O *burst*, kemudian diikuti CPU *burst* yang lain, kemudian I/O *burst*, dan sebagainya. Akhirnya, CPU *burst* berhenti dengan *system request* untuk mengakhiri eksekusi. Pada saat suatu proses dieksekusi, terdapat banyak CPU *burst* yang pendek dan terdapat sedikit CPU *burst* yang panjang. Program yang I/O *bound* biasanya sangat pendek CPU burst nya, sedangkan program yang CPU *bound* kemungkinan CPU *burst* nya sangat lama.

5.3.2. Kriteria Penjadwalan

Algoritma CPU yang berbeda memiliki sifat yang berbeda, dan pilihan algoritma tertentu dapat mendukung satu kelas proses di atas yang lain. Dalam memilih algoritma untuk digunakan dalam situasi tertentu, harus dipertimbangkan sifat-sifat dari berbagai algoritma.

a. Utilisasi CPU

Utilisasi CPU dapat berkisar 0-100 persen. Di sistem yang nyata, range antara 40 – 90 persen.

b. Throughput

Jika CPU sibuk dalam eksekusi proses, kemudian selesai. Salah satu ukuran kerja adalah banyak proses yang diselesaikan per satuan waktu.

c. Turnaround Time

Dari sudut pandang proses tertentu , kriteria yang penting adalah berapa lama waktu yang dibutuhkan untuk eksekusi proses tersebut. Interval dari waktu pengajuan proses ke waktu penyelesaian. Jumlah dari periode yang dihabiskan untuk masuk ke memori, menunggu di *ready queue*, eksekusi di CPU, dan dan melakukan I/O.

d. Waiting Time

Jumlah dari periode yang dihabiskan menunggu di *ready queue*.

e. Response Time

Waktu dari pengajuan permohonan sampai respon pertama diproduksi. Pengukuran ini

disebut response time yang merupakan waktu yang dibutuhkan untuk memulai respon, bukan waktu yang dibutuhkan untuk keluaran respon.

5.3.3. Algoritma Penjadwalan

Penjadwalan CPU menyangkut penentuan proses-proses yang ada dalam ready queue yang akan dialokasikan pada CPU. Terdapat beberapa algoritma penjadwalan CPU seperti di bawah ini:

a. First Come First Served (FCFS)

Proses yang pertama kali meminta jatah waktu untuk menggunakan CPU akan dilayani terlebih dahulu. Pada skema ini, proses yang meminta CPU pertama kali akan dialokasikan ke CPU pertama kali

b. Shortest Job First Scheduler (SJF)

Pada penjadwalan SJF, proses yang memiliki CPU *burst* paling kecil dilayani terlebih dahulu. SJF adalah algoritma penjadwalan yang optimal dengan rata-rata waktu tunggu yang minimal.

c. Priority

Algoritma SJF adalah suatu kasus khusus dari penjadwalan berprioritas. Tiap-tiap proses dilengkapi dengan nomor prioritas (integer). CPU dialokasikan untuk proses yang memiliki prioritas paling tinggi (nilai integer terkecil biasanya merupakan prioritas terbesar). Jika beberapa proses memiliki prioritas yang sama, maka akan digunakan algoritma FCFS.

d. Round Robin

Konsep dasar dari algoritma ini adalah dengan menggunakan *time-sharing*. Pada dasarnya algoritma ini sama dengan FCFS, hanya saja bersifat *preemptive*. Setiap proses mendapatkan waktu CPU yang disebut dengan waktu *quantum* (quantum time) untuk membatasi waktu proses, biasanya 1-100 milidetik. Setelah waktu habis, proses ditunda dan ditambahkan pada *ready queue*.

