

Urszula Chmielewska

300167

EOPSY

Lab 4

Memory Management

TASK

The goal of this task was to configure a command file which maps any 8 pages of physical memory to the first 8 pages of virtual memory. After that it reads from one virtual memory address on each of the 64 virtual pages.

INTRODUCTION

Memory Management is functionality of an operating system to manages primary memory and moves processes between main memory and disk during execution. It has knowledge about each memory localization, no matter if it is assigned to any process or stays free. It also checks how much memory has to be allocated for the process.

While loading and removing processes from the memory, the free memory space is divided into smaller pieces. After some time, memory blocks are so small that there is not enough memory space for the process to be allocated. Such problem is known as **fragmentation**.

To prevent from fragmentation, **paging** was implemented. Paging is a memory management technique in which process address space is divided into blocks of the same size called *pages*. Moreover, main memory is also divided into physical memory blocks, with the same size, what is called *frames*.

Described above method called paging was used during this laboratory. There are several advantages from such approach:

1. reduces external fragmentation
2. simple to implement and efficient
3. swapping between pages and frames is easy due to the fact that they have equal sizes

SOLUTION

In order to configure mapping it was necessary to edit *memory.conf* file.

```
1 // memset virt page #  physical page #  R (read from)  M (modified) inMemTime (ns) lastTouchTime (ns)
2 memset 0 0 0 0 0 0
3 memset 1 1 0 0 0 0
4 memset 2 2 0 0 0 0
5 memset 3 3 0 0 0 0
6 memset 4 4 0 0 0 0
7 memset 5 5 0 0 0 0
8 memset 6 6 0 0 0 0
9 memset 7 7 0 0 0 0
10
11 // enable_logging 'true' or 'false'
12 // When true specify a log_file or leave blank for stdout
13 enable_logging true
14
15 // log_file <FILENAME>
16 // Where <FILENAME> is the name of the file you want output
17 // to be print to.
18 log_file tracefile
19
20 // page size, defaults to 2^14 and cannot be greater than 2^26
21 // pagesize <single page size (base 10)> or <'power' num (base 2)>
22 pagesize 16384
23
24 // addressradix sets the radix in which numerical values are displayed
25 // 2 is the default value
26 // addressradix <radix>
27 addressradix 10
28
29 // numpages sets the number of pages (physical and virtual)
30 // 64 is the default value
31 // numpages must be at least 2 and no more than 64
32 // numpages <num>
33 numpages 64
```

According to our task I map only 8 pages. It is done using *memset* command which assigns virtual pages to the physical pages respectively.

Secondly I configured *commands* file which specifies read operation for 64 virtual pages. In the *memory.conf* file, pagesize was set to 16384 addresses. Due to this fact I execute read command 64 times for addresses being multiples of 16384.

```
1 READ 0
2 READ 16384
3 READ 32768
4 READ 49152
5 READ 65536
6 READ 81920
7 READ 98304
8 READ 114688
9 READ 131072
10 READ 147456
11 READ 163840
12 READ 180224
13 READ 196608
14 READ 212992
15 READ 229376
16 READ 245760
17 READ 262144
18 READ 278528
19 READ 294912
20 READ 311296
21 READ 327680
22 READ 344064
23 READ 360448
24 READ 376832
25 READ 393216
26 READ 409600
27 READ 425984
28 READ 442368
29 READ 458752
30 READ 475136
31 READ 491520
32 READ 507904
33 READ 524288
34 READ 540672
35 READ 557056
36 READ 573440
37 READ 589824
38 READ 606208
39 READ 622592
40 READ 638976
41 READ 655360
42 READ 671744
43 READ 688128
44 READ 704512
45 READ 720896
46 READ 737280
47 READ 753664
48 READ 770048
49 READ 786432
50 READ 802816
51 READ 819200
52 READ 835584
53 READ 851968
54 READ 868352
55 READ 884736
56 READ 901120
57 READ 917504

58 READ 933888
59 READ 950272
60 READ 966656
61 READ 983040
62 READ 999424
63 READ 1015808
64 READ 1032192
```

Below there is presented program running the simulation in its primary state. As it was stated in the *memory.conf* file, first 0-7 virtual pages correspond to first 0-7 physical pages. Rest of the pages are mapped by default.

Memory Management				
virtual	physical	virtual	physical	time: 0
page 0	page 0	page 32		
page 1	page 1	page 33		instruction: NONE
page 2	page 2	page 34		address: NULL
page 3	page 3	page 35		
page 4	page 4	page 36		page fault: NO
page 5	page 5	page 37		
page 6	page 6	page 38		virtual page: x
page 7	page 7	page 39		physical page: 0
page 8	page 8	page 40		R: 0
page 9	page 9	page 41		M: 0
page 10	page 10	page 42		inMemTime: 0
page 11	page 11	page 43		lastTouchTime: 0
page 12	page 12	page 44		low: 0
page 13	page 13	page 45		high: 0
page 14	page 14	page 46		
page 15	page 15	page 47		
page 16	page 16	page 48		
page 17	page 17	page 49		
page 18	page 18	page 50		
page 19	page 19	page 51		
page 20	page 20	page 52		
page 21	page 21	page 53		
page 22	page 22	page 54		
page 23	page 23	page 55		
page 24	page 24	page 56		
page 25	page 25	page 57		
page 26	page 26	page 58		
page 27	page 27	page 59		
page 28	page 28	page 60		
page 29	page 29	page 61		
page 30	page 30	page 62		
page 31	page 31	page 63		

Next, if we start simulation by one step, we can notice that the first mapped page becomes the first replacement. This is caused due to the fact that the used page replacement algorithm was First-In First-Out (FIFO).

