Lumbini City College

(Affiliated to Tribhuvan University)

Tilottama-04 Rupandehi



LAB ASSIGNMENT (2080)

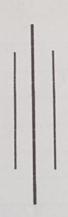
Course code: CACS 251

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

S.N.	Page	Topics	Date	Signature
1.		CPU Scheduling Algorithm	2080/08/22	
	01	a. FCFS		
	02-03	b. SJF		
	04-05	c. Round Robin		
	06-08	d. Priority		
2.		Page Replacement Algorithm	2080/10/24	
	09-10	a. fIFO		
	11-13	b. Optimal Page Replacement		
	14-15	b. Optimal Page Replacement c. LRU Page Replacement		
3.		Disk Scheduling Algorithm	2080/11/08	
	16-	a. FCFS		
	17-18	b. SSTF		
	19 - 21	c. SCAN		

Name of subject teacher:	Final Remark
Rahul Shakya	

```
Objective: First Come First Served (FIFO)/FCFS scheduling
           algorithm in c.
Source cod:
  # include <stdio.hs
    int maines
   int n, b+[20], w+[20], ta+[20], avw+=0, avta+=0, i,j;
   print f(" Enter total no. of process [max=20]: ");
   scont("%d", 4n);
    print + ("In Enter Process Burst Time In");
    tor(1=0;12n;1++){
      print f(" p[%d]; ", i+1);
     scont(" "/d", 16+[1]);
     4
    w1 [0] = 0',
    for (j=0; j <in; j++) {
      w+[i]+=b+[j];
    printf("In Ancess 1+1+ Brust time It Waiting time It Turn oround
            time ");
     for (1=0; i2n; i++) s
       tad[i] = b+[i] + w+[i];
      avwill+= WI[i];
      avtat += tad[i];
      Print f ("In P[%d])+\+ %d\+\+ %d\+\+ %d", 1+1,b+Ei], w+Ei],
                 tat [i]);
        3
      avw1/=1;
      avtal /= i:
      point ("Inin average waiting time: ",d", avwi);
      print # ("In average turn oround time; "id", avtad);
       return 0'.
```

```
Objective: Shortest Job first (SJF) scheduling algorithm in c.
source cod:
 #include Lstdion)
  void main ()
   int bt[20], p[20], wt[20], tat[20], i,j, n, total =0, pos, temp;
   flood ovg. wt, ovg. tat;
   printf(" Enter number of process! ");
   sconf(" "/od", 4n):
   print if ("In Enter brust time: \n");
   tot (1=0; i <n; i++)
        print + (" p'xd:", i+1);
        Scant ("%d", 4 b+[1]);
        P[i] = i+1;
    tor ( i=0; ixn; i++) {
        DOS = 1';
       tor(j=1;j(n;j++) {
         ia (b+[j] (b+[pos])
               pas = j;
       temp = bt[i];
       bt[i] = bt[pas];
       b+ [pos] = temp;
       pemp = P[i];
       p[i] = p[pos];
        P[pos] = temp;
      w+ [0] =0;
     for (i=1) i < n; i++) {
       wat [i] = 0;
        408(j=0)j/1;j++) {
           :[[j]td = +[i]tw
           total + = wot[i];
   avg- iot: (Hood) total/n;
```

```
ang-wt= (#load) total /n',

total =0',

printd("In Process It Brust time It waiting time It Turn oround");

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

tat [i]=bt [i]+wt [i];

total += tat [i];

printd("In p'hal It It had It It had It It had ", p [i], b + [i], wt [i], tat [i]);

ang-tat= (#load) total/n;

printd ("In n exercise waiting time = "ht", ong-wt);

printd("In exercise turn around time = "ht", ong-tat);

3
```

```
Objective: Round Robin (RR) scheduling algorithm in c.
source code:
 # include <5+dio.h)
  void main ()
   int i,j, bu[10], wa[10], tat[10], c+[10], +, max',
   fload aw =0, att =0, temp=0.
   print+ (" Enter the no. of process: ");
   scont(" "d", fn);
    for(1=0;120;1++)$
      print+("In Enter Brust time for process %d: "i+1);
      sconf(" " d", 4 bu [i]);
       CALIJ : bolij',
     printfl" in Enter the size of time slice: "):
    scant (" "3d", + e+)",
     max = bu[o];
      dor (1:1; ikn; i++) {
         iz (max < buli]) {
              max = bu[i];
         3
  for (j=0; j((max/+)+1; j++) {
      for (1:0; 120; 1+1) {
            ia (bu[i]!=0) {
               13 (bu[ 1] 1=+) §
                 tat[i]:temp + bu [i];
                 temp = temp + bulij;
                 buli] 20
              3
             dsex
                bu [i] = bu[i] -+;
                  temp = temp ++;
              3
          3
      3
```