5.4. **Langkah Praktiku**m

Praktikum ini dilakukan dengan terlebih dahulu terhubung dengan layanan aws educate dengan cara mengaktifkan instance dari halaman instance summary. Pilih action dan Start untuk mengaktifkan instance. Lakukan koneksi SSH dengan cara yang sama seperti pada Bab 1. Lakukan percobaan pada kode program berikut.

```
1 #include <stdio.h>
2 #include <string.h>
3
4 main()
5 {
6     // n= banyak proses, AT= Arrival Time, WT= Waiting Time, TAT=
7     // TurnAround Time b= burst time, TotWT= Total Waiting Time, TotTA=
```

```
8     // Total TurnAround time name= nama proses, AvWT= Average Waiting Time,
9     // AvTA= Average TurnAround time
10    int n, AT[100], b[100], i, j, tmp, WT[100], TAT[100], time[100];
11    int TotWT = 0, TotTA = 0;
12    float AvWT, AvTA;
13    char name[20][20], tmpName[20];
14    printf("\t Algoritma CPU Scheduling FCFS \n");
15    puts("");
16    printf("Jumlah Proses\t= ");
17    scanf("%d", &n);
18    puts("");
19
20    // Masukkan data yang diproses
21    for (i = 0; i < n; i++)
22    {
23        fflush(stdin);
24        printf("Nama Proses\t= ");
25        scanf("%s", name[i]);
26        printf("Arrival time\t= ");
27        scanf("%d", &AT[i]);
28        printf("Burst time\t= ");
29        scanf("%d", &b[i]);
30        puts("");
31    }
32
33    // Urutkan Data
34    for (i = 0; i < n; i++)
35    {
36        for (j = i + 1; j < n; j++)
37            if (AT[i] > AT[j])
38            {
39                // tukar nama
40                strcpy(tmpName, name[i]);
41                strcpy(name[i], name[j]);
42                strcpy(name[j], tmpName);
43                // tukar arrival time
44                tmp = AT[i];
45                AT[i] = AT[j];
46                AT[j] = tmp;
47                // tukar burst
48                tmp = b[i];
49                b[i] = b[j];
50                b[j] = tmp;
51            }
```

```
52      }
53
54      time[0] = AT[0];
55      puts("\n\t Tabel Proses ");
56      puts("=========================================");
57      printf("| no | proses\t | time arrival\t | burst |\n");
58      puts("-----------------------------------------");
59
60      for (i = 0; i < n; i++)
61      {
62          printf("| %2d | %s\t | \t%d\t | %d\t |\n", i + 1, name[i], AT[i],
63                 b[i]);
64          time[i + 1] = time[i] + b[i]; // menghitung time pada gant chart
65          WT[i] = time[i] - AT[i];
66          TAT[i] = time[i + 1] - AT[i];
67          TotWT += WT[i];
68          TotTA += TAT[i];
69      }
70
71      puts("=========================================");
72      printf("\tTotal waiting time\t= %d \n", TotWT);
73      printf("\tTurn around time\t= %d \n", TotTA);
74      puts("\n\t Tabel Waktu Proses ");
75      puts("=================================================");
76      printf("| no | proses\t | waiting time\t | turn arround\t |\n");
77      puts("-------------------------------------------------");
78
79      for (i = 0; i < n; i++)
80      {
81          printf("| %2d | %s\t | \t%d\t | \t%d\t |\n", i + 1, name[i], WT[i],
82                 TAT[i]);
83      }
84
85      puts("=================================================");
86      // untuk Gant Chart
87      puts("\n");
88      puts("\t Gant-Chart \n");
89
90      for (i = 0; i < n; i++)
91      {
92          printf(" %s\t ", name[i]);
93      }
94
95      puts("");
```

```
96
97      for (i = 0; i < n; i++)
98      {
99          printf("|=========");
100     }
101
102     printf("|\n");
103
104     for (i = 0; i <= n; i++)
105     {
106         printf(" %d\t ", time[i]);
107     }
108
109     printf("//diperoleh dari penjumlahan Burst");
110     puts("\n");
111     AvWT = (float)TotWT / n;
112     AvTA = (float)TotTA / n;
113     printf("\tAverage Waiting Time : %f\n", AvWT);
114     printf("\tAverage Turn Around TIme : %f\n", AvTA);
115 }
```

