# Canara Engineering College
## Benjanapadavu


# OPERATING SYSTEMS LABORATORY MANUAL

Course Code – 21CB42

PREPARED BY

MRS. YOJANA K.S.

MR. PRAAHAS AMIN

MR. VINAY P

1. Simulate the following CPU scheduling algorithms. a) FCFS b) SJF c) Round Robin d) Priority.
2. Write a C program to simulate producer-consumer problem using Semaphores
3. Write a C program to simulate the concept of Dining-philosophers problem.
4. Write a C program to simulate the following contiguous memory allocation Techniques a) Worst fit b) Best fit c) First fit.
5. Simulate all page replacement algorithms a)FIFO b) LRU c) OPTIMAL
6. Simulate all File Organization Techniques a) Single level directory b) Two level directory
7. Simulate all file allocation strategies a) Sequential b) Indexed c) Linked.
8. Simulate Bankers Algorithm for Dead Lock Avoidance.
9. Simulate Bankers Algorithm for Dead Lock Prevention.
10. Write a C program to simulate disk scheduling algorithms. a) FCFS b) SCAN c) C-SCAN

1. Simulate the following CPU scheduling algorithms. a) FCFS b) SJF c) Round Robin d) Priority.

**a. FCFS**

```
#include <stdio.h>
#include <conio.h>
int main()
{
   int bt[20], wt[20], tat[20], i, n;
   float wtavg, tatavg;
   printf("\nEnter the number of processes -- ");
   scanf("%d", &n);
   for (i = 0; i < n; i++)
   {
      printf("\nEnter Burst Time for Process %d -- ", i);
      scanf("%d", &bt[i]);
   }
   wt[0] = wtavg = 0;
   tat[0] = tatavg = 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("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
   for (i = 0; i < n; i++)
      printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
   printf("\nAverage Waiting Time -- %f", wtavg / n);
   printf("\nAverage Turnaround Time -- %f", tatavg / n);
   getch();
}
```

**b. SJF**

```
#include <stdio.h>
#include <conio.h>
int main()
{
   int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
   float wtavg,
      tatavg;
   printf("\nEnter the number of processes -- ");
   scanf("%d", &n);
   for (i = 0; i < n; i++)
   {
      p[i] = i;
      printf("Enter Burst Time for Process %d -- ", i);
      scanf("%d", &bt[i]);
   }
   for (i = 0; i < n; i++)
      for (k = i + 1; k < n; k++)
         if (bt[i] > bt[k])
         {
            temp = bt[i];
            bt[i] = bt[k];
            bt[k] = temp;
            temp = p[i];
            p[i] = p[k];
            p[k] = temp;
         }
   wt[0] = wtavg = 0;
   tat[0] = tatavg = 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\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
   for (i = 0; i < n; i++)
      printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
   printf("\nAverage Waiting Time -- %f", wtavg / n);
   printf("\nAverage Turnaround Time -- %f", tatavg / n);
   getch();
}
```

**c. Round Robin**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
   float awt = 0, att = 0, temp = 0;
   printf("Enter the no of processes -- ");
   scanf("%d", &n);
   for (i = 0; i < n; i++)
   {
      printf("\nEnter Burst Time for process %d -- ", i + 1);
      scanf("%d", &bu[i]);
      ct[i] = bu[i];
   }
   printf("\nEnter 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("\nThe Average Turnaround time is -- %f", att / n);
   printf("\nThe Average Waiting time is -- %f ", awt / n);
   printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
   for (i = 0; i < n; i++)
      printf("\t%d \t %d \t\t %d \t\t %d \n", i + 1, ct[i], wa[i], tat[i]);
   getch();
}
```

2. Write a C program to simulate producer-consumer problem using Semaphores

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int buffer[10], bufsize, in, out, produce, consume,
      choice = 0;
   in = 0;
   out = 0;
   bufsize = 10;
   while (choice != 3)
   {
      printf("\n1.Produce \t 2. Consume \t3.Exit");
      printf("\nEnter your choice: ");
      scanf("% d", &choice);
      switch (choice)
      {
      case 1:
         if ((in + 1) % bufsize == out)
            printf("\nBuffer is Full");
         else
         {
         }
         break;
         ;
         ;
         printf("\nEnter the value: ");
         scanf("% d", &produce);
         buffer[in] = produce;
         in = (in + 1) % bufsize;
      case 2:
         if (in == out)
            printf("\nBuffer is Empty");
         else
         {
            consume = buffer[out];
            printf("\nThe consumed value is % d", consume);
            out = (out + 1) % bufsize;
         }
         break;
      }
   }
}
```

