19BIT0292
Bhaumik Tandan

ASSESMENT-1
OPERATING SYSTEM
Laboratory

Q1. Basic Linux commands :

mkdir -> create new directory

cd -> Change to new directory

rmdir-> remove empty directory

pwd-> show current directory

```
bhaum@DESKTOP-QIJQUE1 ~
$ mkdir q19BIT0292

bhaum@DESKTOP-QIJQUE1 ~
$ cd q19BIT0292

bhaum@DESKTOP-QIJQUE1 ~/q19BIT0292
$ cd ..

bhaum@DESKTOP-QIJQUE1 ~
$ rmdir q19BIT0292

bhaum@DESKTOP-QIJQUE1 ~
$ pwd
/home/bhaum
```

Check that rmdir on remove empty directory

ls-> list files in a directory and their attributes

```
bhaum@DESKTOP-QIJQUE1 ~/scheule
$ ls
fifo.c  fifo.exe  roundrobin.c  sjfs.c  sjfs.exe  te.c  te.exe

bhaum@DESKTOP-QIJQUE1 ~/scheule
$ cd ..

bhaum@DESKTOP-QIJQUE1 ~
$ rmdir scheule
rmdir: failed to remove 'scheule': Directory not empty
```

history-> list of previously executed commands

date-> show date and time

```
bhaum@DESKTOP-QIJQUE1 ~
$ date
Mon Mar 15 21:41:00 IST 2021

bhaum@DESKTOP-QIJQUE1 ~
$ history
    1  gcc
    2  g++
    3  gcc
    4  g++
    5  g++
    6  ls
    7  code dr
    8  mkdir scheule
    9  ls
   10  cd scheule
   11  code .
   12  code .
   13  ls
   14  cd scheule
   15  code .
   16  code .
   17  cd schedular
   18  cd scedule
   19  ls
   20  cd scheule
   21  code.
   22  code .
   23  code .
   24  ls
   25  cd scheule
   26  code .
   27  code .
   28  mkdir daq19BIT0292
   29  lw
```

clear-> This command clears all the clutter on the terminal and gives you a clean window to work

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ cd ..

bhaum@DESKTOP-QIJQUE1 ~/da
$ cd ..

bhaum@DESKTOP-QIJQUE1 ~
$ clear
```

```
bhaum@DESKTOP-QIJQUE1 ~
$
```

cal month year -> Prints a calendar for the specified month of the specified year

```
bhaum@DESKTOP-QIJQUE1 ~
$ cal 2023
                                   2023

       January                February                March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7              1  2  3  4              1  2  3  4
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    5  6  7  8  9 10 11
15 16 17 18 19 20 21   12 13 14 15 16 17 18   12 13 14 15 16 17 18
22 23 24 25 26 27 28   19 20 21 22 23 24 25   19 20 21 22 23 24 25
29 30 31               26 27 28               26 27 28 29 30 31

        April                  May                   June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                    1       1  2  3  4  5  6                 1  2  3
 2  3  4  5  6  7  8    7  8  9 10 11 12 13    4  5  6  7  8  9 10
 9 10 11 12 13 14 15   14 15 16 17 18 19 20   11 12 13 14 15 16 17
16 17 18 19 20 21 22   21 22 23 24 25 26 27   18 19 20 21 22 23 24
23 24 25 26 27 28 29   28 29 30 31            25 26 27 28 29 30
30
        July                  August                September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                    1          1  2  3  4  5                    1  2
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    3  4  5  6  7  8  9
 9 10 11 12 13 14 15   13 14 15 16 17 18 19   10 11 12 13 14 15 16
16 17 18 19 20 21 22   20 21 22 23 24 25 26   17 18 19 20 21 22 23
23 24 25 26 27 28 29   27 28 29 30 31         24 25 26 27 28 29 30
30 31
       October               November               December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7              1  2  3  4                    1  2
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    3  4  5  6  7  8  9
15 16 17 18 19 20 21   12 13 14 15 16 17 18   10 11 12 13 14 15 16
22 23 24 25 26 27 28   19 20 21 22 23 24 25   17 18 19 20 21 22 23
29 30 31               26 27 28 29 30         24 25 26 27 28 29 30
                                              31
```

```
bhaum@DESKTOP-QIJQUE1 ~
$ cal
     March 2021
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

```
bhaum@DESKTOP-QIJQUE1 ~
$ cal Feb
    February 2021
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28
```

```
bhaum@DESKTOP-QIJQUE1 ~
$ cal Feb 2023
    February 2023
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28
```

tty-> know the terminal name

uname-> know the terminal name

ps-> process status

cat-> view files

head-> show first few lines of a file(s)

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ tty
/dev/pty1

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ uname
CYGWIN_NT-10.0
```

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ cat 19BIT0292q1.sh
echo 'Enter the numbers: '
read a b c
if [ $a -gt $b ]
then
if [ $c -lt $a ]
then echo $a
else
echo $c
fi
else
if [[ $b>$c ]]
then echo $b
else
echo 'Greatech number is: $c'
fi
fi
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ ps
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
      703     610     703       6924  pty1      197609 22:03:18 /usr/bin/ps
      610     609     610      13380  pty1      197609 21:44:42 /usr/bin/bash
      609       1     609       4912  ?         197609 21:44:42 /usr/bin/mintty

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ head 19BIT0292q1.sh
echo 'Enter the numbers: '
read a b c
if [ $a -gt $b ]
then
if [ $c -lt $a ]
then echo $a
else
echo $c
fi
else
```

cp> copy files

rm-> remove files

kill-> kill background job or previous process…

echo $$-> process id of current shell.

