Logo

Description automatically generated

### AMITY UNIVERSITY RAJASTHAN

### AMITY SCHOOL OF ENGINEERING & TECHNOLOGY

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

### OPERATING SYSTEMS LAB RECORD

|  |  |
| --- | --- |
| Submitted To: | Submitted By: |
| **Mr. Amit Kumar Gupta** | **Arpit Agarwal**  **B.Tech CSE Sec-B(2020-24)**  **Enrolment No.-A20405220102** |

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | EXPERIMENT | DATE | PAGE  NO. | REMARKS |
| 1 | Read the content of a file and count no of words line and character in a file | 3/8/2021 | 4 |  |
| 2 | Read the file and find occurrence of a particular character in a file | 3/8/2021 | 5 |  |
| 3 | Basic commands which are using in the shell programming | 10/8/2021 | 6 |  |
| 4 | Write a shell script to print your name | 17/8/2021 | 7 |  |
| 5 | Write a shell script to calculate the sum of three no and find out the average of hem | 17/8/2021 | 8 |  |
| 6 | Write a shell script to find out greatest no between the three number | 24/8/2021 | 9 |  |
| 7 | Write a shell script to find out greatest between the four number | 24/8/2021 | 10 |  |
| 8 | Write a shell script to check whether given no is prime or not | 31/8/2021 | 11 |  |
| 9 | Write a shell script to check whether given no is Armstrong or not | 31/8/2021 | 12 |  |
| 10 | Write a shell script to calculate reverse no of given no | 7/9/2021 | 13 |  |
| 11 | Write a shell script to sum of digit of given no | 7/9/2021 | 14 |  |
| 12 | Write a shell script to print file names one per line in a directory showing serial number of the file | 14/9/2021 | 15 |  |
| 13 | Write a shell script to sort a list of files either in alphabetic order or largest file first, according to user response. | 14/9/2021 | 16 |  |
| 14 | Write a shell script count the lines, words, and characters in its input | 21/9/2021 | 17 |  |
| 15 | Write a shell script to change date format. Show the time taken in execution of this script | 21/9/2021 | 18 |  |
| 16 | Write a shell script to find out pattern in a file and count its occurrence | 28/9/2021 | 19 |  |
| 17 | Write a shell script to count the occurrence of each word in a file. | 28/9/2021 | 20 |  |
| 19 | FCFS(First Come First Serve) Scheduling | 12/10/2021 | 21-22 |  |
| 20 | Write a program for Priority Scheduling with SJF burst time logic | 12/10/2021 | 23-24 |  |
| 21 | Priority Scheduling | 19/10/2021 | 25-28 |  |
| 22 | Write a program to check whether given no. is palindrome or not | 26/10/2021 | 29 |  |
| 23 | Write a program for preemptive SJF and the logic of preemptive algorithm | 26/10/2021 | 30-31 |  |
| 24 | Write a program for preemptive Priority Scheduling and the logic of the preemptive algorithm | 2/11/2021 | 32-34 |  |
| 25 | Write a program for implementing Round Robin Scheduling | 2/11/2021 | 35-37 |  |

**Experiment - 1**

**Aim:** Read the content of a file and count no of words line and character in a file.

#include<stdio.h>

#include<string.h>

int main(){

    int count1= 0;

    int count2= 1;

    int count3= 1;

    char c;

    FILE \*f1;

    f1 = fopen("/Users/DELL/OneDrive/Desktop/git/eii.txt","r");

    fflush(stdin);

    if (f1 == NULL)

    {

        printf("The file dosent exists\n");

    }

    else{

     while((c = fgetc(f1))!= EOF )

    {

        printf("%c",c);

        count1++;

        if(c == '\n')

        {

        count2++ ;

        }

        if(c == ' '||c=='\n')

        {

        count3++ ;

        }

    }

    printf("\nnumber of characters in this file: %d\n",count1);

    printf("number of lines in this file: %d\n",count2);

    printf("number of words in this file: %d\n",count3);

    }

    fclose(f1);

return 0;

}

**Output:**

Good Morning

Shishir Banerjee

number of characters in this file: 30

number of lines in this file: 2

number of words in this file: 4

**Experiment - 2**

**Aim:** Read the file and find occurrence of a particular character in a file.

#include<stdio.h>

#include<string.h>

int main(){

    int count1= 0;

    char c;

    char character;

    printf("Enter the character you want to count\n");

    scanf("%c",&character);

