Lumbini City College

(Affiliated to Tribhuvan University)

Tilottama-04 Rupandehi

LAB ASSIGNMENT (2080)

Course code: CACS 251

Course Title: Operating System

Faculty Of Humanities and Social Science

Bachelor In Computer Application

Fourth Semester

Submitted To:

Mr. Rahul Shakya

Submitted By:

Gaurav Karki

Table of Content

Name of subject teacher:

Rahul shakya

Final Remark

....

Objective: First Come First Served (FIFO)/FCFS scheduling algorithm in C.

Source cod:

```
# include <stdio.h>
 int main()
 {
 int n, bt[20], wt[20], tat[20], avwt=0, avtat =0, i, j;
 printf(" Enter total no. of process [max=20]: ");
 scanf("%d", &n);
 printf("\n Enter Process Burst Time\n");
 for(i=0; i<n; i++){
  printf(" P[%d]: ", i+1);
  scanf("%d", &bt[i]);
  }
 wt [0] = 0;
 for (j=0 ; j<i; j++){
   wt[i] + =bt[j];
  }
 printf("\n Process\t\t Brust time \t Waiting time \t Turnaround
         time ");
 for (i=0 ; i<n; i++ ) {
   tat[i] = bt[i] +wt[i];
  avwt += wt[i];
  avtat += tat[i];
  printf("\n P[%d]\t\t %d\t \t %d \t\t %d", i+1, bt[i], wt[i],
            tat [i] );
  }
 avwt/= i;
 avtat /= i;
 printf("\n\n Average waiting time: %d ". avwt);
 printf("\n Average turn around time: %d ". avtat);
 return 0;
 }
```

Lab 2

Objective: Shortest Job first (SJF) scheduling algorithm in C.

source cod:

```
#include <stdio.h>
void 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("\n Enter brust time:\n");
 for (i=0; i<n; i++)
   {
     printf("p%d:", i+1);
     scanf ("%d", & bt[i]);
     p[i] = i+1;
   }
  for(i=0; i<n; i++) {
     pos = i;
     for(j=i; 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;~~
avg-wt = (float) total /n;
total =0;
printf("\n Process \t Brust time \t Waiting time \t Turn around");
for(i=0; i<n; i++){
   tat[i] = bt[i] + wt[i];
   total + = tat[i];
 printf("\n p%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
   }
avg-tat = (float) total/n;
printf("\n\n Average waiting time = %f", avg-wt);
printf("\n average turnaround time = %f", avg-tat);
 }
```

Lab 3

Objective: Round Robin (RR) scheduling algorithm in c.

source code:

```
#include <stdio.h>
 void main ()
{
  int i,j, bu[10], wa[10], tat[10], ct[10], t, max;
  float awt =0 , att =0, temp=0;
  printf(" Enter the no. of process: ");
  scanf(" %d ", &n);
  for(i=0; i<n; i++){
    printf("\n Enter Brust time for process %d : "i+1);
    scanf(" %d ", &bu[i]);
     ct[i] = bu[i];
   }
   printf("\n Enter the size of time slice: ");
  scanf("%d ", &t);
   max = bu[0];
   for ( i=1; i<n ; i++){
       if(max < bu[i]){
           max = bu[i];
       }
     }
for (j=0 ; j<(max/t) +1; j++){
    for (i=0; i<n ;i++){
          if (bu[i]!=0){
             if(bu[i] <=t){
               tat[i] = temp + bu[i];
               temp = temp + bu[i];
               bu[i] =0
             }
           else{
             bu [i] = bu[i] -t;
               temp = temp +t;
             }
           }
       }
   }
for(i=0; i<n; i++) {
    wa[i] = tat[i] - ct[i]
    att + = tat[i];
    awt + = wa[i];
 }
printf("\n average turnarount time is %f ", att/n);
printf("\n average waiting time is %d", awt/n);
printf("\n\t process \t brust time \t waiting time \t turnaround\n");
for( i=0; i<n; i++) {
  printf("\t %d \t %d \t\t %d\t\t %d ", i+1, ct[i], wa[i], tat[i]);
  getch();
 }
}
```

