**1) Develop a C program to implement system process calls (fork(), exec(), wait(), create process, terminate process).**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

void main()

{

pid\_t pid=fork();

if(pid==-1)

{

printf("Fork Error");

exit(1);

}

if(pid==0)

{

execlp("/bin/date","date",NULL);

exit(1);

}

else

{

int status;

printf("Parent Process\n");

wait();

printf("Child Process Terminates\n");

}

}

**Output:**

**2) Simulate the following CPU scheduling algorithms to find turnaround time and waiting time (a)FCFS, (b)SJF, (c)Priority, (d)Round Robin.**

**(a) FCFS-CPU Scheduling**

#include<stdio.h>

int main() {

int n, bt[20], wt[20], tat[20], avwt = 0, avtat = 0, i, j;

printf("Enter total number of processes (maximum 20): ");

scanf("%d", &n);

printf("\nEnter Process Burst Time:\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] = 0;

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

wt[i] += bt[j];

}

}

printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time");

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

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

avwt += wt[i];

avtat += tat[i];

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

}

// Calculate averages

float avgwt = (float)avwt / n;

float avgtat = (float)avtat / n;

// Display averages

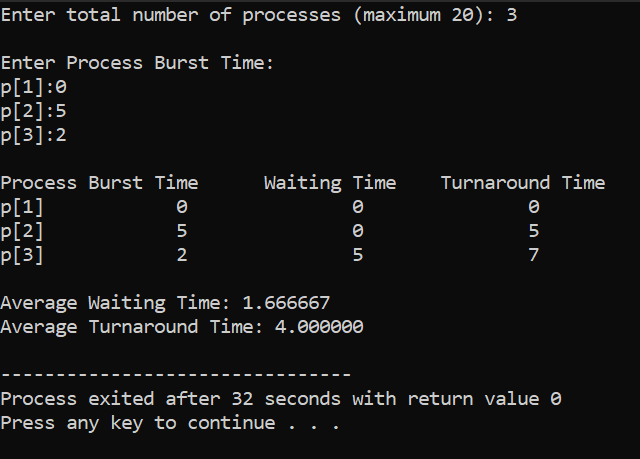
printf("\n\nAverage Waiting Time: %f", avgwt);

printf("\nAverage Turnaround Time: %f\n", avgtat);

return 0;

}

**Output:**



**(b) SJF-CPU Scheduling**

#include <stdio.h>

int main() {

int bt[20], p[20], wt[20], tat[20], i, j, n, totwt = 0, tottat = 0, pos, temp;

float avgwt, avgtat;

printf("Enter the number of processes:");

scanf("%d", &n);

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

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

printf("p%d:", i);

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

p[i] = i;

}

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

pos = i;

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

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

pos = j;

}

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];

}

}

tottat = 0;

printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time");

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

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

tottat += tat[i];

totwt += wt[i];

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

}

avgwt = (float)totwt / n;

avgtat = (float)tottat / n;

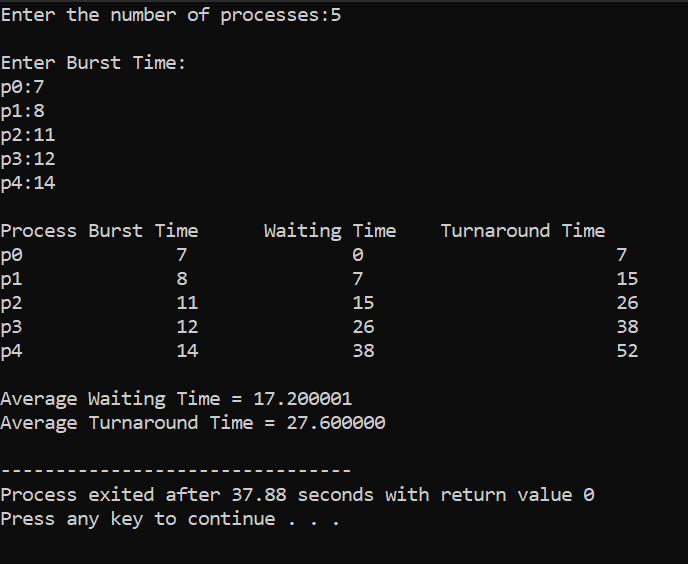
printf("\n\nAverage Waiting Time = %f", avgwt);

printf("\nAverage Turnaround Time = %f\n", avgtat);

return 0;