3. Write a C program to simulate the concept of Dining-philosophers problem.

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

#ifdef _WIN32
#include <Windows.h>
#else
#include <unistd.h>
#endif
#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (phnum + 4) % N
#define RIGHT (phnum + 1) % N

int state[N];
int phil[N] = {0, 1, 2, 3, 4};

sem_t mutex;
sem_t S[N];

void test(int phnum)
{
   if (state[phnum] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING)
   {
      // state that eating
      state[phnum] = EATING;

      Sleep(2);

      printf("Philosopher %d takes fork %d and %d\n",
            phnum + 1, LEFT + 1, phnum + 1);

      printf("Philosopher %d is Eating\n", phnum + 1);

      // sem_post(&S[phnum]) has no effect
      // during takefork
      // used to wake up hungry philosophers
      // during putfork
      sem_post(&S[phnum]);
   }
}

// take up chopsticks
void take_fork(int phnum)
{

```

```
49.         sem_wait(&mutex);
50.
51.         // state that hungry
52.         state[phnum] = HUNGRY;
53.
54.         printf("Philosopher %d is Hungry\n", phnum + 1);
55.
56.         // eat if neighbours are not eating
57.         test(phnum);
58.
59.         sem_post(&mutex);
60.
61.         // if unable to eat wait to be signalled
62.         sem_wait(&S[phnum]);
63.
64.         Sleep(1);
65.      }
66.
67.      // put down chopsticks
68.      void put_fork(int phnum)
69.      {
70.
71.         sem_wait(&mutex);
72.
73.         // state that thinking
74.         state[phnum] = THINKING;
75.
76.         printf("Philosopher %d putting fork %d and %d down\n",
77.               phnum + 1, LEFT + 1, phnum + 1);
78.         printf("Philosopher %d is thinking\n", phnum + 1);
79.
80.         test(LEFT);
81.         test(RIGHT);
82.
83.         sem_post(&mutex);
84.      }
85.
86.      void *philosopher(void *num)
87.      {
88.
89.         while (1)
90.         {
91.
92.            int *i = num;
93.
94.            Sleep(1);
95.
96.            take_fork(*i);
97.
```

```
98.             Sleep(0);
99.
100.            put_fork(*i);
101.        }
102.    }
103.
104.    int main()
105.    {
106.
107.        int i;
108.        pthread_t thread_id[N];
109.
110.        // initialize the semaphores
111.        sem_init(&mutex, 0, 1);
112.
113.        for (i = 0; i < N; i++)
114.
115.            sem_init(&S[i], 0, 0);
116.
117.        for (i = 0; i < N; i++)
118.        {
119.
120.            // create philosopher processes
121.            pthread_create(&thread_id[i], NULL,philosopher, &phil[i]);
122.
123.            printf("Philosopher %d is thinking\n", i + 1);
124.        }
125.
126.        for (i = 0; i < N; i++)
127.
128.            pthread_join(thread_id[i], NULL);
129.    }
```

4. Write a C program to simulate the following contiguous memory allocation Techniques

**a) Worst fit**

