

Gaulthier Gain & Jonathan Francart

INFO0940  
OPERATING SYSTEMS

Project #2

Academic year 2020-2021



# YOUR SECOND AND LAST PROJECT (1)

## You will hack the kernel

1. You will implement several syscalls in order to retrieve specific information about processes in the Linux kernel. The main objective of this assignment is a practice for adding system calls in the Linux kernel.
2. You will mainly deal with some stats about the virtual memory of particular processes(es).
3. All will be done within the kernel space. We will provide you a user-space program[1] in order to test your syscalls.

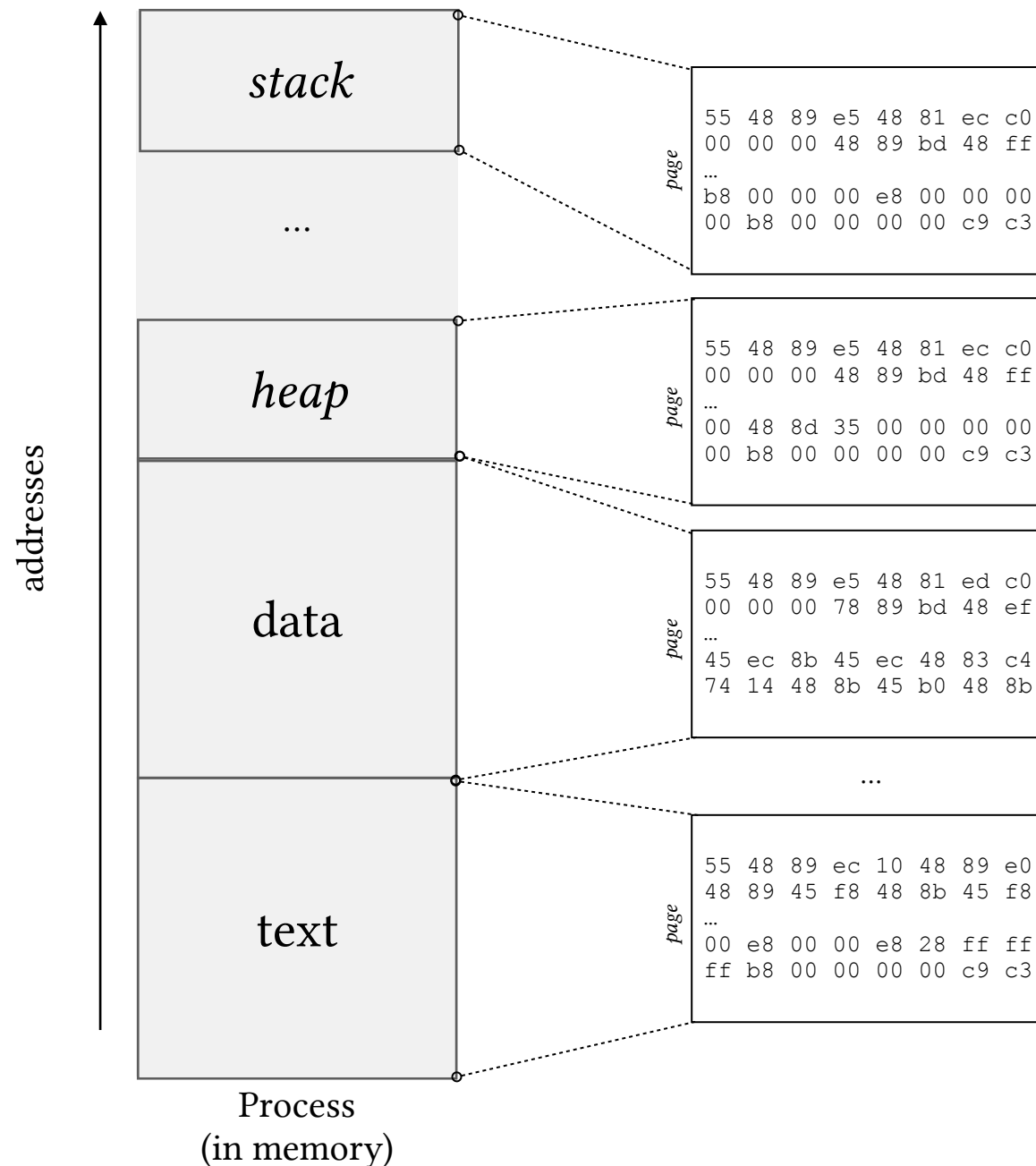
[1] see the tester program at the following link: [https://github.com/gauthiergain/INFO0940-1\\_project2](https://github.com/gauthiergain/INFO0940-1_project2)

# YOUR SECOND AND LAST PROJECT (2)

## It is a **research** project...

1. This means you have to do some research yourself to understand how things work.
2. It is not a waste of time. If you understand how memory works, you might pass the oral exam if you have a similar question.
3. Basically, it seems complicated but you will see that after some investigation, all will be clear.

