|  |  |  |  |
| --- | --- | --- | --- |
|  | **COLLEGE OF COMPUTING AND INFORMATION SCIENCES** | | |
| **Assignment # 01** | | |
| **Course Title** | Operating System | **Total Marks** | 10 |
| **Date** | 5-11-21 | **Class ID** | 108185 |
| **Student Id** | **11403** | **Student Name** | **Sumaiya Saleh** |

**Instructions:**

* Copied work and late submission will be marked as ZERO.
* Attach your code and screenshot of your output in this file.
* Submit hardcopy of your solution in class.

**Submission Deadline: 09-11-2021**

**Question 1:**

Write a program that allocate memory for array and print the elements of array (size and elements of array should be user defined), search the number form array using any searching algorithm.

Next, reallocate the size of array and print the array elements. For memory reallocation write down your own **realloc** function using **malloc** and **free** function.  
Suppose if your array size is of 5 indexes then the minimum memory required is of 20bytes. After reallocation you can increase or decrease the memory. If the memory size is incremented print 0 at new indexes or if the memory size is decremented, then remove the indexes from array.

Code:

#include <stdio.h>

#include <stdlib.h>

int main()

{

int size;

int \*ptr;

int total;

int temp;

int f;

int item;

printf("Size of the Array: ");

scanf("%d",&size);

ptr=(int\*)malloc(size\*sizeof(int));

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

{

printf("Element for %02d: ",i+1);

scanf("%d",(ptr+i));

}

printf("\n Search number from array: ");

scanf("%d",&item);

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

{

if(\*(ptr+i)==item)

{

f=1;

break;

}

}

if(f==1)

{

printf("Number is available");

}

else

{

printf("Number is not available!");

}

printf("\nArray contains :\n");

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

for (int j = i + 1; j < size; j++)

{

if (\*(ptr + j) < \*(ptr + i))

{

temp = \*(ptr + i);

\*(ptr + i) = \*(ptr + j);

\*(ptr + j) = temp;

}

}

}

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

{

printf("%d\n",\*(ptr+i));

}

printf("Reallocating memory size: ");

scanf("%d",&size);ptr = (int\*)realloc(ptr, size \* sizeof(int));

printf("After reallocation array contains : \n");

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

{

printf("%d\n",\*(ptr+i));

}

printf("Reallocating memory size again: ");

scanf("%d",&size);

ptr = (int\*)realloc(ptr, size \* sizeof(int));

//read array elements

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

{

printf("Element for %02d: ",i+1);

scanf("%d",(ptr+i));

}

printf("\nArray contains :\n");

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

for (int j = i + 1; j < size; j++) {

if (\*(ptr + j) < \*(ptr + i)) {

temp = \*(ptr + i);

\*(ptr + i) = \*(ptr + j);

\*(ptr + j) = temp;

}

}

}

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

{

printf("%d\n",\*(ptr+i));

}

printf("Reallocating memory size: ");

scanf("%d",&size);

ptr = (int\*)realloc(ptr, size \* sizeof(int));

printf("After reallocation array contains : \n");

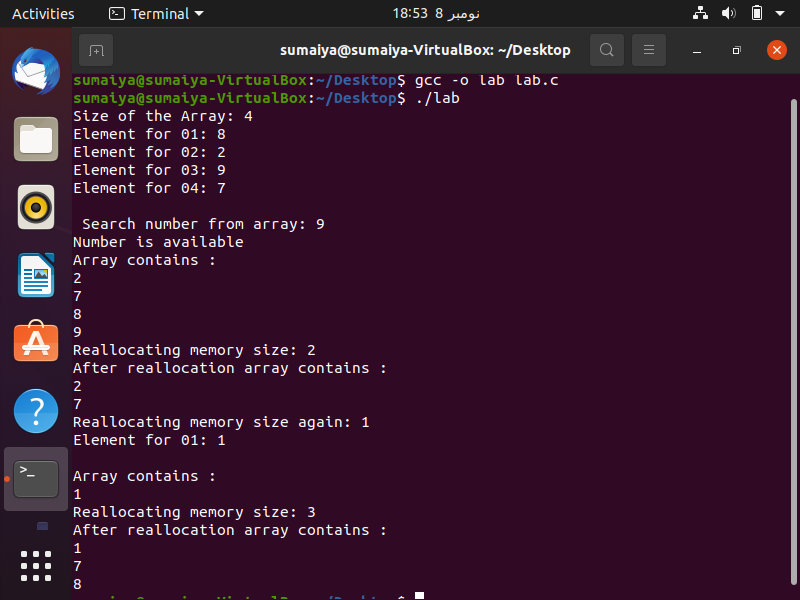
for(int i=0;i<size;i++)