}

**Output:**



**(c) Priority-CPU Scheduling**

#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("\n Enter 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=0;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("\n Process\tBurst 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%d", p[i], bt[i], wt[i], tat[i]);

}

avg\_tat=total/n;

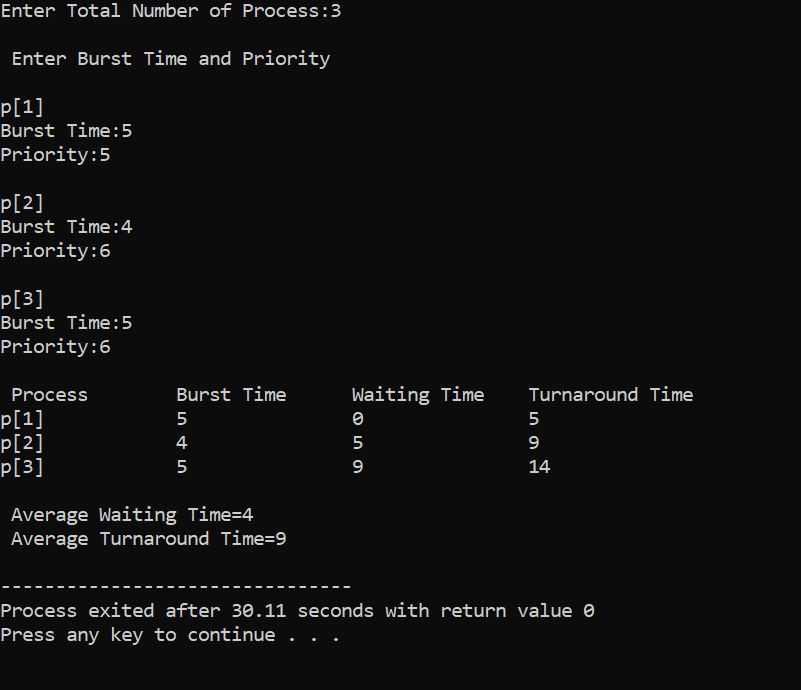
printf("\n\n Average Waiting Time=%d",avg\_wt);

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

return 0;

}

**Output:**



**3) Develop a C program to simulate producer-consumer problem using semaphores.**

#include<stdio.h>

void main()

{

int buffer[10],bufsize,in,out,produce,consume,choice=0;

in=0;

out=0;

bufsize=1;

while(choice!=3)

{

printf("\n1.Produce\t2.Consume\t3.Exit");

printf("\n Enter your choice:");

scanf("%d",&choice);

switch(choice){

case 1:if((in+1)%bufsize==out)

printf("\n Buffer is Full");

else

{

printf("\n Enter the value:");

scanf("%d",&produce);

buffer[in]=produce;

in=(in+1)%bufsize;

}

break;

case 2:if(in==out)

printf("\n Buffer is Empty");

else

{

consume=buffer[out];

printf("\n The consumed value is%d",consume);

out=(out+1)%bufsize;

}

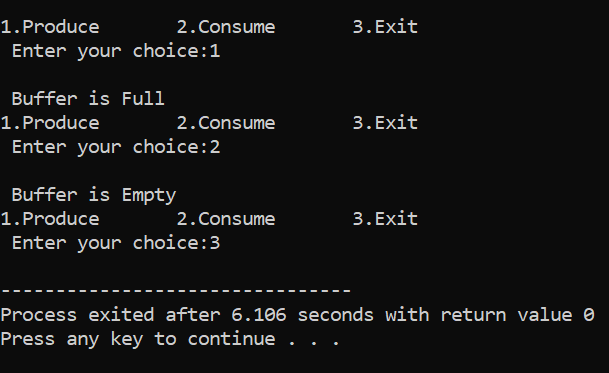
break;