~/da

```
bhaum@DESKTOP
$ echo $$
610

bhaum@DESKTOP
$ kill 610
```

~

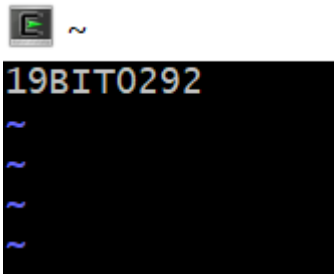
```
bhaum@DESKTOP-QIJQUE1 ~
$ ls
a.txt  da  scheule

bhaum@DESKTOP-QIJQUE1 ~
$ vi a.txt

bhaum@DESKTOP-QIJQUE1 ~
$ cp a.txt da

bhaum@DESKTOP-QIJQUE1 ~
$ cd da

bhaum@DESKTOP-QIJQUE1 ~/da
$ ls
a.txt  da

bhaum@DESKTOP-QIJQUE1 ~/da
$ rm a.txt

bhaum@DESKTOP-QIJQUE1 ~/da
$ ls
da

bhaum@DESKTOP-QIJQUE1 ~/da
$ cd ..

bhaum@DESKTOP-QIJQUE1 ~
$ ls
a.txt  da  scheule

bhaum@DESKTOP-QIJQUE1 ~
$ mv a.txt da

bhaum@DESKTOP-QIJQUE1 ~
$ ls
da  scheule
```

~/da

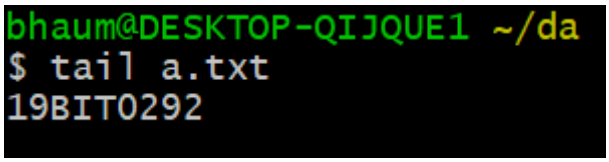
```
bhaum@DESKTOP-QIJQUE1 ~/da
$ ls
a.txt  da

bhaum@DESKTOP-QIJQUE1 ~/da
$ mv a.txt b.txt

bhaum@DESKTOP-QIJQUE1 ~/da
$ ls
b.txt  da
```

mv-> change file name or directory location

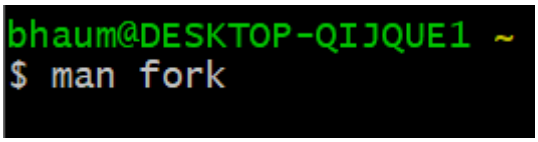
vi-> full-featured screen editor for modifying text files



tail-> show last few lines of a file; or reverse line order



man-> show online documentation by program name



```
FORK(3am)                    GNU Awk Extension Modules                    FORK(3am)

NAME
       fork, wait, waitpid - basic process management

SYNOPSIS
       @load "fork"

       pid = fork()

       ret = waitpid(pid)

       ret = wait();

DESCRIPTION
       The fork extension adds three functions, as follows.

       fork() This  function  creates  a  new process. The return value is the
              zero in the child and the process-id number of the child in  the
              parent,  or  -1  upon error. In the latter case, ERRNO indicates
              the problem.  In the child, PROCINFO["pid"] and PROCINFO["ppid"]
              are updated to reflect the correct values.
```

Q2. Shell Programming

a. Find the smallest of three numbers

CODE

```
echo 'Enter the numbers: '
```

```
read a b c
if [ $a -gt $b ]
then
if [ $c -lt $a ]
then echo $a
else
echo $c
fi
else
if [[ $b>$c ]]
then echo $b
else
echo 'Greatech number is: $c'
fi
fi
```

~/da/da

```
echo 'Enter the numbers: '
read a b c
if [ $a -gt $b ]
then
if [ $c -lt $a ]
then echo $a
else
echo $c
fi
else
if [[ $b>$c ]]
then echo $b
else
echo 'Greatech number is: $c'
fi
fi
```

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q1.sh
Enter the numbers:
3 4 2
4

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q1.sh
Enter the numbers:
9 -5 4
9
```

b. Swapping of two numbers without using third variable

CODE

```
echo 'Enter the numbers: '
read a b
```

~/da/da

```
echo 'Enter the numbers: '
read a b
a=$((a+b))
b=$((a-b))
a=$((a-b))
echo $a $b
```

```
a=$((a+b))

b=$((a-b))

a=$((a-b))

echo $a $b
```

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ vi 19BIT0292q2.sh

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q2.sh
Enter the numbers:
7 5
5 7
```

c. Check the grade of the students based on marks using elif

CODE

```
read a

if [ $a -gt 90 ]

then

echo 'S'

elif [ $a -gt 80 ]

then

echo 'A'

elif [ $a -gt 70 ]

then
```

