Operating Systems – spring 2023
Tutorial-Assignment 6

Instructor: Hans P. Reiser

Submission Deadline: Monday, February 20th, 2023 – 23:59

A new assignment will be published every week. It must be completed before its submission deadline (late policy for programming assignments: up to two days, 10% penalty/day)

Lab Exercisess are theory and programming exercises discussed in the lab class. They are not graded, but should help you solve the graded questions and prepare for the final exam. Make sure to read and think about possible solutions before the lab class.

T-Questions are theory homework assignments and need to be answered directly on Canvas (quiz).

P-Questions are programming assignments. Download the provided template from Canvas. Do not fiddle with the compiler flags. Submission instructions can be found in the introductory section below.

The topic of this assignment is a deeper dive into page fault handling and page replacement.

Question 6.1: Hardware- vs. Software-Walked Page Tables

a. What's the difference between the use of hardware-walked page tables and software-walked page tables? How does this relate to TLBs?

b. What difference can you make out in the contents of software-walked vs. hardware-walked multi-level page tables?

c. Under what circumstances are TLB miss handlers or page fault handlers invoked?

Question 6.2: Page Fault Handling

a. Explain the terms demand-paging and pre-paging. What are the respective strengths and weaknesses?

b. When a thread touches (either reads or writes) a page for the first time with demand-paging, a page fault will occur. Classify the page fault according to where the data for the unmapped page has to be fetched from by the page fault handler.

c. If the process has been running for some time, modifying data along its way, there is one additional case that needs to be covered on a page fault. Which?

d. Discuss which information is required by the page fault handler to correctly setup (or restore) the contents of accessed pages.

e. Can you reuse page table entries to store some of this information? Is it a good idea?

f. What is Copy-on-Write? How can it be implemented?

g. Recap: Describe the steps necessary to handle a page fault in an application's address space.

Question 6.3: Page Replacement Basics

a. The pager (e.g., kswapd on Linux) of some systems tries to always offer a certain amount of free page frames to improve paging. What is the basic idea behind such a pager?

b. Describe the difference between a global and a local page replacement algorithm. Discuss the advantages and disadvantages of each of them.

c. Does a virtual memory system implementing equal allocation require a global or a local page replacement policy? Justify your answer.

d. What is thrashing? When does it occur?

e. What is the working set of a process? How can the working set be used to prevent thrashing?

Question 6.4: Page Replacement Policies

a. A task has four page frames $(0, \dots, 3)$ allocated to it. The virtual page number of each page frame, the time of the last loading of a page into each page frame, the time of the last access to the page frame, and the referenced (R) and modified (M) bits of each page frame are shown in the following table.

frame	virtual page	load time	access time	referenced	modified
0	2	60	161	0	1
1	1	130	160	0	0
2	0	26	162	1	0
3	3	20	163	1	1

A pagefault upon access to virtual page 4 occurs. Which page frame will have its contents replaced for the *FIFO*, *LRU*, *Clock* and *Optimal* (with respect to the number of page replacements) replacement policies?

For the Clock algorithm assume that the circular buffer is ordered ascending by load time and that the next-frame pointer refers to frame 3.

For the Optimal algorithm use the following string for subsequent references:
4, 0, 0, 0, 2, 4, 2, 1, 0, 3, 2.

Explain the reason in each case.

T-Question 6.1: Paging

a. Let's assume that shared memory between two processes A and B is realized on the page table level. Which of the folloing statements are true? **2 T-pt**

T-Question 6.2: Page Replacement

a. A page fault handler has to deal differently with *named entries (memory mapped files, such as code of a program)* and *anonymous mappings (page not backed by a file, such as the heap and stack of a program)* when it comes to eviction and loading of pages from/into physical memory. Which of the following statements are true? **2 T-pt**

b. Consider a system with a total of 4 page frames and an application accessing memory according to this reference string of virtual page numbers (VPNs): **4 1 2 7 3 5**. Complete the mapping table for a process at different times (t_0: initial mapping, t_1: mapping after accessing VPN 4, etc.) if **clock page replacement** is used. Assume that the circular buffer of the clock is in ascending order (i.e., frame 0, 1, 2, 3), that the clock hand at t_0 is positioned at **frame 0** and that the reference bit for **page 7** (in frame 2) is set, and cleared for the other pages. **3 T-pt**

frame	$VPN(t_0)$	$VPN(t_1)$	$VPN(t_2)$	$VPN(t_3)$	$VPN(t_4)$	$VPN(t_5)$	$VPN(t_6)$
0	3						
1	1						
2	7						
3	6						

c. Page faults for different algorithms **3 T-pt**

Note: The access sequence for this question was generated by the following command (see P part): `paging-policy.py -s 6 -n 15 -m 5`

Consider the page access sequence (reference string):

```
3 4 2 1 0 3 2 3 1 3 1 4 3 2 2
```

For each of the three policies FIFO, LRU and OPT, determine which of these access are page faults ("cache misses") in a system with a total of four physical frames ("cache size")! What is the total number of page faults?

P-Question 6.1: Page Replacement Simulation

For this assignment, there is no C programming required. Instead you will be doing experiments with page replacement strategies using the simulator from the OSTEP book (see `https://github.com/xxyzz/ostep-hw/blob/master/22/paging-policy.py` for the "original", there is a version available on Canvas with minor adjustments for Python 3). Please note the terminology used in the script (see also book chapter 22): The main memory of a process is seen as a "cache" of the virtual memory of a process. Consequently, the number of physical pages mapped in an application's virtual address space is called the "cache size".

a. Worst-case reference strings **1 P-pt**

Find a single, arbitrary access pattern with 24 elements that has an worst-case performance on all of these scheduling policies: FIFO, LRU, CLOCK and OPT.

You can simulate the execution of an arbitrary access pattern using the following command: (example: access pattern 1,2,3,1,2,3,1,2,3 on a system with cache size 5 and policy FIFO):

```
./paging-policy.py -a 1,2,3,1,2,3,1,2,3 -C 5 -p FIFO -c
```

For the submission of part (a) and (b), put your worst-case reference strings into the file `def-worstcase.sh`. You can run all simulations (a+b) using `run-worstcase.sh`

b. More worst-case reference strings. **3 P-pt**

For each of the three policies FIFO, LRU and MRU, find one access pattern **(reference string with 24 elements)** that uses **at most 5 different** virtual memory pages (cache size) and has a worst-case beheviour (maximizing page faults) on a system **with four physical frames.**

c. Paging policy silmulation with real workloads. **2 P-pt**

Compile the file `main.c` in the template and run the following command on skel to obtain a real memory access trace:
`valgrind --tool=lackey --trace-mem=yes ./main`
Implement a Python script `transform-lackey.py` that transforms the lackey output in a list of page numberss. The script shall read the lackey output on standard input and write *virtual page numbers* on standard output (note: page size on skel is the default page size of an x86-64 CPU).
Additional note: the valgrind output contains lines with type of access (I:code, L:load, S:store, M:modify), virtual address, and access size. You may discard the type and size of the access, and simply map the virtual address to the virtual page number.

d. Page replacement simulation **4 P-pt**

Use the trace from part (c) and the simulation tool to calculate the page hit rate for several sizes of the physical memory, for each of the policies FIFO, LRU, RAND and OPT. Plot a graph with the results: X-axis: number of pages in physical memory (1..total number of accessed blocks), Y-axis: page hit rate.
For the X-axis select a step size such that you have at least around 100 measurements.
Submission: a single file `simresult.png` with your results (all four policies plotted in a single diagram, and in colour).

Total:
10 T-pt
10 P-pt