

Homework 3 – CS 450 Spring 2022

1a. FIFO with 3 pages of physical memory

P1	P2	P3	P4	P1	P2	P5	P1	P2	P3	P4	P5
P1	P1 P2	P1 P2 P3	P2 P3 P4	P3 P4 P1	P4 P1 P2	P1 P2 P5	P1 P2 P5	P1 P2 P5	P2 P5 P3	P5 P3 P4	P5 P3 P4
Fault	Fault	Fault	Fault	Fault	Fault	Fault			Fault	Fault	

Total: 9 page faults

1b. FIFO with 4 pages of physical memory

P1	P2	P3	P4	P1	P2	P5	P1	P2	P3	P4	P5
P1	P1 P2	P1 P2 P3	P1 P2 P3 P4	P1 P2 P3 P4	P1 P2 P3 P4	P2 P3 P4 P5	P3 P4 P5 P1	P4 P5 P1 P2	P5 P1 P2 P3	P1 P2 P3 P4	P2 P3 P4 P5
Fault	Fault	Fault	Fault			Fault	Fault	Fault	Fault	Fault	Fault

Total: 10 page faults

With 3 pages of physical memory, the number of page faults was 9, and for 4 pages of physical memory, the number of page faults was 10. This is Belady's anomaly, having more page faults when increasing the number of pages of physical memory.

2a. LRU with 3 pages of physical memory

P1	P2	P3	P4	P1	P2	P5	P1	P2	P3	P4	P5
P1 (1)	P1 (2) P2 (1)	P1 (3) P2 (2) P3 (1)	P2 (3) P3 (2) P4 (1)	P3 (3) P4 (2) P1 (1)	P4 (3) P1 (2) P2 (1)	P1 (3) P2 (2) P5 (1)	P1 (1) P2 (3) P5 (2)	P1 (2) P2 (1) P5 (3)	P1 (3) P2 (2) P3 (1)	P2 (3) P3 (2) P4 (1)	P3 (3) P4 (2) P5 (1)
Fault	Fault	Fault	Fault	Fault	Fault	Fault			Fault	Fault	Fault

Total: 10 page faults

2b. LRU with 4 pages of physical memory

P1	P2	P3	P4	P1	P2	P5	P1	P2	P3	P4	P5
P1 (1)	P1 (2) P2 (1)	P1 (3) P2 (2) P3 (1)	P1 (4) P2 (3) P3 (2) P4 (1)	P1 (1) P2 (4) P3 (3) P4 (2)	P1 (2) P2 (1) P3 (4) P4 (3)	P1 (3) P2 (2) P4 (4) P5 (1)	P1 (1) P2 (3) P4 (4) P5 (2)	P1 (2) P2 (1) P4 (4) P5 (3)	P1 (3) P2 (2) P5 (4) P3 (1)	P1 (4) P2 (3) P3 (2) P4 (1)	P2 (4) P3 (3) P4 (2) P5 (1)
Fault	Fault	Fault	Fault			Fault			Fault	Fault	Fault

Total: 8 page faults

This aligns with LRU because with LRU, it is guaranteed to have fewer or the same number of page faults when adding more pages of physical memory.

3a.

```
Swap:      923        1      921
mark@mark-VirtualBox:~/Desktop$ free -m
              total        used         free      shared  buff/cache   available
Mem:          7931        1426          206         63       6298       6156
Swap:         923          1          921
mark@mark-VirtualBox:~/Desktop$ free -m
              total        used         free      shared  buff/cache   available
Mem:          7931        1286          346         63       6298       6296
Swap:         923          1          921
```

Difference: 140

This amount of difference is expected because we allocated 128mb of memory using malloc. Then we used bzero to fill up the allocated memory which is why there is a difference in free memory before and after pressing enter. Before pressing enter, the allocated memory is filled up, and thus the used memory is 1426mb, but after pressing enter, the used memory becomes 1286. So, the difference is 140 and this is to be expected.

3b. Commenting out bzero

```
Swap:      923        1      921
mark@mark-VirtualBox:~/Desktop$ free -m
              total        used         free      shared  buff/cache   available
Mem:          7931        1284          347         64       6298       6298
Swap:         923          1          921
mark@mark-VirtualBox:~/Desktop$ free -m
              total        used         free      shared  buff/cache   available
Mem:          7931        1284          348         64       6298       6299
Swap:         923          1          921
```

Difference: 0

This difference is much smaller than the previous because commenting out bzero is the cause of the difference in free memory. Malloc simply allocates and doesn't do anything with it. However, with bzero, it places n zero-valued bytes in the area pointed to, in this case p, the allocated memory 128mb. With bzero commented out, it no longer fills up the 128mb allocated memory with 0 bytes, so the memory is still free. Thus, the difference is much smaller, and in this case, there is no difference in free memory. Before pressing enter, the used memory is 1284 and after pressing enter, it is still 1284mb.

3c. pmap

```

mark@mark-VirtualBox:~/Desktop$ pmap 75207
75207: ./mem_test
0000555ab94de000      4K r---- mem_test
0000555ab94df000      4K r-x-- mem_test
0000555ab94e0000      4K r---- mem_test
0000555ab94e1000      4K r---- mem_test
0000555ab94e2000      4K rw--- mem_test
0000555aba010000     132K rw--- [ anon ]
00007fef6a128000    131076K rw--- [ anon ]
00007fef72129000     148K r---- libc-2.31.so
00007fef7214e000     1504K r-x-- libc-2.31.so
00007fef722c6000     296K r---- libc-2.31.so
00007fef72310000      4K ---- libc-2.31.so
00007fef72311000     12K r---- libc-2.31.so
00007fef72314000     12K rw--- libc-2.31.so
00007fef72317000     24K rw--- [ anon ]
00007fef72330000      4K r---- ld-2.31.so
00007fef72331000     140K r-x-- ld-2.31.so
00007fef72354000     32K r---- ld-2.31.so
00007fef7235d000      4K r---- ld-2.31.so
00007fef7235e000      4K rw--- ld-2.31.so
00007fef7235f000      4K rw--- [ anon ]
00007ffe1bd41000     132K rw--- [ stack ]
00007ffe1bdf2000     16K r---- [ anon ]
00007ffe1bdf6000      8K r-x-- [ anon ]
fffffffffff60000      4K --x-- [ anon ]
total                133576K

```

Pmap displays the memory map of the process. The columns are the address of the map, size of map, mode (permissions) and the last column is the files backing the map or [anon] or [stack]. [anon] is for allocated memory (malloc), and [stack] is for the program stack.

The first 4 lines are just the mem_test file with 4kb and all of them are having read permissions but one is also executable denoted by the r-x--. The next two lines are allocated memory, with 132kb and 131076kb. I'm assuming the one with 131076kb, or 131mb, is big due to us allocating 128mb in our program. Not sure what the other allocated memories, [anon], are for though. The size of the map for our program stack is 132kb and has read and write permissions. The other lines are the libraries used, libc-2.31 and ld-2.31. So, the process has its own mapping of the library that it uses and needs.