# DEALING WITH VIRTUAL MEMORY

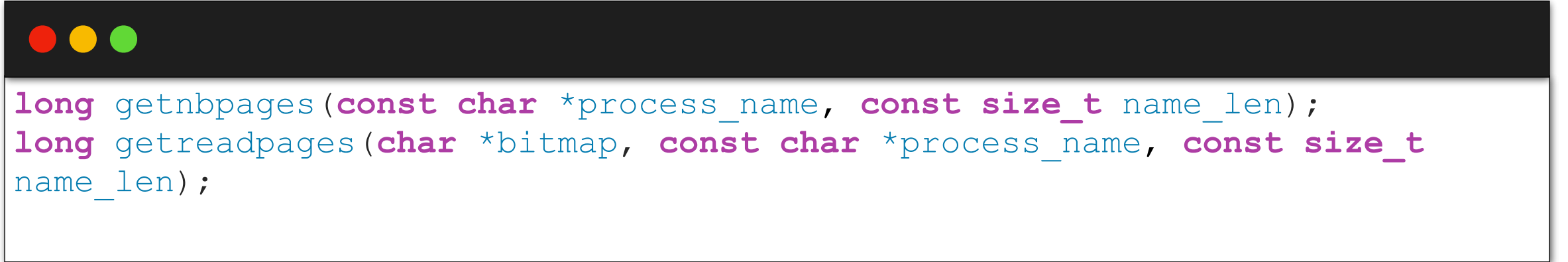


- ❖ Virtual memory is an abstraction which allows to create the illusion of a very large main memory.
- ❖ To manage memory efficiently, the kernel divides the virtual addressing of a process into various blocks of fixed size (by default 4KB), called **pages**.
- ❖ When manipulating memory, the kernel first needs to consult the **page table**[1] which is used by virtual memory to store the mapping between virtual addresses and physical addresses.
- ❖ The page table (managed by the kernel) also contains the permissions associated to pages.

[1] see theoretical courses

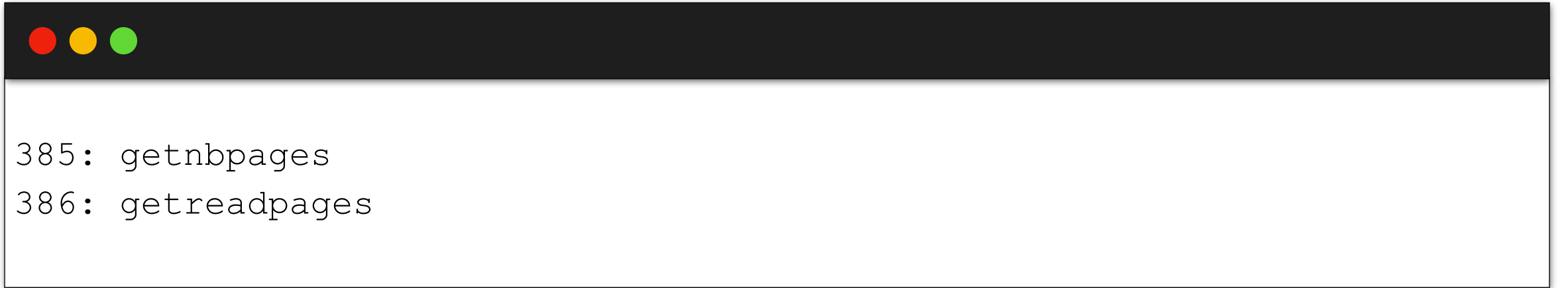
# HIGH OVERVIEW (1)

You need to implement the following syscalls:



```
long getnbpages(const char *process_name, const size_t name_len);  
long getreadpages(char *bitmap, const char *process_name, const size_t  
name_len);
```