    FILE \*f1;

    f1 = fopen("/Users/DELL/OneDrive/Desktop/git/eii.txt","r");

    if (f1 == NULL)

    {

        printf("The file dosent exists\n");

    }

    else{

    while((c = fgetc(f1))!= EOF )

    {

        printf("%c",c);

        if(c == character)

        {

            count1++;

        }

    }

    printf("\nnumber of '%c' characters in this file: %d\n",character,count1);

    }

    fclose(f1);

return 0;

}

**Output:**

Enter the character you want to count

o

Good Morning

Shishir Banerjee

number of 'o' characters in this file:

**Experiment - 3**

**Aim:** Basic commands which are using in the shell programming

**1. touch command**: - used to create a blank file. Used to create file without any content.

touch filename

**2. cat command**: - used to create a file with content.

$cat filename

**3. cp command**: - used to copy content of one file to another. If the second file does not exist it will overwrite the content.

$cp a1 a2

**4. rm command**: - used to remove directories or files.

rm [Option]…File.

**5. mv command: -** used to rename the file.

$mv test sample

**6. ls command: -** it is used for listing files and directories.

$ls

**7. cd command: -** used to change the directory

$cd new dir.

**8. mk dir command: -** used to make directories

$mk dir calendar

**9. expr command: -** used to run an arithmetic operation.

$expr 27+11

**10. bc command: -** used for binary calculator.

$bc

**Experiment - 4**

**Aim:** Write a shell script to print your name.

#name.sh

echo "Enter name"

read a

echo "$a"

**Output:**

Enter name

Arpit Agarwal

Arpit Agarwal

**Experiment - 5**

**Aim:** Write a shell script to calculate the sum of three no and find out the average of them.

echo "Enter  number"

read num1

echo "Enter  number"

read num2

echo "Enter  number"

read num3

c=`expr $(($num1+$num2+$num3))`

d=$(echo $c / 3| bc -l)

echo "Sum of three number =$c"

echo "Avg of three number =$d"

**Output:**

Enter number

1

Enter number

2

Enter number

5

Sum of three number =8

Avg of three number =2.66666666666666666666

**Experiment - 6**

**Aim:** Write a shell script to find out greatest no between the three number.

echo "Enter number n1"

read n1

echo "Enter number n2 "

read n2

echo "Enter number n3"

read n3

if  test  $n1 -gt $n2 -a  $n1 -gt $n3

then

      echo "$n1 is the Greatest Number"

elif test  $n2 -gt $n3

then

     echo "$n2 is the Greatest Number"

else

     echo "$n3 is the Greatest Number"

fi

**Output:**

Enter number n1

3

Enter number n2

6

Enter number n3

4

6 is the Greatest Number

**Experiment - 7**

**Aim:** Write a shell script to find out greatest between the four number.

echo "Enter number n1"

read n1

echo "Enter number n2 "

read n2

echo "Enter number n3”

read n3

echo "Enter number n4"

read n4

if  test  $n1 -gt $n2 -a  $n1 -gt $n2 -a  $n1 -gt $n3 -a $n1 -gt $n4

then

      echo "$n1 is a Greatest Number"

elif test  $n2 -gt $n3  -a $n2 -gt $n3 -a  $n2 -gt $n4

then

     echo "$n2 is a Greatest Number"

elif test  $n3 -gt $n4

then

     echo "$n3 is a Greatest Number"

else

     echo "$n4 is a Greatest Number"

fi

**Output:**

Enter number n1

5

Enter number n2

3

Enter number n3

7

Enter number n4

2

7 is a Greatest Number

**Experiment - 8**

**Aim: Write a shell script to check whether given no is prime or not.**

echo "enter number"

read num

for((i=2; i<=num/2; i++))

do

  if  test $((num%i)) -eq 0

  then

    echo "$num is not a prime number."

    exit

  fi

done

echo "$num is a prime number."

**Output:**

enter number

5

5 is a prime number.

enter number

8

8 is not a prime number.

**Experiment - 9**

**Aim:** Write a shell script to check whether given no is Armstrong or not.

echo "Enter the number"

read n

t=$n

r=0

sum=0

p=0

while [ $t -gt 0 ]

do

   r=`expr $t % 10`

   p=`expr $r \\* $r \\* $r`

   sum=`expr $sum + $p`

   t=`expr $t / 10`

   done

if [ $sum -eq $n ]

then

echo "$n is Amstrong"

else

echo "$n is not Amstrong"

fi

**Output:**

Enter the number

153

153 is Amstrong

Enter the number

93

93 is not Amstrong

**Experiment - 10**

**Aim:** Write a shell script to calculate reverse no of given number.

echo "Enter Numnber"

read n

sd=0

rev=0

while [ $n -gt 0 ]

do

    sd=$(( $n % 10 );

    rev=$(( $rev \* 10 + $sd ))

    n=$(( $n / 10 ))

done

echo "Reverse number of entered digit is $rev"

**Output:**

Enter Number

9876

Reverse number of entered digit is 6789

**Experiment - 11**

**Aim:** Write a shell script to sum of digit of given number.

echo "Enter the number"

read n

sd=0

sum=0

while [ $n -gt 0 ]

do

    sd=$(( $n % 10 ))

    sum=`expr $sum + $sd`

    n=$(( $n / 10 ))

done

echo "Sum=$sum"

**Output:**

Enter the number

456

Sum=15

**Experiment - 12**

**Aim:** Write a shell script to print file names one per line in a directory showing serial number of the file.

echo "File Names in the Directory"

ls | nl

**Output:**

File Names in the Directory

1 Armstrong.sh

2 Namedirectory.sh

3 SumDigit.sh

4 fileby\_Alp.sh

5 formatedate.sh

6 gretest.sh

7 gretest3.sh

8 name.sh

9 prime.sh

10 reverse.sh

11 sum.sh

12 tempCodeRunnerFile.sh

13 word.sh

**Experiment - 13**

**Aim:** Write a shell script to sort a list of files either in alphabetic order or largest file first, according to user response.

echo "In which type you want print file names in this directory"

echo "enter 1 for alphabatical and 2 for size"

read n

if  test $n -eq 1

then

   ls -l;

else

   ls -S;

fi

**Output:**

In which type you want print file names in this directory

Enter 1 for alphabetical and 2 for size

2

gretest.sh gretest3.sh sum.sh reverse.sh prime.sh name.sh tempCodeRunnerFile.sh

formatedate.sh Armstrong.sh word.sh fileby\_Alp.sh SumDigit.sh Namedirectory.sh

In which type you want print file names in this directory

Enter 1 for alphabetical and 2 for size

1

Armstrong.sh Namedirectory.sh SumDigit.sh fileby\_Alp.sh formatedate.sh gretest.sh gretest3.sh

name.sh prime.sh reverse.sh sum.sh tempCodeRunnerFile.sh word.sh word1.sh

**Experiment - 14**

**Aim:** Write a shell script count the lines, words, and characters in its input.

echo "Enter the filename"

read file

words=$(cat $file | wc -w)

chars=$(cat $file | wc -c)

lines=$(grep -c "." $file)

echo "Number of characters in $file is $chars"

echo "Number of words in $file is $words"

echo "Number of lines in $file is $lines"

File inside

Arpit Agarwal

good morning

**Output:**

Enter the filename

test

Number of characters in test is 23

Number of words in test is 4

Number of lines in test is 2

**Experiment - 15**

**Aim:** Write a shell script to change date format. Show the time taken in execution of this script.

#changing date formate

echo  "Choose date formate"

echo "dd-mm-yy"

echo "dd-mm-yyyy"

echo "mm-dd-yyyy"

echo "mm-dd-yy

Enter 1,2,3,4 according"

read N

if test $N -eq 1

then

  date +"%d-%m-%y"

  fi

  if test $N -eq 2

then

  date +"%d-%m-%Y"

  fi

  if test $N -eq 3

then

  date +"%m-%d-%Y"

  fi

  if test $N -eq 4

then

  date +"%m-%d-%y"

  fi

**Output:**

Choose date format

dd-mm-yy

dd-mm-yyyy

mm-dd-yyyy

mm-dd-yy

Enter 1,2,3,4 according

1

23-09-21

**Experiment-16**

**Aim-** Write a shell script to find out pattern in a file and count its occurrence

if ( { grep -q '<Pattern>' '<file>' } ) then

 echo "Pattern found"

else

 echo "Pattern not found"

 endif

**Output-** Pattern Found

**Experiment-17**

**Aim-** Write a shell script to count the occurrence of each word in a file.

**Input-** Consider the file(demo.txt) with the following content:

This is first line.

This is second line

This is third line.

**Program-**

#!/usr/bin/bash

# path to the file

file\_path="/home/shishir/Desktop/demo.txt"

# using wc command to count number of lines

number\_of\_lines=`wc --lines < $file\_path`

# using wc command to count number of words

number\_of\_words=`wc --word < $file\_path`

# Displaying number of lines and number of words

echo "Number of lines: $number\_of\_lines"

echo "Number of words: $number\_of\_words"

**Output-**

Number of lines:3

Number of words:12

This is third line.

**Experiment-19**

**Aim-FCFS (First Come First Serve) Scheduling**

#include <stdio.h>

int waitingtime(int proc[], int n,

int burst\_time[], int wait\_time[]) {

   wait\_time[0] = 0;

   for (int i = 1; i < n ; i++ )

   wait\_time[i] = burst\_time[i-1] + wait\_time[i-1] ;

   return 0;

}

int turnaroundtime( int proc[], int n,

int burst\_time[], int wait\_time[], int tat[]){

   int i;

   for ( i = 0; i < n ; i++)

   tat[i] = burst\_time[i] + wait\_time[i];

   return 0;

}

int avgtime( int proc[], int n, int burst\_time[]) {

   int wait\_time[n], tat[n], total\_wt = 0, total\_tat = 0;

   int i;

   waitingtime(proc, n, burst\_time, wait\_time);

   turnaroundtime(proc, n, burst\_time, wait\_time, tat);

   printf("Processes  Burst   Waiting Turn around \n");

   for ( i=0; i<n; i++) {

      total\_wt = total\_wt + wait\_time[i];

      total\_tat = total\_tat + tat[i];

      printf(" %d\t  %d\t\t %d \t%d\n", i+1, burst\_time[i], wait\_time[i], tat[i]);

   }

   printf("Average waiting time = %f\n", (float)total\_wt / (float)n);

   printf("Average turn around time = %f\n", (float)total\_tat / (float)n);

   return 0;

}

int main() {

   int proc[] = { 1, 2, 3};

   int n = sizeof proc / sizeof proc[0];

   int burst\_time[] = {5, 8, 12};

   avgtime(proc, n, burst\_time);

   return 0;

}

**Output-** Average waiting time = 6.000000

Average turnaround time = 14.333333

**Experiment-20**

**Aim-** Write a program for Priority scheduling with SJF burst time logic

#include<stdio.h>

 int main()

{

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

    float avg\_wt,avg\_tat;

    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;

    }

   //sorting of burst times

    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];

        total+=wt[i];

    }

    avg\_wt=(float)total/n;

    total=0;

    printf("nProcesst    Burst Time    tWaiting TimetTurnaround Time");

    for(i=0;i<n;i++)

    {

        tat[i]=bt[i]+wt[i];

        total+=tat[i];

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

    }

    avg\_tat=(float)total/n;

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

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

}

**Output-** Enter the number of processes: 5

Enter Burst Time:

P1:4

P2:3

P3:7

P4:1

P5:2

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Burst time | Waiting time | Turnaround time |
| P4 | 1 | 0 | 1 |
| P5 | 2 | 1 | 3 |
| P2 | 3 | 3 | 6 |
| P1 | 4 | 6 | 10 |
| P3 | 7 | 10 | 17 |

Average Waiting Time: 4.000000

Average Turnaround Time: 7.400000

**Experiment-21**

**Aim-** Priority Scheduling

#include <stdio.h>

#include<conio.h>

void main()

 {

   int x,n,p[10],pp[10],pt[10],w[10],t[10],awt,atat,i;

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

   scanf("%d",&n);

   printf("\n Enter process : time priorities \n");

   for(i=0;i<n;i++)

    {

      printf("\nProcess no %d : ",i+1);

      scanf("%d  %d",&pt[i],&pp[i]);

      p[i]=i+1;

    }

  for(i=0;i<n-1;i++)

   {

     for(int j=i+1;j<n;j++)

     {

       if(pp[i]<pp[j])

       {

         x=pp[i];

         pp[i]=pp[j];

         pp[j]=x;

         x=pt[i];

         pt[i]=pt[j];

         pt[j]=x;

         x=p[i];

         p[i]=p[j];

         p[j]=x;

      }

   }

}

w[0]=0;

awt=0;

t[0]=pt[0];

atat=t[0];

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

 {

   w[i]=t[i-1];

   awt+=w[i];

   t[i]=w[i]+pt[i];

   atat+=t[i];