{

printf("%d\n",\*(ptr+i));

}return 0;

}



Output:

**Question 2:**

Write a program that implements **Process Scheduling Algorithm** (FCFS, SJF – Preemptive , SJF – Non Preemptive). First your program need to ask what scheduling algorithm you need to execute. After selecting process scheduling algorithm, ask user to input Process ID, Arrival Time and Burst Time. In the end, Calculate Start Time, Completion Time, Waiting Time, Turnaround Time, Average Waiting Time and Average Turnaround Time.

Print all the values in the tabular form.

Code:

#include <stdio.h>

#include <string.h>

#define MAX 9999

struct proc

{

int no,at,bt,rt,ct,tat,wt;

};

struct proc read(int i)

{

struct proc p;

printf("\nNumber of process: %d\n",i);

p.no=i;

printf("Arrival Time: ");

scanf("%d",&p.at);

printf("Burst Time: ");

scanf("%d",&p.bt);

p.rt=p.bt;

return p;

}

int main()

{

printf(" Enter 11 for FCFS \n Enter 22 for PSA\n Enter 33 for SJF\n Enter 44 for SRTF\n");

int choose;

scanf("%d",&choose);

if(choose==11)

{

char pr\_num[10][10],t[10];

int array[10],br[10],st[10],ct[10],tat[10],wt[10],i,j,n,temp;

int t\_wt=0,t\_tat=0;

printf("<<<<FCFS Schedualing Algorithm>>>>\n");

printf("Number of processes:");

scanf("%d",&n);

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

{

printf("Enter Process Name, AT & BT: ");

scanf("%s%d%d",pr\_num[i],&array[i],&br[i]);

}

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

{

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

{

if(array[i]<array[j])

{

temp=array[i];

array[i]=array[j];

array[j]=temp;

temp=br[i];

br[i]=br[j];

br[j]=temp;

strcpy(t,pr\_num[i]);

strcpy(pr\_num[i],pr\_num[j]);

strcpy(pr\_num[j],t);

}

}

}

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

if(i==0)

st[i]=array[i];

else

st[i]=ct[i-1];

wt[i]=st[i]-array[i];

ct[i]=st[i]+br[i];

tat[i]=ct[i]-array[i];

}

printf("\nProcess Ar Br Wt St TAT Ct");

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

{

printf("\n%s\t%3d\t%3d\t%3d\t%3d\t%6d\t%6d",pr\_num[i],array[i],br[i],wt[i],st[i],tat[i],ct[i]);

t\_wt+=wt[i];

t\_tat+=tat[i];

}

printf("\nAverage Waiting time:%f",(float)t\_wt/n);

printf("\nAverage Turnaround Time:%f",(float)t\_tat/n);

}

else if(choose==22){

int pr\_number = 0;

int cr\_time = 0;

printf("<<<<Priority Schedualing Algorithm>>>>\n");

printf("Number of processes: ");

scanf("%d",&pr\_number);

int ar\_time[pr\_number];

int ATt[pr\_number];

int NoP = pr\_number;

int PT[pr\_number];

int ppr[pr\_number];

int PPt[pr\_number];

int wt[pr\_number];

int tat[pr\_number];

int i=0;

for(i=0 ;i<pr\_number ;i++){

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

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

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

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

PPt[i] = ppr[i];

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

scanf("%d",&ar\_time[i]);

ATt[i] = ar\_time[i];

}

int last\_arrival = 0;

for(i = 0; i < pr\_number; i++)

if(ar\_time[i] > last\_arrival)

last\_arrival = ar\_time[i];

int max\_priority = 0;

for(i = 0; i < pr\_number; i++)

if(PPt[i] > max\_priority)

max\_priority = PPt[i];

int ATi = 0;

int P1 = PPt[0];

int P2 = PPt[0];

int j = -1;

while(NoP > 0 && cr\_time <= 1000){

for(i = 0; i < pr\_number; i++){

if((ATt[i] <= cr\_time) && (ATt[i] != (last\_arrival+10))){

if(PPt[i] != (max\_priority+1)){

P2 = PPt[i];

j= 1;

if(P2 < P1){

j= 1;