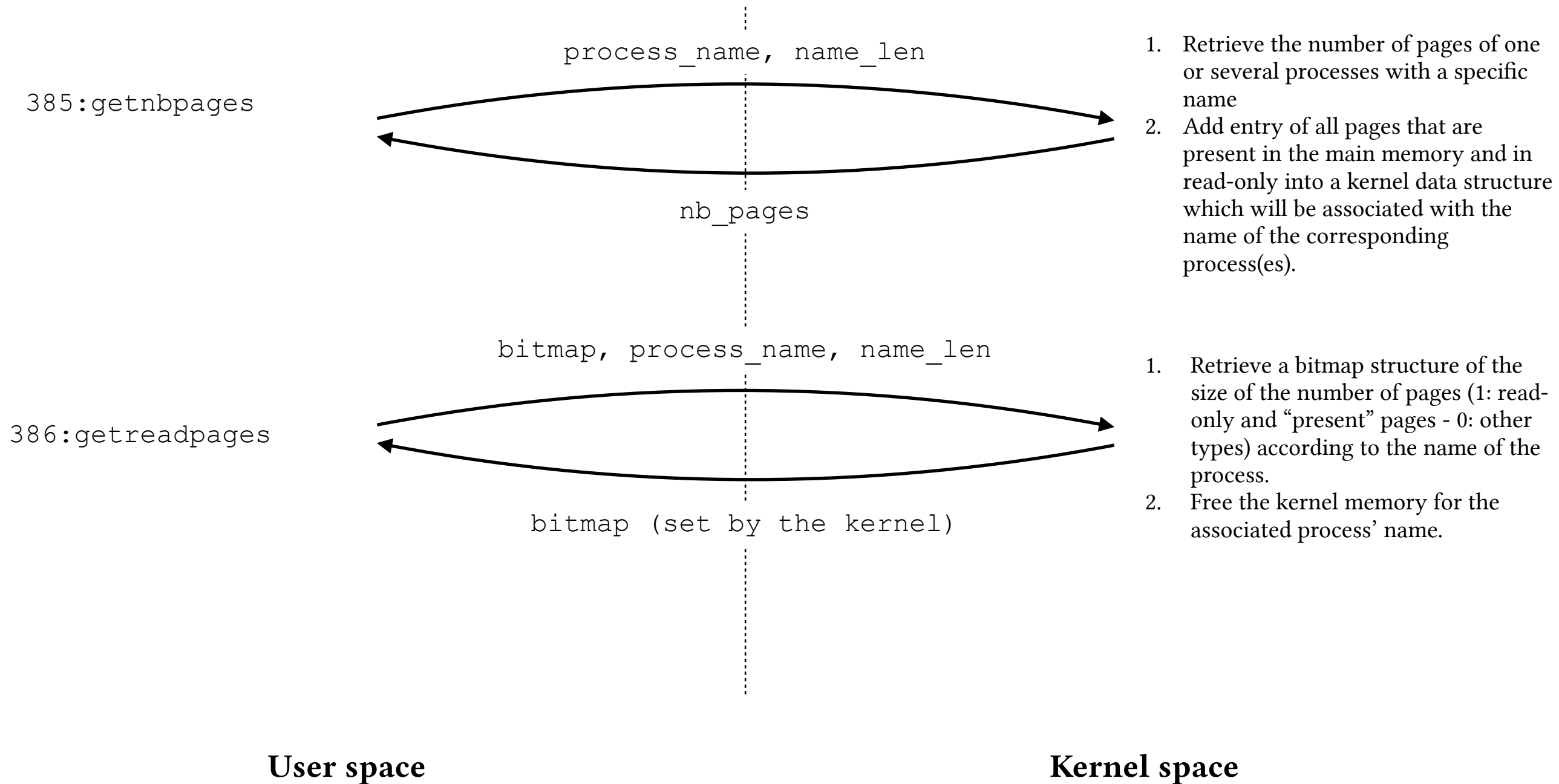
The system call number assignment is as follows (**do not change it**).



```
385: getnbpages  
386: getreadpages
```

