Operating Systems

Experiment No. 9

Name: Ayush Jain
SAPID: 60004200132
Batch: B1
Branch: Computer Engineering

Aim: Implement various page replacement policies

Problem Statement-
Perform following page replacement policies and perform comparative assessment them.

1) LEAST RECENTLY USED (LRU)
2) FIRST-IN-FIRST-OUT

Description:

In multiprogramming system using dynamic partitioning there will come a time when all of the processes in the main memory are in a blocked state and there is insufficient memory. To avoid wasting processor time waiting for an active process to become unblocked. The OS will swap one of the process out of the main memory to make room for a new process or for a process in Ready-Suspend state. Therefore, the OS must choose which process to replace.
Thus, when a page fault occurs, the OS has to change a page to remove from memory to make room for the page that must be brought in. If the page to be removed has been modified while in memory it must be written to disk to bring the disk copy up to date.
Replacement algorithms can affect the system's performance. Following are the three basic page replacement algorithms:

Least Recently Used Algorithm
This paging algorithm selects a page for replacement that has been unused for the longest time.

First-In-First-Out

Replace the page that has been in memory longest, is the policy applied by FIFO. Pages from memory are removed in round-robin fashion. Its advantage is it's simplicity.

Code-

Least Recently Used

```
#include<stdio.h>

int findLRU (int time[], int n)
{
  int i, minimum = time[0], pos = 0;

  for (i = 1; i < n; ++i)
    {
      if (time[i] < minimum)
         {
           minimum = time[i];
           pos = i;
         }
    }
  return pos;
}

int main ()
{
  int no_of_frames, no_of_pages, frames[10], pages[30], counter =
    0, time[10], flag1, flag2, i, j, pos, faults = 0;
printf ("Enter number of frames: ");   scanf
("%d", &no_of_frames);   printf ("Enter
number of pages: ");   scanf ("%d",
&no_of_pages);   printf ("Enter Pages in
Order: ");
  for (i = 0; i < no_of_pages; ++i)
    {
       printf("\nEnter : ")
      scanf ("%d", &pages[i]);
    }
```

```
  for (i = 0; i < no_of_frames; ++i)
    {
       // empty frames are identified by -1
      frames[i] = -1;
    }

  for (i = 0; i < no_of_pages; ++i)
    {
      flag1 = flag2 = 0;

      for (j = 0; j < no_of_frames; ++j)
         {
           if (frames[j] == pages[i])
             {          counter++;
      time[j] = counter;
flag1 = flag2 = 1;
break;             }
         }

      if (flag1 == 0)
         {
           for (j = 0; j < no_of_frames; ++j)
             {
               if (frames[j] == -1)
                  {
                    counter++;
  faults++;
    frames[j] = pages[i];     time[j] =
counter;
                    flag2 = 1;
  break;
                  }
             }
         }

      if (flag2 == 0)
         {
           pos = findLRU (time, no_of_frames);
           counter++;
faults++;       frames[pos] =
```

```
pages[i];       time[pos] =
counter;
        }

     printf ("\n");

     for (j = 0; j < no_of_frames; ++j)
        {
          printf ("%d\t", frames[j]);
        }
   }
  printf ("\n\nTotal Page Faults = %d", faults);

  return 0;
}
```

FiFO

```
#include <stdio.h>
int main()
{
int referenceString[10], pageFaults = 0, m, n, s, pages, frames;
printf("\nEnter the number of Pages:\t"); scanf("%d", &pages);
printf("\nEnter reference string values:\n");
for( m = 0; m < pages; m++)
{
   printf("Value No. [%d]:\t", m + 1);
   scanf("%d", &referenceString[m]);
}
printf("\n What are the total number of frames:\t");
{
   scanf("%d", &frames);
}
int temp[frames]; for(m = 0;
m < frames; m++)
{
  temp[m] = -1;
}
for(m = 0; m < pages; m++)
{   s =
0;
  for(n = 0; n < frames; n++)
```

```
  {
    if(referenceString[m] == temp[n])
      {          s++;
pageFaults--;
      }
  }
  pageFaults++;
  if((pageFaults <= frames) && (s == 0))
    {
      temp[m] = referenceString[m];
    }
  else if(s == 0)
    {
      temp[(pageFaults - 1) % frames] = referenceString[m];
    }
printf("\n");
    for(n = 0; n < frames; n++)
     {
       printf("%d\t", temp[n]);
     }
}
printf("\nTotal Page Faults:\t%d\n", pageFaults); return
0;
}
```

Output:

Least Recently Used:

FIFO:

```
Enter the number of Pages:      10

Enter reference string values:
Value No. [1]:  1
Value No. [2]:  2
Value No. [3]:  3
Value No. [4]:  4
Value No. [5]:  2
Value No. [6]:  1
Value No. [7]:  5
Value No. [8]:  3
Value No. [9]:  2
Value No. [10]: 4

 What are the total number of frames:   3

1       -1      -1
1       2       -1
1       2       3
4       2       3
4       2       3
4       1       3
4       1       5
3       1       5
3       2       5
3       2       4
Total Page Faults:      9
```

Conclusion: Implemented various page replacement policies.