MODUL PRAKTIKUM SISTEM OPERASI

PRAKTIKUM Vb

PENJADWALAN PROSES

A. T U J U A N

Memahami algoritma penjadwalan proses.

B. DASAR TEORI

Asumsikan semua proses tiba pada saat yang bersamaan.

1. FCFS
   Algoritma penjadwalan FCFS, yaitu membaca sejumlah proses/ pekerjaan dalam sistem, CPU *burst time*-nya. Penjadwalan dilakukan berdasarkan waktu kedatangan proses pada antrian, terlepas dari parameter lainnya. Setiap proses akan dijalankan sesuai waktu kedatangannya. Hitung waktu tunggu dan waktu perputaran dari masing-masing proses yang sesuai.

2. SJF
   Algoritma penjadwalan SJF, yaitu membaca sejumlah proses / pekerjaan dalam sistem, CPU *burst time*-nya. Pekerjaan diatur sesuai dengan *burst time* mereka. Mungkin ada dua pekerjaan dalam antrian dengan waktu eksekusi yang sama, hal ini harus dilakukan pendekatan FCFS yaitu urutan berdasarkan waktu kedatangannya. Setiap proses akan dijalankan sesuai dengan lamanya *burst time* mereka. Kemudian hitung waktu tunggu dan waktu penyelesaian masing-masing proses yang sesuai.

3. ROUND ROBIN
   Algoritma penjadwalan round robin, yaitu membaca sejumlah proses / pekerjaan dalam sistem, CPU *burst time*-nya, serta besarnya *time slice*. *Time slice* ditugaskan untuk menentukan porsi yang sama dalam menjalankan setiap proses yang dijalankan pada urutan melingkar. Sehingga setiap proses mendapatkan kesempatan yang sama. Hitung waktu tunggu dan waktu perputaran dari masing-masing proses yang sesuai.

C. LANGKAH – LANGKAH

1. FCFS CPU Scheduling Algorithm. Beri nama `fcfs.c`.

```
#include<stdio.h>

int main()
{
	int bt[20], wt[20], tat[20], i, n;
	float wtavg, tatavg;
	printf("\nEnter the number of processes -- ");
	scanf("%d", &n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter Burst Time for Process %d -- ", i); scanf("%d",
	&bt[i]);
	}
	wt[0] = wtavg = 0;
	tat[0] = tatavg = bt[0];
	for(i=1;i<n;i++)
	{
		wt[i] = wt[i-1] +bt[i-1];
		tat[i] = tat[i-1] +bt[i];
		wtavg = wtavg + wt[i];
		tatavg = tatavg + tat[i];
	}
	printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
	for(i=0;i<n;i++)
		printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i],
	tat[i]);
	printf("\nAverage Waiting Time -- %f", wtavg/n);
	printf("\nAverage Turnaround Time -- %f", tatavg/n);
}
```

Amati dan catat hasil yang ditampilkan, bandingkan dengan perhitungan manual algoritma FCFS.

2. SJF CPU Scheduling Algorithm. Beri nama `sjf.c`.

```
#include<stdio.h>

int main()
{
      int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
      float wtavg, tatavg;
      printf("\nEnter the number of processes -- ");
      scanf("%d", &n);
      for(i=0;i<n;i++)
      {
            p[i]=i;
            printf("Enter Burst Time for Process %d -- ", i);
            scanf("%d", &bt[i]);
      }
      /* mengurutkan proses dengan urutan burst time terkecil */
      for(i=0;i<n;i++)
            for(k=i+1;k<n;k++)
                  if(bt[i]>bt[k])
                  {
                        temp=bt[i];
                        bt[i]=bt[k];
                        bt[k]=temp;

                        temp=p[i];
                        p[i]=p[k];
                        p[k]=temp;
                  }
      wt[0] = wtavg = 0;
      tat[0] = tatavg = bt[0];
      for(i=1;i<n;i++)
      {
            wt[i] = wt[i-1] +bt[i-1];
            tat[i] = tat[i-1] +bt[i];
            wtavg = wtavg + wt[i];
            tatavg = tatavg + tat[i];
      }
      printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND
      TIME\n");
      for(i=0;i<n;i++)
            printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i],
      tat[i]);
      printf("\nAverage Waiting Time -- %f", wtavg/n);
      printf("\nAverage Turnaround Time -- %f", tatavg/n);
}
```

Amati dan catat hasil yang ditampilkan, bandingkan dengan perhitungan manual algoritma SJF.

3. Roundrobin CPU Scheduling Algorithm. Beri nama `roundrobin.c`.

```
#include<stdio.h>

main()
{
	int i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
	float awt=0,att=0,temp=0;
	printf("Enter the no of processes -- ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter Burst Time for process %d -- ", i+1);
		scanf("%d",&bu[i]);
		ct[i]=bu[i];
	}
	printf("\nEnter the size of time slice -- ");
	scanf("%d",&t);

	max=bu[0];
	for(i=1;i<n;i++) /* temukan proses dengan burst time terbesar */
		if(max<bu[i])
			max=bu[i];
	for(j=0;j<(max/t)+1;j++)
		for(i=0;i<n;i++) /* iterasi pada setiap proses */
			if(bu[i]!=0)
				if(bu[i]<=t) /*jika kurang dari time slice*/
				{
					tat[i]=temp+bu[i]; /* catat ta-time */
					temp=temp+bu[i];
					bu[i]=0;
				}
				else /*jika lebih besar dari time slice*/
				{
					bu[i]=bu[i]-t;
					temp=temp+t;
				}
	for(i=0;i<n;i++)
	{
		wa[i]=tat[i]-ct[i]; /* catat waiting time tiap proses*/
		att+=tat[i]; /* catat total turn around time */
		awt+=wa[i];  /* catat total waiting time */
	}
	printf("\nThe Average Turnaround time is -- %f",att/n);
	printf("\nThe Average Waiting time is -- %f ",awt/n);
	printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
	for(i=0;i<n;i++)
		printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);
}
```

Amati dan catat hasil yang ditampilkan, bandingkan dengan perhitungan manual algoritma Round-Robin.

D. EVALUASI

1. Bandingkan implementasi program FCFS dengan SJF, mana yang lebih ringkas? Beri penjelasannya.
2. Dengan jumlah proses dan nilai burst time yang sama pada percobaan FCFS, SJF, dan RR, bandingkan nilai **average waiting-time** dan **average turn-around** time, algoritma mana yang terbaik?
3. Berdasarkan hasil pengamatan terhadap ketiga algoritma penjadwalan, apakah algoritma RR memiliki kelebihan dibandingkan FCFS dan SJF? Beri penjelasannya (silahkan merujuk buku referensi)