LABORATORIUM JARINGAN KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

PRAKTIKUM SISTEM OPERASI

SEMESTER : GENAP **TAHUN : 2015/2016**

BAB IV

JUDUL BAB : SCHEDULING

DISUSUN OLEH : MOH ARIF ANDRIAN

NIM : 156150600111002

ASISTEN : SISKA PERMATASARI
ZAENAL KURNIAWAN

KOORDINATOR ASISTEN : DANY RAHMANA

LAPORAN PRAKTIKUM SISTEM OPERASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Laporan : BAB IV
Asisten : Siska Permatasari
Zaenal Kurniawan

BAB IV
SCHEDULING

4.1 DASAR TEORI

4.1.1 Konsep Dasar

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 sistem *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.

4.1.2 *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.

4.1.3 Kriteria Penjadwalan

Algoritma CPU *scheduling* 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.

4.1.4 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*.

4.2 MATERI PRAKTIKUM

1. Login ke sistem GNU/Linux kemudian buka terminal.

```
andrian@156150600111002:~$ sudo su
[sudo] password for andrian:
root@156150600111002:/home/andrian#
```

2. Lakukan percobaan terhadap coding-coding FCFS berikut.

```
  GNU nano 2.5.3                File: fcfs.c                      Modified

#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


^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

3. Compile menggunakan **gcc –o fcfs fcfs.c** lalu run menggunakan **./fcfs**

```
root@156150600111002:/home/andrian/Scheduling# gcc -o fcfs fcfs.c
fcfs.c: In function ‘main’:
fcfs.c:24:34: warning: format ‘%s’ expects argument of type ‘char *’, b
ut argument 2 has type ‘char (*)[20]’ [-Wformat=]
 printf("Nama Proses\t= "); scanf("%s", &name[i]);
                                  ^
root@156150600111002:/home/andrian/Scheduling# ./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

```
root@156150600111002:/home/andrian/Scheduling# ./fcfs
         Algoritma CPU Scheduling FCFS

Jumlah Proses    = 3

Nama Proses      = p1
Arrival time     = 0
Burst time       = 3

Nama Proses      = p2
Arrival time     = 1
Burst time       = 2

Nama Proses      = p3
Arrival time     = 2
Burst time       = 1
```

5. Amati hasil outputnya dan buat kesimpulannya!

```
         Tabel Proses
=========================================
| no | proses    | time arrival  | burst |
-------------
|  1 | p1        |      0        | 3     |
|  2 | p2        |      1        | 2     |
|  3 | p3        |      2        | 1     |
=========================================
        Total waiting time      = 5
        Turn around time        = 11

         Tabel Waktu Proses
=================================================
| no | proses    | waiting time  | turn arround  |
-----------------
|  1 | p1        |      0        |      3        |
|  2 | p2        |      2        |      4        |
|  3 | p3        |      3        |      4        |
=================================================


         Gant-Chart

 p1       p2      p3
|=========|=========|=========|
 0        3        5        6

        Average Waiting Time : 1.666667
        Average Turn Around TIme : 3.666667
root@156150600111002:/home/andrian/Scheduling#
```

Jawaban:

Karena program diatas adalah FCFS jadi proses yang mempunyai arrival time yang lebih rendah akan dilayani terlebih dahulu.

6. Algoritma FCFS termasuk mode *preemptive* atau *non-preemptive*? Jelaskan berdasarkan hasil dari keluaran program ini!

   Jawaban:

   Algoritma FCFS termasuk non preemtive karena FCFS tidak memberlakukan prioritas yang berarti algoritma FCFS non preemptive.

```
1   #include<stdio.h>
2   #include<string.h>
3   main()
4   {
5   // n= banyak proses, AT= Arrival Time, WT= Waiting Time,
6   TAT= TurnAround Time
7   // b= burst time, TotWT= Total Waiting Time, TotTA=
8   Total TurnAround time
9   // name= nama proses, AvWT= Average Waiting Time, AvTA=
10  Average TurnAround time
11
12  int n, AT[100], b[100], i, j, tmp, WT[100], TAT[100],
13  time[100];
14  int TotWT=0, TotTA=0;
15  float AvWT, AvTA;
16  char name[20][20], tmpName[20];
17
18  printf("\t Algoritma CPU Scheduling FCFS \n");
19  puts("");
20  printf("Jumlah Proses\t= "); scanf("%d", &n);
21  puts("");
22
23  // Masukkan data yang diproses
24
25  for(i=0; i<n; i++){
26  fflush(stdin);
27
28  printf(Nama Proses\t= "); scanf("%s", &name[i]);
29  printf("Arrival time\t= "); scanf("%d", &AT[i]);
30  printf("Burst time\t= "); scanf("%d", &b[i]);
31  puts("");
32
33  }
34
35  // Urutkan Data
36
37  for(i=0; i<n; i++){
```

```
38  for(j=i+1; j<n; j++)
39  if(AT[i]>AT[j]){
40
41  //tukar nama
42
43  strcpy(tmpName, name[i]);
44  strcpy(name[i], name[j]);
45  strcpy(name[j], tmpName);
46
47  //tukar arrival time
48
49  tmp=AT[i];
50  AT[i]=AT[j];
51  AT[j]=tmp;
52
53  //tukar burst
54
55  tmp=b[i];
56  b[i]=b[j];
57  b[j]=tmp;
58
59  }
60
61  }
62
63  time[0]=AT[0];
64
65  puts("\n\t Tabel Proses ");
66  puts("=========================================");
67  printf("| no | proses\t | time arrival\t | burst |\n");
68  puts("————————————");
69
70  for (i=0; i<n; i++){
71
72  printf("| %2d | %s\t |   \t%d\t | %d\t |\n", i+1,
73  name[i], AT[i], b[i]);
74  time[i+1]=time[i]+b[i]; //menghitung time pada gant
75  chart
76  WT[i]=time[i]-AT[i];
77  TAT[i]=time[i+1]-AT[i];
78  TotWT+=WT[i];
79  TotTA+=TAT[i];
80
81  }
82
83  puts("=========================================");
```

```
84  printf("\tTotal waiting time\t= %d \n", TotWT);
85  printf("\tTurn around time\t= %d \n", TotTA);
86  puts("\n\t Tabel Waktu Proses ");
87  puts("=============================================
88  ==");
89  printf("| no | proses\t | waiting time\t | turn
90  arround\t |\n");
91  puts("——————————————-");
92
93  for(i=0; i<n; i++){
94
95  printf("| %2d | %s\t |  \t%d\t | \t%d\t |\n", i+1,
96  name[i], WT[i], TAT[i]);
97
98  }
99
100 puts("=============================================
101 ==");
102
103 //untuk Gant Chart
104
105 puts("\n");
106 puts("\t Gant-Chart \n");
103 for(i=0; i<n; i++){
104 printf(" %s\t ", name[i]);
105
106 }
107
108 puts("");
109 for(i=0; i<n; i++){
110 printf("|=========");
111
112
113 printf("|\n");
114 for(i=0; i<=n; i++){
115 printf(" %d\t ", time[i]);
116
    } printf("//diperoleh dari penjumlahan Burst");

    puts("\n");
    AvWT=(float)TotWT/n;
    AvTA=(float)TotTA/n;
    printf("\tAverage Waiting Time : %f\n", AvWT);
    printf("\tAverage Turn Around TIme : %f\n", AvTA);
    }
```

LAPORAN PRAKTIKUM SISTEM OPERASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Tugas : BAB IV
Asisten : Siska Permatasari
Zaenal Kurniawan

TUGAS PRAKTIKUM

1. Login ke sistem GNU/Linux kemudian buka terminal.

```
andrian@156150600111002:~$ sudo su
[sudo] password for andrian:
root@156150600111002:/home/andrian#
```

2. Lakukan percobaan terhadap coding-coding Round Robin yang ada di bawah.

```
  GNU nano 2.5.3                File: rr.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;

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

3. Compile menggunakan **gcc –o rr rr.c** lalu run menggunakan **./rr**

```
root@156150600111002:/home/andrian/Scheduling# gcc -o rr rr.c
root@156150600111002:/home/andrian/Scheduling# ./rr
```

4. 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

```
root@156150600111002:/home/andrian/Scheduling# ./rr
Masukkan jumlah proses : 3
Masukkan arrival time untuk Proses P1 :0
Masukkan burst time untuk Proses P1 :3
Masukkan arrival time untuk Proses P2 :1
Masukkan burst time untuk Proses P2 :2
Masukkan arrival time untuk Proses P3 :2
Masukkan burst time untuk Proses P3 :1
Masukkan time quantum 2


Process |Turnaround time|waiting time

P[2]    |       3       |       1
P[3]    |       3       |       2
P[1]    |       6       |       3

Average Waiting Time = 2.000000
Average Turnaround Time = 4.000000
root@156150600111002:/home/andrian/Scheduling#
```

5. Amati hasil outputnya untuk buat kesimpulannya!

   Jawaban:

   Pada algoritma round robin diatas time quantum mempunyai pelayanan pada suatu proses. Jika brust tume memiliki quantum maka proses tersebut akan memundanya dan melakukan proses yang lainnya

6. Apa yang menjadi perbedaan dengan algoritma Round Robin ini dengan percobaan sebelumnya yaitu FCFS? Jelaskan!

   Jawaban:

   Perhitungan dari keduanya sangatlah berbeda. Algoritma round robin memproses dengan mekanisme time quantum dan selalu berpindah proses setiap time quantum sedangkan FCFS langsung menyelesaikan proses yang dilakukan.

7. Jika variabel pada pada nomor 4 di atas diubah menjadi: jumlah proses sebanyak 3, time quantum 2, dan

Nama Proses	Arrival Time	Burst Time
p1	0	5
p2	20	4
p3	25	5

```
root@156150600111002:/home/andrian/Scheduling# ./rr
Masukkan jumlah proses : 3
Masukkan arrival time untuk Proses P1 :0
Masukkan burst time untuk Proses P1 :5
Masukkan arrival time untuk Proses P2 :20
Masukkan burst time untuk Proses P2 :4
Masukkan arrival time untuk Proses P3 :25
Masukkan burst time untuk Proses P3 :5
Masukkan time quantum 2


Process |Turnaround time|waiting time

P[1]    |       5       |       0
```

8. Amati hasil outputnya dan buat kesimpulannya pada nomor 7. Apa yang menjadi perbedaan dengan hasil pada nomor 5, jelaskan!

   Jawaban:

   Proses tidak akan berlangsung setelah P1 karena tidak memenuhi source code yang harus dipenuhi pada saat p1 selesai waktunya adalah 5 sedangkan p2 baru akan mulai berjalan pada detik 20, sehingga hal itu membuat p2 tidak berjalan.

9. Buatlah tambahan program untuk menampilkan Gant-Chart pada algoritma Round Robin yang ada di bawah ini.