ATi = i;

P1 = PPt[i];

P2 = PPt[i];

}}}}

if(j == -1){

cr\_time = cr\_time+1;

continue;

}else{

wt[ATi] = cr\_time - ATt[ATi];

cr\_time = cr\_time + PT[ATi];

tat[ATi] = cr\_time - ATt[ATi];

ATt[ATi] = last\_arrival +10;

j = -1;

PPt[ATi] = max\_priority + 1;

ATi = 0;

P1 = max\_priority+1;

P2 = max\_priority+1;

NoP = NoP - 1;

}

}

printf("\nPN\tPT\tPP\tAT\tWT\tTT\n");

for(i = 0; i < pr\_number; i++){

printf("P%d\t%d\t%d\t%d\t%d\t%d\n",i+1,PT[i],ppr[i],ar\_time[i],wt[i],tat[i]);

}

int Avg\_WT = 0;

int AVG\_TaT = 0;

for(i = 0; i < pr\_number; i++){

Avg\_WT = wt[i] + Avg\_WT;

AVG\_TaT = tat[i] + AVG\_TaT;

}

printf("Average Waiting Time = %d\nAverage Turnaround Time = %d\n",Avg\_WT/pr\_number,AVG\_TaT/pr\_number);

}

else if(choose==33) {

int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0;

int bt[10],temp,j,at[10],wt[10],tt[10],ta=0,sum=0;

float wavg=0,tavg=0,tsum=0,wsum=0;

printf("<<<<Shortest Job First Scheduling>>>>\n");

printf("\nNumber of processes :");

scanf("%d",&n);

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

printf("\tBT of %d process :",i+1);

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

printf("\tAT of %d process :",i+1);

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

}

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

{

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

{

if(at[i]<at[j])

{

temp=p[j];

p[j]=p[i];

p[i]=temp;

temp=at[j];

at[j]=at[i];

at[i]=temp;

temp=bt[j];

bt[j]=bt[i];

bt[i]=temp;

}

}

}

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

btime=btime+bt[j];

min=bt[k];

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

if (btime>=at[i] && bt[i]<min){

temp=p[k];

p[k]=p[i];

p[i]=temp;

temp=at[k];

at[k]=at[i];

at[i]=temp;

temp=bt[k];

bt[k]=bt[i];

bt[i]=temp;

}

}

k++;

}

wt[0]=0;

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

sum=sum+bt[i-1];

wt[i]=sum-at[i];

wsum=wsum+wt[i];

}

wavg=(wsum/n);

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

ta=ta+bt[i];

tt[i]=ta-at[i];

tsum=tsum+tt[i];

}

tavg=(tsum/n);

printf("\n");

printf("\nProcess\t Ar\t Br\t Wt\t TAT" );

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

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

}

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

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

}

else if(choose==44){

struct proc p[10],temp;

float avgtat=0,avgwt=0;

int n,s,remain=0,time;

printf("<<<<Shortest Remaining Time First Scheduling Algorithm>>>>\n");

printf("Number of Processes: ");

scanf("%d",&n);

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

p[i]=read(i+1);

for(int i=0;i<n-1;i++)

for(int j=0;j<n-i-1;j++)

if(p[j].at>p[j+1].at) {

temp=p[j];

p[j]=p[j+1];

p[j+1]=temp;

}

printf("\nPr\t\tAT\tBT\tCT\tTAT\tWT\n");

p[9].rt=MAX;

for(time=0;remain!=n;time++){

s=9;

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

if(p[i].at<=time&&p[i].rt<p[s].rt&&p[i].rt>0)

s=i;

p[s].rt--;

if(p[s].rt==0){

remain++;

p[s].ct=time+1;

p[s].tat=p[s].ct-p[s].at;

avgtat+=p[s].tat;

p[s].wt=p[s].tat-p[s].bt;

avgwt+=p[s].wt;

printf("P%d\t\t%d\t%d\t%d\t%d\t%d\n",p[s].no,p[s].at,p[s].bt,p[s].ct,p[s].tat,p[s].wt);

}

}

avgtat/=n,avgwt/=n;

