```
Script started on 2020-03-29 20:57:41+0530
Harshini@Harshini: ~/Desktop/paging
Harshini@Harshini [00m: [01;34m~/Desktop/paging $ gcc paging.c -o p
Harshini@Harshini [00m: [01;34m~/Desktop/paging $ cat pageing.c
#include<stdio.h>
#include<stdlib.h>

int pagesize;
int no_of_frames;
int free_frames_avail;
int physical_memory_size;
typedef struct pagetable
{
	int pageno;
	int frameno;
	int process;
	struct pagetable* next;
}pt;
typedef struct freeframes
{
	int frame;
	int avail;
	struct freeframes* next;
}ff;
pt* create_p()
{
	pt* newTable = (pt*)malloc(sizeof(pt));
	newTable->next = NULL;
	return newTable;
}
ff* create_f()
{
	ff* FrameList = malloc(sizeof(ff));
	FrameList->next = NULL;
	return FrameList;
}
pt* newMap(int process,int pageno,int frameno)
{
	pt* link = malloc(sizeof(pt));
	link->pageno = pageno;
	link->frameno = frameno;
	link->process = process;
	link->next = NULL;
	return link;
}
ff* newFrame(int frame, int avail)
{
	ff* newFrame = malloc(sizeof(ff));
	newFrame->frame = frame;
	newFrame->avail = avail;
	newFrame->next = NULL;
	return newFrame;
}
void insertLast_frame(ff* head, ff* newNode)
{
	ff* temp = head;
	while(temp->next!=NULL)
		temp = temp->next;
	newNode->next = temp->next;
	temp->next = newNode;
}
void insertLast_table(pt* head, pt* newNode)
{
	pt* temp = head;
	while(temp->next!=NULL)
		temp = temp->next;
	newNode->next = temp->next;
	temp->next = newNode;
}
void display_table(pt* table)
{
	pt * temp_table = table->next;
	while(temp_table!=NULL)
	{
		printf("Process : %d\t Page: %d\t Frameno : %d \n",temp_table->process,
temp_table->pageno,temp_table->frameno);
		temp_table = temp_table->next;
	}
}
void display_frames(ff* framelist)
{
	ff* temp_frame = framelist->next;
	while(temp_frame!=NULL)
	{
		printf("Frame : %d\t avail:%d \n",temp_frame->frame,temp_frame->avail);
		temp_frame = temp_frame->next;
	}
}
void delete_frame(ff* framelist, ff* delNode)
{
	ff* temp = framelist;
	while(temp->next != NULL)
	{
		if(temp->next->avail == delNode->avail)
			break;
		temp = temp->next;
	}
	temp->next = temp->next->next;
}
void delete_link(pt* framelist, pt* delNode)
{
	pt* temp = framelist;
	while(temp->next != NULL)
	{
		if(temp->next->process == delNode->process)
			break;
		temp = temp->next;
	}
	temp->next = temp->next->next;
}
void request(int process_id, int process_size, pt* table, ff* framelist)
{
	int reqd_frames = process_size / pagesize;
	ff* temp;
	pt* link;
	if(reqd_frames  > free_frames_avail)
	{
		printf("Request Denied : Not enough memory \n");
		return;
	}
	else
	{
		for(int i = 0;i < reqd_frames;i++)
		{
			temp = framelist->next;
			while(temp->avail != 1)
				temp = temp->next;
			temp->avail = 0;
			link = newMap(process_id,i,temp->frame);
```

```
                    delete_frame(framelist, temp);
                    insertLast_table(table, link);
                    free_frames_avail--;
            }
        }
}
void delc(int process_id, pt* table, ff* framelist)
{
        pt* temp = table->next;
        ff* fram;
        while(temp!=NULL)
        {
            if(temp->process == process_id)
            {
                    int frameno = temp->frameno;
                    int avail = 1;
                    fram = newFrame(frameno,avail);
                    delete_link(table,temp);
                    insertLast_frame(framelist,fram);
                    free_frames_avail++;
            }
            temp = temp->next;
        }
}
int fno=0;
void addressmap()
{
        int pid, logical, offset, physical,pageno;
        printf("Enter PID: ");
        scanf("%d", &pid);
        printf("Enter Logical address: ");
        scanf("%d", &logical);
        pageno = logical / (pagesize * 1024);
        offset = logical % (pagesize * 1024);
        physical = fno * pagesize * 1024 + offset;
        printf("Page no : %d\t Offset : %d\t Frameno : %d \n",pageno,offset,fno);
        printf("Physical address: %d\n", physical);
        fno++;
}

void main()
{
        int process_id;
        int process_size;
        int choice;
        pt* table = create_p();
        ff* framelist = create_f();
        ff* temp_frame;
        pt* temp_table;

        printf("Enter the physical memory size \n");