FIFO is the simplest page replacement algorithm. Operating System keeps track of all pages in the memory in a queue. In the front of the queue is the oldest page. When a page needs to be replaced, the first page from the queue is used.

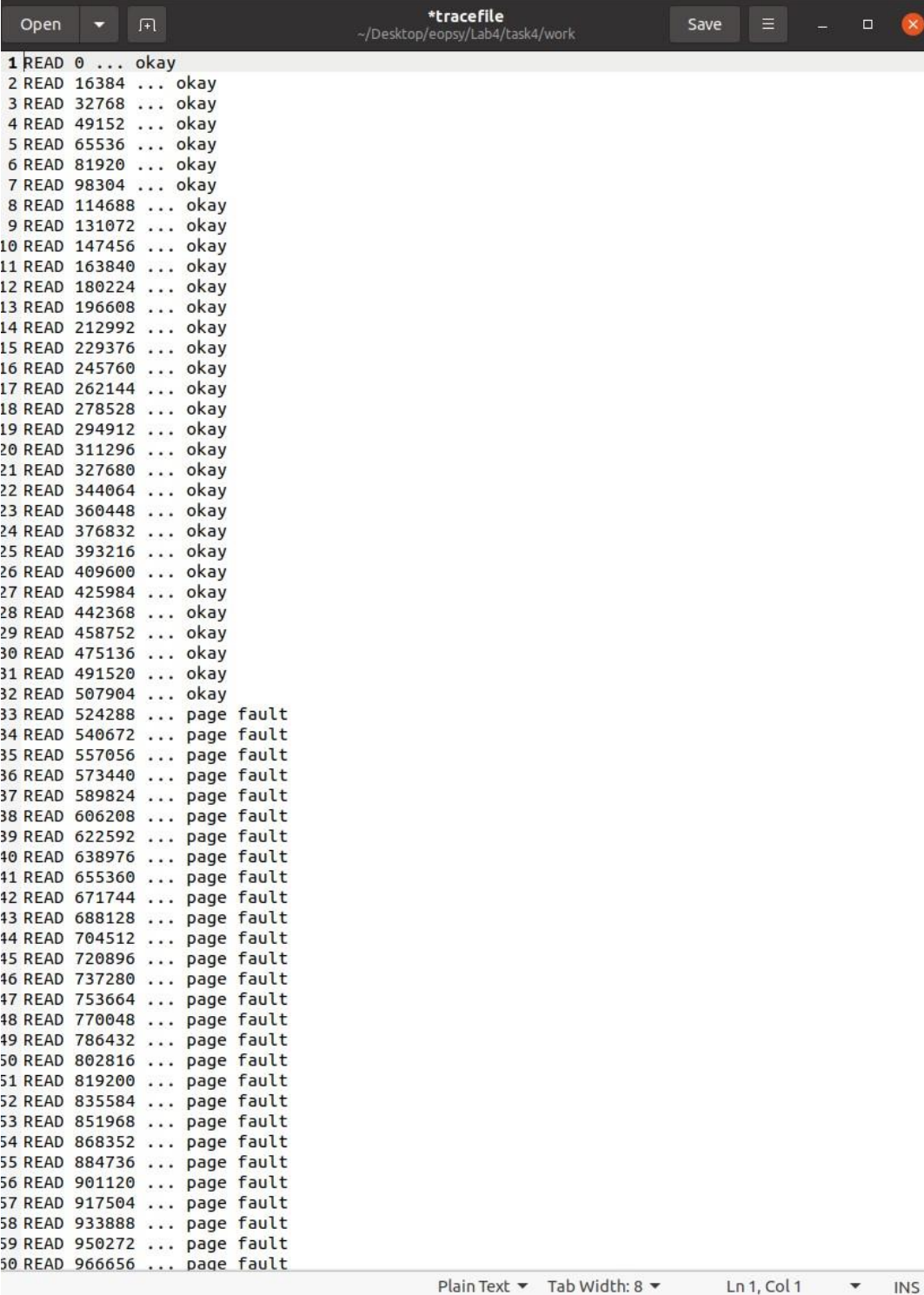
Due to the fact that there are 32 virtual pages mapped to physical pages and the rest virtual pages are not mapped to any physical pages, I expected faults while trying to access not mapped virtual pages. My suspicion has been confirmed and it can be seen on the picture below:

Memory Management					
virtual	physical	virtual	physical	time: 330 (ns)	
page 0		page 32	page 0	instruction: READ	
page 1	page 1	page 33		address: 524288	
page 2	page 2	page 34		page fault: YES	
page 3	page 3	page 35		virtual page: 32	
page 4	page 4	page 36		physical page: -1	
page 5	page 5	page 37		R: 0	
page 6	page 6	page 38		M: 0	
page 7	page 7	page 39		inMemTime: 0	
page 8	page 8	page 40		lastTouchTime: 0	
page 9	page 9	page 41		low: 524288	
page 10	page 10	page 42		high: 540671	
page 11	page 11	page 43			
page 12	page 12	page 44			
page 13	page 13	page 45			
page 14	page 14	page 46			
page 15	page 15	page 47			
page 16	page 16	page 48			
page 17	page 17	page 49			
page 18	page 18	page 50			
page 19	page 19	page 51			
page 20	page 20	page 52			
page 21	page 21	page 53			
page 22	page 22	page 54			
page 23	page 23	page 55			
page 24	page 24	page 56			
page 25	page 25	page 57			
page 26	page 26	page 58			
page 27	page 27	page 59			
page 28	page 28	page 60			
page 29	page 29	page 61			
page 30	page 30	page 62			
page 31	page 31	page 63			

The first page fault occurs at the 32nd page. Page used as a replacement was the oldest page in the queue so the 0th page. The rest of the pages will be also replaced with the next page from the queue respectively. Below there is a result after going through all pages.

Memory Management					
virtual	physical	virtual	physical	time: 640 (ns)	
page 0		page 32	page 0	instruction: READ	
page 1		page 33	page 1	address: 1032192	
page 2		page 34	page 2	page fault: YES	
page 3		page 35	page 3	virtual page: 63	
page 4		page 36	page 4	physical page: -1	
page 5		page 37	page 5	R: 0	
page 6		page 38	page 6	M: 0	
page 7		page 39	page 7	inMemTime: 0	
page 8		page 40	page 8	lastTouchTime: 0	
page 9		page 41	page 9	low: 1032192	
page 10		page 42	page 10	high: 1048575	
page 11		page 43	page 11		
page 12		page 44	page 12		
page 13		page 45	page 13		
page 14		page 46	page 14		
page 15		page 47	page 15		
page 16		page 48	page 16		
page 17		page 49	page 17		
page 18		page 50	page 18		
page 19		page 51	page 19		
page 20		page 52	page 20		
page 21		page 53	page 21		
page 22		page 54	page 22		
page 23		page 55	page 23		
page 24		page 56	page 24		
page 25		page 57	page 25		
page 26		page 58	page 26		
page 27		page 59	page 27		
page 28		page 60	page 28		
page 29		page 61	page 29		
page 30		page 62	page 30		
page 31		page 63	page 31		

Moreover, the prove, that the first page fault occurs for the 32nd page, can be found in the *tracefile* with results. All pages below 32nd one caused also a fault:



```
Open  [icon]  *tracefile
~/Desktop/eopsy/Lab4/task4/work  Save  [icon]  [icon]  [icon]

1 READ 0 ... okay
2 READ 16384 ... okay
3 READ 32768 ... okay
4 READ 49152 ... okay
5 READ 65536 ... okay
6 READ 81920 ... okay
7 READ 98304 ... okay
8 READ 114688 ... okay
9 READ 131072 ... okay
10 READ 147456 ... okay
11 READ 163840 ... okay
12 READ 180224 ... okay
13 READ 196608 ... okay
14 READ 212992 ... okay
15 READ 229376 ... okay
16 READ 245760 ... okay
17 READ 262144 ... okay
18 READ 278528 ... okay
19 READ 294912 ... okay
20 READ 311296 ... okay
21 READ 327680 ... okay
22 READ 344064 ... okay
23 READ 360448 ... okay
24 READ 376832 ... okay
25 READ 393216 ... okay
26 READ 409600 ... okay
27 READ 425984 ... okay
28 READ 442368 ... okay
29 READ 458752 ... okay
30 READ 475136 ... okay
31 READ 491520 ... okay
32 READ 507904 ... okay
33 READ 524288 ... page fault
34 READ 540672 ... page fault
35 READ 557056 ... page fault
36 READ 573440 ... page fault
37 READ 589824 ... page fault
38 READ 606208 ... page fault
39 READ 622592 ... page fault
40 READ 638976 ... page fault
41 READ 655360 ... page fault
42 READ 671744 ... page fault
43 READ 688128 ... page fault
44 READ 704512 ... page fault
45 READ 720896 ... page fault
46 READ 737280 ... page fault
47 READ 753664 ... page fault
48 READ 770048 ... page fault
49 READ 786432 ... page fault
50 READ 802816 ... page fault
51 READ 819200 ... page fault
52 READ 835584 ... page fault
53 READ 851968 ... page fault
54 READ 868352 ... page fault
55 READ 884736 ... page fault
56 READ 901120 ... page fault
57 READ 917504 ... page fault
58 READ 933888 ... page fault
59 READ 950272 ... page fault
60 READ 966656 ... page fault

Plain Text  Tab Width: 8  Ln 1, Col 1  INS
```