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

wa[i]: +at[i]:
at+ + = +at[i];
aw+ = wa[i];
}

print f("\n xverage tuin or own time is ".t", at+/n);
print f("\n xverage waiting time is ".t", awt/n);

print f("\n xverage waiting time is ".t", awt/n);

print f("\n\t process \+ brust time \+ waiting time \+ tuin or own d'n");

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

print f("\t ".t ".t \+".t \+".t \+".t \+".t \-".t \".i \+".t \-".t \-".i \+".t \-".t \-".i \+".t \-".t \-".i \-".i \-".i \+".t \-".t \-".i \-".i
```

```
algorithm for Priority scheduling in e.
Objective:
source code:
# include < stdio:n>
   void main ()
   ind p[20], 6+[20], poi [20], wol [20], tat [20], i, k, n, temp;
   flood wtavg, tatavg;
   print+(" Enter the no. of process: ")",
   scont("hd", +n);
   for (1:0; izn; i+1) {
         P[i]=i?
         print+(" Enter the brust time of priority of process "d; "i);
         Scont (" 12d 1/d", 4 b+[i], + pin[i]),
   tor (1=0;1<0 1;1++) {
      for ( K= 1+1; K<n; K++) {
            if (prin [i] > pri [k] {
                 temp = p [i];
                 PLIJ : P[K];
                 P[K]: temp!
                 temp = b+[i];
                 b+[1] = b+[K]:
                 bf[K] = temp.
                 temp = pri [ kJ;
                 prilij: pri[x];
                 pri[k] : temp;
          3
    wtong = w+[0]=0;
    tatang : tat co] = 6+ co];
    for (i=1; 320; i++) &
      Q WALIJ = MALI-1] +BALI-1];
        talli]: tat[i-1] +b+[i];
        wtayg = wtayg + wt [i];
        tadang = tatang +tat [i];
```

```
print + (" \n process \t\t priority \t burst time\t waiting time\t

turn oround time ");

tor(i=0; i<n; i+t) \( \)

print + (" \n in it t) \( \)

print + (" \n in it t) \( \)

print + (" \n average waiting time is "it", whove in );

print + (" \n average turn oround time is "it", tatary in);

getch ();

2
```

```
Objective: C- Program to demonstrate FIFO page replacement.
source codec
 # include (stdio.h)
 # include comonhy
   int 18[3];
   voids main ()
     void display ();
      int p[12] = { 2,3,2,1,5,2,4,5,3,2,5,23,1,1,45[3];
      int index, K, J, Hag2 =0, Hag2 =0, Pt =0, tsize =3;
      tor (1=0; 1<3;1++) {
          for [i] =- 1;
     for (j=0;j<12;j+1)5
        Hag 1 = 0; Hag 2 = 0;
         tor(1:0;123; 1++) {
              iz (+r [i] == p[j]) s
                  11091=1;
                  flag2 = 1;
                  break!
          3 in CAREIT:
        iz ( flag 2 = = 0) {
           dor(1:0;123;1++) {
              13( Ar [i] = = -1) {
                 Ar [i] = p[i],
                  flag 2 = 1;
                break;
          4
        iz ( flog 2 = = 0) {
          for (1=0;123;1++) {
              45[i] =0 ;
             for (K=j,-1, J=1,1K= frsize-1; J++, K--) {
```

```
for (1:0;123;1++) &
       ig( {r [i] = = P[K] ) {
             15[i] = j:3
    dor(1:0: 123 :1++) {
       ig ( 45 [i] = 20) {
         3 index si;
    ft]q = [x shori] xt rot
         P# ++'
   display ();
 Pront & ("In No. of page gaults: 1/d", pt+ trsize );
  getch ();
void display ()
   ind is
   prind+("in");
   for(1=0;123;14+) {
    print f(" \ + 1/2 d", fr[1]);
3
```