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

}

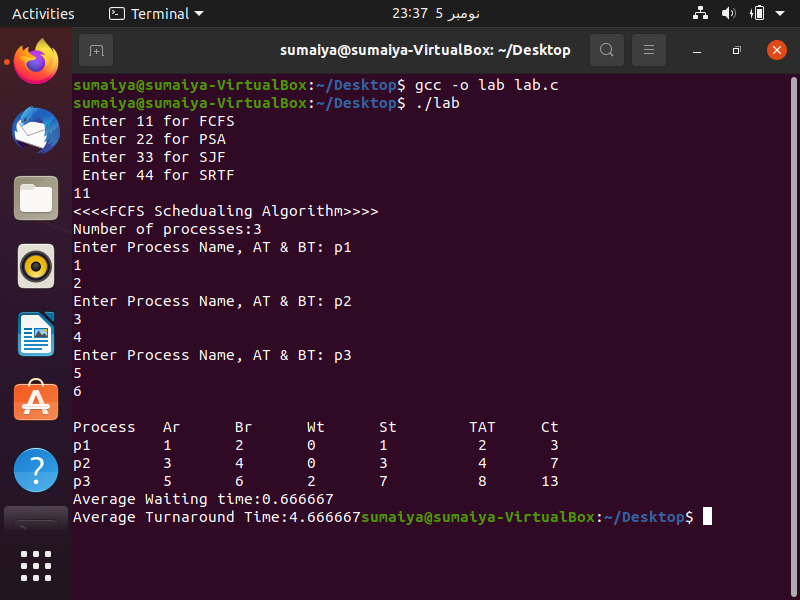
else {

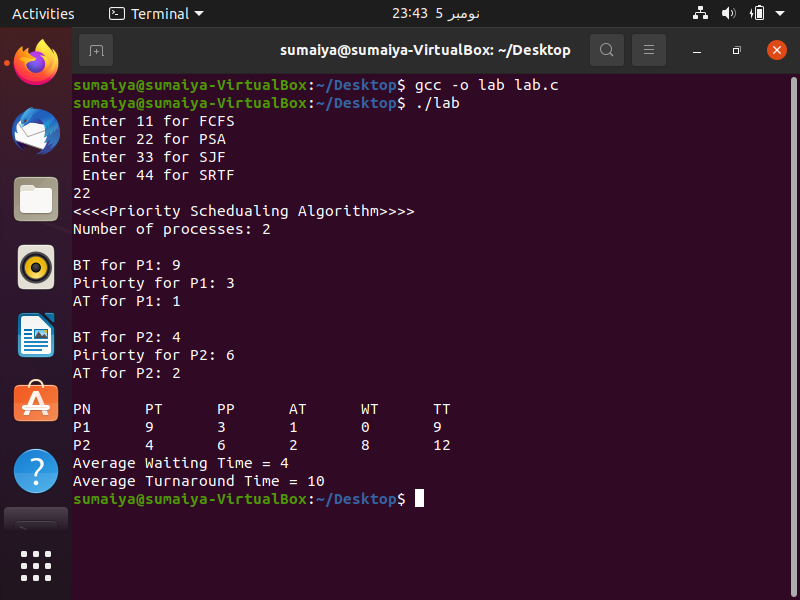
printf("Incorrect number");

