

# 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.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #include<time.h>
5
6 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));

```

```

109         printf("\n Enter process ID to deallocate: ");
110         scanf(" %s",PID);
111         processDeallocation(p,&ctr,PID,&FreeFrameList);
112     }
113     else if(option==3){
114         for(int i=0;i<ctr;i++){
115             printf("\n Process %s: \n",p[i].PID);
116             for(int j=0;j<p[i].no_of_pages;j++){
117                 printf("\n Page %d ---> Frame %d\n",j,p[i]
118 ].page_table[j]);
119             }
120         }
121     else if(option==4){
122         displayQueue(&FreeFrameList);
123     }
124     else if(option!=0){
125         printf("\nInvalid option. \n");
126     }
127     else;
128
129     }while(option);
130
131 }

```

### ***Output:***

```

1
2 Enter size of Physical Memory: 32
3
4 Enter size of page: 1
5
6 Enter option:
7 1.Process Request
8 2.Deallocation
9 3.Display Page Table
10 4.Display Free Frame List
11 0.Exit
12 Your choice: 4
13 30 11 19 26 18 5 31 3 2
14 Enter option:
15 1.Process Request
16 2.Deallocation
17 3.Display Page Table
18 4.Display Free Frame List

```

```

19 0.Exit
20 Your choice: 1
21
22 Enter process ID: P1
23
24 Enter size of process: 4
25
26 Enter option:
27 1.Process Request
28 2.Deallocation
29 3.Display Page Table
30 4.Display Free Frame List
31 0.Exit
32 Your choice: 1
33
34 Enter process ID: P2
35
36 Enter size of process: 2
37
38 Enter option:
39 1.Process Request
40 2.Deallocation
41 3.Display Page Table
42 4.Display Free Frame List
43 0.Exit
44 Your choice: 1
45
46 Enter process ID: P3
47
48 Enter size of process: 2
49
50 Enter option:
51 1.Process Request
52 2.Deallocation
53 3.Display Page Table
54 4.Display Free Frame List
55 0.Exit
56 Your choice: 4
57 2
58 Enter option:
59 1.Process Request
60 2.Deallocation
61 3.Display Page Table
62 4.Display Free Frame List
63 0.Exit

```



```

64 Your choice: 3
65
66 Process P1:
67
68 Page 0 ---> Frame 30
69
70 Page 1 ---> Frame 11
71
72 Page 2 ---> Frame 19
73
74 Page 3 ---> Frame 26
75
76 Process P2:
77
78 Page 0 ---> Frame 18
79
80 Page 1 ---> Frame 5
81
82 Process P3:
83
84 Page 0 ---> Frame 31
85
86 Page 1 ---> Frame 3
87
88 Enter option:
89 1.Process Request
90 2.Deallocation
91 3.Display Page Table
92 4.Display Free Frame List
93 0.Exit
94 Your choice: 2
95
96 Enter process ID to deallocate: P2
97
98 Enter option:
99 1.Process Request
100 2.Deallocation
101 3.Display Page Table
102 4.Display Free Frame List
103 0.Exit
104 Your choice: 4
105 2 18 5
106 Enter option:
107 1.Process Request
108 2.Deallocation

```

```
109 3.Display Page Table
110 4.Display Free Frame List
111 0.Exit
112 Your choice: 3
113
114 Process P1:
115
116 Page 0 ---> Frame 30
117
118 Page 1 ---> Frame 11
119
120 Page 2 ---> Frame 19
121
122 Page 3 ---> Frame 26
123
124 Process P3:
125
126 Page 0 ---> Frame 31
127
128 Page 1 ---> Frame 3
129
130
131 Enter option:
132 1.Process Request
133 2.Deallocation
134 3.Display Page Table
135 4.Display Free Frame List
136 0.Exit
137 Your choice: 0
```

---