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ vi 19BIT0292q3.sh

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q3.sh
55
E

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q3.sh
32
F

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q3.sh
90
A

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q3.sh
91
S

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q3.sh
75
B
```

```
echo 'B'
elif [ $a -gt 60 ]
then
echo 'C'
elif [ $a -gt 55 ]
then
echo 'D'
elif [ $a -gt 50 ]
then
echo 'E'
else
echo 'F'
fi
```

d. Perform basic arithmetic operations based on user choice

CODE

```
read a b
echo 'Enter the operation to be performed
```

```
(+,-,*,/,%)'
read o
case "$o" in
'+') echo $((a+b))
;;
'-') echo $((a-b))
;;
'*') echo $((a*b))
;;
'/') echo $((a/b))
;;
'%') echo $((a%b))
;;

esac
```

~/da/da

```
read a b
echo 'Enter the operation to be performed (+,-,*,/,%)'
read o
case "$o" in
'+') echo $((a+b))
;;
'-') echo $((a-b))
;;
'*') echo $((a*b))
;;
'/') echo $((a/b))
;;
'%') echo $((a%b))
;;
esac
```

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q4.sh
4 6
Enter the operation to be performed (+,-,*,/,%)
+
10

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q4.sh
8 5
Enter the operation to be performed (+,-,*,/,%)
-
3

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q4.sh
5 2
Enter the operation to be performed (+,-,*,/,%)
*
10

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q4.sh
6 2
Enter the operation to be performed (+,-,*,/,%)
/
3

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q4.sh
5 9
Enter the operation to be performed (+,-,*,/,%)
%
5
```

e. Find the sum of first n natural numbers

CODE

```
read n

echo $(((n*(n+1))/2))
```

~/da/da

```
read n
echo $(((n*(n+1))/2))
```

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ vi 19BIT0292q5.sh

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q5.sh
5
15

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q5.sh
9
45
```

f. Find the sum of first n natural numbers

CODE

```
read n

s=0

p=1

i=0

while [ $i -lt $n ]

do

  echo $p
```

~/da/da

```
read n
s=0
p=1
i=0
while [ $i -lt $n ]
do
  echo $p
  t=$p
  p=$((s+p))
  s=$t
  i=`expr $i + 1`
done
```

```
    t=$p
    p=$((s+p))
    s=$t
    i=`expr $i + 1`
done
```

~/da/da

```
bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q6.sh
6
1
1
2
3
5
8

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q6.sh
10
1
1
2
3
5
8
13
21
34
55

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q6.sh
5
1
1
2
3
5

bhaum@DESKTOP-QIJQUE1 ~/da/da
$ sh 19BIT0292q6.sh
3
1
1
2
```

CLICK HERE

FOR

GITHUB LINK

FOR

SHELL PROGRAMING

Q3. Process Creation

CREATING A CHID PROCESS

CODE:

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
void main()
{
 int a=0;
 int b=fork();
 if(b==0)
 printf("\nMy pid = %d and my parent pid= %d and value of a=\
n",getpid(),getppid(),++a);
 //The above line will never print the value of a because in child
process a is missing
 //fork only create child for succeding lines
 printf("\nMy pid=%d and a=%d\n",getpid(),++a);
}
```

~/procr

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
void main()
{
 int a=0;
 int b=fork();
 if(b==0)
 printf("\nMy pid = %d and my parent pid= %d and value of a=\n",getpid(),getppid(),++a);
 //The above line will never print the value of a because in child process a is missing
 //fork only create child for succeding lines
 printf("\nMy pid=%d and a=%d\n",getpid(),++a);
}
```

~/procr

```
bhaum@DESKTOP-QIJQUE1 ~/procr
$ vi 1.c

bhaum@DESKTOP-QIJQUE1 ~/procr
$ gcc 1.c -o ex.exe
1.c: In function ‘main’:
1.c:7:8: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
    7 |  int b=fork();
      |        ^~~~
1.c:9:65: warning: implicit declaration of function ‘getpid’ [-Wimplicit-function-declaration]
    9 |  printf("\nMy pid = %d and my parent pid= %d and value of a=\n",getpid(),getppid(),++a);
      |                                                                 ^~~~~~
1.c:9:74: warning: implicit declaration of function ‘getppid’ [-Wimplicit-function-declaration]
    9 |  printf("\nMy pid = %d and my parent pid= %d and value of a=\n",getpid(),getppid(),++a);
      |                                                                          ^~~~~~~

bhaum@DESKTOP-QIJQUE1 ~/procr
$ ./ex.exe


My pid=1336 and a=1
My pid = 1337 and my parent pid= 1336 and value of a=

My pid=1337 and a=2

bhaum@DESKTOP-QIJQUE1 ~/procr
$ |
```

PERFROMING DIFFERENT OPERATIONS

CODE:

```
#include<stdio.h>
#include <unistd.h>
#include<string.h>


void main()
{

 int p=fork();
 int c=fork();