```
#include <stdio.h>
#include <conio.h>
#define max 25
void main()
{
   int frag[max], b[max], f[max], i, j, nb, nf, temp;
   static int bf[max], ff[max];
   printf("\n\tMemory Management Scheme - First Fit");
   printf("\nEnter the number of blocks:");
   scanf("%d", &nb);
   printf("Enter the number of files:");
   scanf("%d", &nf);
   printf("\nEnter the size of the blocks:-\n");
   for (i = 1; i <= nb; i++)
   {
      printf("Block %d:", i);
      scanf("%d", &b[i]);
   }
   printf("Enter the size of the files :-\n");
   for (i = 1; i <= nf; i++)
   {
      printf("File %d:", i);
      scanf("%d", &f[i]);
   }
   for (i = 1; i <= nf; i++)
   {
      for (j = 1; j <= nb; j++)
      {
         if (bf[j] != 1)
         {
            temp = b[j] - f[i];
            if (temp >= 0)
            {
               ff[i] = j;
               break;
            }
         }
      }
      frag[i] = temp;
      bf[ff[i]] = 1;
   }
   printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
   for (i = 1; i <= nf; i++)
      printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
   getch();
}
```

**b) Best fit**

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  #define max 25
4.  void main()
5.  {
6.     int frag[max], b[max], f[max], i, j, nb, nf, temp, lowest = 10000;
7.     static int bf[max], ff[max];
8.     printf("\nEnter the number of blocks:");
9.     scanf("%d", &nb);
10.    printf("Enter the number of files:");
11.    scanf("%d", &nf);
12.    printf("\nEnter the size of the blocks:-\n");
13.    for (i = 1; i <= nb; i++)
14.       printf("Block %d:", i);
15.    scanf("%d", &b[i]);
16.    printf("Enter the size of the files :-\n");
17.    for (i = 1; i <= nf; i++)
18.    {
19.       printf("File %d:", i);
20.       scanf("%d", &f[i]);
21.    }
22.    for (i = 1; i <= nf; i++)
23.    {
24.       for (j = 1; j <= nb; j++)
25.       {
26.          if (bf[j] != 1)
27.          {
28.             temp = b[j] - f[i];
29.             if (temp >= 0)
30.                if (lowest > temp)
31.                {
32.                   ff[i] = j;
33.                   lowest = temp;
34.                }
35.          }
36.       }
37.       frag[i] = lowest;
38.       bf[ff[i]] = 1;
39.       lowest = 10000;
40.    }
41.    printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
42.    for (i = 1; i <= nf && ff[i] != 0; i++)
43.       printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
44.    getch();
45. }
```

**c) First Fit**

```
#include <stdio.h>
#include <conio.h>
#define max 25
void main()
{
   int frag[max], b[max], f[max], i, j, nb, nf, temp, highest = 0;
   static int bf[max], ff[max];
   printf("\n\tMemory Management Scheme - Worst Fit");
   printf("\nEnter the number of blocks:");
   scanf("%d", &nb);
   printf("Enter the number of files:");
   scanf("%d", &nf);
   printf("\nEnter the size of the blocks:-\n");
   for (i = 1; i <= nb; i++)
   {
      printf("Block %d:", i);
      scanf("%d", &b[i]);
   }
   printf("Enter the size of the files :-\n");
   for (i = 1; i <= nf; i++)
   {
      printf("File %d:", i);
      scanf("%d", &f[i]);
   }
   for (i = 1; i <= nf; i++)
   {
      for (j = 1; j <= nb; j++)
      {
         if (bf[j] != 1) // if bf[j] is not allocated
         {
            temp = b[j] - f[i];
            if (temp >= 0)
               if (highest < temp)
               {
               }
         }
         frag[i] = highest;
         bf[ff[i]] = 1;
         highest = 0;
      }
      ff[i] = j;
      highest = temp;
   }
   printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragement");
   for (i = 1; i <= nf; i++)
      printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
   getch();
}
```

5. Simulate all page replacement algorithms

**a) FIFO**

```
#include <stdio.h>
#include <conio.h>
int fr[3];
void main()
{
  void display();
  int i, j, page[12] = {2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2};
  int flag1 = 0, flag2 = 0, pf = 0, frsize = 3, top = 0;
  for (i = 0; i < 3; i++)
  {
    fr[i] = -1;
  }
  for (j = 0; j < 12; j++)
  {
    flag1 = 0;
    flag2 = 0;
    for (i = 0; i < 12; i++)
    {
      if (fr[i] == page[j])
      {
        flag1 = 1;
        flag2 = 1;
        break;
      }
    }
    if (flag1 == 0)
    {
      for (i = 0; i < frsize; i++)
      {
        if (fr[i] == -1)
        {
          fr[i] = page[j];
          flag2 = 1;
          break;
        }
      }
    }
    if (flag2 == 0)
    {
      fr[top] = page[j];
      top++;
      pf++;
      if (top >= frsize)
        top = 0;
    }
    display();
  }
  printf("Number of page faults : %d ", pf + frsize);
  getch();
}
void display()
{
  int i;
```

```
54.             printf("\n");
55.             for (i = 0; i < 3; i++)
56.                 printf("%d\t", fr[i]);
57.         }
```

