**CSA0465 – OPERATING SYSTEMS FOR HANDLING DEADLOCKS**

**LAB EXPERIMENTS – Slot B**

**Name :- Aswini .P**

**Reg no :- 192011399**

**1.Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.**

#include<stdio.h>

#include<unistd.h>

main()

{

int pid,pid1,pid2;

if(pid==-1)

{

printf("ERROR IN PROCESS CREATION\n");

exit(1);

}

if(pid!=0)

{

pid1=getpid();

printf("\n the parent process ID is %d\n",pid1);

}

else

{

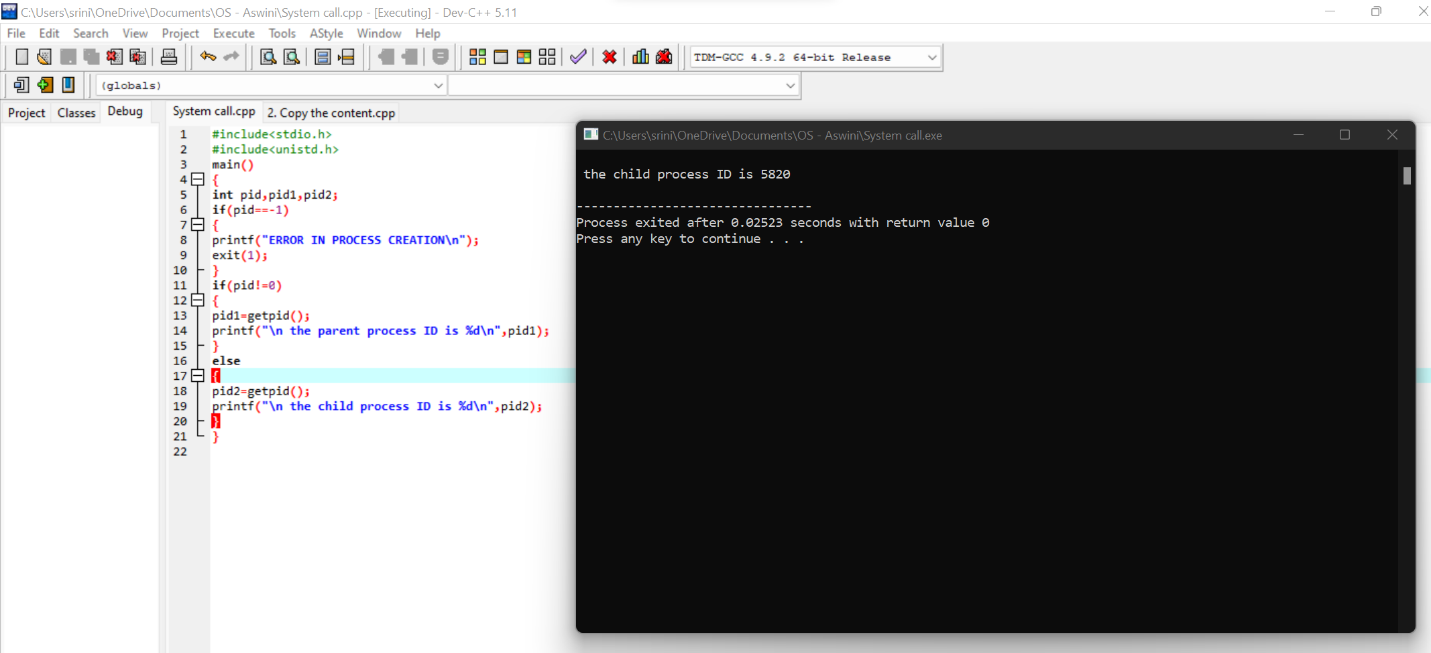
pid2=getpid();

printf("\n the child process ID is %d\n",pid2);

}

}

**Output :-**



**2.Identify the system calls to copy the content of one file to another and illustrate the same using a C program.**

**Program :-**

#include<stdio.h>

#include<unistd.h>

int main()

{

FILE \*fptr1, \*fptr2;

char filename[100], c;

printf("Enter the filename to open for reading \n");

scanf("%s", filename);

fptr1 = fopen(filename, "r");

if (fptr1 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(0);

}

printf("Enter the filename to open for writing \n");

scanf("%s", filename);

fptr2 = fopen(filename, "w");

if (fptr2 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(0);

}

c = fgetc(fptr1);

while (c != EOF)

{

fputc(c, fptr2);

c = fgetc(fptr1);

}

printf("\nContents copied to %s", filename);