```
1   #include<stdio.h>
2   int main()
3   {
4   int i,j,n,time,remain,flag=0,tq;
5   int TotWT=0,TotTA=0,AT[100],b[100],rt[100];
6   printf("Masukkan jumlah proses : ");
7   scanf("%d",&n);
8   remain=n;
9   for(i=0;i<n;i++)
10  {
11  printf("Masukkan arrival time untuk Proses P%d :",i+1);
12  scanf("%d",&AT[i]);
13  printf("Masukkan burst time untuk Proses P%d :",i+1);
14  scanf("%d",&b[i]);
15  rt[i]=b[i];
16  }
17  printf("Masukkan time quantum ");
18  scanf("%d",&tq);
19  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;
}
```

Setelah penambahan

```
#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;
}
puts("\n");
puts("\t Gant-Chart \n");
for (i = 0; i < n; i++) {
printf("P[%d]\t ", i + 1);
}
puts("");
for (i = 0; i < n; i++) {
printf("|=========");
}
printf("|\n");
for (i = 0; i <= n; i++) {
printf(" %d\t ", AT[i]);
}
```

```
puts("\n");
printf("\nAverage Waiting Time = %f\n", TotWT * 1.0 / n);
printf("Average Turnaround Time = %f\n", TotTA * 1.0 / n);
return 0;
}
```

```
  GNU nano 2.5.3                File: rr2.c                          Modifiec

#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;

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell
```

Output program:

```
root@156150600111002:/home/andrian/Scheduling# nano rr2.c
root@156150600111002:/home/andrian/Scheduling# gcc -o rr2 rr2.c
root@156150600111002:/home/andrian/Scheduling# ./rr2
Masukkan jumlah proses : 3
Masukkan arrival time untuk Proses P1 :0
Masukkan burst time untuk Proses P1 :3
Masukkan arrival time untuk Proses P2 :1
Masukkan burst time untuk Proses P2 :2
Masukkan arrival time untuk Proses P3 :2
Masukkan burst time untuk Proses P3 :1
Masukkan time quantum 2


Process |Turnaround time|waiting time

P[2]    |       3       |       1
P[3]    |       3       |       2
P[1]    |       6       |       3


         Gant-Chart

P[1]     P[2]    P[3]
|=========|=========|=========|
 0        1       2       32765


Average Waiting Time = 2.000000
Average Turnaround Time = 4.000000
root@156150600111002:/home/andrian/Scheduling#
```

LAPORAN PRAKTIKUM SISTEM OPERASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Kesimpulan : BAB IV
Asisten : Siska Permatasari
Zaenal Kurniawan

KESIMPULAN

Scheduling adalah sebuah metode yang mengatur proses-proses yang akan berjalan. Bagian dari sistem operasi yang membuat pilihan dinamakan scheduller, sedangkan algoritma yang digunakan dinamakan schedulling algorithm.
Strategi penjadwalan adalah :

a. Penjadwalan Nonpreemtive.
   Strategi di mana suatu proses bisa diganggu, artinya isi penjadwalan bisa berubah saat suatu penjadwalan sedang berjalan.
b. Penjadwalan preemtive.
   Strategi di mana suatu proses tidak bisa diganggu, artinya isi penjadwalan tidak berubah dan langsung diselesaikan tanpa adanya penambahan suatu proses.

Algoritma digunakan dalam situasi tertentu harus diperhatikan sifat sifat dari kriteria, yang dilihat dari utilisasi CPU, Throughput,1 Turnaround time, Waiting Time dan Respon Time. Algoritma penjadwalan antara lain :

a. First Comme First Served (FCFS)
   Penjadwalan ini dipengaruhi oleh proses mana yang datang lebih awal dan meminta untuk diproses oleh CPU.
b. Shortest Job First (SJF)
   Penjadwalan ini dipengaruhi oleh besar-kecilnya Burst Time. Burst Time yang nilainya lebih kecil akan diproses terlebih dahulu.
c. Priority Schedulling
   Penjadwalan ini dipengaruhi oleh besar-kecilnya Priority. Priority dengan nilai terkecil akan didahulukan, jika terdapat dua proses yang memiliki nilai priority sama, akan digunakan algoritma FCFS.
d. Round-Robin Schedulling.
   Sama dengan konsep time sharing, yang berupa algoritma FCFS, hanya bersifat preemptive menggunakan quantum time untuk membatasi waktu proses.