```
Objective: C- Program to demonstrate Optimal page resplacement.
source code;
  #include <stdio.hs
  # include Lconio. hs
  int 18[3], n, m;
  void display (),
   void main (
    int i,j, page[20], ts[20];
    int max, found =0, 1g [3], index, k, 1, #1ag1=0, #1ag2=0, pt=0;
    float pr;
    print # (" Enter length of reference storing: ");
    scong(" ".d", 4n)",
    print+1" Ents the regerence stolog: ");
     for(i=0; ixn; i++) {
        scont (" ", d", 1 page [i])",
     print+ ("Enter no. of frames: ");
    scont (" ",d ", +m)",
    tor (1=0; ixm; i++) {
        Ar[i] = -1;
    Pf=m;
   for (j=0; j<n;j++){
      flag1 =0',
      Alag2 =0',
      tor(1=0;1<m;1++) {
           iz(tr[i] == page[j]) {
               Hag 1=1'
               Hag 2 = 1;
               break?
      iz ( + 1 ag 1 = = 0) {
         for (1=0312m;1++) {
              ix ( fr [ i] == -1) {
                   to [i] = page [j];
                   flag2 = 1;
```

```
break;
    4
   3
3
iz ( +1992 == 0) {
   tor (1=0; 12m; 1++) &
       tor(k= j+1; K(=n; K++) {
           iz(ACi] == page [K]) s
                 1g [i] = k-j's
                  break;
         3
    3
   tound = 0'
   for (1:0; 12m; 1++) {
       iz (dg[i] == 0) {
             index = i',
              tound = 1;
               break.
    iz (tound = = 0) &
        max = 1g [O];
         index =0'
         tor (100;12m;1++) {
             iz (max 2 lg[i]) {
                   max = 1g[i];
                    index = i;
               4
         3
      dr [index] = page[j];
       p+++2,
  display ()',
 print + ("In Humber of page faults; ",d", p+);
  pr = ( $100d) p $ / n * 100;
```

```
print + ("page tault rate = 1/2+ \n", pr);

getch ();

void display () {

int i;

foo (i=0; izm; i++) {

print + ("?.d\+", fr [i]); }

print + ("?.d\+", fr [i]); }

print + ("\n");
```

```
Objectives C-program to demonstrate IRU page replacement.
source code;
 #include < stdio. hs
 # include (whio ) hs
   int to [3];
   void display ();
   void main ()
     void display ();
      int p[12] = { 2,3, 2, 1, 5, 2, 4, 5, 3, 2,5, 23, 1, 1, $563];
      int index, k, 1, +1991=0, +1992=0, Pd=0, trsize=3',
      tor(1=03 123; 1++) {
           88 Ei7 = -1;
     for (j=0', j<12; j++) $
       dlag1 = 0, Hag2 = 0;
        for(1=0;123:1++){
           }([t]q==[[]]){i
                flag 1: 1;
                dlag 2 = 1;
              3 break;
       iz (+lag1 == 0) {
             for (1=0; 143; 1++) {
                  ia (41[1]==-1) {
                       dreij = PEj];
                       Hag2 = 1;
                       break;
               3 3
        in (Hag2 = = 0) {
              for (1:0; 123; 1+1) {
                    $S[i] =0;
                 for ( K=j-1,1=1; 1<= trsize-1:1++, K--) {
                        for (1=0;123;1++) {
```

```
8 ( LX ] = = P[K]) {
                      AS[1]=1;
      3
     for (1=0;123;1++) {
            13 (45 [1] ==0) {
                 index = i,
      Ar Cindex] = p[j];
            P#++3
  3
   display ();
  print+ ("in No. of page fault: "id". pt + tosize ),
   getch ();
3
void disploye) {
     inti;
      print d ("\n");
      tor(i=0; 123; 1++) {
         print+ ("\+ %d", to[i]);
      3
3
```

```
Objective: C- Program zor First Come First Serve Disk scheduling
Source code:
  #include Latdions
   int main ()
   înt i,j, n, h, a[10], dm=0, rm, k, tdm;
   print A(" Enta the no. of queue; ");
   sconf (" "hd", 4 n);
   print of " Enter the head of disk: ").
   Sconf (" "d", 4h);
   iz (n>10) {
     print+("Error: Number of queues exceed size of orray");
      redurn 1:
   for(i=0; i < n; i++) $
     print f("Enter number of ".d?", "+1);
    scont(" " d', 4 a [ i ] );
    if (h> a[0]) {
          K= h-a[0];
     elseif (h < a[0]) {
         k: alog-h;
  4
  for (j=0 ; j < n-1 ; j++) {
    rm = a[j+1]-a[j];
    if (rm <0) {
        rm = - rm; 3
     dm + = rm;
  tdm = dm + K;
  print+ (" Total disk moment: ".d", tdm);
   return o',
```