#Lab 4

Objective: algorithm for Priority scheduling in c.

source code:

```
# include<stdio.h>
  void main ()
 {
  int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
  float wtavg, tatavg;
  printf(" Enter the no. of process: ");
  scanf("%d",&n);
  for(i=0; i<n; i++){
        p[i] = i;
        printf(" Enter the brust time & priority of process %d: ",i);
        scanf("%d %d", &bt[i], &pri[i]);
   }

  for (i=0; i<n ; i++){
     for(k=i+1; k<n; k++){
          if(prin[i] > pri[k] {
               temp = p[i];
               p[i] = p[k];
               p[k] = temp;
               temp = bt[i];
               bt[i] = bt[k];
               bt[k] = temp;
               temp = pri[k];
               pri[i] = pri[k];
               pri[k] = temp;
          }
       }
   }
  wtavg = wt[0] = 0;
  tatavg = tat[0] = bt[0];
  for (i=1 ; i<n;i++){
   { wt[i] = wt[i-1] +bt[i-1];
     tat[i]= tat[i-1] +bt[i];
     wtavg= wtavg + wt[i];
     tatavg= tatavg +tat[i];
   }
printf("\n process \t\t priority \t burst time\t waiting time\t
      turn around time ");
for(i=0; i<n ; i++){
    printf("\n %d \t\t %d \t\t %d \t\t %d\t\t %d", p[i],pri[i],bt[i],
          wt[i], tat[i]);
  }
printf("\n Average waiting time is %f", wtavg/n);
printf("\n Average turnaround time is %f", tatavg/n);
getch ();
}
```

#Lab5

Objective: C- Program to demonstrate FIFO page replacement.

Source code:

```
 # include <stdio.h>
 # include <conio.h>
  int fr[3];
  void main ()
 {
    void display ();
    int p[12] = {2,3,2,1,5,2,4,5,3,2,5,2}, i, j, fs[3];
    int index, k, l, flag1 =0, flag2 =0, pf =0, fsize =3;
    for (i=0; i<3; i++){
        for[i] =-1;
     }
    for (j=0; j<12 ; j++){
       flag1 =0; flag2 =0;
       for(i=0; i<3; i++){
           if (fr[i] == p[j]){
               flag1 = 1;
               flag2 = 1;
               break;
           }
       } if ~~(fr[i] =~~
      if (flag1 ==0){
        for( i=0; i<3; i++){
            if( fr[i] == -1){
                fr[i] = p[i];
                flag2 = 1;
                 break;
             }
        }
     }
     if (flag2 ==0){
      for (i=0; i<3; i++){
          fs[i] =0;
          for (k=j,-1, l=1, l<= frsize-1; l++, k--){
```

```
   for(i=0; i<3; i++){
      if( fr[i] = = p[k] ){
           fs[i] = j;}
     }
   for(i=0; i<3 ;i++){
      if( fs[i] = =0){
          index =i;
        }
     }
    ~~for~~ fr[index] = p[j];
        pf ++;
   }
   display ();
 }
 printf("\n No. of page faults: %d", pf+ frsize);
  getch();
}
void display()
{  int i;
   printf("\n");
   for(i=0;i<3;i++){
    printf(" \t%d ", fr[i]);
     }
}
```

#Lab6

Objective: C-Program to demonstrate Optimal Page replacement.

