Assignment 9: Implementation of Paging Technique

Aim:

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

Algorithm:

1. Generate a random number using the srand() function.

2. Create an array of processes of type table.

3. Input the size of the physical memory and the size of each page.

4. Assign the ceil value of the quotient when the size of physical memory is divided by size of page as number of frames.

5. Initially, assign 0 to all elements of the frames array.

6. Assign -1 to random frames to indicate that the frames are free.

7. Declare an integer p and an integer m and assign 1 to m.

8. Input an integer option.

9. If option is equal to 1, execute steps 9.1 to 9.6.

9.1. Input the process ID and its size.

9.2. Calculate the number of frames that will be occupied by the process.

9.3. If the number of frames required by the process is less than the number of free frames, execute steps 9.4 to 9.6. Else, print that there aren’t enough frames.

9.4. Assign 0 to p and increment the number of processes.

9.5. Add the process ID to the list of processes.

9.6. Declare an integer i and set it to 1. Repeat steps 9.6.1 to 9.6. until i is greater than the number of frames required by the process.

9.6.1. Declare an integer j and set it to m. Declare an integer k and set it to 1. Repeat steps 9.6.1.1 to 9.6.1.7 till k is greater than the total number of frames present.

9.6.1.1. If the (j+1)th frame is free, execute steps 9.6.1.2 to 9.6.1.4.

9.6.1.2. Decrement the number of free frames and assign the process to the (j+1)th frame.

9.6.1.3. Assign i-1 to the page number and j to the frame number of the process.

9.6.1.4. Increment i and p by 1 and break out of the loops.

9.6.1.5. Assign the next frame number to j.

9.6.1.6. Assign j to m.

9.6.1.7. Increment k by 1.

10. If option is equal to 2, execute steps 10.1 to 10.2.

10.1. Input the ID of the process to be deallocated.

10.2. Assign -1 to all the frames allocated to the process to indicate that the process frames have been deallocated. Remove the process from the process list as well.

11. If option is equal to 3, print the page table for all processes.

12. If option is equal to 4, print all the free frames, i.e., print all the frames where the (j+1)th element in the frames array is equal to -1.

13. If option is equal to 5, exit the program.

14. Else, declare that the user has entered an invalid input.

15. Repeat steps 8 to 14 until option is equal to 5.

Code:

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<time.h>

#define MAX 50

struct table

{ int page[MAX];

int frame[MAX];

int no\_of\_pages;

};

int main()

{ srand(time(0));

int no\_of\_frames,option,process,frames\_req,no\_of\_free\_frames=0,no\_of\_process=0,m=1;

double size,memory\_size,page\_size;

int frames[MAX],process\_[MAX];

struct table processes\_[MAX];

printf("\nPAGING TECHNIQUE\n");

printf("\nEnter the physical memory size: ");

scanf("%lf KB",&memory\_size);

printf("Enter the page size: ");

scanf("%lf KB",&page\_size);

no\_of\_frames=ceil(memory\_size/page\_size);

printf("\nPhysical memory is divided into %d frames.",no\_of\_frames);

for(int i=1;i<=no\_of\_frames;i++)

frames[i]=0;

int n=10;

for(int i=1;i<=n;i++)

{ frames[rand()%no\_of\_frames+1]=-1;

no\_of\_free\_frames++;

}

printf("\n\nAfter Initialisation: \n\nFree Frames: ");

for(int i=1;i<=no\_of\_frames;i++)

{ if(frames[i]==-1)

printf("%d ",i);

}

int p=0;

printf("\n\nMENU:\n\t1. Process request\n\t2. Deallocation\n\t3. Page Table display for all input process\n\t4. Free Frame list display\n\t5. Exit\n");

do

{ printf("\nEnter the option: ");

scanf("%d",&option);

switch(option)

{ case 1:

printf("\nEnter the process requirement (ID,size): P");

scanf("%d, %lf KB",&process,&size);

frames\_req=ceil(size/page\_size);

processes\_[process].no\_of\_pages=frames\_req;

printf("\nProcess is divivded into %d pages.\n\nPage Table for P%d:\n",frames\_req,process);

if(frames\_req<=no\_of\_free\_frames)

{ p=0;

no\_of\_process++;

process\_[no\_of\_process]=process;

for(int i=1;i<=frames\_req;)

{ for(int j=m,k=1;k<=no\_of\_frames;k++)

{ if(frames[j]==-1)

{ printf("\n\tPage %d : Frame %d \n",i-1,j);

no\_of\_free\_frames--;

frames[j]=process;

processes\_[process].page[p]=i-1;

processes\_[process].frame[p]=j;

i++;

p++;

break;

}

j=j%no\_of\_frames+1;

m=j;

}

}

}

else

printf("\nThere is no enough free frames to allocate for this process!\n");

break;

case 2:

printf("\nEnter the process to be deallocated: P");

scanf("%d",&process);

for(int i=1;i<=no\_of\_frames;i++)

