OS ASSIGNMENT-5 GROUP NO.-15

IIT2019204 - Mitta Lekhana Reddy
IIT2019205 - Sanskar Patro
IIT2019208 - Dhanush Vasa
IIT2019219 - Gitika Yadav
IIT2019234 - Kodi Pravallika

New initialisations

```
/*GLOBALLY DECLARED OR INITIALISED NEW VARIABLES*/
int counter;
int time[100];
int frequency[no_of_frames];
int pos;

/*INITIALISED VARIABLES IN INITIALISE( ) FUNCTION*/
for (int i=0; i<no_of_frames; i++)
  {
        frequency[i] = 0;
  }
```

New initialisation explanation

Globally we declare variables such as:

counter=> as soon as a entry we increment the counter which represents the time at which that frame is used

time[100]=>used to store the pages according to the time they entered

frequency[no_of_frames]=>used to store the frequency of pages which are in frame table.

pos=>pos determines the frame to which a new entry has to be read i.e., which satisfies low frequency , oldest entered

We initialize all the frequency array elements to zero

New functions

```
void printFreq()
{
        for (int i=0;i<no_of_frames;i++)
        {
                printf(">%d > frame %d \n",frequency[i],i);
        }
        printf("\n");
        printf("The Page Fault Count = %d\n\n",page_fault_count);
}
```

Printfreq function explanation

-In this function, firstly we run the for loop from int i=0 to i<no_of_frames

-For every iteration we print frequency of corresponding frame number, along with it’s frame number

-Lastly, in this function we will print the page fault count

```
int findLFU(int time[],int n, int freq[])
{
        int min_freq = freq[0];
        int min_time = 1000;
        int pos = 0;

        for (int i=0; i<n; i++)
        {
                if (freq[i]<min_freq)
                {
                        min_freq = freq[i];
                }
        }

        for (int i=0; i<n; i++)
        {
                if (time[i] < min_time && min_freq == freq[i])
                {
                        pos = i;
                        min_time = time[i];
                }

        }
        freq[pos] = 1;
        return FrameTable[pos];
}
```

findLFU function explanation

-Firstly, we declare min_freq, which is used to find the minimum frequency of the pages in the frame table.So, we find the minimum by firstly assigning freq[0] to it.

-In the same way we define min_time, which is used when we find two or more page numbers with minimum frequency.We initialize it to random number(say 1000)

-min_time=> used to know which page entered first

- pos=> determines the frame to which a new entry has to be read i.e., which satisfies low frequency , oldest entered

-Now we run a for loop for finding minimum frequency of pages in frame table

-Other for loop is to find if there are more than one with less frequency.If there are more than one,then we use time array to solve the problem.The one which entered time array first will be the victim page. We set the frequency of frame where we read the page to 1.Lastly, we return victim page number.

GetFrameNo function

```
if (PageTable[pno].valid_bit == 1)
{
        fno = PageTable[pno].frm_no;
        counter++;
        time[fno] = counter;
        var1 = 1;
        var2 = 1;
        frequency[fno]++;
        return fno;
}
```

explanation

-We declare two variables var1,var2 and initialize both to 0.Where var1 is used to check if the page no. is present in memory and var2 is to check if frame table is free.We use fno for frame number

-Firstly to get the frame number, we first scan through the page table to check if the page is in memory by checking the valid bit of our page number.

-If the valid bit is 1,then we store the corresponding frame number in fno, along with this we increment the counter value. We store it in the time array, and assign 1 to both var1 and var2.Now, increase frequency of the frame no. Lastly, we return frame number.

GetFrameNo function

```
if (var1 == 0)
{
        for(int i=0; i<no_of_frames; i++)
        {
                if (FrameTable[i] == -1)
                {
                        counter++;
                        time[i] = counter;
                        fno = i;
                        var2 = 1;
                        PageTable[pno].valid_bit = 1;
                        PageTable[pno].frm_no = fno;
                        readPage(pno,fno);
                        frequency[i]++;
                        break;
                }
        }
}
```

explanation

-We enter this part if we don’t find the required pno in memory.Now, in this code we search for the free frame, so that we can store this pno in that free frame

-For that we run the loop for the given 3 frames, and we check if they are free

-If we find the free frame, then we increment the counter and store it in time array and change the var2 to 1.