}

}

}

**Output:**



**4) Develop a C program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.**

**5) Develop a C program to simulate Bankers Algorithm for DeadLock Avoidance.**

#include <stdio.h>

struct process

{

int allocation[3], max[3], need[3], finish;

} p[10];

int main()

{

int n, i, j, avail[3], work[3], flag, count = 0, sequence[10], k = 0;

printf("\n Enter the number of processes:");

scanf("%d", &n);

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

{

printf("\n Enter the %dth process allocated resources:", i);

scanf("%d%d%d", &p[i].allocation[0], &p[i].allocation[1], &p[i].allocation[2]);

printf("\n Enter the %dth process max resources:", i);

scanf("%d%d%d", &p[i].max[0], &p[i].max[1], &p[i].max[2]);

p[i].finish = 0;

p[i].need[0] = p[i].max[0] - p[i].allocation[0];

p[i].need[1] = p[i].max[1] - p[i].allocation[1]; // Corrected line

p[i].need[2] = p[i].max[2] - p[i].allocation[2]; // Corrected line

}

printf("\n Enter the available vector:");

scanf("%d%d%d", &avail[0], &avail[1], &avail[2]);

for (i = 0; i < 3; i++)

work[i] = avail[i];

while (count != n)

{

count = 0;

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

{

flag = 1;

if (p[i].finish == 0 &&

p[i].need[0] <= work[0] &&

p[i].need[1] <= work[1] &&

p[i].need[2] <= work[2])

{

for (j = 0; j < 3; j++)

work[j] += p[i].allocation[j];

p[i].finish = 1;

sequence[k++] = i;

flag = 0;

}

if (flag == 1)

count++;

}

}

count = 0;

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

if (p[i].finish == 1)

count++;

printf("\n The Safe Sequence is:");

if (count == n)

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

printf("%d\t", sequence[i]);

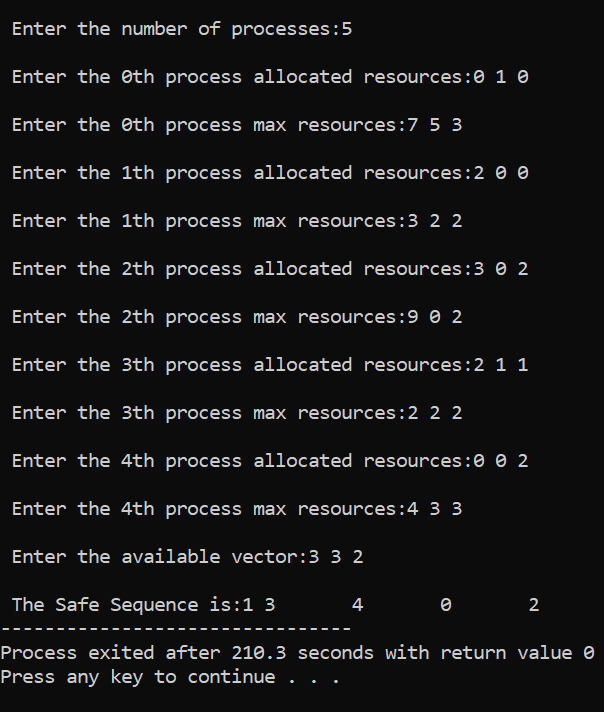
else

printf("System is not in a safe state\n\n");

return 0;

}

**Output:**



**6) Develop a C program to simulate the following contiguous memory allocation Techniques: a) Worst fit b) Best fit c) First fit**

#include<stdio.h>

#include<conio.h>

void main()

{

int frag[10],b[10],f[10],i,j,nb,nf,temp;

int bf[10],ff[10];

printf("\n\t ------Memory Mangement scheme First Fit---------");

printf("\n Enter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of files:");

scanf("%d",&nf);

printf("\n Enter the size of the blocks:\n");

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

{

printf("Block%d:",i);

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

}

printf("Enter the size of the files:\n");

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

{

printf("File%d:",i);

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

}

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

{

temp=-1;

for(j=1;j<=nb;j++)

{

if(bf[j]!=1)

{

if(b[j]>=f[i])

{

frag[i]=b[j]-f[i];

ff[i]=j;

temp=j;

bf[j]=1;

break;

