4ITRC2 Operating System Lab

Lab Assignment 5

**Aim**: To create C programs for the different scheduling algorithms.

**To perform**: Create and execute C programs for following CPU Scheduling Algorithms:

1. First Come First Serve (FCFS)

**File Name: fcfs.c**

#include <stdio.h>

int main() {

int n, i;

printf("Enter number of processes: ");

scanf("%d", &n);

int bt[n], wt[n], tat[n];

printf("Enter burst time for each process:\n");

for (i = 0; i < n; i++) {

printf("P[%d]: ", i + 1);

scanf("%d", &bt[i]);

}

wt[0] = 0;

for (i = 1; i < n; i++)

wt[i] = wt[i - 1] + bt[i - 1];

for (i = 0; i < n; i++)

tat[i] = wt[i] + bt[i];

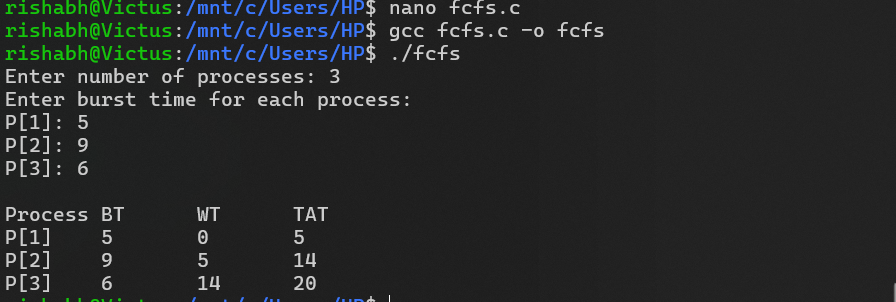
printf("\nProcess\tBT\tWT\tTAT\n");

for (i = 0; i < n; i++)

printf("P[%d]\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);

return 0;

}



1. Shortest Job First (SJF)

File Name**: sjf.c**

#include <stdio.h>

int main() {

int n, i, j;

printf("Enter number of processes: ");

scanf("%d", &n);

int bt[n], wt[n], tat[n], p[n], temp;

printf("Enter burst time for each process:\n");

for (i = 0; i < n; i++) {

printf("P[%d]: ", i + 1);

scanf("%d", &bt[i]);

p[i] = i + 1;

}

// Sort based on burst time

for (i = 0; i < n; i++) {

for (j = i + 1; j < n; j++) {

if (bt[i] > bt[j]) {

temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;

temp = p[i]; p[i] = p[j]; p[j] = temp;

}

}

}

wt[0] = 0;

for (i = 1; i < n; i++)

wt[i] = wt[i - 1] + bt[i - 1];

for (i = 0; i < n; i++)

tat[i] = wt[i] + bt[i];

printf("\nProcess\tBT\tWT\tTAT\n");

for (i = 0; i < n; i++)

printf("P[%d]\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);

return 0;

}

wt[0] = 0;

for (i = 1; i < n; i++)

wt[i] = wt[i - 1] + bt[i - 1];

for (i = 0; i < n; i++)

tat[i] = wt[i] + bt[i];

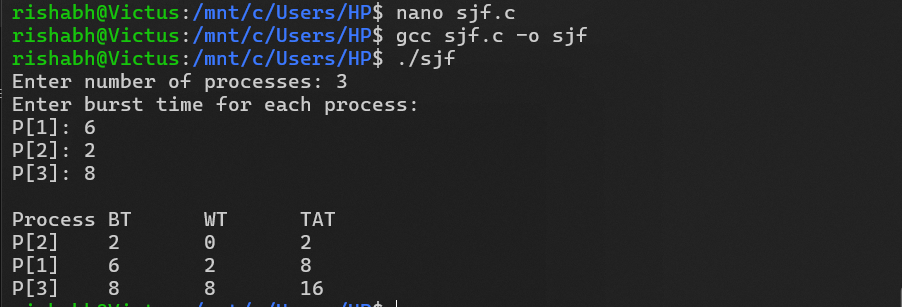
printf("\nProcess\tBT\tWT\tTAT\n");

for (i = 0; i < n; i++)

printf("P[%d]\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);

return 0;

}



1. Round Robin Scheduling

File Name: **round\_robin.c**

#include <stdio.h>

int main() {

int n, i, tq;

printf("Enter number of processes: ");

scanf("%d", &n);

int bt[n], rt[n], wt[n], tat[n];

for (i = 0; i < n; i++) {

printf("Enter burst time for P[%d]: ", i + 1);

scanf("%d", &bt[i]);

rt[i] = bt[i];

}

printf("Enter time quantum: ");

scanf("%d", &tq);

int time = 0, done;

do {

done = 1;

for (i = 0; i < n; i++) {

if (rt[i] > 0) {

done = 0;

if (rt[i] > tq) {

time += tq;

rt[i] -= tq;

} else {

time += rt[i];

wt[i] = time - bt[i];

rt[i] = 0;

}

}

}

} while (!done);

for (i = 0; i < n; i++)

tat[i] = bt[i] + wt[i];

printf("\nProcess\tBT\tWT\tTAT\n");

for (i = 0; i < n; i++)

printf("P[%d]\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);

return 0;

}

printf("Enter time quantum: ");

scanf("%d", &tq);

int time = 0, done;

do { done = 1;

for (i = 0; i < n; i++) {

if (rt[i] > 0) {

done = 0;

if (rt[i] > tq) {

time += tq;

rt[i] -= tq;

} else {

time += rt[i];

wt[i] = time - bt[i];

rt[i] = 0;

}

}

}

} while (!done);

for (i = 0; i < n; i++)

tat[i] = bt[i] + wt[i];

printf("\nProcess\tBT\tWT\tTAT\n");

for (i = 0; i < n; i++)

printf("P[%d]\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);

return 0; }