{ if(frames[i]==process)

{ frames[i]=-1;

no\_of\_free\_frames++;

}

}

for(int i=1;i<=no\_of\_process;i++)

{ if(process\_[i]==process)

{ process\_[i]=-1;

break;

}

}

break;

case 3:

for(int k=1;k<=no\_of\_process;k++)

{ if(process\_[k]!=-1)

{ printf("\nPage table for P%d:\n",process\_[k]);

int i=0;

for(;i<processes\_[process\_[k]].no\_of\_pages;i++)

printf("\n\tPage %d : Frame %d \n",processes\_[process\_[k]].page[i],processes\_[process\_[k]].frame[i]);

}

}

break;

case 4:

printf("\nFree Frames: ");

for(int i=1,j=m;i<=no\_of\_frames;i++)

{ if(frames[j]==-1)

printf("%d ",j);

j=j%no\_of\_frames+1;

m=j;

}

printf("\n");

break;

case 5:

printf("\nProgram terminated\n");

break;

default:

printf("\nInvalid option\n");

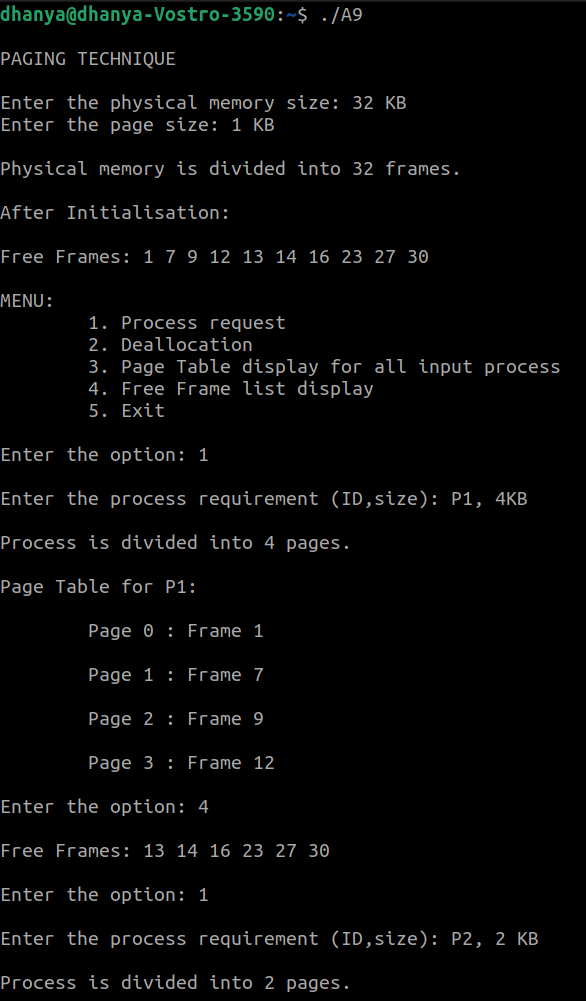
break;

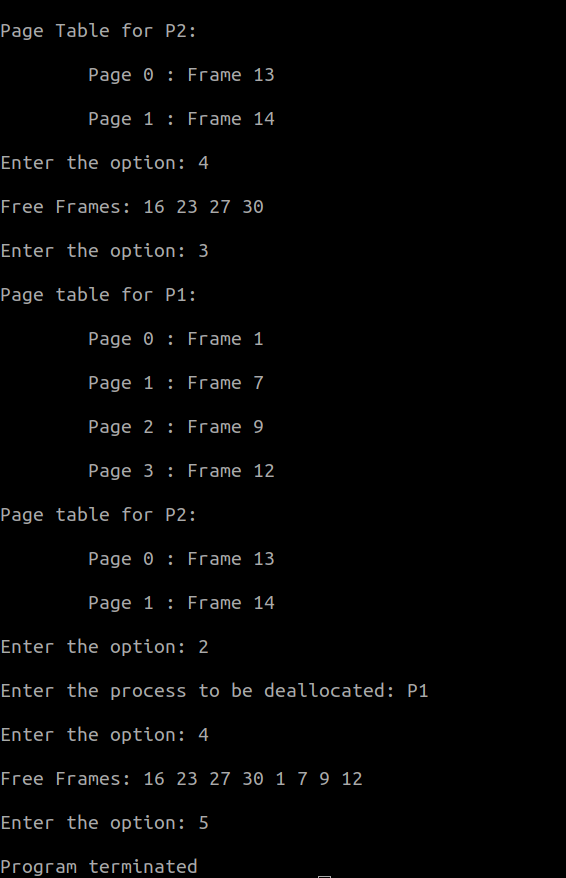
}

}while(option!=5);

}

Output:





Learning outcomes:

* Learned about paging.
* Learned about the use of paging technique.
* Learned what frames and pages are.
* Learned how to allocate memory to processes using paging technique.