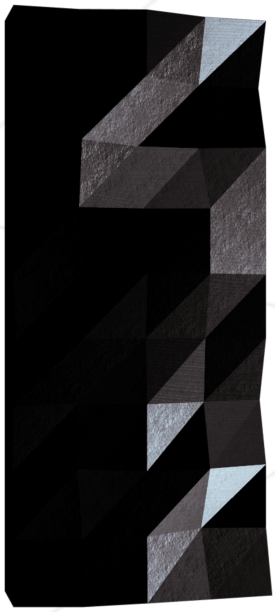

A completely UNIX project

ft_nm

Summary: This project is about recoding the command nm.

Contents

Chapter I

Foreword

theorem 1 (Lagrange)**.** *For any finite group G, the order (number of elements) of every subgroup H of G divides the order of G.*

Proof. Let $\sim$, be the relation defined by: for everything $x, y \in G$, $x \sim y$ if and only if there is a in H such as $ax = y$. Let's show that $\sim$ east an equivalence relation.

reflexivity $1x = x$.

Symmetry if $ax = y$ then $x = a^{-1}y$.

transitivity If $ax = y$ and $by = z$ then $(ba)x = z$

The following equivalence classes $\sim$ form a partition of G. By $x \in G$, $cl(x) = Hx$. If we show that all classes have the same cardinal, so we show that the cardinal of $cl(1) = H$ divides the cardinal from G.

Let $a, b \in G$. Let's explain a bijection of Ha dans Hb. Let $f : Ha \longrightarrow Hb$ as for all x in G, $f(x) = xa^{-1}b$. Let $g : Hb \longrightarrow Ha$ as for all x in G, $g(x) = xb^{-1}a$. For all $x \in G$, $f(g(x)) = xb^{-1}aa^{-1}b = x$ et $g(f(x)) = xa^{-1}bb^{-1}a = x$. So $g = f^{-1}$.

□

Chapter II

Sujet

Program name	`ft_nm`
Turn in files	
Makefile	`Yes`
Arguments	
External functs.	`open(2) close(2) mmap(2) munmap(2) write(2) fstat(2) malloc(3) free(3)`
Libft authorized	
Description	`You have to recode the nm (with no options)`

To do this project, you will have to use a linux docker/VM. You have to work with ELF binaries. You have to handle x86_32, x64, object files, .so

Use the `file` command to view details on a file. You can use the binaries located in your system (`/usr/bin/, /usr/lib/`...).

```
man nm
```

- This project will be corrected by humans only. You're allowed to organise and name your files as you see fit, but you must follow the following rules.
- The executable must be named `ft_nm`
- You must code in C and use a Makefile.
- Your Makefile must compile the project and must contain the usual rules.
- If you are clever, you will use your library for your `ft_nm`. Submit also your folder `libft` including its own `Makefile` at the root of your repository. Your Makefile will have to compile the library, and then compile your project.
- Output is to be similar to nm on the symbols list (order, offset, padding...). Other aspects can have some small differences (filename, etc).
- Your program must behave like the system nm on every other aspect. You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).

- Be cautious. There are many ways to lead your program out of the mapped content, may it be with non-null terminated string, incorrect offsets... Check everything.
- Make sure that every value that you parse is correct: for instance, when parsing flags, architecture, match your result with a reference to ensure its value is correct.

Chapter III

Bonus part

We will look at your bonus part if and only if your mandatory part is EXCELLENT. This means that your must complete the mandatory part, beginning to end, and your error management needs to be flawless, even in cases of twisted or bad usage. If that's not the case, your bonuses will be totally IGNORED.

Managing the following options:

- -a
- -g
- -u
- -r
- -p

Chapter IV

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.