source code:

```
 #include <stdio.h>
 # include <conio.h>
 int fr[3], n, m;
 void display ();
 void main ()
 {
   int i, j, page[20], fs[10];
   int max, found =0, lg [3], index, k, l, flag1=0, flag2 =0, pf=0;
   float pr;
   printf ("Enter length of reference string: ");
   scanf(" %d", &n);
   printf(" Enter the reference string: ");
   for(i=0; i<n ; i++) {
      scanf("%d", &page [i]);
      }
   printf("Enter no. of frames: ");
   scanf ("%d ", &m);
  for (i=0; i<m; i++){
     fr[i] = -1;
   }
  pf = m;
  for(j=0 ; j<n ;j++){
    flag1 =0;
    flag2 =0;
    for(i=0; i<m; i++) {
        if(fr[i] == page[j]){
            flag1 = 1;
            flag2 = 1;
        }  break;
    }
    if (flag1 = =0){
      for (i=0; i <m; i++){
          if( fr[i] = = -1){
              fr[i] = page [j];
              flag2 = 1;
          break;
        }
      }
  }
  if(flag2==0){
     for(i=0; i<m; i++){
         for(k=j+1; k<=n; k++){
             if(fr[i]==page[k]){
                 lg[i]=k-j;
                  break;
             }
         }
     }
     found=0;
     for(i=0; i<m; i++){
         if(lg[i]==0){
              index=i;
               found=1;
                break;
         }
     }
      if(found==0){
          max=lg[0];
          index=0;
          for(i=0; i<m; i++){
              if(max<lg[i]){
                   max=lg[i];
                   index=i;
              }
          }
      }
       fr[index]=page[j];
       pf++;
   }
   display();
  }
  printf("\n Number of page faults: %d", pf);
  pr=(float)pf/n*100;
```

```
 printf("page fault rate= %f \n", pr);
  getch ();
}
void display(){
   int i;
   for(i=0; i<m; i++){
       printf("%d\t", fr[i]); }
       printf("\n");
   }
```

#Lab 7

Objective: C-Program to demonstrate LRU page replacement.

source code:

```
 #include<stdio.h>
 # include <conio.h>
  int fr[3];
  void display();
  void main()
  {
    void display();
    int p[12] = {2,3,2,1,5,2,4,5,3,2,5,2}, i,j,fs[3];
    int index ,k,l, flag1=0 ,flag2=0 ,pf=0, frsize =3;
    for(i=0; i<3; i++){
        fr[i] =-1;
    }
    for(j=0; j<12 ; j++) {
      flag1 = 0, flag2=0;
      for(i=0;i<3; i++){
         if(fr[i]==p[j]){
             flag1= 1;
             flag2 =1;
             break;
         }
      }
      if(flag1 ==0){
          for(i=0; i<3 ;i++){
              if(fr[i]==-1) {
                  fr[i] = p[j];
                  flag2 =1;
                  break;
              }
          }
      }
      if(flag2==0){
          for(i=0; i<3;i++){
              fs[i]=0;
          for( k=j-1,l=1; l<=frsize-1; l++, k--) {
              for (i=0;i<3; i++) {
                  if(fr[i]==P[K]){
                        fs[i]=1;
                     }
                  }
            }
        }
      for(i=0;i<3;i++){
            if(fs[i]==0){
                  index=i;
            }
      }
      fr[index]=p[j];
            pf++;
   }
   display();
  }
  printf("\n No. of page fault: %d". pf + frsize);
  getch();
}
void display(){
     int i;
     printf("\n");
     for(i=0; i<3; i++){
         printf("\t%d", fr[i]);
     }
}
```

Lab 8

Objective: C Program for First Come First Serve Disk Scheduling.

Source code:

```
#include <stdio.h>
int main()
{
 int i, j, n, h, a[10], dm=0, rm, k, tdm;
 printf("Enter the no. of queue: ");
 scanf("%d", &n);
 printf("Enter the head of disk: ");
 scanf("%d", &h);
 if (n>10) {
   printf("Error: Number of queues exceed size of array");
   return 1;
  }
 for(i=0; i<n; i++) {
   printf("Enter number at %d: ", i+1);
   scanf("%d", &a[i]);
   if (h > a[0]) {
        k = h-a[0];
   elseif (h < a[0]) {
        k = a[0]-h;
    }
 }
 for(j=0 ; j<n-1 ; j++) {
   rm = a[j+1]-a[j];
   if (rm<0) {
       rm = -rm; }
   dm += rm;
   }
 tdm = dm+k;
 printf("Total disk moment: %d", tdm);
  return 0;
 }
```

#Lab9

Objective: C- Program for Shortest Seek Time first disk scheduling.