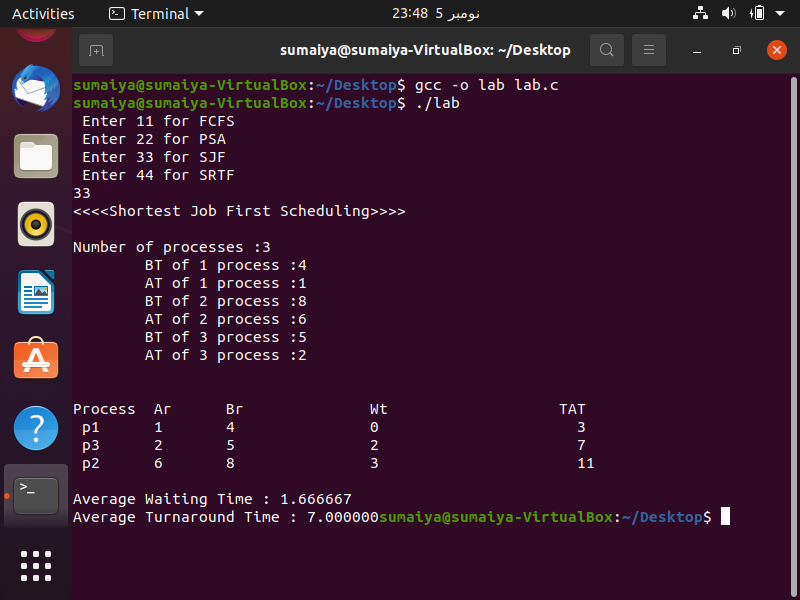
}

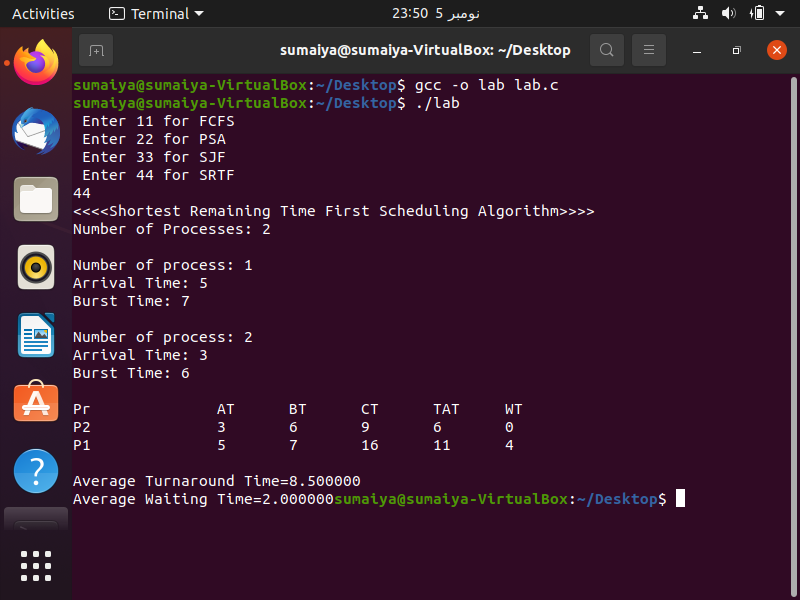
return 0;

Output:

****

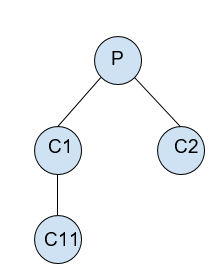
****

****

****

**Question 3:**

Implement the C program in which we create process hierarchy using **fork** as given below:



* Parent Process**(P)** will accepts the integers from the user to perform the operations.
* First child**(C1)** of parent will perform addition.
* Child **(C11)** will perform subtraction.
* Second child**(C2)** of parent will perform multiplication.

You also need to print the process ID along with their parent process ID.

**Sample Input:**

Parent Process 6741:  
 Enter number 1 to perform operations: 10  
 Enter number 2 to perform operations: 2

**Sample Output:**

C1 process 6742, parent 6741 performs addition: 12  
 C2 process 6743, parent 6741 performs subtraction: 8  
 C11 process 6744, parent 6742 performs multiplication: 20

Code:

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int main()

{

int val1;

int val2;

printf("Enter First Number: ");

scanf("%d", &val1);

printf("Enter Second Number: ");

scanf("%d",&val2);

pid\_t C\_a,C\_b,C\_a1,C\_b1;

printf("\n Parent Process with ID %d \n",getpid());

printf("First number: %d\n",val1);

printf("Second number: %d\n",val2);

C\_a=fork();

if (C\_a == 0 ) {

printf("\nChild C1 with PID %d and Parent ID is %d\n",getpid(),getppid());

int c = val1+val2;

printf("Add: %d\n",c);

C\_a1=fork();

if (C\_a1 == 0 ){

printf("\nChild C11 with PID %d and Parent ID is %d\n",getpid(),getppid());

int e = val1\*val2;

printf("Multiply: %d\n",e);

} }

else{

C\_b = fork();

if (C\_b == 0)

{

printf("\nChild C2 with ID %d and Parent ID is %d\n",getpid(),getppid());

int d = val1-val2;

printf("Subtract: %d\n",d);

C\_b1=fork();

if (C\_b1 == 0 ){

printf("\nChild C21 with PID %d and Parent ID is %d\n",getpid(),getppid());

int f = val1/val2;

printf("Divide: %d\n",f);

} }

else{

sleep(1);

}

}

}

Output:

