UCS 1411 - Operating Systems Lab

Exercise 9 - Paging Technique

Mahesh Bharadwaj K - 185001089

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

Queue Header File

```
typedef int Data;

typedef struct Node{
    Data d;
    struct Node *next;
}Node;

typedef Node* Queue;

int isEmpty(Queue front, Queue rear){
    if (front == NULL)
        return 1;
    return 0;
}

int size(Queue front, Queue rear){
    if(isEmpty(front, rear))
        return 0;
    int c = 0;
    Node *tmp = front;
    while(tmp){
        tmp = tmp -> next;
        c++;
    }
    return c;
}

void enqueue(Queue *front, Queue *rear, const Data d){
    Node* new = (Node*)malloc(sizeof(Node));
    new -> d = d;
    new -> next = NULL;

    if(isEmpty(*front, *rear))
        (*front) = (*rear) = new;
    else{
        (*rear) -> next = new;
        (*rear) = new;
    }
}

Data dequeue(Queue *front, Queue *rear){
    Data rVal =0;
    if(isEmpty(*front, *rear))
```

```
        return rVal;

    Node *tmp = (*front);
    rVal = (*front) -> d;

    if (*front == *rear)
        (*rear) = NULL;

    (*front) = (*front) -> next;
    free(tmp);
    return rVal;
}

void display(Queue front, Queue rear){
    if(isEmpty(front, rear)){
        printf(" Empty Queue!\n");
        return;
    }

    Queue tmp = front;
    while(tmp){
        printf("% d", tmp -> d);
        tmp = tmp -> next;
    }
    printf("\n");
}
```

Main Program

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

#define MAX 10

typedef struct Process{
    unsigned int pid;
    unsigned int pages[MAX];
    unsigned int n_pages;
}Process;


#include "Queue.h"

void allocate(Queue*, Queue*, Process*, int, int, int);
void deallocate(Queue*, Queue*, Process*, const int);
void pagetable(Process*, const int);

int main(){
    Process p[MAX];



    for(int i = 0; i < MAX; i++){
        p[i].pid = -1;
        p[i].n_pages=0;
```

```
}

Queue front = NULL,
      rear = NULL;

int phy_mem = -1,
    frame_size = -1,
    choice = -1,
    no_frames = -1,
    pid = -1,
    mem_reqd = -1;

printf(" Enter the Size of Physical Memory: ");
scanf("%d", &phy_mem);
printf(" Enter the Page Size: ");
scanf("%d", &frame_size);

no_frames = phy_mem / frame_size;

for(int i = 0; i < 12; i++)
    enqueue(&front, &rear, (random() % no_frames) +1);


while(1){
    printf("\t\t\t\tPAGING TECHNIQUES\n");
    printf(" 1 - Process Request\n");
    printf(" 2 - De-Allocation\n");
    printf(" 3 - Page table for all input processes\n");
    printf(" 4 - Free Frame List\n");
    printf(" 0 - Exit\n");
    printf(" ----------------------------------------\n");
    printf(" Enter your choice: ");
    scanf("%d", &choice);

    switch(choice){
        case 0:
            exit(0);
        case 1:
            printf("\n Enter the PID of the Process & Memory Required: ");
            scanf("%d %d", &pid, &mem_reqd);
            allocate(&front, &rear, p, pid, mem_reqd, frame_size);
            break;
        case 2:
            printf("\n Enter the PID of Process to De-Allocate: ");
            scanf("%d", &pid);
            deallocate(&front, &rear, p, pid);
            break;
        case 3:
            for(int i = 0; i < MAX; i++)
                pagetable(p, i);
            break;
        case 4:
            printf("\n The List of Free Frames:\n");
            display(front, rear);
            break;
        default:
            printf(" Invalid Input!\n");
            break;
```

```
        }
    }
}

void allocate(Queue *front, Queue *rear, Process *p, int pid, int mem_reqd, int page_size){
    int no_pages = mem_reqd / page_size;

    if(no_pages > size(*front, *rear)){
        printf(" Insufficient Memory!\n");
        return;
    }

    if(p[pid].pid != -1){
        printf(" Duplicate PID!\n");
        return;
    }

    printf("\n Process is divided into %d Pages", no_pages);
    p[pid].pid = pid;
    p[pid].n_pages = no_pages;

    for(int i = 0; i < no_pages; i++)
        p[pid].pages[i] = dequeue(front, rear);

    pagetable(p, pid);

    printf(" Successfully Allocated Pages!\n");
}

void deallocate(Queue *front, Queue *rear, Process *p, const int pid){
    if(!p[pid].n_pages){
        printf(" No Such Process exists!\n");
        return;
    }

    p[pid].pid = -1;
    int n_pages = p[pid].n_pages;
    p[pid].n_pages = 0;

    for(int i = 0; i < n_pages; i++)
        enqueue(front, rear, p[pid].pages[i]);

    printf(" Successfully De-Allocated Process!\n");
}

void pagetable(Process* p, const int pid){
    if(p[pid].n_pages == 0)
        return;

    printf("\n\n Page Table for Process %d\n", pid);
    for(int i = 0; i < p[pid].n_pages; i++)
        printf(" Page %-2d : Frame %-2d\n", i, p[pid].pages[i]);
    printf("\n");
}
```

Output

```
Enter the Size of Physical Memory: 32
Enter the Page Size: 1
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 4

The List of Free Frames:
8 7 10 20 18 32 11 13 10 14 27 12
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 1

Enter the PID of the Process & Memory Required: 1 4

Process is divided into 4 Pages

Page Table for Process 1
Page 0  : Frame 8
Page 1  : Frame 7
Page 2  : Frame 10
Page 3  : Frame 20

Successfully Allocated Pages!
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 4

The List of Free Frames:
18 32 11 13 10 14 27 12
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 1

Enter the PID of the Process & Memory Required: 2 3

Process is divided into 3 Pages
```

```
Page Table for Process 2
Page 0  : Frame 18
Page 1  : Frame 32
Page 2  : Frame 11

Successfully Allocated Pages!
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 4

The List of Free Frames:
13 10 14 27 12
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 2

Enter the PID of Process to De-Allocate: 1
Successfully De-Allocated Process!
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 4

The List of Free Frames:
13 10 14 27 12 8 7 10 20
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 1

Enter the PID of the Process & Memory Required: 3 6

Process is divided into 6 Pages

Page Table for Process 3
Page 0  : Frame 13
Page 1  : Frame 10
Page 2  : Frame 14
Page 3  : Frame 27
Page 4  : Frame 12
```

```
Page 5  : Frame 8

Successfully Allocated Pages!
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 4

The List of Free Frames:
7 10 20
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 1

Enter the PID of the Process & Memory Required: 5 2

Process is divided into 2 Pages

Page Table for Process 5
Page 0  : Frame 7
Page 1  : Frame 10

Successfully Allocated Pages!
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 4

The List of Free Frames:
20
                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 3


Page Table for Process 2
Page 0  : Frame 18
Page 1  : Frame 32
Page 2  : Frame 11
```

```
Page Table for Process 3
Page 0  : Frame 13
Page 1  : Frame 10
Page 2  : Frame 14
Page 3  : Frame 27
Page 4  : Frame 12
Page 5  : Frame 8



Page Table for Process 5
Page 0  : Frame 7
Page 1  : Frame 10

                                PAGING TECHNIQUES
1 - Process Request
2 - De-Allocation
3 - Page table for all input processes
4 - Free Frame List
0 - Exit
----------------------------------------
Enter your choice: 0
```