sourcecode:

```
  #include <stdio.h>
   struct head{
      int num;
      int flag;
    };
   int main(){
    struct head h[33];
     int array_1[33], array_2[33];
     int count=0, j, x, limit, minimum.location.disk_head,sum =0;
    printf("\n Enter total no. of locations:\t ");
   scanf("%d ", &limit);
    printf("\n Enter position of disk head:\t ");
    scanf("%d", &disk_head);
     printf("\n Enter elements of disk head queue\n ");
     while (count<limit)
     {
      scanf ("%d ",&h[count].num);
      h[count].flag=0;
      count ++;
      }
    for (count=0; count < limit; count ++){
       x =0;
       minimum =0, location =0 ;
       for (j=0 ;j <limit ; j++){
            if (h[j].flag ==0){
                if (x==0){
                   array_1 [j] =disk_head-h[j].num;
                   if (array_1[j] <0) {
                       array_1[j] =h[j].num -disk_head;
                    }
                   minimum = array_1[j];
                   location =j;
                   x++;
                }
              else {
                 array_1[j] = disk_head - h[j].num;
                  if(array_1[j] <0) {
                          array_1[j] = h[j].num - disk_head;
            }
         }
        if (minimum > array_1[j]){
            minimum= array_1[j];
             location=j;
         }
      }
  }
  h[location].flag=1;
  array_2[count]=h[location].num-disk_head];
  if (array_2[count] <0){
     array_2[count] =disk-head -h[location].num;
   }
   disk_head = h[location].num;

}
count =0;
while(count <limit ){
    sum = sum + array_2[count];
    count ++;
    }
  printf("\n Total movements of the cylinders:\t %d", sum);
  return 0;
}
```

Lab10

Objective: C-program for SCAN disk scheduling.

source code:

```
#include <stdio.h>
# include <conio.h>
  void scan_algo(int left[], int right[], int count, int limit)
  {
   int arr [20];
   int x= count -1, y= count+1, c=0, d=0, j;
   while (x> -1)
   {
     printf("\n x: \t %d", x);
     printf(("\n Left[x]: \t %d ", left[x]);
     arr [d] = left[x];
      x--;
      d++;
    }
   arr [d] =0;
   while (y <limit + 1)
   {
    arr[y] = right [c];
     c++;
     y++;
    }
   printf("\n Scanning Order: \n");
   for (j =0; j< limit +1; j++){
        printf("\n%d", arr [j]);
    }
 }
void division (int elements[], int limit, int disk_head){
    int count =0, p, q, m, x;
    int left [20], right [20];
    for (count =0; count <limit; count++){
      if (elements[count] > disk_head)
      {  printf("\n Break Position: \t %d\n", elements [count]);
         break;
      }
    }
   printf("\n value: \t %d\n", count);
   q = 1;
p=0;
m=limit;
left[0]=element[0];
printf("\n Left:\t %d", left[0]);
while(q < count)
 {
   printf("\n Element[1] value:\t %d", elements[q]);
   left[q]=elements[q];
   printf("\n Left:\t %d", left[q]);
       q++;
       printf("n1:\t %d", q);
  }
 r=count;
 while (r <m)
  {
   right[p]= elements[r];
   printf("\n Right:\t %d", right[p]);
   printf("\n Element:\t %d", elements[r]);
     p++;
     r++;
   }
 scan_algo(left, right, count, limit);
}
void sorting (int elements[], int limit)
 {
  int location, count, j, temp, small;
  for (count=0; count < limit-1; count++)
    { small = elements[count];
      location = count;
      for(j=count+1; j<limit; j++){
           if(small >elements[j]){
              small = elements [j];
               location= j;
           }
      }
     temp= elements [location];
     elements[location] = elements [count];
     elements [count] = temp;
    }
}
```

```
int main ()
{
int count, disk_head, elements[20], limit;
printf("Enter total number of locations: ");
scanf("%d", &limit);
printf("\n Enter position of disk head: ");
scanf("%d", &disk_head);
printf("\n Enter elements of disk head queue: ");
 for(count=0; count<limit; count++)
  {
   printf("Element[%d]: ", count+1);
   scanf("%d", &elements[count]);
  }
sorting(elements, limit);
division(elements, limit, disk_head);
getch();
return 0;
}
```