-Next we assign corresponding frame no. into fno and then change the valid bit of this pno to 1. Then we read the page into the memory

-Lastly, we increase the frequency of that frame no. and break the loop.

GetFrameNo function

```
if (var2 == 0)
  {
          int victim_pno;
          victim_pno = findLFU(time,no_of_frames,frequency);
          fno = PageTable[victim_pno].frm_no;
          PageTable[victim_pno].valid_bit = 0;
          PageTable[victim_pno].frm_no = -1;
          PageTable[pno].valid_bit = 1;
          PageTable[pno].frm_no = fno;
          counter++;
          time[fno] = counter;
          if (PageTable[victim_pno].modify_bit == 1)
                  writeFrame(fno, victim_pno);
          readPage(pno, fno);
  }
```

explanation

-We enter this code section when we don’t find any free frame to store our pno. , so our task is to find the victim page no and replace it with our page no. using least frequently used page algorithm

-For this we call findLFU function to find the victim page number and store it in victim_pno and corresponding frame number into fno

-We change the valid bit of victim_pno to 0 and change the corresponding frame number to -1, as we are removing it from page table.

-We change the valid bit of our pno to 1 and store fno in corresponding frame number in page table.

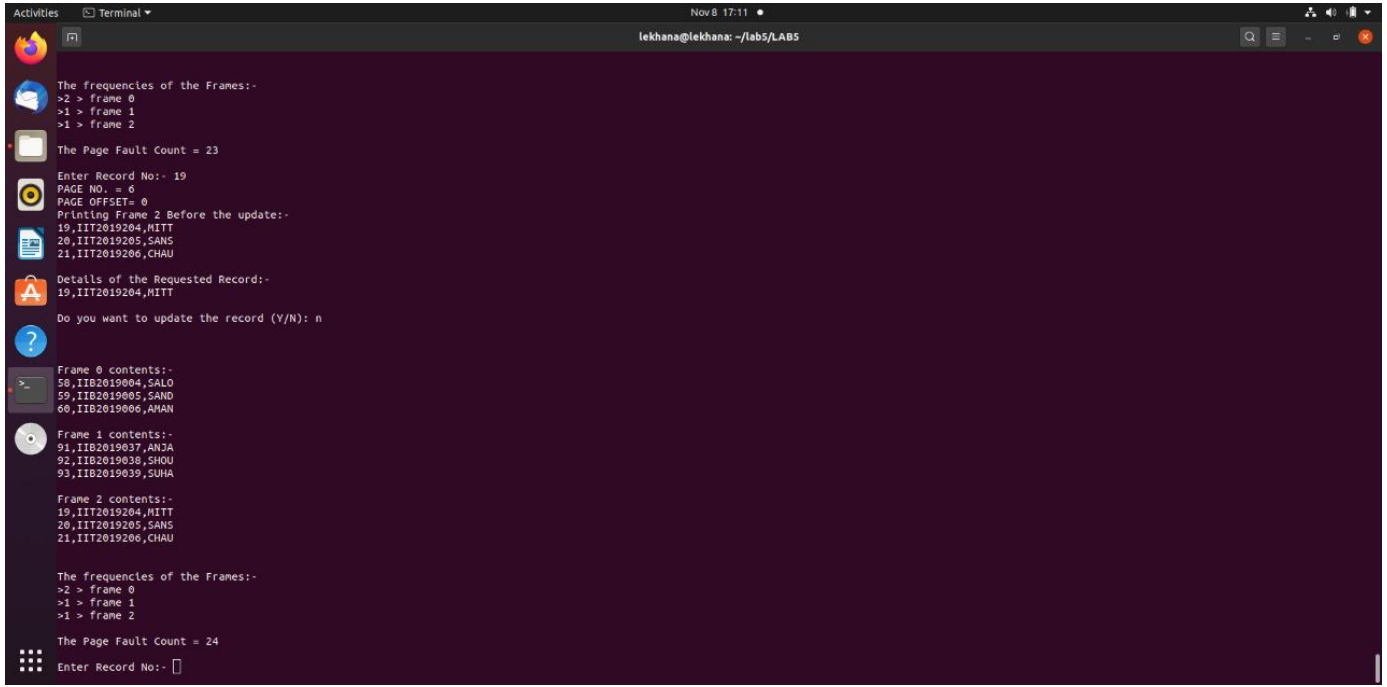
-Increment the counter and as-usually store into time array

-if modify bit of victim_pno is equal to 1, then we call writeFrame function, we write the victim page back to disk file and then call readPage() function.

