BUBT | BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Committed to Academic Excellence

Lab Assignment-2

Course Code: CSE 310

Course Title: Operating Systems Lab

Submitted to:

Name: Suman Saha

Assistant Professor

Dept. of CSE

at Bangladesh University of Business and Technology.

Submitted by:

Name: Syeda Nowshin Ibnat

ID: 17183103020

Intake: 39

Section: 01

Program: B.Sc. in CSE

Date of Submission: 16.01.2021

Q1 Solution

```
#!/bin/bash
for (( i=1; i<=50; i++ ))
do
mkdir $i
flag=true
if (( $i==1 ))
then
flag=false
fi
for (( j=2; j<=i/2; j++ ))
do
if (( $i%$j == 0 ))
then
flag=false
fi
done
if [ "$flag" = true ] ;
then
rm -r $i
fi
done
```

Q2 Solution

```
#include <stdio.h>
#include <unistd.h>
int main() {
int p_id,p_pid;
for(int i=0;i<10;i++)
p_id=getpid();  /*process id*/
int flag = 1;
for(int j=2;j<=p_id/2;j++){
if(p_id%j==0){
flag=0;
break; } }
if(flag==1){
printf("Process ID: %d\n",p_id); } }
return 0; }
```

Q3 Solution

```
# !/bin/bash
echo -n "Enter the Total numbers :"
read n
echo "Enter numbers :"
i=0
while [ $i -lt $n ]
do
```

```
# To take input from user
read a[$i]
# Increment the i = i + 1
i=`expr $i + 1`
done
echo "Output :"
i=0
sum=0
while [ $i -lt $n ]
do
if (( i%2==1 ))
then
sum=`expr $sum + ${a[$i]}`
fi
# To increment index
# by 1, i=i+1
i=`expr $i + 1`
done
echo $sum
```

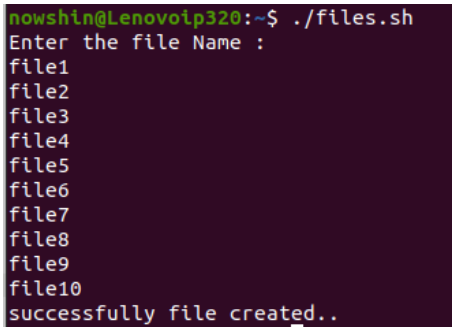
Q4 Solution

Source code:

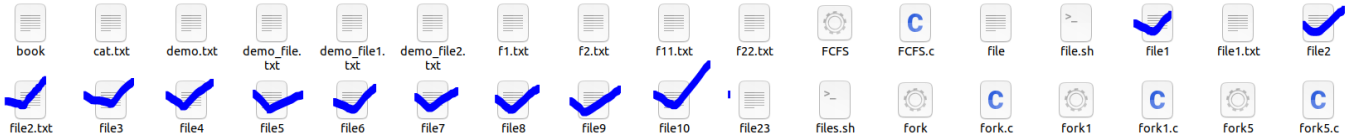
```
# !/bin/bash
echo "Enter the file Name :"
n=10
```

```
i=0
while [ $i -lt $n ]
do
read a[$i] # To take input from user
i=`expr $i + 1` # Increment the i = i + 1
done
echo "successfully file created.."
i=0
sum=0
while [ $i -lt $n ]
do
touch ${a[$i]}
# To increment index
# by 1, i=i+1
i=`expr $i + 1`
done
```

Output:



File created:



Q5 Solution

Source code:

```
#include <stdio.h>
#include  <sys/types.h>
int arr[5];
int n=5;
void insertionSort(){
for(int j=1; j<n; j++){
int key = arr[j];
int i= j-1;
while(i>=0 && arr[i] > key){
arr[i+1] = arr[i];
i--;    }
arr[i+1] = key;   }
return; }
void parentProcess(){
printf("\n--------Starting of the parent process--------\n");
printf("Printing the odd elements: ");
for(int i=0; i<n; i++){
if(arr[i] %2){
printf("%d ", arr[i]); } }
printf("\n--------Ending of the parent process--------\n\n\n"); }
void childProcess(){
```

```
printf("\n--------Starting of the child process--------\n");
printf("Sorting the arry...\n");
insertionSort();
printf("Printing the array: ");
for(int i=0; i<n; i++){
printf("%d ", arr[i]); }
printf("\n--------Ending of the child process--------\n"); }
int main(){
pid_t pid;
printf("Taking input for the array: ");
for(int i=0; i<n; i++){
scanf("%d", &arr[i]); }
// calling fork();
pid = fork();
if(pid == 0){ // calling parent process
childProcess();
}else{
parentProcess();}
return 0; }
```

Output:

```
nowshin@Lenovoip320:~$ gcc odd_sort.c -o odd_sort
odd_sort.c: In function ‘main’:
odd_sort.c:59:8: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
   59 |  pid = fork();
      |        ^~~~
nowshin@Lenovoip320:~$ ./odd_sort
Taking input for the array: 1 2 3 6 3

--------Starting of the parent process--------
Printing the odd elements: 1 3 3
--------Ending of the parent process--------


nowshin@Lenovoip320:~$
--------Starting of the child process--------
Sorting the arry...
Printing the array: 1 2 3 3 6
--------Ending of the child process--------
```

Q6 Solution

Source code:

```
#include<stdio.h>

void main() {

int p[20],bt[20],wt[20],tat[20],i,j,n,total=0,pos,temp; // p=process, bt=brust time, wt=waiting time, tat= turn around time.

float avg_wt,avg_tat;       // average waiting time, average turn around time.

printf("Enter number of process:");

scanf("%d",&n);

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

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

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

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

p[i]=i+1;     }      //contains process number

//sorting burst time in ascending order using selection sort

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;            //waiting time for first process will be zero

//calculate waiting time
for(i=1;i<n;i++) {
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i]; }
avg_wt=(float)total/n;      //average waiting time
total=0;
printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++) {
tat[i]=bt[i]+wt[i];     //calculate turnaround time
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=(float)total/n;     //average turnaround time
printf("\n\nAverage Waiting Time=%.2f",avg_wt);
printf("\nAverage Turnaround Time=%.2f\n",avg_tat); }
```

Output:

```
nowshin@Lenovoip320:~$ gcc p_SJF.c -o p_SJF
nowshin@Lenovoip320:~$ ./p_SJF
Enter number of process:4

Enter Burst Time:
p1:12
p2:4
p3:6
p4:5

Process     Burst Time          Waiting Time    Turnaround Time
p2                4                 0                   4
p4                5                 4                   9
p3                6                 9                   15
p1                12                15                  27

Average Waiting Time=7.00
Average Turnaround Time=13.75
```