SJF (pre-emptive) 30/05/24

```
#include <stdio.h>
#include <stdlib.h>
struct Process {
    int pid;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int waiting_time;
    int turnaround_time;
    int response_time;
};
int findShortestProcess(struct Process processes[],
int n, int current_time){
int shortest_index = -1;
int shortest_remaining_time = INT_MAX;
for(int i=0; i<n; i++){
if(processes[i].arrival_time <= current_time
&& processes[i].remaining_time <
shortest_remaining_time && processes[i].
remaining_time > 0){
shortest_index = i;
shortest_remaining_time = processes[i].
remaining_time;
}
}
return shortest_index;
}
```

```
void SRTF (struct Process processes[], int n){
  int total_time = 0;
  int completed = 0;
  int current_process = -1;
  int total_waiting_time = 0;
  int total_turnaround_time = 0;
  printf("Gantt Chart: \n");
  while (completed != n) {
    int shortest_index = findShortestProcess(
    processes, n, total_time);
    if (shortest_index == -1) {
      printf("Idle ");
      total_time++;
      continue;
    }
    if (current_process != shortest_index) {
      printf("P%d ", processes[shortest_index
        ].pid);
    }
    if (processes[shortest_index].remaining_
    time == processes[shortest_index].burst_time)
    processes[shortest_index].response_time = total_time
    - processes[shortest_index].arrival_time;
  }
  processes[shortest_index].remaining_time--;
  total_time++;
```

```
processes[shortest_index].remaining_time--;
total_time++;
printf("process\t Turnaround time\t waiting
Time\t Response Time\n");
for(int i=0; i<n; i++){
printf("%d\t%d\t\t%d\t\t\t\n",
processes[i].pid, processes[i].turnaround_
processes[i].waiting_time, processes[i].
response_time);
}
float avg_waiting_time = (float) total_
float(float.SATF :: avg_waiting_time
total_turnaround_time)/n;
for(int i=0; i<n; i++){
}
avg_response_time/=n;
printf("Average waiting time: %.2f\n",
avg_waiting_time);
printf("Avg Turnaround Time: %.2f\n",
avg_turnaround_time);
printf("Average Response Time: %.2f\n",
avg_response_time);
}
```

```
int main () {
int n;
printf ("Enter the number of processes : ");
scanf ("%d", &n);
struct Process *processes = malloc (n * sizeof
(struct Process));
for (int i = 0; i < n; i++) {
printf ("Enter arrival time for process %d : "
n, i + 1);
scanf ("%d", &processes (i). arrival_time);
printf ("Enter burst time for process %d : "
n, i + 1);
scanf ("%d", &processes (i). burst_time);
processes (i) .pid = i + 1;
processes (i). remaining_time = processes (i).
burst_time;
}
SRTF (processes, n);
free (processes);
return 0;
}
```

Output

Enter the number of processes: 4
Enter arrival time for process 1: 0
Enter burst time for process 1: 8
Enter arrival time for process 2: 1
Enter burst time for process 2: 4
Enter arrival time for process 3: 2
Enter burst time for process 3: 9
Enter arrival time for process 4: 3
Enter burst time for process 4: 5

Gantt chart

process	Turnaround Time	waiting Time	Response Time
P_1	17	9	0
P_2	4	0	0
P_3	24	15	15
P_4	7	2	2

Avg waiting Time: 6.50
Avg Turnaround Time: 13.00
Avg Response Time: 4.25.

Shashank-SP
1BM22CS256

Round robin

```
#include<stdio.h>
typedef struct{
    int pid;
    int bt;
    int at;
    int wt;
    int tat;
    int rem_bt;
    int start_time;
} process;
void calculateTimes(Process proc[],
int n, int Quantum) {
    int t=0;
    int completed = 0;
    int total_wt =0, total_tat =0, total_
    int is_first_run[n];
}
while (completed != n){
    int all_idle = 1;
    for (int i = 0; i<n; i++){
        if (proc[i].at <= t && proc[i].
rem_bt > 0){
all_idle = 0;
if (is_first_run[i]){
proc[i].start = time = t;
```

```
        is_first_run[i]=0;
    }
    if (proc[i].rem_bt > quantum){
        t.t+=quantum;
        proc[i].rem_bt -=quantum;
    }else{
        t=t+proc[i].rem_bt;
        proc[i].rem_bt=0;
        total_wt+=proc[i].wt;
        total_tat+=proc[i].tat;
        total_rt+=proc[i].tt;
        Completed++;
    }
  }
 }
 if(all_idle){
    t++t;
  }
 }
 printf("\nPID\tBT\tAT\tWT\tTAT\
    \tRT\n");
int main(){
  int n, quantum;
  printf("\nEnter the number of processes
                    \n");
```

```
scanf("%d", &n);

process proc[n];
for(int i = 0; i < n; i++){
  proc[i].pid = i + 1;
}
printf("Enter arrival time for process
%d: ", i + 1);
scanf("%d", &proc[i].at);
printf("Enter burst time for process
%d: ", i + 1);
scanf("%d", &proc[i].bt);

}
calculateTimes(proc, n, Quantum);
return 0;

}
```

Outputs

Enter time Quantum: 3

PID	BT	AT	WT	TAT	RT
1	3	2	7	4	1
2	5	4	5	10	2
3	6	0	6	10	0

Average waiting time: 4.03

Average Turn Around Time = 8.67

Average Beyond time 1.00

Shashank SP

1Bm22CS276

3) Priority (pre-emptive)

```
struct process {
int pid;
int priority;
int burst time;
}
void ~~primary~~ priority Scheduling
(struct process.
process[], (int n) {
int waiting, time[n], turn around time[n];
total waiting time=0 .turn around time
waiting time [0]=0;
turn around time [0]= process[0].
burst-time
for (int i=1, i<n, i++ ){
waiting time [1] = turn around time[i-1];
turn around time[i] = waiting time[i].
processes (i) burst time;
}
for (int i=0; i<n; i++ ){
total waiting time += waiting time (i)
total turnaround time += turn around
time (i);
}
```

```
for (int i=0; i<n; i++)
printf("%d \t %d \t %d \t %d \n",
process[i] pid, process[i] priority
processes[i].burst_time; waiting_time[i],
turnaround time[i]);
}
int main(){
int n;
printf("Enter the number of process");
scanf("%d", &n);
struct process processes[n];
for(int i=0; i<n; i++){
printf("Enter details for process
processes[i].pid=i+1;
printf("priority");
scanf("%d", &processes[i].priority);
printf("Burst time:");
scanf("%d", &processes[i].burst
priority Scheduling(processes, n);
return 0;
```

Output

Enter the no of processes: 3
Enter details for process 1
priority: 1
Burst time 3
Enter details for process 2
priority 5
Burst time -10
Enter details for process -3
priority 4
Burst time: 12.

process	priority	Burst	Waiting	TT
1	1	3	0	3
2	5	10	3	13
3	4	13	13	26

Average waiting time: 5.33
Average Turn-Around time 14.00.

Shathanik SP
1Bm22CS256

Priority (non-preemptive)

```
#include<stdio.h>
struct process{
int id;
int burstTime;
int priority;
int waitingTime;
int turnaroundTime;
};
void SortProcessesByPriority(struct process
processes[], int n){
for(int i=0; i<n-1; i++){
for(int j=i+1; j<n; j++){
if(processes[i].priority > processes[j].
priority){
struct Process temp=processes[i];
processes[i]=processes[j];
processes[j]=temp;
}
}
}
}
void calculateTimes(struct Process process
[], int n){
processes[0].waitingTime=0;
processes[0].turnaroundTime=processes
[0].burstTime;
```

```
for (int i=1; i<n; i++){
processes[i].waitingTime = processes[i-1].
waitingTime + processes[i-1].burstTime;
processes[i].turnaroundTime = processes
[i].waitingTime + processes[i].burstTime;
}
}
void calculateAverage(struct process processes
[], int n, float * avgWaitingTime,
float * avgTurnaroundTime){
int totalWaitingTime=0;
int totalTurnaroundTime=0;
for (int i=0; i<n; i++){
totalWaitingTime += processes[i].waiting
Time;
totalTurnaroundTime += process[i].turnaround
Time;
}
*avgWaitingTime = (float) totalWaiting Time/n;
*avgTurnaroundTime = (float) total Turnaround Time/n;
void printResults(struct process processes[],
int n, float avgWaitingTime, float
avgTurnaroundTime){
printf("Process ID \t Burst Time \t Priority
\t Waiting Time \t Turnaround Time \n");
```

~~for int~~

int main() {
int n;
printf("Enter the number of processes:");
scanf("%d", &n);
struct Process processes[n];
for (int i=0; i<n; i++) {
processes[i].id = i+1;
printf("Enter burst time & priority for process %d:", i+1);
scanf("%d %d", &processes[i].burstTime, &processes[i].priority);
}
printResults(processes, n, avgWaitingTime, avgTurnaroundTime);
return 0;
}

Output

Enter the number of processes: 6
Enter burst time & priority for process 1:
Enter burst time & priority for process 2: 5
Enter burst time & priority for process 3: 4 3
Enter burst time & priority for process 4: 2 5
Enter burst time & priority for process 5: 7 1
process 6: 4 4

process ID	Burst Time	priority	Waiting Time	T.A.T
P1	3	2	0	3
P2	4	3	3	7
P6	4	4	7	11
P4	2	5	11	13
P3	5	6	13	18
P5	7	7	18	25

Average waiting Time = 8.67
Average Turnaround time = 12.83

Shashank-SP
1Bm22cs256