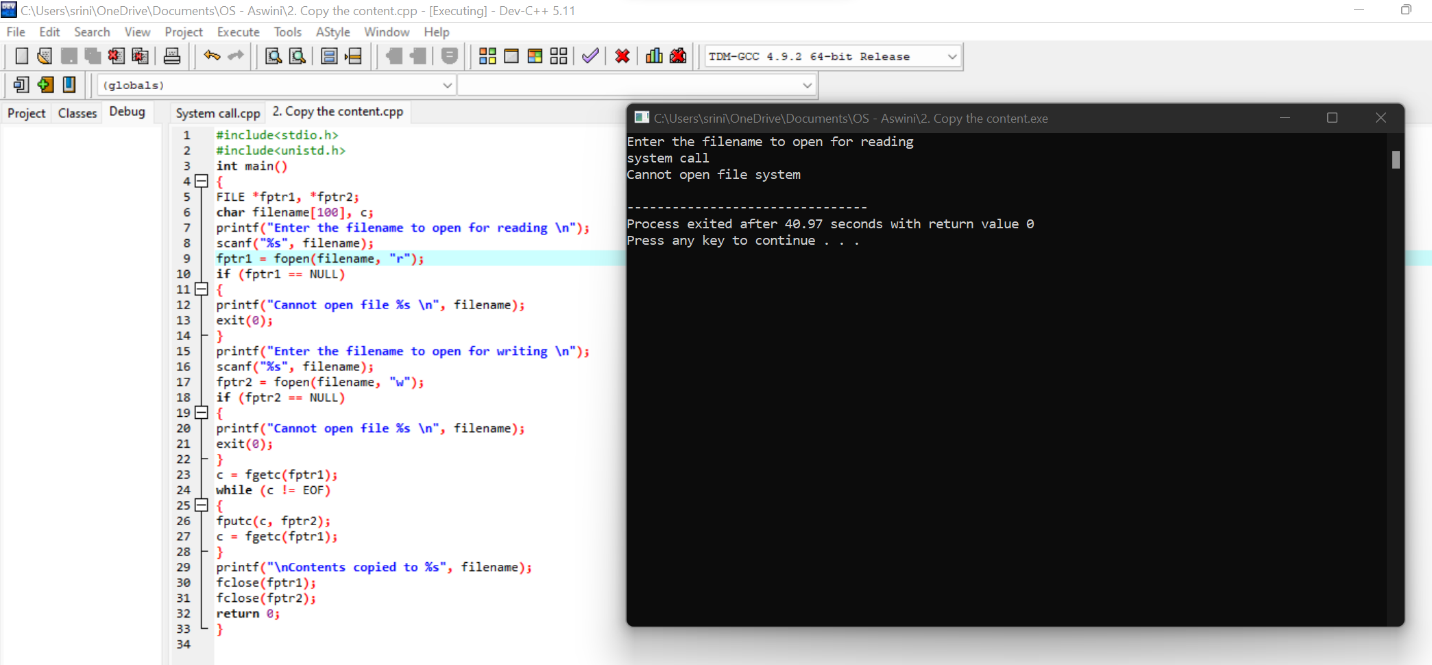
fclose(fptr1);

fclose(fptr2);

return 0;

}

**Output :-**



**3.** **Design a CPU scheduling program with C using First Come First Served technique with the following considerations.**

**a. All processes are activated at time 0.**

**b. Assume that no process waits on I/O devices.**

**Program :-**

#include<stdio.h>

#include<stdlib.h>

struct fcfs

{

int pid;

int btime;

int wtime;

int ttime;

}

p[10];

int main()

{

int i,n;

int totwttime=0,totttime=0;

printf("\n fcfs scheduling...\n");

printf("enter the no of process");

scanf("%d",&n);

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

{

p[i].pid=1;

printf("\n burst time of the process");

scanf("%d",&p[i].btime);

}

p[0].wtime=0;

p[0].ttime=p[0].btime;

totttime+=p[i].ttime;

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

{

p[i].wtime=p[i-1].wtime+p[i-1].btime;

p[i].ttime=p[i].wtime+p[i].btime;

totttime+=p[i].ttime;

totwttime+=p[i].wtime;

}

printf("\n total waiting time :%d", totwttime );