        scanf("%d",&physical_memory_size);
        printf("Enter page size");
        scanf("%d",&pagesize);
        int no_of_frames = physical_memory_size / pagesize;
        printf("\tPhysical memory is divided into %d frames \n",no_of_frames);
        printf("Initializing physical memory and frame list\n");
        // Generating random numbers
        for (int i = 0; i < no_of_frames; i++)
        {
                int avail = (rand() % (2)); // generates values between 0 and 1 - 0 implies
unavailable;
                temp_frame = newFrame(i,avail);
                if(avail == 1)
                        {
                                insertLast_frame(framelist,temp_frame);
                                free_frames_avail++;
                        }
                else
                {
                        int random_pid = (rand() % (11));
                        int random_page = (rand() & no_of_frames + 1);
                        temp_table = newMap(random_pid,random_page,i);
                        insertLast_table(table,temp_table);
                }
        }
        do
        {
                printf("\t\t\t PAGING IMPLEMENTATION\n");
                printf("1.Process Request \n");
                printf("2.Dealloation \n");
                printf("3.Display Page table \n");
                printf("4.Display free frames\n");
                printf("5.Display logical to physical memory conversion\n");
                printf("6.Exit \n");
                printf("Enter your choice\n");
```

```
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:printf("\n\nEnter Process ID\n");
				scanf("%d",&process_id);
				printf("Enter size of the process \n");
				scanf("%d",&process_size);
				request(process_id,process_size,table,framelist);
				break;
			case 2:
				printf("\n\nEnter Process ID to deallocated\n");
				scanf("%d",&process_id);
				delc(process_id,table,framelist);
				break;
			case 3:
				display_table(table);
				break;
			case 4:
				display_frames(framelist);
				break;
			case 5:   addressmap();
				break;
		}
	}while(choice!=6);
}
```

```
 ]0;Harshini@Harshini: ~/Desktop/paging  [01;32mHarshini@Harshini [00m:
[01;34m~/Desktop/paging $./p
Enter the physical memory size
32
Enter page size1
	Physical memory is divided into 32 frames
Initializing physical memory and frame list
				PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
4
```

```
Frame : 0       avail:1
Frame : 2       avail:1
Frame : 3       avail:1
Frame : 5       avail:1
Frame : 7       avail:1
Frame : 8       avail:1
Frame : 13      avail:1
Frame : 14      avail:1
Frame : 15      avail:1
Frame : 16      avail:1
Frame : 18      avail:1
Frame : 19      avail:1
Frame : 20      avail:1
Frame : 22      avail:1
Frame : 23      avail:1
Frame : 24      avail:1
Frame : 25      avail:1
Frame : 28      avail:1
                        PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
1


Enter Process ID
871
Enter size of the process
16
                        PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
3
Process : 6     Page: 33      Frameno : 1
Process : 6     Page: 33     Frameno : 4
Process : 7     Page: 32     Frameno : 6
Process : 9     Page: 0      Frameno : 9
Process : 0     Page: 1      Frameno : 10
Process : 5     Page: 1      Frameno : 11
Process : 6     Page: 32     Frameno : 12
Process : 10    Page: 32     Frameno : 17
Process : 2     Page: 0      Frameno : 21
Process : 10    Page: 0      Frameno : 26
Process : 7     Page: 32     Frameno : 27
Process : 2     Page: 32     Frameno : 29
Process : 6     Page: 0      Frameno : 30
Process : 5     Page: 1      Frameno : 31
Process : 871   Page: 0      Frameno : 0
Process : 871   Page: 1      Frameno : 2
Process : 871   Page: 2      Frameno : 3
Process : 871   Page: 3      Frameno : 5
Process : 871   Page: 4      Frameno : 7
Process : 871   Page: 5      Frameno : 8
Process : 871   Page: 6      Frameno : 13
Process : 871   Page: 7      Frameno : 14
Process : 871   Page: 8      Frameno : 15
Process : 871   Page: 9      Frameno : 16
Process : 871   Page: 10     Frameno : 18
Process : 871   Page: 11     Frameno : 19
Process : 871   Page: 12     Frameno : 20
Process : 871   Page: 13     Frameno : 22