**b) LRU**

```
#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, 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 faults :%d", pf + frsize);
        getch();
    }
    void display()
    {
        int i;
        printf("\n");
        for (i = 0; i < 3; i++)
            printf("\t%d", fr[i]);
    }
```

**c) OPTIMAL**

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  int fr[3], n, m;
4.  void display();
5.  void main()
6.  {
7.    int i, j, page[20], fs[10];
8.    int
9.      max,
10.     found = 0, lg[3], index, k, l, flag1 = 0, flag2 = 0, pf = 0;
11.   float pr;
12.   printf("Enter length of the reference string: ");
13.   scanf("%d", &n);
14.   printf("Enter the reference string: ");
15.   for (i = 0; i < n; i++)
16.     scanf("%d", &page[i]);
17.   printf("Enter no of frames: ");
18.   scanf("%d", &m);
19.   for (i = 0; i < m; i++)
20.     fr[i] = -1;
21.   pf = m;
22.   for (j = 0; j < n; j++)
23.   {
24.     flag1 = 0;
25.     flag2 = 0;
26.     for (i = 0; i < m; i++)
27.     {
28.       if (fr[i] == page[j])
29.       {
30.         flag1 = 1;
31.         flag2 = 1;
32.         break;
33.       }
34.     }
35.     if (flag1 == 0)
36.     {
37.       for (i = 0; i < m; i++)
38.       {
39.         if (fr[i] == -1)
40.         {
41.           fr[i] = page[j];
42.           flag2 = 1;
43.           break;
44.         }
45.       }
46.     }
47.     if (flag2 == 0)
48.     {
49.       for (i = 0; i < m; i++)
50.         lg[i] = 0;
51.       for (i = 0; i < m; i++)
52.       {
53.         for (k = j + 1; k <= n; k++)
54.         {
```

```
                    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("Number of page faults : %d\n", 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");
}
```

6. Simulate all File Organization Techniques

**a) Single level directory**

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
struct
{
  char dname[10], fname[10][10];
  int fcnt;
} dir;
int main()
{
  int i, ch;
  char
    f[30];
  dir.fcnt = 0;
  printf("\nEnter name of directory -- ");
  scanf("%s", dir.dname);
  while (1)
  {
    printf("\n\n1. Create File\t2. Delete File\t3. Search File \n4. Display Files\t5.Exit\nEnter your choice-- ");
    scanf("%d", &ch);
    switch (ch)
    {
    case 1:
      printf("\nEnter the name of the file -- ");
      scanf("%s", dir.fname[dir.fcnt]);
      dir.fcnt++;
      break;
    case 2:
      printf("\nEnter the name of the file -- ");
      scanf("%s", f);
      for (i = 0; i < dir.fcnt; i++)
      {
        if (strcmp(f, dir.fname[i]) == 0)
        {
          printf("File %s is deleted ", f);
          strcpy(dir.fname[i], dir.fname[dir.fcnt - 1]);
          break;
        }
      }
      if (i == dir.fcnt)
        printf("File %s not found", f);
      else
        dir.fcnt--;
      break;
    case 3:
      printf("\nEnter the name of the file -- ");
      scanf("%s", f);
      for (i = 0; i < dir.fcnt; i++)
      {
        if (strcmp(f, dir.fname[i]) == 0)
        {
```

```
53.                     printf("File %s is found ", f);
54.                     break;
55.                   }
56.                 }
57.                 if (i == dir.fcnt)
58.                   printf("File %s not found", f);
59.                 break;
60.               case 4:
61.                 if (dir.fcnt == 0)
62.                   printf("\nDirectory Empty");
63.                 else
64.                 {
65.                   printf("\nThe Files are -- ");
66.                   for (i = 0; i < dir.fcnt; i++)
67.                     printf("\t%s", dir.fname[i]);
68.                 }
69.                 break;
70.               default:
71.                 exit(0);
72.               }
73.           }
74.           getch();
75.       }
```

**b) Two level directory**

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
struct
{
   char dname[10], fname[10][10];
   int fcnt;
} dir[10];
int main()
{
   int i, ch, dcnt, k;
   char
      f[30],
      d[30];
   dcnt = 0;
   while (1)
   {
      printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");
      printf("\n4. Search File\t\t5. Display\t6. Exit\t Enter your choice --");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         printf("\nEnter name of directory -- ");
         scanf("%s", dir[dcnt].dname);
         dir[dcnt].fcnt = 0;
         dcnt++;
         printf("Directory created");
         break;
      case 2:
         printf("\nEnter name of the directory -- ");
         scanf("%s", d);
         for (i = 0; i < dcnt; i++)
            if (strcmp(d, dir[i].dname) == 0)
            {
               printf("Enter name of the file -- ");
               scanf("%s", dir[i].fname[dir[i].fcnt]);
               dir[i].fcnt++;
               printf("File created");
            }
         if (i == dcnt)
            printf("Directory %s not found", d);
         break;
      case 3:
         printf("\nEnter name of the directory -- ");
         scanf("%s", d);
         for (i = 0; i < dcnt; i++)
            for (i = 0; i < dcnt; i++)
            {
               if (strcmp(d, dir[i].dname) == 0)
               {
                  printf("Enter name of the file -- ");
                  scanf("%s", f);
```

```
                for (k = 0; k < dir[i].fcnt; k++)
                {
                  if (strcmp(f, dir[i].fname[k]) == 0)
                  {
                    printf("File %s is deleted ", f);
                    dir[i].fcnt--;
                    strcpy(dir[i].fname[k], dir[i].fname[dir[i].fcnt]);
                    goto jmp;
                  }
                }
                printf("File %s not found", f);
                goto jmp;
              }
            }
          printf("Directory %s not found", d);
        jmp:
          break;
        case 4:
          printf("\nEnter name of the directory -- ");
          scanf("%s", d);
          for (i = 0; i < dcnt; i++)
          {
            if (strcmp(d, dir[i].dname) == 0)
            {
              printf("Enter the name of the file -- ");
              scanf("%s", f);
              for (k = 0; k < dir[i].fcnt; k++)
              {
                if (strcmp(f, dir[i].fname[k]) == 0)
                {
                  printf("File %s is found ", f);
                  goto jmp1;
                }
              }
              printf("File %s not found", f);
              goto jmp1;
              printf("Directory %s not found", d);
            jmp1:
              break;
            case 5:
              if (dcnt == 0)
                printf("\nNo Directory's ");
              else
              {
                printf("\nDirectory\tFiles");
                for (i = 0; i < dcnt; i++)
                {
                  printf("\n%s\t\t", dir[i].dname);
                  for (k = 0; k < dir[i].fcnt; k++)
                    printf("\t%s", dir[i].fname[k]);
                }
              }
              break;
            default:
              exit(0);
```

```
110.                 }
111.             }
112.             getch();
113.         }
114.     }
115. }
```

7. Simulate all file allocation strategies

**a) Sequential**

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

int main()
{
   int f[50], i, st, j, len, c, k;
   for (i = 0; i < 50; i++)
      f[i] = 0;
X:
   printf("\n Enter the starting block & length of file");
   scanf("%d%d", &st, &len);
   for (j = st; j < (st + len); j++)
      if (f[j] == 0)
      {
         f[j] = 1;
         printf("\n%d->%d", j, f[j]);
      }
      else
      {
         printf("Block already allocated");
         break;
      }
   if (j == (st + len))
      printf("\n the file is allocated to disk");
   printf("\n if u want to enter more files?(y-1/n-0)");
   scanf("%d", &c);
   if (c == 1)
      goto X;
   else
      exit(0);
   getch();
}
```