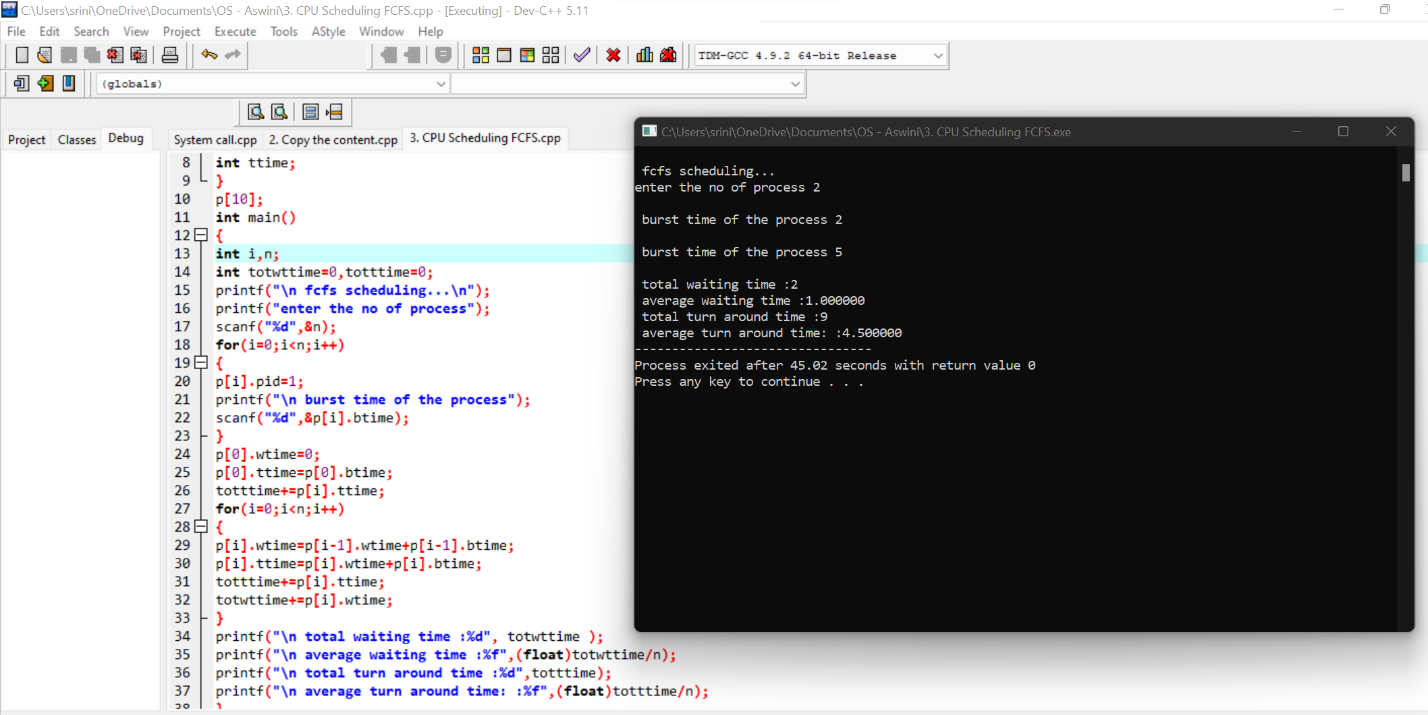
printf("\n average waiting time :%f",(float)totwttime/n);

printf("\n total turn around time :%d",totttime);

printf("\n average turn around time: :%f",(float)totttime/n);

}

**Output :-**



**4.Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.**

**Program :-**

#include <stdio.h>

int main()

{

int A[100][4];

int i, j, n, total = 0, index, temp;

float avg\_wt, avg\_tat;

printf("Enter number of process: ");

scanf("%d", &n);

printf("Enter Burst Time:\n");

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

{

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

scanf("%d", &A[i][1]);

A[i][0] = i + 1;

}

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

{

index = i;

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

if (A[j][1] < A[index][1])

index = j;

temp = A[i][1];

A[i][1] = A[index][1];

A[index][1] = temp;

temp = A[i][0];

A[i][0] = A[index][0];

A[index][0] = temp;

}

A[0][2] = 0;

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

A[i][2] = 0;

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

A[i][2] += A[j][1];

total += A[i][2];

}

avg\_wt = (float)total / n;

total = 0;

printf("P BT WT TAT\n");

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

{

A[i][3] = A[i][1] + A[i][2];

total += A[i][3];

printf("P%d %d %d %d\n", A[i][0],A[i][1], A[i][2], A[i][3]);

