Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester IV

26 March 2020

UCS1411 - Operating Systems Laboratory

Lab Exercise 9: Implementation of Paging Technique

Objective:

Develop a C program to implement the paging technique in memory management.

Code:

Q.To write a C program to implement the paging technique in memory management.

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

typedef struct{
```

```
7       int front,rear;
8       int data[100];
9       int capacity,size;
10  }Queue;
11
12  void initialise(Queue *q);
13
14  int isFull(Queue *q);
15
16  int isEmpty(Queue *q);
17
18  void enqueue(Queue *q,int x);
19
20  int dequeue(Queue *q);
21
22  void display(Queue *q);
23
24  void initialise(Queue *q){
25      q->front=q->rear=-1;
26      q->capacity=100;
27      q->size=0;
28  }
29
30  int isFull(Queue *q){
31      if((q->rear==q->capacity-1&&q->front==0)||(q->rear==q->
        front-1))
32          return 1;
33      else
34          return 0;
35  }
36
37  int isEmpty(Queue *q){
38      if(q->front==-1)
39          return 1;
40      else
41          return 0;
42  }
43
44  void enqueue(Queue *q,int x){
45      if(isFull(q))
46          printf("Queue is full ");
47      else{
48          if(q->front==-1)
49              q->front++;
50
```

```
51          if(q->rear==q->capacity-1)
52              q->rear=0;
53          else
54              q->rear++;
55
56          q->size++;
57          q->data[q->rear]=x;
58      }
59  }
60
61  int dequeue(Queue *q){
62      int x=0;
63      if(isEmpty(q))
64          return -1;
65      else{
66          x=q->data[q->front];
67          q->size--;
68          if(q->front==q->rear)
69              q->front=q->rear=-1;
70          else if(q->front==q->capacity-1)
71              q->front=0;
72          else
73              q->front++;
74      }
75      return x;
76  }
77
78  void displayQueue(Queue *q){
79      if(isEmpty(q))
80          printf("Queue is empty");
81      else{
82              int i=q->front;
83              while(i!=q->rear){
84                  printf("%d ",q->data[i]);
85
86                  if(i==q->capacity-1)
87                      i=0;
88                  else
89                      i++;
90              }
91
92              printf("%d ",q->data[i]);
93      }
94  }
95
```

```
96 //Check if a value is already in the queue
97 int checkQueue(Queue *q,int x){
98     if(isEmpty(q))
99         return 0;
100     else{
101         int i=q->front;
102         while(i!=q->rear){
103             if(x==q->data[i])
104                 return 1;
105 
106             if(i==q->capacity-1)
107                 i=0;
108             else
109                 i++;
110         }
111         if(x==q->data[i])
112             return 1;
113     }
114     return 0;
115 }
```

```
1 
2 #include "Queue.h"
3 
4 struct Job{
5     char *PID;
6     int size;
7     int no_of_pages;
8     int page_table[20];
9 };
10 
11 typedef struct Job Process;
12 
13 void initialiseProcess(Process *p){
14 
15     p->PID=(char*)malloc(sizeof(15));
16     p->size=0;
17     p->no_of_pages=0;
18     for(int i=0;i<20;i++)
19         p->page_table[i]=0;
20 }
21 
22 void acceptProcess(Process *p){
23     printf("\nEnter process ID: ");scanf(" %s",p->PID);
24     printf("\nEnter size of process: ");scanf("%d",&p->size);
25 }
```

```
26
27 void processDeallocation(Process p[],int *no_of_processes,
       char *PID,Queue *FreeFrameList){
28     int pno=0;
29     for(pno=0;pno<*no_of_processes;pno++){
30         if(strcmp(p[pno].PID,PID)==0)
31             break;
32     }
33     if(pno>=*no_of_processes)
34         printf("\nSpecified process not found. \n");
35     else{
36         for(int i=0;i<p[pno].no_of_pages;i++){
37             enqueue(FreeFrameList,p[pno].page_table[i]);
38         }
39
40         for(int i=pno;i<(*no_of_processes)-1;i++)
41             p[i]=p[i+1];
42         *(no_of_processes)--;
43     }
44 }
45
46 void main(){
47
48     srand(time(0));
49     int Mem_size;
50     int page_size;
51     int no_of_frames;
52     //Initial number of free frames
53     int noff;
54
55     Process p[10];
56
57     int ctr=0;
58
59     Queue FreeFrameList;
60     initialise(&FreeFrameList);
61
62     printf("\n Enter size of Physical Memory: ");scanf("%d",&
       Mem_size);
63     printf("\n Enter size of page: ");scanf("%d",&page_size);
64
65     if(Mem_size%page_size){
66         no_of_frames = (Mem_size/page_size)+1;
67     }
68     else{
```