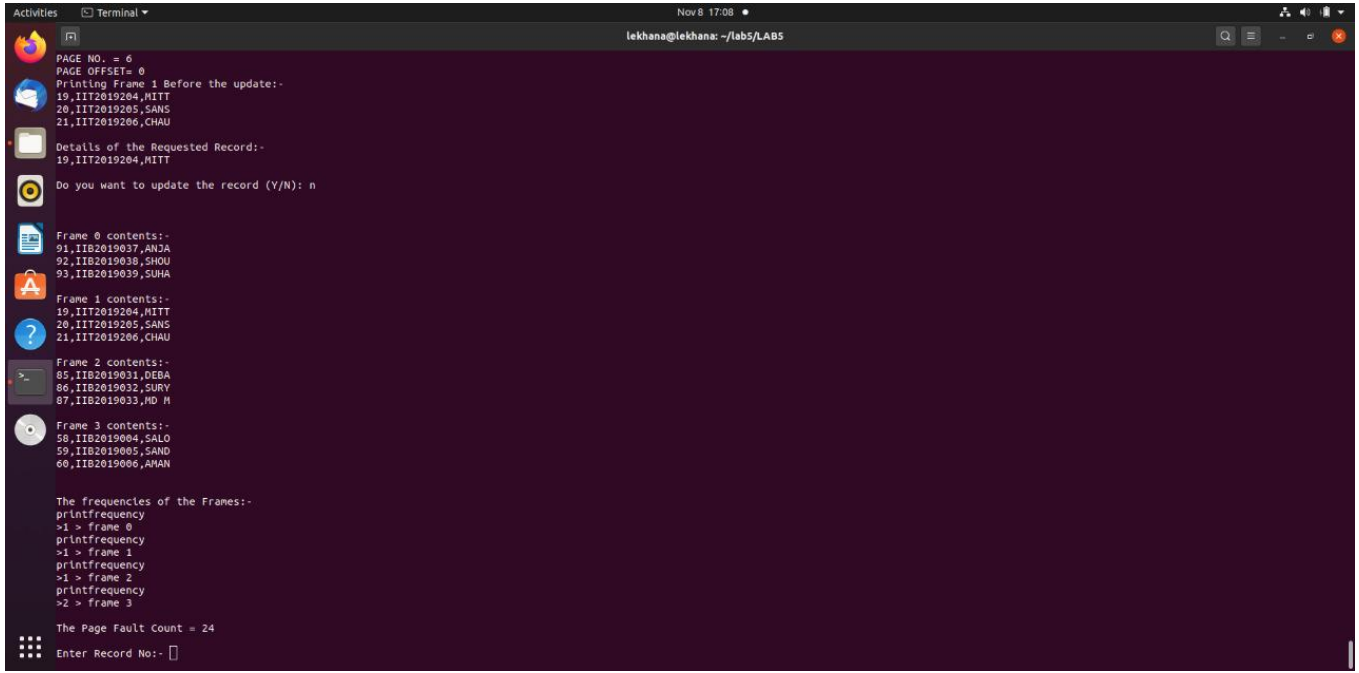
-Lastly, we return fno.

The page replacement algorithms (LFU algo. & optimal algo) in a graph.