**b) Indexed**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int f[50], i, k, j, inde[50], n, c, count = 0, p;
int main()
{
   for (i = 0; i < 50; i++)
      f[i] = 0;
x:
   printf("enter index block\t");
   scanf("%d", &p);
   if (f[p] == 0)
   {
      f[p] = 1;
      printf("enter no of files on index\t");
      scanf("%d", &n);
   }
   else
   {
      printf("Block already allocated\n");
      goto x;
   }
   for (i = 0; i < n; i++)
      scanf("%d", &inde[i]);
   for (i = 0; i < n; i++)
      if (f[inde[i]] == 1)
      {
         printf("Block already allocated");
         goto x;
      }
   for (j = 0; j < n; j++)
      f[inde[j]] = 1;
   printf("\n allocated");
   printf("\n file indexed");
   for (k = 0; k < n; k++)
      printf("\n %d->%d:%d", p, inde[k], f[inde[k]]);
   printf(" Enter 1 to enter more files and 0 to exit\t");
   scanf("%d", &c);
   if (c == 1)
      goto x;
   else
      exit(0);
   getch();
}
```

**c) Linked**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main()
{
   int f[50], p, i, j, k, a, st, len, n, c;
   for (i = 0; i < 50; i++)
      f[i] = 0;
   printf("Enter how many blocks that are already allocated");
   scanf(" % d", &p);
   printf("\nEnter the blocks no.s that are already allocated");
   for (i = 0; i < p; i++)
   {
      scanf("%d", &a);
      f[a] = 1;
   }
X:
   printf("Enter the starting index block & length");
   scanf(" % d % d", &st, &len);
   k = len;
   for (j = st; j < (k + st); j++)
   {
      if (f[j] == 0)
      {
         f[j] = 1;
         printf("\n%d->%d", j, f[j]);
      }
      else
      {
         printf("\n %d->file is already allocated", j);
         k++;
      }
   }
   printf("\n If u want to enter one more file ? (yes - 1 / no - 0)");
   scanf("%d", &c);
   if (c == 1)
      goto X;
   else
      exit(0);
   getch();
}
```

8. Simulate Bankers Algorithm for Dead Lock Avoidance.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
   int alloc[10][10], max[10][10];
   int avail[10], work[10], total[10];
   int i, j, k, n, need[10][10];
   int m;
   int count = 0, c = 0;
   char finish[10];
   printf("Enter the no. of processes and resources:");
   scanf("%d%d", &n, &m);
   for (i = 0; i <= n; i++)
      finish[i] = 'n';
   printf("Enter the claim matrix:\n");
   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         scanf("%d", &max[i][j]);
   printf("Enter the allocation matrix:\n");
   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         scanf("%d", &alloc[i][j]);
   printf("Resource vector:");
   for (i = 0; i < m; i++)
      scanf("%d", &total[i]);
   for (i = 0; i < m; i++)
      avail[i] = 0;
   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         avail[j] += alloc[i][j];
   for (i = 0; i < m; i++)
      work[i] = avail[i];
   for (j = 0; j < m; j++)
      work[j] = total[j] - work[j];
   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         need[i][j] = max[i][j] - alloc[i][j];
A:
   for (i = 0; i < n; i++)
   {
      c = 0;
      for (j = 0; j < m; j++)
         if ((need[i][j] <= work[j]) && (finish[i] == 'n'))
            c++;
      if (c == m)
      {
         printf("All the resources can be allocated to Process %d", i + 1);
         printf("\n\nAvailable resources are:");
         for (k = 0; k < m; k++)
         {
            work[k] += alloc[i][k];
            printf("%4d", work[k]);
         }
```

```
            printf("\n");
            finish[i] = 'y';
            printf("\nProcess %d executed?:%c \n", i + 1, finish[i]);
            count++;
          }
        }
        if (count != n)
          goto A;
        else
          printf("\n System is in safe mode");
        printf("\n The given state is safe state");
        getch();
    }
```

9. Simulate Bankers Algorithm for Dead Lock Prevention.