}

}

}

if(temp==-1)

ff[i]=-1;

}

printf("\n File\_no:\tFile\_size:\tBlock\_no:\tBlock\_size:\tFragement:");

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

{

if(ff[i]!=-1)

{

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);

}

else

{

printf("\n%d\t\t%d\t\t",i,f[i]);

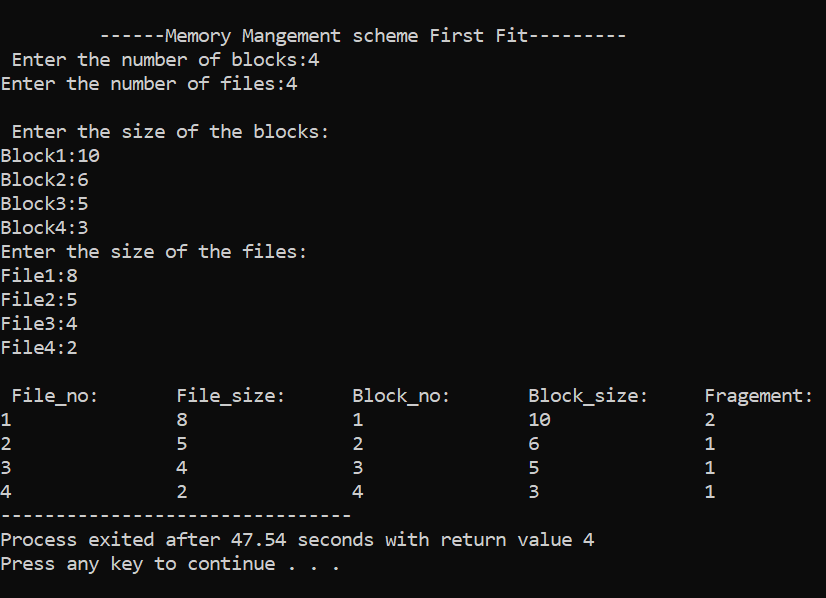
printf("not allocated\n");

}

}

}

**Output:**



**7) Develop a C program to simulate page replacement algorithms: a) FIFO b) LRU**

#include <stdio.h>

#include <stdlib.h>

void FIFO(char s[], char F[], int l, int f) {

int i, j = 0, k, flag;

printf("PAGE\tFRAMES\tFAULTS");

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

flag = 0;

for (k = 0; k < f; k++)

if (F[k] == s[i])

flag = 1;

printf("\n%c\t", s[i]);

if (flag == 0) {

F[j++] = s[i];

printf("%s", F);

printf("\tPage Fault");

} else {

printf("%s", F);

printf("\tPage Hit");

}

if (j == f)

j = 0;

}

}

void LRU(char s[], char F[], int l, int f) {

int i, j = 0, k, m, flag, top = 0;

printf("\nPAGE\tFRAMES\tFAULTS");

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

flag = 0;

for (k = 0; k < f; k++)

if (F[k] == s[i])

flag = 1;

printf("\n%c\t", s[i]);

if (j != f && flag != 1) {

F[top] = s[i];

if (++j != f)

top++;

} else {

if (flag != 1) {

for (k = 0; k < top; k++)

F[k] = F[k + 1];

F[top] = s[i];

} else {

for (m = k; m < top; m++)

F[m] = F[m + 1];

F[top] = s[i];

}

}

printf("%s", F);

if (flag == 0)

printf("\tPage Fault");

else

printf("\tPage Hit");

}

}

int main() {

int ch, i, l, f;

char F[10], s[25];

printf("Enter the number of frames: ");

scanf("%d", &f);

printf("Enter the length of the string: ");

scanf("%d", &l);

printf("Enter the string: ");

scanf("%s", s);

while (1) {

printf("\nEnter:\n1:FIFO\n2:LRU\n3:EXIT\n");

scanf("%d", &ch);

switch (ch) {

case 1:

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

F[i] = -1;

FIFO(s, F, l, f);

break;

case 2:

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

F[i] = -1;

LRU(s, F, l, f);

break;

case 3:

exit(0);

}

}

return 0;

}

**Output:**