}

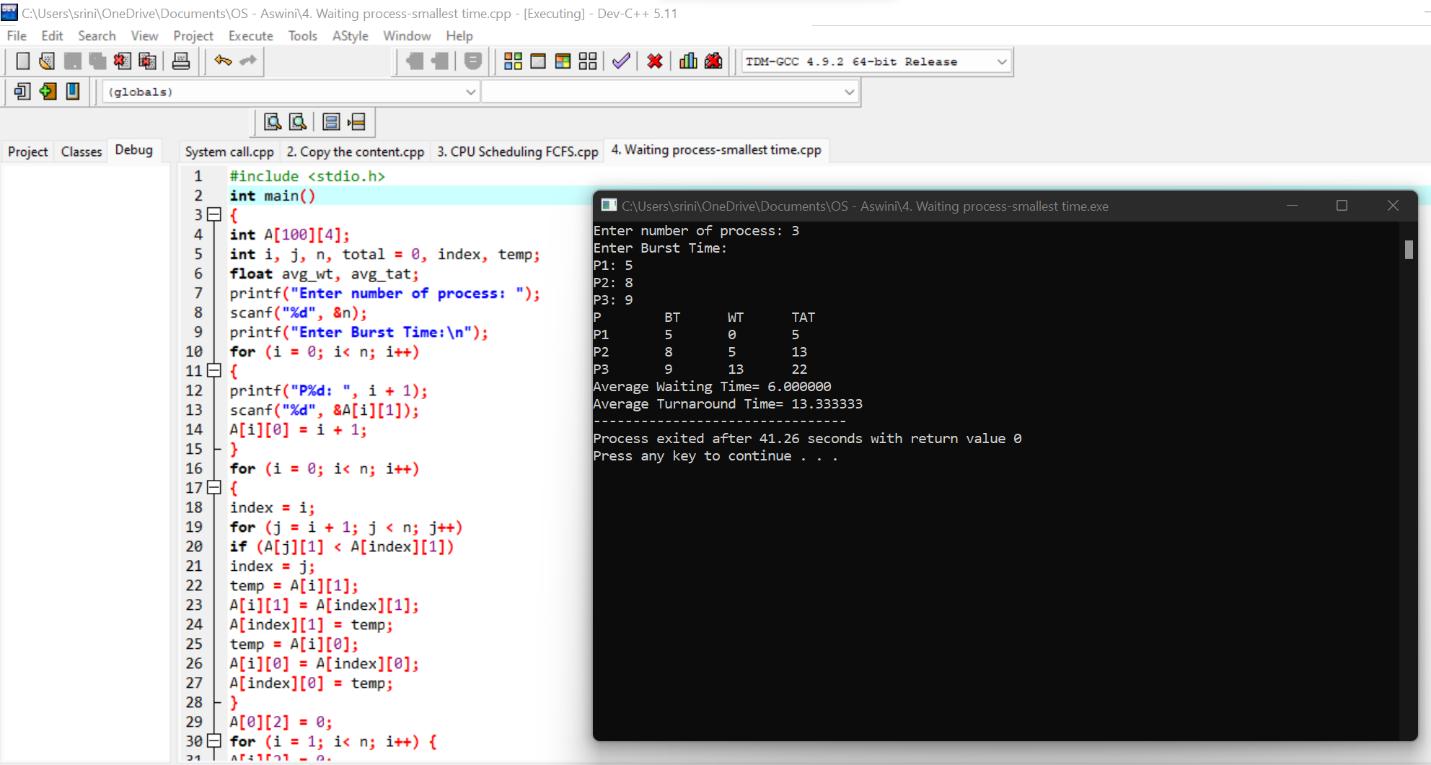
avg\_tat = (float)total / n;

printf("Average Waiting Time= %f", avg\_wt);

printf("\nAverage Turnaround Time= %f", avg\_tat);

}

**Output :-**



**5. Construct a scheduling program with C that selects the waiting process with the highest priority to execute next.**

**Program :-**

#include<stdio.h>

int main()

{

int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg\_wt,avg\_tat;

printf("Enter Total Number of Process:");

scanf("%d",&n);

printf("\nEnter Burst Time and Priority\n");

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

{

printf("\nP[%d]\n",i+1);

printf("Burst Time:");

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

printf("Priority:");

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

p[i]=i+1;

}

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

{

pos=i;

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

{

if(pr[j]<pr[pos])

pos=j;

}

temp=pr[i];

pr[i]=pr[pos];

pr[pos]=temp;

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0;

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

{

wt[i]=0;

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

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=total/n;

total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

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

total+=tat[i];

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

}

avg\_tat=total/n;

printf("\n\nAverage Waiting Time=%d",avg\_wt);

printf("\nAverage Turnaround Time=%d\n",avg\_tat);

return 0;

}

**Output :-**

