CHRISTOFOROU ANTHONY

RAPPORT

Difference between `sha1sum` (`md5sum`) on a file and `echo | sha1sum` (`md5sum`)

We can see that we obtain 2 different hash. Why? the answer is pretty simple, one character difference: `\n`

Usually, files don't have the newline character at the end but simply using echo does. A simple way to make it the same is add a newline inside the file or use:

```
echo -n "this is a test" | sha1sum
```

> the `-n` option removes the newline character

Implementation

Ce programme est composer de 2 modules differents et une fonction principale:

- `digest.c`
- `options.c`
- `main.c`

Digest

Ce module consiste de 2 fonctions qui vont s'occuper de hasher le message.

EVP String:

Cette fonction prend en entree le message a hasher et le nom du digest. On va utiliser des fonction de la librairie `openssl`, `crypto` qui nous permettrons de hasher le message tres rapidement et le print.

EVP File:

Meme principe que `EVP File` sauf on va commencer par ouvrir le fichier en lecture puis en utilisant une loop et `EVP_DigestUpdate` qui va permettre de hasher plusieurs messages on va lire le fichier ligne par ligne et rajouter le hash a chaque fois.

Options

Ce module va s'occuper de verifier les options entree par l'user. On utilise `getopt` (disponible dans le header `getopt.h`). La fonction va prendre en entree `argc`, `argv`, et une structure contenant un flag et le nom du digest. La suite est simple, on utilise getopt et selon ce que retourne la fonction si on a:

- `f` : on met le flag `is_file` a 1
- `t` : on recupere l'argument en t'en que nom de hash

- `?` : une erreur s'est produite

Main

Dans la fonction main, on va commencer par initialiser nos valeurs par default de notre structure `Options` par `{0, sha1}`. On appel `check_options` pour savoir sur quoi on va devoir operer. Finalement on a une boucle for si `is_file` est vrai au cas ou on aurait plusieurs nom de fichiers. Dans le cas ou c'est un String on va creer un buffer qui va recuperer chaque string pour en faire une phrase en ajoutant des espaces entre chaques mots.

Utilisation

Pour utiliser ce script il suffit de faire:

```
make
```

> Raccourci de `make all` : va permettre de compiler tout en liant les modules

Fichiers:

```
./digest -f file1 file2 ...
```

> Grace a l'option `-f` le programme va reconnaitre les arguments comme des fichiers

String:

```
./digest this is text
```

> Le script va hasher directement "this is text"