```
#include <stdio.h>
#include <conio.h>
void main()
{
   char job[10][10];
   int time[10], avail, tem[10], temp[10];
   int safe[10];
   int ind = 1, i, j, q, n, t;
   printf("Enter no of jobs: ");
   scanf("%d", &n);
   for (i = 0; i < n; i++)
   {
      printf("Enter name and time: ");
      scanf("%s%d", &job[i], &time[i]);
   }
   printf("Enter the available resources:");
   scanf("%d", &avail);
   for (i = 0; i < n; i++)
   {
      temp[i] = time[i];
      tem[i] = i;
   }
   for (i = 0; i < n; i++)
      for (j = i + 1; j < n; j++)
      {
         if (temp[i] > temp[j])
         {
            t = temp[i];
            temp[i] = temp[j];
            temp[j] = t;
            t = tem[i];
            tem[i] = tem[j];
            tem[j] = t;
         }
      }
   for (i = 0; i < n; i++)
   {
      q = tem[i];
      if (time[q] <= avail)
      {
         safe[ind] = tem[i];
         avail = avail - tem[q];
         printf("%s", job[safe[ind]]);
         ind++;
      }
      else
      {
         printf("No safe sequence\n");
      }
   }
   printf("Safe sequence is:");
   for (i = 1; i < ind; i++)
      printf("%s %d\n", job[safe[i]], time[safe[i]]);
   getch();
}
```

10. Write a C program to simulate disk scheduling algorithms.

**a) FCFS**

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

int main() {
   int queue[20], head, n, i, seekTime = 0;

   printf("Enter the number of disk requests: ");
   scanf("%d", &n);

   printf("Enter the disk request queue: ");
   for (i = 0; i < n; i++) {
      scanf("%d", &queue[i]);
   }

   printf("Enter the initial head position: ");
   scanf("%d", &head);

   printf("\n");

   printf("Seek Sequence: ");

   for (i = 0; i < n; i++) {
      printf("%d ", queue[i]);
      seekTime += abs(head - queue[i]);
      head = queue[i];
   }

   printf("\n\nTotal Seek Time: %d\n", seekTime);

   getch();
}
```

**b) SCAN**

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

int main() {
    int queue[20], head, n, i, j, seekTime = 0, direction, maxTrack;

    printf("Enter the number of disk requests: ");
    scanf("%d", &n);

    printf("Enter the disk request queue: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
    }

    printf("Enter the initial head position: ");
    scanf("%d", &head);

    printf("Enter the maximum track number: ");
    scanf("%d", &maxTrack);

    printf("Enter the direction (0 for left, 1 for right): ");
    scanf("%d", &direction);

    printf("\n");

    int temp;
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (queue[i] > queue[j]) {
                temp = queue[i];
                queue[i] = queue[j];
                queue[j] = temp;
            }
        }
    }

    int currentTrack = head;

    printf("Seek Sequence: ");

    if (direction == 0) { // Left
        for (i = head; i >= 0; i--) {
            printf("%d ", i);
            seekTime += abs(currentTrack - i);
            currentTrack = i;
        }
        printf("0 ");
        seekTime += currentTrack;

        for (i = 1; i <= maxTrack; i++) {
            printf("%d ", i);
            seekTime += abs(currentTrack - i);
            currentTrack = i;
        }
    } else { // Right
        for (i = head; i <= maxTrack; i++) {
            printf("%d ", i);
            seekTime += abs(currentTrack - i);
```

```
            currentTrack = i;
        }
        printf("%d ", maxTrack);
        seekTime += abs(currentTrack - maxTrack);

        for (i = maxTrack - 1; i >= 0; i--) {
            printf("%d ", i);
            seekTime += abs(currentTrack - i);
            currentTrack = i;
        }
    }

    printf("\n\nTotal Seek Time: %d\n", seekTime);

    getch();
}
```

**c) C-SCAN**

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

int main()
{
    int queue[20], head, n, i, j, seekTime = 0, maxTrack;

    printf("Enter the number of disk requests: ");
    scanf("%d", &n);

    printf("Enter the disk request queue: ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &queue[i]);
    }

    printf("Enter the initial head position: ");
    scanf("%d", &head);

    printf("Enter the maximum track number: ");
    scanf("%d", &maxTrack);

    printf("\n");

    int temp;
    for (i = 0; i < n - 1; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (queue[i] > queue[j])
            {
                temp = queue[i];
                queue[i] = queue[j];
                queue[j] = temp;
            }
        }
    }

    int currentTrack = head;

    printf("Seek Sequence: ");
```

```
43.
44.     // Scanning to the right
45.     for (i = head; i <= maxTrack; i++)
46.     {
47.        printf("%d ", i);
48.        seekTime += abs(currentTrack - i);
49.        currentTrack = i;
50.     }
51.
52.     // Moving to the beginning
53.     printf("%d ", maxTrack);
54.     seekTime += abs(currentTrack - maxTrack);
55.     currentTrack = 0;
56.
57.     // Scanning to the right again
58.     for (i = 0; i <= head; i++)
59.     {
60.        printf("%d ", i);
61.        seekTime += abs(currentTrack - i);
62.        currentTrack = i;
63.     }
64.
65.     printf("\n\nTotal Seek Time: %d\n", seekTime);
66.
67.     getch();
68.  }
```