Operating Systems
Jacobs University Bremen
Arsenij Percov

Course: CO20-320202
03 November 2019

# Homework 4

## Problem 4.1

**Solution:**

a)

| reference string | 1 | 2 | 3 | 4 | 1 | 1 | 4 | 2 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| frame 0 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| frame 1 | | 2 | 2 | 4 | 4 | 4 | 4 | 2 | 2 | 2 |
| page fault | * | * | * | * | * | | | * | | |

Total 6 page faults.

b)

| 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 6 page faults.

c)

| 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 5 page faults.

d)

| 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 4 page faults.

e)

| 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 7 page faults.

f)

| 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 6 page faults.

## Problem 4.2

**Solution:**

a)

The program loads transformation in the order it reads the arguments, we use getopt, that goes from left to right, then we save functions starting from 0 index, and execute them in the same order, so the order of transformations is defined by the order of provided arguments (transformations).

In this case we use first rot13 transformation, which either adds 13 to the first 13 letter of alphabet, or deducts 13 if it is letter from last 13 letters of alphabet.
After first transformation we have string uryyb jbeyq.
Next, we do the upper case transformation, which makes the string change every lower case letter to its upper case substitute.
The string will become URYYB JBEYQ.
Last we apply, again the rot13 transformation. Since letters that were before 13th became with index (index in context of alphabet) bigger than 13, when we deduct 13 from them, we will recieve the original letter. Therefore, two or any other even number of rot13 transformations on the same string will be canceled out.
The final string is:HELLO WORLD

b)
From the results of the pmap calls, I could observe that every time we load the new library 4 memory segments are added, 3 of the size 4k and 1 of the size 2044k. Also when we add same library multiple times nothing happens to the memory, and the same library is reused. You can find output from the cat++ with pmap calls inside in the archive (result.txt).

c)
Regarding the cat -n, I believe the answer is yes. We can create a function which we will use to make a library for that transformation. The function has one static int, which works as a count for the number of lines, and then we concatenate the count value with the original line, and we returned modified line.
For wc, we cannot implement it properly, because we output same number of lines, and there are in input, however for wc, we expect to receive number of lines equal to the number of files provided. However, we can make similar program,, but that will print the the counters' values for numbers, but every time updated value for each line.