//Child tree
```

```
// parent
// |_____________
// |             |________
// |             |        |
// |______       |       p=0
// |      |      |       c=0
// |      |      |
// |      |      p=0
// |     p>0     c>0
// p>0   c=0
// c>0

int a=5, b=-9;

 if(p>0 && c>0) //PARENT process IS BEING EXECUTED.
 {
 printf(" THE ADDITION IS : %d\n",a+b);
 }
 else if(p>=0 && c==0) //FIRST CHILD IS BEING EXECUTED.
 {
 printf(" THE SUBSTRACTION IS : %d\n",a-b);
 }
 else if(p==0 && c>0) //SECOND CHILD IS BEIGN EXECUTED.
 {
 printf(" THE MULTIPLICATION IS : %d\n",a*b);
 }
 else if(p==0 && c==0) //THIRD CHILD IS BEING EXECUTED.
 {
 printf(" THE DIVISION IS :
%d\n",a/b);
 }
}
```

```
bhaum@DESKTOP-QIJQUE1 ~/procr
$ vi 2.c

bhaum@DESKTOP-QIJQUE1 ~/procr
$ gcc 2.c -o ex.exe

bhaum@DESKTOP-QIJQUE1 ~/procr
$ ./ex.exe
 THE ADDITION IS : -4
 THE SUBSTRACTION IS : 14
 THE MULTIPLICATION IS : -45
 THE SUBSTRACTION IS : 14
```

~/procr

```
#include<stdio.h>
#include <unistd.h>
#include<string.h>


void main()
{

 int p=fork();
 int c=fork();


//Child tree
// parent
// |_____________
// |              |________
// |              |        |
// |______        |       p=0
// |      |       |       c=0
// |      |       |
// |      |      p=0
// |     p>0     c>0
// p>0   c=0
// c>0

int a=5, b=-9;

 if(p>0 && c>0) //PARENT process IS BEING EXECUTED.
 {
 printf(" THE ADDITION IS : %d\n",a+b);
 }
 else if(p>=0 && c==0) //FIRST CHILD IS BEING EXECUTED.
 {
 printf(" THE SUBSTRACTION IS : %d\n",a-b);
 }
 else if(p==0 && c>0) //SECOND CHILD IS BEIGN EXECUTED.
 {
 printf(" THE MULTIPLICATION IS : %d\n",a*b);
 }
 else if(p==0 && c==0) //THIRD CHILD IS BEING EXECUTED.
 {
 printf(" THE DIVISION IS : %d\n",a/b);
 }
}
~
~
~
~
"2.c" [dos format] 44 lines, 830 characters
```

ORPHAN CHILD

CODE:

```
#include<stdio.h>
#include <unistd.h>
#include<string.h>


void main()
{

 int p=fork();
 if(p==0)
 {
    sleep(3);
    printf("My pid is %d and parent pid is %d",getpid(),getppid());
    //here ppid will be different because prant will get terminated and child is "orphan"
 }
 else
 printf("I am parent My pid is %d",getpid());
 //above statement will print true parent pid
```

```
bhaum@DESKTOP-QIJQUE1 ~/procr
$ vi 3.c

bhaum@DESKTOP-QIJQUE1 ~/procr
$ gcc 3.c -o ex.exe

bhaum@DESKTOP-QIJQUE1 ~/procr
$ ./ex.exe
I am parent My pid is 1375
bhaum@DESKTOP-QIJQUE1 ~/procr
$ My pid is 1376 and parent pid is 1
```

}

~/procr

```
#include<stdio.h>
#include <unistd.h>
#include<string.h>


void main()
{

 int p=fork();
 if(p==0)
 {
     sleep(3);
     printf("My pid is %d and parent pid is %d",getpid(),getppid());
     //here ppid will be different because prant will get terminated and child is "orphan"
 }
 else
 printf("I am parent My pid is %d",getpid());
 //above statement will print true parent pid
}
```

ZOMBIE CHILD

CODE:

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
void main()
{
 pid_t c =fork();

if(c==0)
{
 printf("\nMy pid = %d and my parent pid= %d\n",getpid(),getppid());
}
 else{
    sleep(15);//here parent will sleep for 15 sec while paent will get executed
    printf("\nMy pid=%d\n",getpid());
```

```
  }
}
```

~/procr

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
void main()
{
 pid_t c =fork();

if(c==0)
{
 printf("\nMy pid = %d and my parent pid= %d\n",getpid(),getppid());
}
 else{
     sleep(15);//here parent will sleep for 15 sec while paent will get executed
     printf("\nMy pid=%d\n",getpid());
 }
}
```

```
bhaum@DESKTOP-QIJQUE1 ~/procr
$ vi 4.c

bhaum@DESKTOP-QIJQUE1 ~/procr
$ gcc 4.c -o ex.exe
4.c: In function 'main':
4.c:6:11: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
    6 |  pid_t c =fork();
      |           ^~~~
4.c:10:49: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
   10 |  printf("\nMy pid = %d and my parent pid= %d\n",getpid(),getppid());
      |                                                  ^~~~~~
4.c:10:58: warning: implicit declaration of function 'getppid' [-Wimplicit-function-declaration]
   10 |  printf("\nMy pid = %d and my parent pid= %d\n",getpid(),getppid());
      |                                                            ^~~~~~~
4.c:13:6: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
   13 |      sleep(15);//here parent will sleep for 15 sec while paent will get executed
      |      ^~~~~

bhaum@DESKTOP-QIJQUE1 ~/procr
$ ./ex.exe

My pid = 1417 and my parent pid= 1416
```

AFTER 15 SECONDS

```
bhaum@DESKTOP-QIJQUE1 ~/procr
$ ./ex.exe

My pid = 1417 and my parent pid= 1416

My pid=1416
```

CLICK HERE FOR GITHUB LINK OF

PROCESS CREATION

SCHEDULING ALGORITHMS

NOTE: This table is used in all the algorithms, in case arrival time or priority not required given 3rd and 4th column can be ignored respectively.

19BIT0292

File | D:/web_dev/ostable.html

Process Number	Burst Time	Arrival Time	Priority
1	5	3	2
2	8	2	4
3	14	1	3
4	8	2	5
5	4	0	1

Q4. First-Come, First-Served Scheduling

ARRIVAL TIME NOT GIVEN

CODE:

```
#include<stdio.h>
#include <stdlib.h>
int main()
{
```

```
 int *b,n,i,j,w=0,tr=0,br=0;
 printf("(19BIT0292) Enter the number of processes: ");
 scanf("%d",&n);
 b=(int*)malloc(n*sizeof(int));
 for(i=0;i<n;i++)
    {
        printf("Enter the burst time for process %d: ",i+1);
        scanf("%d",b+i);
        br+=b[i];
    }
  printf("\n    Process No.\t  Burst Time\tWaiting Time\t  Turn
around Time\n");
   for(j=0;j<n;j++)
   {
       printf("\t%d\t\t%d\t\t%d\t\t%d\n",j+1,b[j],w,w+b[j]);
       tr+=w;
       w+=b[j];
   }
   printf("\nAverage waiting time: %f\n",(tr*0.1)/(n*0.1));
   printf("\nAverage turnarround time:
%f\n",((tr+br)*0.1)/(n*0.1));
}
```

~/scheule

```c
#include<stdio.h>
#include <stdlib.h>


int main()
{
 int *b,n,i,j,w=0,tr=0,br=0;
 printf("(19BIT0292) Enter the number of processes: ");
 scanf("%d",&n);
 b=(int*)malloc(n*sizeof(int));
 for(i=0;i<n;i++)
    {
        printf("Enter the burst time for process %d: ",i+1);
        scanf("%d",b+i);
        br+=b[i];
    }

  printf("\n    Process No.\t  Burst Time\tWaiting Time\t  Turn around Time\n");
   for(j=0;j<n;j++)
   {
       printf("\t%d\t\t%d\t\t%d\t\t%d\n",j+1,b[j],w,w+b[j]);
       tr+=w;
       w+=b[j];
   }
   printf("\nAverage waiting time: %f\n",(tr*0.1)/(n*0.1));
   printf("\nAverage turnarround time: %f\n",((tr+br)*0.1)/(n*0.1));
}
```

~/scheule

```
bhaum@DESKTOP-QIJQUE1 ~/scheule
$ vi fifo_nar.c

bhaum@DESKTOP-QIJQUE1 ~/scheule
$ gcc fifo_nar.c -o ex.exe

bhaum@DESKTOP-QIJQUE1 ~/scheule
$ ./ex.exe
(19BIT0292) Enter the number of processes: 5
Enter the burst time for process 1: 5
Enter the burst time for process 2: 8
Enter the burst time for process 3: 14
Enter the burst time for process 4: 8
Enter the burst time for process 5: 4

    Process No.    Burst Time    Waiting Time      Turn around Time
        1              5              0                 5
        2              8              5                 13
        3              14             13                27
        4              8              27                35
        5              4              35                39

Average waiting time: 16.000000

Average turnarround time: 23.800000
```

ARRIVAL TIME GIVEN

CODE:

```
#include<stdio.h>
#include <stdlib.h>

struct n
{
    int b;
    int a;
    int n;
};

typedef struct n sn;

int main()
{
 sn *a,t;
 int n,i,j,s=0,w=0,tr=0;
 printf("(19BIT0292) Enter the number of processes: ");
 scanf("%d",&n);
 a=(sn*)malloc(n*sizeof(sn));
 for(i=0;i<n;i++)
    {
        printf("Enter the burst time for process %d: ",i+1);
        scanf("%d",&a[i].b);
        printf("Enter the arrival time for process %d: ",i+1);
        scanf("%d",&a[i].a);
        a[i].n=i+1;
    }
//used bubble sort because it is stable
```

```
 for (i=0;i<n-1;i++)
      for (j = 0; j < n-i-1; j++)
        if (a[j].a>a[j+1].a)
              {
                  t=a[j];
                  a[j]=a[j+1];
                  a[j+1]=t;
              }
  printf("\n    Process No.\t  Burst Time\tWaiting Time\t Arrival
Time\t  Turn around Time\n");
   for(j=0;j<n;j++)
   {
       i=((s-a[j].a)<0)?0:s-a[j].a;
       printf("\t%d\t\t%d\t\t%d\t\t%d\t\t%d\
n",a[j].n,a[j].b,i,a[j].a,i+a[j].b);
       s+=a[j].b;
       w+=i;
       tr+=i+a[j].b;
   }
   printf("\nAverage waiting time: %f\n",(w*0.1)/(n*0.1));
   printf("\nAverage turnarround time: %f\n",(tr*0.1)/(n*0.1));
}
```

~/scheule

