**SSN College of Engineering, Kalavakkam**

**Department of Computer Science and Engineering**

**Semester - IV**

**UCS1411 – Operating Systems Lab**

**Academic Year: 2019-2020 Batch: 2018-2022**

**Name : Rahul Ram M**

**Class : CSE – B**

**Register Number : 185001121**

**Lab Exercise 9 Implementation of Paging Technique**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <time.h>

bool checkRepetition(int arr[], int size, int value){

for(int i = 0; i < size; i++){

if(arr[i] == value)

return true;

}

return false;

}

void printFreeFrames(int arr[], int size){

printf("Free Frames : ");

for(int i = 0; i < size; i++){

printf("%d ",arr[i]);

}

printf("\n");

}

int\* getFreeFrames(int \*arr, int size, int frames){

srand(time(0));

int i = 0;

int temp;

while(i != size){

temp = (rand() % frames) + 1;

if(!checkRepetition(arr, i, temp)){

arr[i] = temp;

i++;

}

}

return arr;

}

int main(){

int memSize;

int pageSize;

char ms[3];

int frames;

int choice = 1;

printf("Paging Technique :\n");

printf("Enter the physical memory size: ");

scanf("%d %s",&memSize, ms);

printf("Enter the page size: ");

scanf("%d %s",&pageSize, ms);

frames = memSize/pageSize;

printf("Physical memory is divided into %d frames.\n",frames);

printf("After Initialization\n");

srand(time(0));

int length = (rand() % ((int)(frames\*0.2) - (int)(frames\*0.4) + 1)) + (int)(frames\*0.2);

int tlength = length;

int randNum[length];

int \*freeFrames = getFreeFrames(randNum, length, frames);

printFreeFrames(freeFrames, length);

int lengths[20];

for(int i = 0; i < 20; i++){

lengths[i] = 0;

}

int pages[20][10];

char IDs[20][3];

int count = 0;

while(choice != 5){

printf("\n1.Process request\n2.Deallocation\n3.Page Table display for all input process.\n4.Free Frame List display\n5.Exit\nEnter Choice : ");

scanf("%d",&choice);

switch(choice){

case 1:{

int size;

char id[3];

printf("Enter the Process requirement(ID ,size): ");

scanf("%s ,%d",id, &size);

int no\_pages = size/pageSize;

printf("Process is divided into %d pages\n",no\_pages);

if(no\_pages > tlength){

printf("Insufficient frames!\n");

break;

}

strcpy(IDs[count],id);

lengths[count] = no\_pages;

printf("\nPage Table for %s\n",id);

for(int i = 0; i < no\_pages; i++){

printf("Page %d : Frame %d\n",i, freeFrames[i]);

pages[count][i] = freeFrames[i];

}

printf("\n");

count++;

for(int i = 0; (i+no\_pages) < length; i++){

freeFrames[i] = freeFrames[i+no\_pages];

}

tlength -= no\_pages;

break;

}

case 2:{

char id[3];

int index;

printf("Enter the process id to be deallocated: ");

scanf("%s",id);

for(index = 0; index < count; index++){

if(strcmp(IDs[index],id) == 0)

break;

}

if(index == count){

printf("Process id not found!\n");

break;

}

for(int i = 0; i < lengths[index]; i++){

freeFrames[i+tlength] = pages[index][i];

}

tlength += lengths[index];

for(int i = index; (i+1) < length; i++){

strcpy(IDs[i], IDs[i+1]);

for(int j = 0; j < lengths[i+1]; j++){

pages[i][j] = pages[i+1][j];

}

lengths[i] = lengths[i+1];

}

count--;

printf("%s deallocated successfully!\n",id);

break;

}

case 3:{

for(int i = 0; i < count; i++){

printf("Page Table for %s:\n",IDs[i]);

for(int j = 0; j < lengths[i]; j++){

printf("Page %d : Frame %d\n",j, pages[i][j]);

}

printf("\n");

}

break;

}

case 4:{

printFreeFrames(freeFrames, tlength);

break;

}

case 5:{

printf("Program terminated successfully!\n");

break;

}

default:printf("Invalid Input!\n");

}

}

return 0;

}

**Sample I/O:**

Paging Technique :

Enter the physical memory size: 32 KB

Enter the page size: 1 KB

Physical memory is divided into 32 frames.

After Initialization

Free Frames : 8 30 18 3 23 20 29

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 1

Enter the Process requirement(ID ,size): p1 ,4

Process is divided into 4 pages

Page Table for p1

Page 0 : Frame 8

Page 1 : Frame 30

Page 2 : Frame 18

Page 3 : Frame 3

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 3

Page Table for p1:

Page 0 : Frame 8

Page 1 : Frame 30

Page 2 : Frame 18

Page 3 : Frame 3

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 4

Free Frames : 23 20 29

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 1

Enter the Process requirement(ID ,size): p2 ,2

Process is divided into 2 pages

Page Table for p2

Page 0 : Frame 23

Page 1 : Frame 20

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 3

Page Table for p1:

Page 0 : Frame 8

Page 1 : Frame 30

Page 2 : Frame 18

Page 3 : Frame 3

Page Table for p2:

Page 0 : Frame 23

Page 1 : Frame 20

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 4

Free Frames : 29

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 2

Enter the process id to be deallocated: p1

p1 deallocated successfully!

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 3

Page Table for p2:

Page 0 : Frame 23

Page 1 : Frame 20

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 4

Free Frames : 29 8 30 18 3

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 2

Enter the process id to be deallocated: p3

Process id not found!

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 2

Enter the process id to be deallocated: p2

p2 deallocated successfully!

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 3

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 4

Free Frames : 29 8 30 18 3 23 20

1.Process request

2.Deallocation

3.Page Table display for all input process.

4.Free Frame List display

5.Exit

Enter Choice : 5

Program terminated successfully!