

Instituto Superior de Engenharia de Lisboa
BEng in Computer Science and Engineering
System Virtualization Techniques, Autumn/Winter 2023/2024
Second coursework assignment

Place the `cw2.tar.gz` file in the **cw2** directory of your local repository and execute: `tar -xvzf cw2.tar.gz`

After executing that command, delete the `cw2.tar.gz` file, then commit and push the extracted files and directories. Each exercise has a specific directory: `cw2/ex1`, `cw2/ex2`, and `cw2/ex3`. When you need to write answers in text, use **.md** (Markdown) or **.adoc** (Asciidoc) files. You may answer in English or Portuguese.

Before starting this assignment, install additional packages in your Ubuntu 22.04 environment, by executing:


```
sudo apt update
sudo apt install qemu-user qemu-user-static
sudo apt install gcc-aarch64-linux-gnu binutils-aarch64-linux-gnu binutils-aarch64-linux-gnu-dbg
or, if you have a machine with an ARM64 processor, like a MacBook with M1 or M2:
sudo apt install gcc-x86-64-linux-gnu binutils-x86-64-linux-gnu binutils-x86-64-linux-gnu-dbg
```

In this assignment, instructions in light grey are specific for students using machines with ARM64 processors.

1. Consider the System Programming Manual for the AMD64 architecture, then answer the questions below:
<https://www.amd.com/content/dam/amd/en/documents/processor-tech-docs/programmer-references/24593.pdf>
 - a. Compare Figure 5-17 with Figure 5-18. The change from 4-level mapping to 5-level mapping increases the maximum size of a virtual address space. Calculate the maximum size for both cases.
 - b. Subsections 1.1.3 and 5.3.1 specify a *canonical address form* for virtual addresses. With 4-level mapping, canonical addresses go from `0x0000000000000000` to `0x_____` and then from `0x_____` to `0xFFFFFFFFFFFFFFFF`. Complete the missing values and write the canonical ranges for 5-level mapping.
 - c. [OPTIONAL] Find information in the manual to explain the following line of Linux kernel code:
https://github.com/torvalds/linux/blob/8cb1f10d8c4b716c88b87ae4402a3305d96e5db2/arch/x86/kernel/head_64.S#L188 (That is, what happens when line 188 is executed, followed by line 191?)

Tag these answers on the GitHub repository with: **CW2-1**

2. Observe the `x86-64` assembly code in `cw2/ex2/x86-64/prog.s`, then consult the documentation about Linux system calls to determine what the program does. <https://man7.org/linux/man-pages/man2/syscalls.2.html>
https://chromium.googlesource.com/chromiumos/docs/+master/constants/syscalls.md#x86_64-bit
You may also build (`make -f Makefile.arm64`) and run (`qemu-x86_64 ./prog`) the program to complement your analysis. It may be particularly useful to run the program with `strace`.
 - a. In the text file `cw2/ex2/a.md`, list and explain the sequence of calls performed by the program.
 - b. In the source file `cw2/ex2/arm64/prog.s`, replace `?` and `??` with adequate numbers, such that it becomes a working ARM64 version of `cw2/ex2/x86-64/prog.s`.

 Note that the sequence of system calls is exactly the same, but the instructions that prepare each particular system call are not in the same order in both files.

Build the program (`make -f Makefile.arm64`) and run it with an emulator: `qemu-aarch64 ./prog` (or just run `./prog` directly if you have a machine with an ARM64 processor)

- c. [OPTIONAL] In a text file, *cw2/ex2/c.md* or *cw2/ex2/c.adoc*, write a short text explaining what is QEMU User Emulation doing and, in particular, why is it relevant that it has a system call translator, as mentioned in the documentation: <https://www.qemu.org/docs/master/user/main.html>

Tag these answers on the GitHub repository with: **CW2-2**

3. Complete the source code in *cw2/ex3/prog.c* in the indicated places, in order to cause the following effects in the process virtual address space, which can be observed in `/proc/<pid>/smaps` :
- Increase the *resident set* (*Rss*) by about 2MB in the `.bss` region.
 - Access 128 bytes of initialized data (`.data`) with maximum impact in `Private Clean` pages.
 - Reduce the *Pss* of non-initialized data (`.bss`) to around 1MB for 30 seconds, while keeping *Rss*.
 - Add about 512KB of data region without increasing the total *Rss* for data.
 - Increase the *Rss* of the new data region by around 256KB.
 - Create two new regions in the address space, one for code (~ 4KB) and one for data (~ 256KB).
 - Increase `Private dirty` pages by about 128KB in the region for the data section created in f).

Solve this exercise using *cw2/ex3/prog.c* as the main file. Add other source and/or auxiliary files if needed.

Tag this solution on the GitHub repository with: **CW2-3**

Do not submit binaries and other unneeded files to the repository.

For the absolute final version, use the tag **CW2-DONE** on the GitHub repository.

ISEL, October 14th, 2023

Submission last date: October 25th, 2023