```
#include<stdio.h>
#include <stdlib.h>

struct n
{
    int b;
    int a;
    int n;
};

typedef struct n sn;
```

~/scheule

```c
int main()
{
 sn *a,t;
 int n,i,j,s=0,w=0,tr=0;
 printf("(19BIT0292) Enter the number of processes: ");
 scanf("%d",&n);
 a=(sn*)malloc(n*sizeof(sn));
 for(i=0;i<n;i++)
    {
        printf("Enter the burst time for process %d: ",i+1);
        scanf("%d",&a[i].b);
        printf("Enter the arrival time for process %d: ",i+1);
        scanf("%d",&a[i].a);
        a[i].n=i+1;
    }
//used bubble sort because it is stable
 for (i=0;i<n-1;i++)
      for (j = 0; j < n-i-1; j++)
        if (a[j].a>a[j+1].a)
              {
                  t=a[j];
                  a[j]=a[j+1];
                  a[j+1]=t;
              }
  printf("\n    Process No.\t  Burst Time\tWaiting Time\t Arrival Time\t  Turn around Time\n");
  for(j=0;j<n;j++)
  {
      i=((s-a[j].a)<0)?0:s-a[j].a;
      printf("\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",a[j].n,a[j].b,i,a[j].a,i+a[j].b);
      s+=a[j].b;
      w+=i;
      tr+=i+a[j].b;
  }
  printf("\nAverage waiting time: %f\n",(w*0.1)/(n*0.1));
  printf("\nAverage turnarround time: %f\n",(tr*0.1)/(n*0.1));
}
```

OUTPUT

```
bhaum@DESKTOP-QIJQUE1 ~/scheule
$ gcc fifo_ar.c -o ex.exe

bhaum@DESKTOP-QIJQUE1 ~/scheule
$ ./ex.exe
(19BIT0292) Enter the number of processes: 5
Enter the burst time for process 1: 5
Enter the arrival time for process 1: 3
Enter the burst time for process 2: 8
Enter the arrival time for process 2: 2
Enter the burst time for process 3: 14
Enter the arrival time for process 3: 1
Enter the burst time for process 4: 8
Enter the arrival time for process 4: 2
Enter the burst time for process 5: 4
Enter the arrival time for process 5: 0

    Process No.    Burst Time    Waiting Time     Arrival Time     Turn around Time
        5               4               0               0               4
        3               14              3               1               17
        2               8               16              2               24
        4               8               24              2               32
        1               5               31              3               36

Average waiting time: 14.800000

Average turnarround time: 22.600000
```

Q5. Shortest-Job-First Scheduling

NON-PREEMPTIVE

CODE:

```
#include<stdio.h>
#include <stdlib.h>
struct n
{
    int b;
    int a;
    char n;
};

typedef struct n sn;
int main()
{
 sn *a,t;
 int n,i,j,s=0,p=0,o,k,w=0,tr=0;
 printf("(19BIT0292) Enter the number of processes: ");
 scanf("%d",&n);
 a=(sn*)malloc(n*sizeof(sn));

 for(i=0;i<n;i++)
    {
        printf("Enter the burst time for process %d: ",i+1);
        scanf("%d",&a[i].b);
        printf("Enter the arrival time for process %d: ",i+1);
```

```
        scanf("%d",&a[i].a);
        a[i].n=i+1;

    }
    //used bubble sort because it is stable
 for (i=0;i<n-1;i++)
      for (j=0;j<n-i-1;j++)
        if (a[j].a>a[j+1].a)
              {
                  t=a[j];
                  a[j]=a[j+1];
                  a[j+1]=t;
              }
 printf("\n    Process No.\t  Burst Time\tWaiting Time\t Arrival
Time\t  Turn around Time\n");
for (o=0;o<n;o++)
{
    k=p;
    while(a[k].a<=s)
    k++;
    for (i=p;i<k;i++)
      for (j=p;j<k-i;j++)
        if (a[j].b>a[j+1].b)
              {
                  t=a[j];
                  a[j]=a[j+1];
                  a[j+1]=t;
              }
    printf("\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",a[p].n,a[p].b,s-
a[p].a,a[p].a,a[p].b+s-a[p].a);
    s+=a[p].b;
    w+=s-a[p].a;
    tr=s+a[p].b-a[p].a;
    p++;
}
   printf("\nAverage waiting time: %f\n",(w*0.1)/(n*0.1));
```

```
    printf("\nAverage turnarround time: %f\n",(tr*0.1)/(n*0.1));
}
```

~/scheule

```
#include<stdio.h>
#include <stdlib.h>

struct n
{
    int b;
    int a;
    char n;
};

typedef struct n sn;

int main()
{
 sn *a,t;
 int n,i,j,s=0,p=0,o,k,w=0,tr=0;
 printf("(19BIT0292) Enter the number of processes: ");
 scanf("%d",&n);
 a=(sn*)malloc(n*sizeof(sn));

 for(i=0;i<n;i++)
    {
        printf("Enter the burst time for process %d: ",i+1);
        scanf("%d",&a[i].b);
        printf("Enter the arrival time for process %d: ",i+1);
        scanf("%d",&a[i].a);
        a[i].n=i+1;