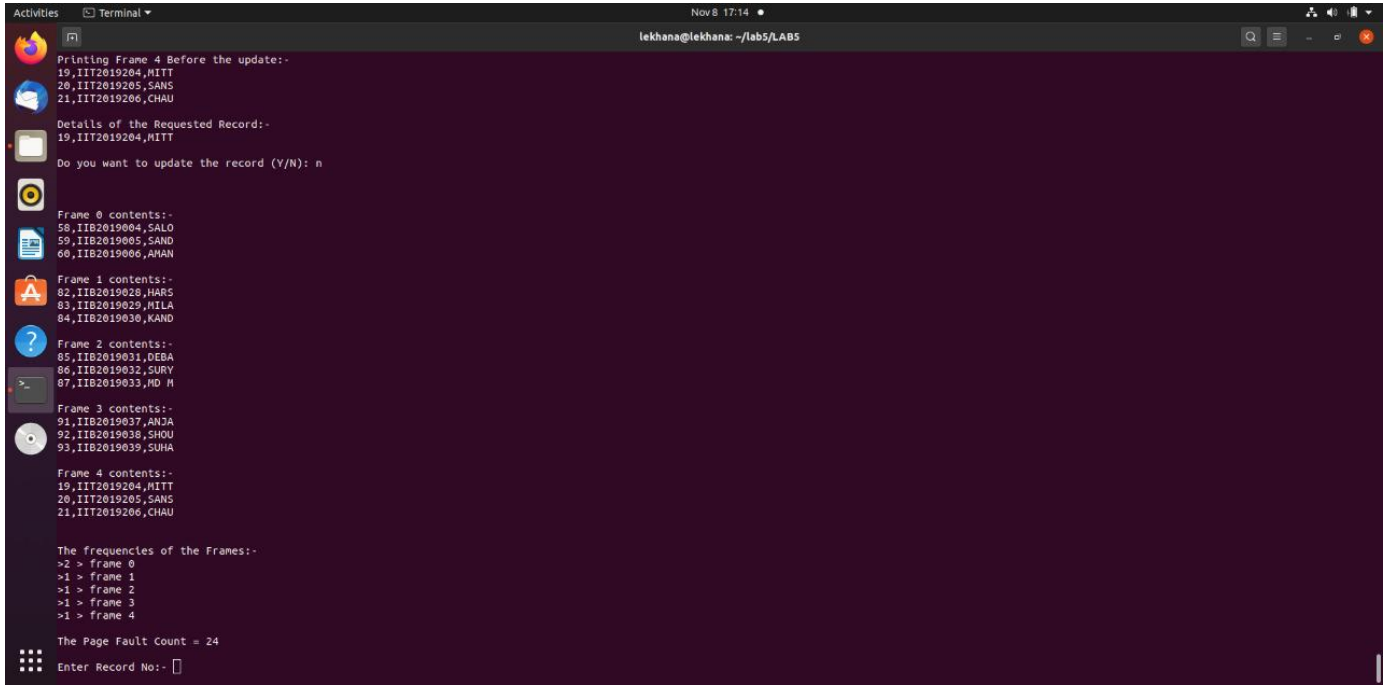
WITH 3 FRAMES WITH LFU algo.
The return of page_fault_count: 24
For:3,23,15,26,76,20,17,37,46,42,09,93,26,76,20,58,62,58,36,28,72,82,85,92,19



WITH 4 FRAMES WITH LFU algo.
The return of page_fault_count:24
For:3,23,15,26,76,20,17,37,46,42,09,93,26,76,20,58,62,58,36,28,72,82,85,92,19



WITH 5 FRAMES WITH LFU algo.
The return of page_fault_count: 24
For:3,23,15,26,76,20,17,37,46,42,09,93,26,76,20,58,62,58,36,28,72,82,85,92,19



WITH 3 FRAMES WITH Optimal algo.
The return of page_fault_count: 21
For:3,23,15,26,76,20,17,37,46,42,09,93,26,76,20,58,62,58,36,28,72,82,85,92,19

	3 Frames - Optimal Paging Algorithm																								
	3	23	15	26	76	20	17	37	46	42	9	93	26	76	20	58	62	58	36	28	72	82	85	92	19
Frame 1	1	1	1	9	9	9	9	9	9	9	9	9	9	9	7	7	21	21	12	12	24	24	29	29	29
Frame 2		8	8	8	26	26	26	26	26	26	26	26	26	26	26	20	20	20	20	10	10	28	28	28	28
Frame 3			5	5	5	7	6	13	16	14	3	31	31	31	31	31	31	31	31	31	31	31	31	31	7
	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	hit	hit	fault	fault	fault	hit	fault	fault	fault	fault	fault	hit	fault

Number of Page Faults = 21 Number of Page Hits = 4

WITH 4 FRAMES WITH Optimal algo.
The return of page_fault_count: 19
For:3,23,15,26,76,20,17,37,46,42,09,93,26,76,20,58,62,58,36,28,72,82,85,92,19