[1] see podcast for further explanations

# HIGH OVERVIEW (2)



[1] error handling is not shown in this diagram

[2] see statements on eCampus for further information

*Demonstration*

*(see podcast)*

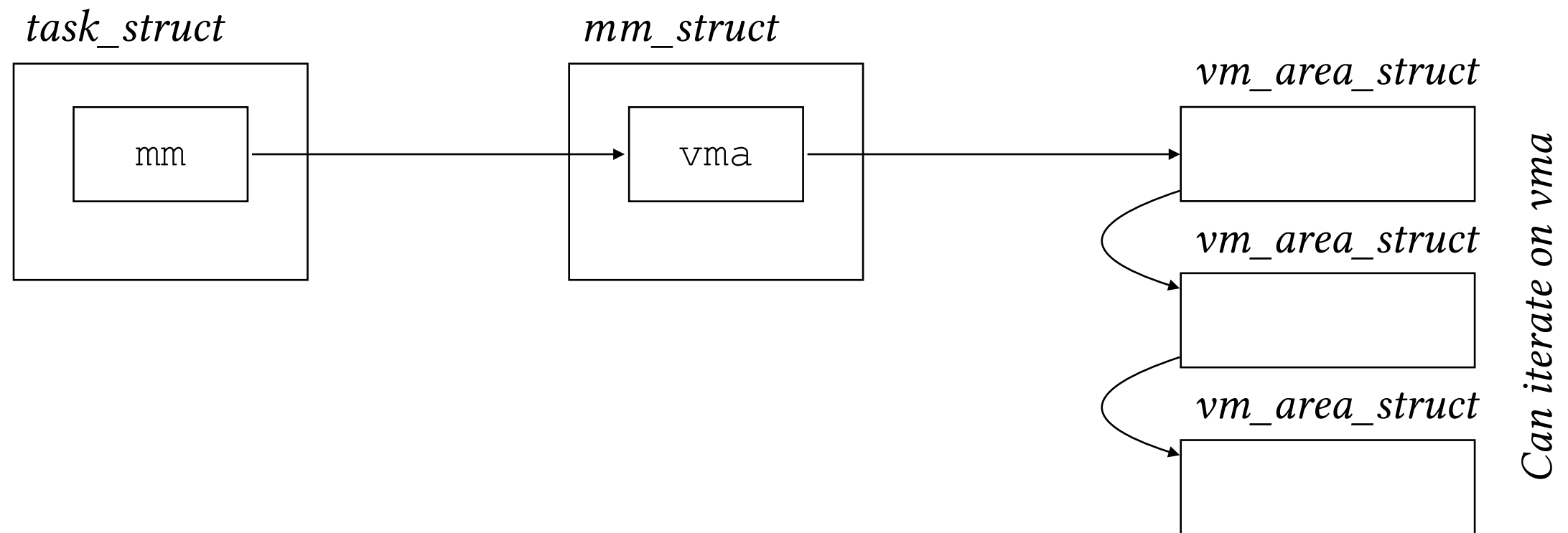
# HINTS (1)

For this project, you need to do some research but here are some hints to help you:

- ❖ You need to add 2 syscalls... think that we only consider **x86 32bits** architecture.
- ❖ You can investigate within the Linux source code and analyse existing system calls such as `read`, `write`, `fork`, ... (+ official documentation).
- ❖ Do not forget to check user inputs (see labs 5/6).
- ❖ Use `printf` to debug and `goto` for errors handling (see labs 5/6).
- ❖ Check if your kernel is upgraded (`uname -a`).
- ❖ You can use `make twice` to be sure that you see all the errors (if any).



# HINTS (2)



- ❖ A process is represented by a `task_struct` within the linux kernel. This one has a `mm_struct` which contains memory areas to keep track of the process's memory mappings (VMA).
- ❖ Each VMA has a start address, a length, and their sizes are always a multiple of the page size (4KB).
- ❖ As an entry point, it will be interesting to have a look at these files: [include/linux/sched.h](#) and [include/linux/mm\\_types.h](#) then do some googling.
- ❖ It is related to memory management so have also a look within the `mm/` subfolder (*pte*, *pgd*, etc.).
- ❖ Memory regions described by VMA are always virtually contiguous. You can check all VMAs associated with a process through the `/proc/<pid>/maps` file (see tutorial 3).

# SUBMISSION

- ❖ Submit all patches that have diffs from Linux-4.15 (the first commit).
- ❖ Patches should be made by `git format-patch` (refer to the slides/tutorial page to know how to use it). Note that we setup a test environment on the submit platform (see the last tutorial).
- ❖ The patch file(s) should be saved in a directory named `patch`, and it should be compressed into an archive called `patch.tar.gz`
- ❖ As you have seen the submission platform might crash sometimes. Try to submit as soon as possible (tests are already there).

# THE 32BITS REFERENCE MACHINE

Use the provided reference machine (32bits) to test  
your project.

- ❖ Do not implement your syscalls on another version of Linux than the one provided with the reference machine.
- ❖ Otherwise your program may fail some tests on the submit platform...
- ❖ Please refer to the online [tutorial](#) to setup the required environment.

# REQUIREMENTS

## Additional Information:

- Group of **two** that you will **keep** the whole semester.
- Submit a *tar.gz* archive on the submission platform (patch(es) & report).
- You are asked to write a very short report (max 2 pages) in which you briefly explain your implementation and answer to one additional question.
- Further information in the statements available on eCampus.

Do not forget: We want clean code and patches, without error, and warning.

Do not forget too: We detect **plagiarism** so don't try...

Plagiarism = **0 for the course!**

**Deadline: 12th May 2021**

*Happy Coding!*