Operating Systems
HW # 4

Submitted by: Neeha Hammad

Problem 4.1:

(a) FIFO:

Reference string	**1**	**2**	**3**	**4**	**1**	**1**	**4**	**2**	**1**	**2**
Frame 0	1	1	3	3	1	1	1	1	1	1
Frame 1		2	2	4	4	4	4	2	2	2
Page Fault	*	*	*	*	*			*		

Total Faults: 6

(b) FIFO:

Reference string	**1**	**2**	**3**	**4**	**1**	**1**	**4**	**2**	**1**	**2**
Frame 0	1	1	1	4	4	4	4	4	4	4
Frame 1		2	2	2	1	1	1	1	1	1
Frame 2			3	3	3	3	3	2	2	2
Page Fault	*	*	*	*	*			*		

Total Faults: 6

(c) BO:

Reference string	**1**	**2**	**3**	**4**	**1**	**1**	**4**	**2**	**1**	**2**
Frame 0	1	1	1	1	1	1	1	1	1	1
Frame 1		2	3	4	4	4	4	2	2	2
Page Fault	*	*	*	*			*			

Total Faults: 5

(d) BO:

Reference string	**1**	**2**	**3**	**4**	**1**	**1**	**4**	**2**	**1**	**2**
Frame 0	1	1	1	1	1	1	1	1	1	1
Frame 1		2	2	2	2	2	2	2	2	2
Frame 2			3	4	4	4	4	4	4	4
Page Fault	*	*	*	*						

Total Faults: 4

(e) LRU:

Reference string	**1**	**2**	**3**	**4**	**1**	**1**	**4**	**2**	**1**	**2**
Frame 0	1	1	3	3	1	1	1	2	2	2
Frame 1		2	2	4	4	4	4	4	1	1
Page Fault	*	*	*	*	*			*	*	

Total Faults: 7

(f) LRU:

Reference string	**1**	**2**	**3**	**4**	**1**	**1**	**4**	**2**	**1**	**2**
Frame 0	1	1	1	4	4	4	4	4	4	4
Frame 1		2	2	2	1	1	1	1	1	1
Frame 2			3	3	3	3	3	2	2	2
Page Fault	*	*	*	*	*			*		

Total Faults: 6

Problem 4.2:

(a) The shell prints “HELLO WORLD”.

The rot13 transformation reversibly maps all 26 letters of the alphabet. This is done by adding 13 to the current alphabet, as far as the sum is still less than Z. For instance, A is swapped with N. B is swapped with O etc.

The upper transformation changes lower-case alphabets to upper-case alphabets.

First, librot13 is implemented, the letters are swapped. Next, they are converted to upper-case by libupper. Finally, they are converted back to their original letters with the librot13. The transformations are applied in the same order as they’re written on the command line.

(b) At first, I try to test for a single transformation by running the following command:

./cat++ -l ./librot13.so

and obtaining the PID in another terminal window by using:

ps aux | grep cat

In the output, I can see that librot13.so appears 4 times with 4 different logical addresses. Thus, we can say that there are 4 memory segments. However, in total there are 23 mappings with different logical addresses. The output remains the same even if I use librot13 twice.

Alternative Method:

A single process would be too quick to observe so I use the following command:

(echo "hello world" ; sleep 100) | ./cat++ -l ./librot13.so -l ./libupper.so -l ./librot13.so & pmap -x $!

Here, I observe a total of 10704 KBs. If we evaluate a single library’s line , we can see that the max kbytes are 2044.

When the same library is loaded multiple times, we see that the logical addresses change but the number of times the library name appears remains the same. For instance, if I only use one transformation (librot13), it appears 8 times with this command. When I use it twice, it still appears 8 times so we can say that 8 memory segments are being added for each library transform being called.

In general, for each library, we can also see that the mode changes from -r - x initially to just -r and then -r -w by the end usually. Total memory segments (lines) are about 60 for the above mentioned command.

If we run another command where we only apply one transform, for example, rot13, total Kbs are approximately 8648. There are 52 memory segments.

(c) -n can be used but we might have to dynamically allocate memory for it. Both programs perform the same function and can copy multiple (n) files to the standard output.

However, wc can't be used. This may be because in the cat program we're using buffers to keep track of what's being read but we don't do so in cat++.