	4 Frames - Optimal Paging Algorithm																								
	3	23	15	26	76	20	17	37	46	42	9	93	26	76	20	58	62	58	36	28	72	82	85	92	19
Frame 1	1	1	1	1	26	26	26	26	26	26	26	26	26	26	26	26	21	21	21	10	10	28	28	28	28
Frame 2		8	8	8	8	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
Frame 3			5	5	5	5	6	13	16	14	3	31	31	31	31	31	31	31	31	31	31	31	31	31	31
Frame 4				9	9	9	9	9	9	9	9	9	9	9	9	20	20	20	12	12	24	24	29	29	29
	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	hit	hit	hit	fault	fault	hit	fault	fault	fault	fault	fault	hit	hit

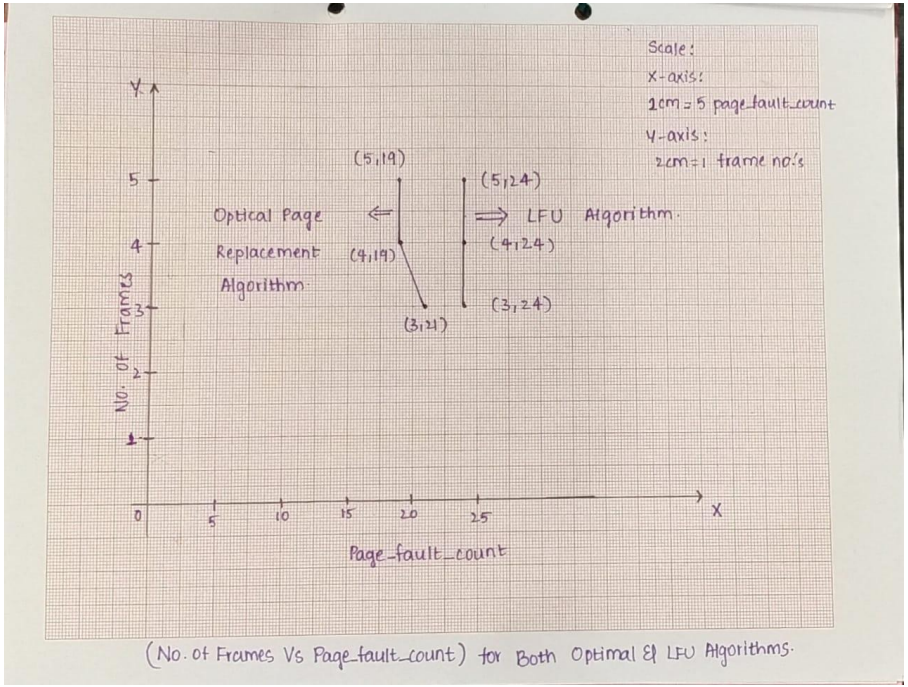
Number of Page Faults = 19 Number of Page Hits = 6

WITH 5 FRAMES WITH Optimal algo.
The return of page_fault_count: 19
For:3,23,15,26,76,20,17,37,46,42,09,93,26,76,20,58,62,58,36,28,72,82,85,92,19

	5 Frames - Optimal Paging Algorithm																								
	3	23	15	26	76	20	17	37	46	42	9	93	26	76	20	58	62	58	36	28	72	82	85	92	19
Frame 1	1	1	1	1	1	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
Frame 2		8	8	8	8	8	6	6	16	16	3	3	3	3	3	3	3	3	12	12	12	28	28	28	28
Frame 3			5	5	5	5	5	13	13	14	14	31	31	31	31	31	31	31	31	31	31	31	31	31	31
Frame 4				9	9	9	9	9	9	9	9	9	9	9	9	20	20	20	20	10	10	10	29	29	29
Frame 5					26	26	26	26	26	26	26	26	26	26	26	26	21	21	21	21	24	24	24	24	24
	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	fault	hit	hit	hit	fault	fault	hit	fault	fault	fault	fault	fault	hit	hit

Number of Page Fault = 19 Number of Page Hits = 6

(no.-of-frames vs. no.-of-page-faults) Graph for LFU algo. and Optimal algo.



THANK YOU