```
Objective: C- Program for Shortest Seek Time first disk scheduling
sourcecode:
  # include 2stdions
   Struct head &
      int hum:
     int flag;
    int main() {
     struct head h [33]:
      int orray -1 [33], array -2 [33];
      int count = 0, j, m. limit, minumum. location. disk-head, sum =0;
     print H" \n Enter total no of locations: \4");
     scant ("%d",4 limit);
      print#("In Enter position of disk head: \1");
      scant ("%d", tdisk-head)
      printf("In Enter elements of disk head queue In")",
      while (count < limit)
       scont ("%d", th [count] num);
        h [count ]. flag = 0;
        count ++',
      for (count=0; count < limit; count ++) {
          minimum =0,10 cation=0;
          for ([j=0 ;j < limit; j++) {
              A (h[j] . 1/09 ==0) {
                 12(8==0)5
                    array-1 [j]=disk-head-h[j].num;
                     if (01804-1[j] <0) {
                        orroy - 1 [j] = h[j] · num - disk head;
                      minimum = array_1[j];
                      location: j;
                      X++;
                 Use {
                    array_1[j] = disk_head - h[j].num;
                     iflarray_1[j] 10) {
                            orray-1[j]=h[j]·num-disk-head;
```

```
3
           if (minimum > array_1[j]) {
                minimum= array_1[j],
                 lo cation=j;
          3
    h [location]. Hag = 1;
    array-2 [count]= h[location].num-disk_head];
     if (orray-2 [count ] 20) {
        arroy-2 [count] = disk-head - h [10 cation] num;
     disk_head = h[location].num;
count = 0',
 while (count < 1 imit ) {
     sum = sum + array 2 [count];
     count ++ ",
   print+ ("In Total movements of the cylinders: \4 %d", sum)-
   return D;
3
```

```
Objective: C-program for SCAN disk scheduling,
source code:
 #include Lotdions
 # include Loonio. hs
   void scan_algolint lest [], int right [], int count, int limit]
    ind orr [207'
     int on= count -1, y= count+1, c=0, d=0, j;
     while ( 3) -1)
       print + (" In x: \+ 1.d", n);
       point+(("nleft[x]: H %d", left[x]);
       arr [d]= legt[n];
        37 -- :
        d++',
    arr [d] =0;
     while (y < limit + 1)
      arty] = right [c];
       C++3,
       y++',
     print + ("In Scanning Order: In");
     tor (j=0;j< limit +1; j++) {
         print # ("In %d", ar [j]);
  3
 void division (Int dements [], Int limit, int disk-head) {
     int count = 0, piq i m, m',
     int kgt [20], night [20];
     for (count =0', count alimit; count ++) §
       if (elements[count] > disk_head)
            print+("In Break Position: \1% dyn", element [count]);
            break;
    print H"Invalue: 17 % d In ", count );
     9 = 21
```

```
P=0',
 m= limit;
 k ft[0] = element[0];
print & ("In Left: It ".d", left[o]),
 while (2 4 count)
   print + (" In Element [t] value: \t ".d", elements [9]);
   left[9] = elements[9];
    print + ("In lest: 4 %d", lest [9]);
       print+("n1: 1+ %, d", 9);
 on = count;
 while ( & Lm)
    night[p]= elements[h],
    point & ("In Alght: It %d", night[p]);
     print + ("In Element: It ", elements [m]);
      P++ ;
      2044;
 scan_algo (left, night, count, limit);
3
void sorting (int elements[], int limit)
   int location, count is, temp, small;
   tor (count =0', wount < limit-1', wount ++)
     small = elements [count];
       location = count;
        for (j=count+1; j < limit) j++) {
            il (small > elements [j]) {
               small : element [j];
                location = j;
       temp: elements [location];
       elements[location] = elements [count];
        elements [count] = temp;
3
```

```
int main ()

int wunt, disk_head, elements[20], limit;

print+(" Enter total number of location: ");

scant(" "d", flimit);

print+(" In Enter position of disk head: ");

scant(" "d", f disk_head);

print+(" In Enter elements of disk head queue: ");

for (wunt = 0; count < limit; count ++)

E print+(" Element["d]; "! count+1);

scant(" ",d", f elements[wunt]);

division (elements, limit, disk_head);

getch ();

return 0;
```