    }
    //used bubble sort because it is stable
 for (i=0;i<n-1;i++)
      for (j=0;j<n-i-1;j++)
        if (a[j].a>a[j+1].a)
              {
                  t=a[j];
                  a[j]=a[j+1];
                  a[j+1]=t;
              }
 printf("\n    Process No.\t  Burst Time\tWaiting Time\t Arrival Time\t  Turn around Time\n");
```

```
for (o=0;o<n;o++)
{
    k=p;
    while(a[k].a<=s)
    k++;
    for (i=p;i<k;i++)
      for (j=p;j<k-i;j++)
        if (a[j].b>a[j+1].b)
              {
                  t=a[j];
                  a[j]=a[j+1];
                  a[j+1]=t;
              }
    printf("\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",a[p].n,a[p].b,s-a[p].a,a[p].a,a[p].b+s-a[p].a);
    s+=a[p].b;
    w+=s-a[p].a;
    tr=s+a[p].b-a[p].a;
    p++;
}

   printf("\nAverage waiting time: %f\n",(w*0.1)/(n*0.1));
   printf("\nAverage turnarround time: %f\n",(tr*0.1)/(n*0.1));
}
```

OUTPUT

```
PS D:\app\crygin\home\bhaum\scheule> cd "d:\app\crygin\home\bhaum\scheule\" ; if ($?) { gcc sjfs_np.c -o sjfs_np } ; if ($?) { .\sjfs_np }
(19BIT0292) Enter the number of processes: 5
Enter the burst time for process 1: 5
Enter the arrival time for process 1: 3
Enter the burst time for process 2: 8
Enter the arrival time for process 2: 2
Enter the burst time for process 3: 14
Enter the arrival time for process 3: 1
Enter the burst time for process 4: 8
Enter the arrival time for process 4: 2
Enter the burst time for process 5: 4
Enter the arrival time for process 5: 0

    Process No.   Burst Time    Waiting Time     Arrival Time     Turn around Time
        5              4              0              0              4
        1              5              1              3              6
        2              8              7              2              15
        4              8              15             2              23
        3              14             24             1              38

Average waiting time: 17.200000

Average turnarround time: 10.400000
```

PREEMPTIVE

CODE:

```
#include<iostream>
using namespace std;
int main()
{
 int n,temp,tt=0,min,d,i,j;
 float atat=0,awt=0,stat=0,swt=0;
 cout<<"enter no of process"<<endl;
 cin>>n;
 int a[n],b[n],e[n],tat[n],wt[n];

 for(i=0;i<n;i++)
 {
 cout<<"enter arival time "; //input
 cin>>a[i];
```

```
}
for(i=0;i<n;i++)
{
cout<<"enter brust time "; //input
cin>>b[i];
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(b[i]>b[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
temp=b[i];
b[i]=b[j];
b[j]=temp;
}
}
}
min=a[0];
for(i=0;i<n;i++)
{
if(min>a[i])
{
min=a[i];
d=i;
}
}
tt=min;
e[d]=tt+b[d];
tt=e[d];
for(i=0;i<n;i++)
{
```

```
 if(a[i]!=min)
 {
 e[i]=b[i]+tt;
 tt=e[i];
 }
 }
 for(i=0;i<n;i++)
 {
 tat[i]=e[i]-a[i];
 stat=stat+tat[i];
 wt[i]=tat[i]-b[i];
 swt=swt+wt[i];
 }
 atat=stat/n;
 awt=swt/n;
 cout<<"Process Arrival-time(s) Burst-time(s) Waiting-time(s)
Turnaround-time(s)\n";
 for(i=0;i<n;i++)
 {
 cout<<"P"<<i+1<<" "<<a[i]<<" "<<b[i]<<" "<<wt[i]<<"
"<<tat[i]<<endl;
 }
 cout<<"awt="<<awt<<" atat="<<atat; //average waiting time and
turn around time
}
```

Q6. Priority Scheduling

PREEMPTIVE

CODE:

```
#include<stdio.h>
#include<string.h>
int main(){
 int bt[20],at[10],n,i,j,temp,p[10],st[10],ft[10],wt[10],ta[10];
 int totwt=0,totta=0;
 float awt,ata;
 char pn[10][10],t[10];
 printf(" Enter the number of process:");
 scanf("%d",&n);
 for(i=0; i<n; i++)
 {
 printf("(19BIT0292) Enter process name, ArrivalTime, BurstTime & Priority:");
 scanf("%s%d%d%d",pn[i],&at[i],&bt[i],&p[i]);
 }
 for(i=0; i<n; i++){
 for(j=0; j<n; j++)
 {
 if(p[i]<p[j])
 {
 temp=p[i];
 p[i]=p[j];
 p[j]=temp;
 temp=at[i];
 at[i]=at[j];
 at[j]=temp;
 temp=bt[i];bt[i]=bt[j];
 bt[j]=temp;
 strcpy(t,pn[i]);
 strcpy(pn[i],pn[j]);
 strcpy(pn[j],t);
 }
 }
 }
 for(i=0; i<n; i++)
 {
 if(i==0)
 {
 st[i]=at[i];
 wt[i]=st[i]-at[i];
 ft[i]=st[i]+bt[i];
 ta[i]=ft[i]-at[i];
 }
 else
 {
```