Process : 871   Page: 14     Frameno : 23
Process : 871   Page: 15     Frameno : 24
                        PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
4
Frame : 25      avail:1
Frame : 28      avail:1
                        PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
```

```
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
1


Enter Process ID
236
Enter size of the process
10
Request Denied : Not enough memory
                        PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
2


Enter Process ID to deallocated
871
                        PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
3
Process : 6     Page: 33      Frameno : 1
Process : 6     Page: 33      Frameno : 4
Process : 7     Page: 32      Frameno : 6
Process : 9     Page: 0       Frameno : 9
Process : 0     Page: 1       Frameno : 10
Process : 5     Page: 1       Frameno : 11
Process : 6     Page: 32      Frameno : 12
Process : 10    Page: 32      Frameno : 17
Process : 2     Page: 0     Frameno : 21
Process : 10    Page: 0     Frameno : 26
Process : 7     Page: 32    Frameno : 27
Process : 2     Page: 32    Frameno : 29
Process : 6     Page: 0     Frameno : 30
Process : 5     Page: 1     Frameno : 31
                    PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
4
Frame : 25      avail:1
Frame : 28      avail:1
Frame : 0       avail:1
Frame : 2       avail:1
Frame : 3       avail:1
Frame : 5       avail:1
Frame : 7       avail:1
Frame : 8       avail:1
Frame : 13      avail:1
Frame : 14      avail:1
Frame : 15      avail:1
Frame : 16      avail:1
Frame : 18      avail:1
Frame : 19      avail:1
Frame : 20      avail:1
Frame : 22      avail:1
Frame : 23      avail:1
Frame : 24      avail:1
                    PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
1
```

```
Enter Process ID
234        912
Enter size of the process
18
                          PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
3
Process : 6     Page: 33     Frameno : 1
Process : 6     Page: 33     Frameno : 4
Process : 7     Page: 32     Frameno : 6
Process : 9     Page: 0      Frameno : 9
Process : 0     Page: 1      Frameno : 10
Process : 5     Page: 1      Frameno : 11
Process : 6     Page: 32     Frameno : 12
Process : 10    Page: 32     Frameno : 17
Process : 2     Page: 0      Frameno : 21
Process : 10    Page: 0      Frameno : 26
Process : 7     Page: 32     Frameno : 27
Process : 2     Page: 32     Frameno : 29
Process : 6     Page: 0      Frameno : 30
Process : 5     Page: 1      Frameno : 31
Process : 912   Page: 0      Frameno : 25
Process : 912   Page: 1      Frameno : 28
Process : 912   Page: 2      Frameno : 0
Process : 912   Page: 3      Frameno : 2
Process : 912   Page: 4      Frameno : 3
Process : 912   Page: 5      Frameno : 5
Process : 912   Page: 6      Frameno : 7
Process : 912   Page: 7      Frameno : 8
Process : 912   Page: 8      Frameno : 13
Process : 912   Page: 9      Frameno : 14
Process : 912   Page: 10     Frameno : 15
Process : 912   Page: 11     Frameno : 16
Process : 912   Page: 12     Frameno : 18
Process : 912   Page: 13     Frameno : 19
Process : 912   Page: 14     Frameno : 20
Process : 912   Page: 15        Frameno : 22
Process : 912   Page: 16        Frameno : 23
Process : 912   Page: 17        Frameno : 24
                              PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
4
                              PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
5
Enter PID: 982
Enter Logical address: 43
Page no : 0     Offset : 43     Frameno : 0
Physical address: 43
                              PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
5.Display logical to physical memory conversion
6.Exit
Enter your choice
5
Enter PID: 4371
Enter Logical address: 78867
Page no : 77    Offset : 19     Frameno : 1
Physical address: 1043
                              PAGING IMPLEMENTATION
1.Process Request
2.Dealloation
3.Display Page table
4.Display free frames
```

```
5.Display logical to physical memory conversion
6.Exit
Enter your choice
6
 ]0;Harshini@Harshini: ~/Desktop/paging  [01;32mHarshini@Harshini [00m:
[01;34m~/Desktop/paging [00m$ exit
exit

Script done on 2020-03-29 21:00:10+0530
```