PROGRAM – 5

AIM: Write a program using pipes to emulate the first come first serve scheduling algorithm . Find average waiting time and turn around time .

INTRODUCTION:

C PROGRAM :

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

int timestamp = 0 ;

struct pro{

int at;

int bt;

int st;

int wt;

int tat;

int ct;

};

typedef struct pro pro ;

int main(){

int n,i;

float w,t;

printf("\nEnter no. of processes : ");

scanf("%d",&n);

pro pros[n];

//enter arrival time and burst time of all processes

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

printf("\nEnter process no. %d .",i+1);

printf("\nArrival time : ");

scanf("%d",&pros[i].at);

printf("Burst time : ");

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

pros[i].st = 0;

pros[i].wt = 0;

pros[i].tat = 0;

pros[i].ct = 0;

}

//schedule processes

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

int p1[2],p2[2];

//pipe from parent to child

if (pipe(p1) == -1){

printf("\nPipe creation failed.");

exit(0);

}

//pipe from child to parent

if (pipe(p2) == -1){

printf("\nPipe creation failed.");

exit(0);

}

int p = fork();

if (p < 0){

printf("\nError creating Child %d",i+1);

exit(0);

}else if (p == 0){

//child process

close(p1[1]);

close(p2[0]);

printf("\n\nChild : %d id : %d",i+1,getpid());

pro child;

read(p1[0],&child.at,sizeof(int));

read(p1[0],&child.bt,sizeof(int));

read(p1[0],&child.st,sizeof(int));

read(p1[0],&child.wt,sizeof(int));

printf("\nI Arrived at %d time units.", child.at );

printf("\nI waited for %d time units.", child.wt );

printf("\nI started at %d time units.", child.st);

printf("\nI am sleeping for %d time units.",child.bt);

printf("\n");

for (int i = 0; i < child.bt; ++i){

printf(".");

sleep(1);

}

child.ct = child.st + child.bt;

child.tat = child.ct - child.at;

printf("\nI am completed at %d time units and my turn around time is %d. ",child.ct,child.tat);

write(p2[1],&child.ct,sizeof(int));

write(p2[1],&child.tat,sizeof(int));

exit(0);

}else{

//parent process

close(p1[0]);

close(p2[1]);

if (pros[i].at > timestamp ){

timestamp = pros[i].at;

}else{

pros[i].wt += (timestamp - pros[i].at) ;

}

pros[i].st = timestamp;

write(p1[1],&pros[i].at,sizeof(int));

write(p1[1],&pros[i].bt,sizeof(int));

write(p1[1],&pros[i].st,sizeof(int));

write(p1[1],&pros[i].wt,sizeof(int));

wait(NULL);

read(p2[0],&pros[i].ct,sizeof(int));

read(p2[0],&pros[i].tat,sizeof(int));

timestamp = pros[i].ct;

}

}

printf("\n");

//calculating average waiting time and average

w=0;

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

w+=pros[i].wt;

}

w = w/n;

t=0;

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

t+=pros[i].tat;

t = t/n;

printf("\nAverage Waiting Time : %f",w);

printf("\nAverage Turn Around Time : %f",t);

printf("\n");

return 0;

}

LEARNING OUTCOMES:

OUTPUT :