 }

printf("\n\n Job \t Burst Time \t Wait Time \t Turn Around Time   Priority \n");

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

  printf("\n %d \t\t %d  \t\t %d \t\t %d \t\t %d \n",p[i],pt[i],w[i],t[i],pp[i]);

awt/=n;

atat/=n;

printf("\n Average Wait Time : %d \n",awt);

printf("\n Average Turn Around Time : %d \n",atat);

getch();

}

**Output-** Enter the number of processes:4

Enter process: time priorities

Process no 1:3

1

Process 2:4

2

Process 3:5

3

Process no 4:6

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Job | Burst Time | Wait Time | Turnaround Time | Priority |
| 4 | 6 | 0 | 6 | 4 |
| 3 | 5 | 6 | 11 | 3 |
| 2 | 4 | 11 | 15 | 2 |
| 1 | 3 | 15 | 18 | 1 |

Average Wait Time: 8

Average Turnaround Time: 12

**Experiment-22**

**Aim-** Write a program to check whether given no. is palindrome or not.

#include <stdio.h>

int main() {

  int n, reversed = 0, remainder, original;

    printf("Enter an integer: ");

    scanf("%d", &n);

    original = n;

    while (n != 0) {

        remainder = n % 10;

        reversed = reversed \* 10 + remainder;

        n /= 10;

    }

    if (original == reversed)

        printf("%d is a palindrome.", original);

    else

        printf("%d is not a palindrome.", original);

    return 0;

}

**Output-** Enter an integer: 1001

1001 is a palindrome

**Experiment-23**

**Aim-** Write a program of the preemptive SJF and the logic of the preemptive algorithm

#include<stdio.h>

struct process

{

    int WT,AT,BT,TAT;

};

struct process a[10];

int main()

{

    int n,temp[10];

    int count=0,t=0,short\_P;

    float total\_WT=0, total\_TAT=0,Avg\_WT,Avg\_TAT;

    printf("Enter the number of the process\n");

    scanf("%d",&n);

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

    printf("AT WT\n");

    for(int i=0;i<n;i++)

    {

        scanf("%d%d",&a[i].AT,&a[i].BT);

        temp[i]=a[i].BT;

    }

    a[9].BT=10000;

    for(t=0;count!=n;t++)

    {

        short\_P=9;

        for(int i=0;i<n;i++)

        {

            if(a[i].BT<a[short\_P].BT && (a[i].AT<=t && a[i].BT>0))

            {

                short\_P=i;

            }

        }

        a[short\_P].BT=a[short\_P].BT-1;

        if(a[short\_P].BT==0)

        {

            // one process complete

            count++;

            a[short\_P].WT=t+1-a[short\_P].AT-temp[short\_P];

            a[short\_P].TAT=t+1-a[short\_P].AT;

            total\_WT=total\_WT+a[short\_P].WT;

            total\_TAT=total\_TAT+a[short\_P].TAT;

        }

    }

    Avg\_WT=total\_WT/n;

    Avg\_TAT=total\_TAT/n;

    printf("Id WT TAT\n");

    for(int i=0;i<n;i++)

    {

        printf("%d\t%d\t%d\n",i+1,a[i].WT,a[i].TAT);

    }

    printf("Avg waiting time of the process is %f\n",Avg\_WT);

    printf("Avg turn around time of the process %f\n",Avg\_TAT);

}

**Output-** Enter the number of process: 3

Enter the arrival time and burst time of the process:

|  |  |
| --- | --- |
| AT | WT |
| 0 | 7 |
| 2 | 3 |
| 7 | 2 |

|  |  |  |
| --- | --- | --- |
| Id | WT | TAT |
| 1 | 5 | 12 |
| 2 | 0 | 3 |
| 3 | 0 | 2 |

Avg waiting time of the process is 1.666667

Avg turnaround time of the process is 5.666667

**Experiment-24**

**Aim-** Write a program for preemptive priority scheduling and the logic of the preemptive algorithm.

#include<stdio.h>

struct process

{

    int WT,AT,BT,TAT,PT;

};

struct process a[10];

int main()

{

    int n,temp[10],t,count=0,short\_p;

    float total\_WT=0,total\_TAT=0,Avg\_WT,Avg\_TAT;

    printf("Enter the number of the process\n");

    scanf("%d",&n);

    printf("Enter the arrival time , burst time and priority of the process\n");

    printf("AT BT PT\n");

    for(int i=0;i<n;i++)