```
69          no_of_frames = (Mem_size/page_size);
70      }
71
72      noff=rand()%15;
73
74      for(int i=0;i<noff;i++){
75          int x = rand()%no_of_frames;
76          if(!checkQueue(&FreeFrameList,x))
77              enqueue(&FreeFrameList,x);
78          else
79              i--;
80      }
81      int option;
82      do{
83
84          printf("\n Enter option: \n 1.Process Request \n 2.
    Deallocation ");
85          printf("\n 3.Display Page Table \n 4.Display Free
    Frame List ");
86          printf("\n 0.Exit \n Your choice: ");scanf("%d",&
    option);
87
88          if(option==1){
89              initialiseProcess(&p[ctr]);
90              acceptProcess(&p[ctr]);
91
92              if(p[ctr].size % page_size)
93                  p[ctr].no_of_pages = (p[ctr].size/page_size)
    +1;
94              else
95                  p[ctr].no_of_pages = (p[ctr].size/page_size);
96
97              for(int i=0;i<p[ctr].no_of_pages;i++){
98                  int x=dequeue(&FreeFrameList);
99                  if(x<0){
100                     printf("\nProcess cannot be accomodated
    fully.\n");
101                     break;
102                 }
103                 p[ctr].page_table[i]=x;
104             }
105             ctr++;
106         }
107         else if(option==2){
108             char *PID=(char*)malloc(sizeof(15));
```

```
            printf("\n Enter process ID to deallocate: ");
            scanf(" %s",PID);
            processDeallocation(p,&ctr,PID,&FreeFrameList);
        }
        else if(option==3){
            for(int i=0;i<ctr;i++){
                printf("\n Process %s: \n",p[i].PID);
                for(int j=0;j<p[i].no_of_pages;j++){
                    printf("\n Page %d ---> Frame %d\n",j,p[i
    ].page_table[j]);
                }
            }
        }
        else if(option==4){
            displayQueue(&FreeFrameList);
        }
        else if(option!=0){
            printf("\nInvalid option. \n");
        }
        else;

     }while(option);

}
```

Output:

```

 Enter size of Physical Memory: 32

 Enter size of page: 1

 Enter option:
 1.Process Request
 2.Deallocation
 3.Display Page Table
 4.Display Free Frame List
 0.Exit
 Your choice: 4
30 11 19 26 18 5 31 3 2
 Enter option:
 1.Process Request
 2.Deallocation
 3.Display Page Table
 4.Display Free Frame List
```

```
 0.Exit
 Your choice: 1

Enter process ID: P1

Enter size of process: 4

 Enter option:
 1.Process Request
 2.Deallocation
 3.Display Page Table
 4.Display Free Frame List
 0.Exit
 Your choice: 1

Enter process ID: P2

Enter size of process: 2

 Enter option:
 1.Process Request
 2.Deallocation
 3.Display Page Table
 4.Display Free Frame List
 0.Exit
 Your choice: 1

Enter process ID: P3

Enter size of process: 2

 Enter option:
 1.Process Request
 2.Deallocation
 3.Display Page Table
 4.Display Free Frame List
 0.Exit
 Your choice: 4
2
 Enter option:
 1.Process Request
 2.Deallocation
 3.Display Page Table
 4.Display Free Frame List
 0.Exit
```

```
Your choice: 3

Process P1:

Page 0 ---> Frame 30

Page 1 ---> Frame 11

Page 2 ---> Frame 19

Page 3 ---> Frame 26

Process P2:

Page 0 ---> Frame 18

Page 1 ---> Frame 5

Process P3:

Page 0 ---> Frame 31

Page 1 ---> Frame 3

Enter option:
1.Process Request
2.Deallocation
3.Display Page Table
4.Display Free Frame List
0.Exit
Your choice: 2

Enter process ID to deallocate: P2

Enter option:
1.Process Request
2.Deallocation
3.Display Page Table
4.Display Free Frame List
0.Exit
Your choice: 4
2 18 5
Enter option:
1.Process Request
2.Deallocation
```

```
3.Display Page Table
4.Display Free Frame List
0.Exit
Your choice: 3

Process P1:

Page 0 ---> Frame 30

Page 1 ---> Frame 11

Page 2 ---> Frame 19

Page 3 ---> Frame 26

Process P3:

Page 0 ---> Frame 31

Page 1 ---> Frame 3


Enter option:
1.Process Request
2.Deallocation
3.Display Page Table
4.Display Free Frame List
0.Exit
Your choice: 0
```