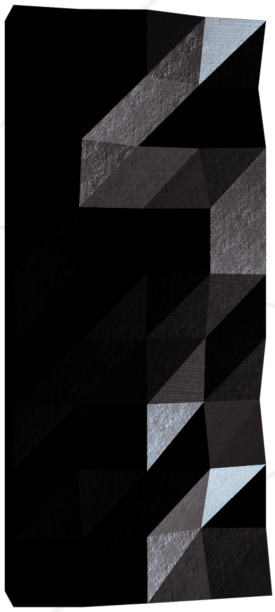

Projet complètement UNIX

ft_nm

Résumé: Ce projet consiste à recoder la commande nm (historiquement otool aussi sur OSX !)

Table des matières

I Préambule **2**

II Sujet **3**

Chapitre I

Préambule

Théorème 1 (Lagrange)**.** *Soit G un groupe fini et H un sous-groupe de G. Le cardinal de H divise le cardinal de G.*

Démonstration. Soit $\sim$, la relation définie par : Pour tout $x, y \in G$, $x \sim y$ si et seulement si il existe a dans H tel que $ax = y$. Montrons que $\sim$ est une relation d'équivalence.

Réflexivité $1x = x$.

Symétrie Si $ax = y$ alors $x = a^{-1}y$.

Transitivité Si $ax = y$ et $by = z$ alors $(ba)x = z$

Les classes d'équivalence suivant $\sim$ forment une partition de G. Pour $x \in G$, $cl(x) = Hx$. Si on montre que toutes les classes ont le même cardinal, alors on montre que le cardinal de $cl(1) = H$ divise le cardinal de G.

Soit $a, b \in G$. Explicitons une bijection de Ha dans Hb. Soit $f : Ha \longrightarrow Hb$ telle que pour tout x dans G, $f(x) = xa^{-1}b$. Soit $g : Hb \longrightarrow Ha$ telle que pour tout x dans G, $g(x) = xb^{-1}a$. Pour tout $x \in G$, $f(g(x)) = xb^{-1}aa^{-1}b = x$ et $g(f(x)) = xa^{-1}bb^{-1}a = x$. Ainsi $g = f^{-1}$.

□

Chapitre II

Sujet

Vous devez recoder la commande `nm` (sans option)

Vous devez travailler sur des fichiers ELF. Voous devez gérer x86_32, x64, universal binaries, object files, .dylib, .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
```

- Ce projet ne sera corrigé que par des humains. Vous êtes donc libres d'organiser et nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes listées ici.
- Vous pourrez en bonus, faire les options de nm.
- L'executables devra se nommer `ft_nm`
- Vous devez coder en C et rendre un Makefile.
- Si vous êtes malin et que vous utilisez votre biliothèque `libft`, vous devez en copier les sources et le `Makefile` associé dans un dossier nommé `libft` qui devra être à la racine de votre dépôt de rendu. Votre `Makefile` devra compiler la librairie, en appelant son `Makefile`, puis compiler votre projet.
- Vous devez gérer les erreurs de façon raisonnée. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, etc...).

- Vous avez le droit d'utiliser les fonctions suivantes :
  - open(2)
  - close(2)
  - mmap(2)
  - munmap(2)
  - write(2)
  - fstat(2)
  - malloc(3)
  - free(3)
- Vous pouvez poser vos questions sur le forum, sur jabber, IRC, ...