1. Lakukan kompilasi kode program dengan menggunakan perintah **gcc main.c –o main**
2. Jalankan hasil kompilasi menggunakan perintah **./main**
3. Masukkan variabel berikut dengan jumlah proses sebanyak 3:

Nama Proses	**Arrival Time**	**Burst Time**
P1	**0**	**3**
P2	**1**	**2**
P3	**2**	**1**

4. *Capture* / *Snapshot output*-nya dan simpan sebagai laporan.
5. Jalankan program tersebut beberapa kali dengan nilai variable masukan yang berbeda-beda.
6. Amati *output* hasil percobaannya

Dari hasil percobaan tersebut, simpulkanlah jenis algoritma penjadwalan apa yang digunakan dalam program tersebut?

5.5. Pembahasan

1. Login ke sistem GNU/Linux kemudian buka terminal.
2. Lakukan percobaan terhadap kode program berikut.

```c
#include <stdio.h>
int main()
{
    int i, j, n, time, remain, flag = 0, tq;
    int TotWT = 0, TotTA = 0, AT[100], b[100], rt[100];
    printf("Masukkan jumlah proses : ");
    scanf("%d", &n);
    remain = n;
    for (i = 0; i < n; i++)
    {
        printf("Masukkan arrival time untuk Proses P%d :", i + 1);
        scanf("%d", &AT[i]);
        printf("Masukkan burst time untuk Proses P%d :", i + 1);
        scanf("%d", &b[i]);
        rt[i] = b[i];
    }
    printf("Masukkan time quantum ");
    scanf("%d", &tq);
    printf("\n\nProcess\t|Turnaround time|waiting time\n\n");
    for (time = 0, i = 0; remain != 0;)
    {
        if (rt[i] <= tq && rt[i] > 0)
        {
            time += rt[i];
            rt[i] = 0;
            flag = 1;
        }
        else if (rt[i] > 0)
        {
            rt[i] -= tq;
            time += tq;
        }
        if (rt[i] == 0 && flag == 1)
        {
            remain--;
```

```
            printf("P[%d]\t|\t%d\t|\t%d\n", i + 1, time - AT[i],
                   time - AT[i] - b[i]);
            TotWT += time - AT[i] - b[i];
            TotTA += time - AT[i];
            flag = 0;
        }
        if (i == n - 1)
            i = 0;
        else if (AT[i + 1] <= time)
            i++;
        else
            i = 0;
    }
    printf("\nAverage Waiting Time = %f\n", TotWT * 1.0 / n);
    printf("Average Turnaround Time = %f\n", TotTA * 1.0 / n);
    return 0;
}
```

3. Lakukan kompilasi kode program dengan menggunakan perintah **gcc main.c –o main**
4. Jalankan hasil kompilasi menggunakan perintah **./main**
5. Masukkan variabel berikut dengan: jumlah proses sebanyak 3, time quantum 2, dan

Nama Proses	Arrival Time	Burst Time
P1	0	3
P2	1	2
P3	2	1

6. Amati hasil outputnya!
7. Jika variabel pada pada nomor 4 di atas diubah menjadi: jumlah proses sebanyak 3, time quantum 2

Nama Proses	Arrival Time	Burst Time
P1	0	5
P2	20	4

P3	**25**	**5**

8. Amati hasil outputnya!
9. Berdasarkan hasil pengamatan anda, maka algoritma apa yang digunakan dalam program tersebut?

5.6. Kesimpulan

Jelaskan kesimpulan dari hasil percobaan pada Bab 5 ini

UNTUK DIPERHATIKAN

Setiap selesai melakukan praktikum, jangan lupa menonaktfikan instance agar billing kuota tidak terus berjalan. Lakukan dengan cara memilih button Action pada Instance Summary. Pilih Stop sebagai perintah yang harus dijalankan.