```c
 st[i]=ft[i-1];
 wt[i]=st[i]-at[i];
 ft[i]=st[i]+bt[i];
 ta[i]=ft[i]-at[i];

 }
 totwt+=wt[i];totta+=ta[i];
 }
 awt=(float)totwt/n;
 ata=(float)totta/n;

printf("\nPName\tArrivalTime\tBurstTime\tPriority\tWaitingTime\tTotalTurnAroundTime");
 for(i=0; i<n; i++)

printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],bt[i],p[i],wt[i],ta[i]);
 printf("\nAverage waiting time is:%f",awt);
 printf("\nAverage turnaroundtime is:%f",ata);
}
```

```
PS D:\app\crygin\home\bhaum> cd "d:\app\crygin\home\bhaum\" ; if ($?) { gcc a.c -o a } ; if ($?) { .\a }
 Enter the number of process:6
(19BIT0292) Enter process name, ArrivalTime, BurstTime & Priority:1 2 5 1
(19BIT0292) Enter process name, ArrivalTime, BurstTime & Priority:3 9 4 3
(19BIT0292) Enter process name, ArrivalTime, BurstTime & Priority:2 6 4 2
(19BIT0292) Enter process name, ArrivalTime, BurstTime & Priority:4 6 1 6
(19BIT0292) Enter process name, ArrivalTime, BurstTime & Priority:5 9 3 4
(19BIT0292) Enter process name, ArrivalTime, BurstTime & Priority:6 7 1 4

PName   ArrivalTime     BurstTime       Priority        WaitingTime     TotalTurnAroundTime
1           2               5               1               0               5
2           6               4               2               1               5
3           9               4               3               2               6
5           9               3               4               6               9
6           7               1               4              11              12
4           6               1               6              13              14
Average waiting time is:5.500000
Average turnaroundtime is:8.500000
PS D:\app\crygin\home\bhaum>
```

NON PREEMPTIVE

CODE:

```
#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(19BIT0292) 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; //contains process number
 }
 //sorting burst time, priority and process number in ascending order using
 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; //waiting time for first process is 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=total/n; //average waiting time
 total=0;
```

```c
 printf("\nProcess\t Burst Time \tWaiting Time\tTurnaroundTime");
 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=total/n; //average turnaround time
 printf("\n\nAverage Waiting Time=%d",avg_wt);
 printf("\nAverage Turnaround Time=%d\n",avg_tat);
return 0;
}
```

```
PS D:\app\crygin\home\bhaum> cd "d:\app\crygin\home\bhaum\" ; if ($?) { gcc a.c -o a } ; if ($?) {
Enter Total Number of Process:3

(19BIT0292) Enter Burst Time and Priority

P[1]
Burst Time:6
Priority:2

P[2]
Burst Time:8
Priority:5

P[3]
Burst Time:10
Priority:4

Process  Burst Time     Waiting Time    TurnaroundTime
P[1]            6               0                    6
P[3]            10              6                    16
P[2]            8               16                   24

Average Waiting Time=7
Average Turnaround Time=15
```

Q7. ROUND ROBIN

CODE:

```
#include<stdio.h>
int main()
{
 int count,j,n,time,remain,flag=0,time_quantum;
 int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
 printf("(19BIT0292) Enter Total Process:\t ");
 scanf("%d",&n);
 remain=n;
 for(count=0;count<n;count++)
 {
 printf(" Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
 scanf("%d",&at[count]);
 scanf("%d",&bt[count]);
 rt[count]=bt[count];
 }
 printf("Enter Time Quantum:\t");
 scanf("%d",&time_quantum);
 printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
 for(time=0,count=0;remain!=0;)
 {
 if(rt[count]<=time_quantum && rt[count]>0)
 {
 time+=rt[count];
 rt[count]=0;
 flag=1;
 }
 else if(rt[count]>0)
 {
 rt[count]-=time_quantum;
 time+=time_quantum;
 }
 if(rt[count]==0 && flag==1)
 {
 remain--;
 printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-
bt[count]);
 wait_time+= time -at[count]-bt[count];
 turnaround_time+= time -at[count];
 flag=0;
 }
 if(count==n-1)
 count=0;
 else if(at[count+1]<=time)
 count++;
 else
 count=0;
```

```
}
printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);
return 0;
}
```

OUTPUT

```
PS C:\Users\bhaum> cd "d:\app\crygin\home\bhaum\" ; if ($?) { gcc a.c -o a } ; if ($?) { .\a }
(19BIT0292) Enter Total Process:         5
 Enter Arrival Time and Burst Time for Process Process Number 1 :2 7
 Enter Arrival Time and Burst Time for Process Process Number 2 :5 9
 Enter Arrival Time and Burst Time for Process Process Number 3 :1 6
 Enter Arrival Time and Burst Time for Process Process Number 4 :4 2
 Enter Arrival Time and Burst Time for Process Process Number 5 :1 5
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[4]    |       8       |       6
P[1]    |       13      |       6
P[3]    |       24      |       18
P[5]    |       25      |       20
P[2]    |       24      |       15

Average Waiting Time= 13.000000
Avg Turnaround Time = 18.800000
PS D:\app\crygin\home\bhaum>
```