    {

        scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

        temp[i]=a[i].BT;

    }

    // we initialize the burst time

    // of a process with maximum

    a[9].PT=10000;

    for(t=0;count!=n;t++)

    {

        short\_p=9;

        for(int i=0;i<n;i++)

        {

            if(a[short\_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)

            {

                short\_p=i;

            }

        }

        a[short\_p].BT=a[short\_p].BT-1;

        // if any process is completed

        if(a[short\_p].BT==0)

        {

            // one process is completed

            // so count increases by 1

            count++;

            a[short\_p].WT=t+1-a[short\_p].AT-temp[short\_p];

            a[short\_p].TAT=t+1-a[short\_p].AT;

            // total calculation

            total\_WT=total\_WT+a[short\_p].WT;

            total\_TAT=total\_TAT+a[short\_p].TAT;

        }

    }

    Avg\_WT=total\_WT/n;

    Avg\_TAT=total\_TAT/n;

    // printing of the answer

    printf("ID WT TAT\n");

    for(int i=0;i<n;i++)

    {

        printf("%d %d\t%d\n",i+1,a[i].WT,a[i].TAT);

    }

    printf("Avg waiting time of the process  is %f\n",Avg\_WT);

    printf("Avg turn around time of the process is %f\n",Avg\_TAT);

    return 0;

}

**Output-** Enter the number of process: 3

Enter the arrival time, burst time and priority of the process:

|  |  |  |
| --- | --- | --- |
| AT | BT | PT |
| 0 | 3 | 3 |
| 1 | 5 | 1 |
| 2 | 2 | 2 |

|  |  |  |
| --- | --- | --- |
| ID | WT | TAT |
| 1 | 7 | 10 |
| 2 | 0 | 5 |
| 3 | 4 | 6 |

Avg waiting time of the process is 3.666667

Avg waiting time of the process is 7.000000

**Experiment-25**

**Aim-** Write a program to implement Round Robin Scheduling

#include<stdio.h>

#include<conio.h>

void main()

{

    int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];

    float avg\_wt, avg\_tat;

    printf(" Total number of process in the system: ");

    scanf("%d", &NOP);

    y = NOP; // Assign the number of process to variable y

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

{

printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);

printf(" Arrival time is: \t");  // Accept arrival time

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

printf(" \nBurst time is: \t"); // Accept the Burst time

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

temp[i] = bt[i]; // store the burst time in temp array

}

printf("Enter the Time Quantum for the process: \t");

scanf("%d", &quant);

printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");

for(sum=0, i = 0; y!=0; )

{

if(temp[i] <= quant && temp[i] > 0) // define the conditions

{

    sum = sum + temp[i];

    temp[i] = 0;

    count=1;

    }

    else if(temp[i] > 0)

    {

        temp[i] = temp[i] - quant;

        sum = sum + quant;

    }

    if(temp[i]==0 && count==1)

    {

        y--;

        printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);

        wt = wt+sum-at[i]-bt[i];

        tat = tat+sum-at[i];

        count =0;

    }

    if(i==NOP-1)

    {

        i=0;

    }

    else if(at[i+1]<=sum)

    {

        i++;

    }

    else

    {

        i=0;

    }

}

avg\_wt = wt \* 1.0/NOP;

avg\_tat = tat \* 1.0/NOP;

printf("\n Average Turn Around Time: \t%f", avg\_wt);

printf("\n Average Waiting Time: \t%f", avg\_tat);

getch();

}

**Output-** Total number of processes in the system: 4

Enter the arrival and Burst time of the Process 1

Arrival time is: 0

Burst time is: 8

Enter the Arrival and Burst time of the Process 2

Arrival time is: 1

Burst time is : 5

Enter the Arrival and Burst time of the Process 3:

Arrival time is: 2

Burst time is: 10

Enter the Arrival and Burst time of Process 4:

Arrival time is: 3

Burst time is : 11

Enter the Time Quantum for the process: 6

|  |  |  |  |
| --- | --- | --- | --- |
| Process No. | Burst time | TAT | Waiting time |
| 2 | 5 | 10 | 5 |
| 1 | 8 | 25 | 17 |
| 3 | 10 | 27 | 17 |
| 4 | 11 | 31 | 20 |

Avg Turnaround time: 14.750000

Avg Waiting time: 23.250000