PROGRAMMATION ORIENTÉE SYSTÈME

Exercice Noté 1

5 mars 12:00–17 mars 23:59

INSTRUCTIONS (à lire attentivement)

IMPORTANT ! Veuillez suivre les instructions suivantes à la lettre sous peine de voir votre note annulée dans le cas contraire.

1. Cet exercice doit être réalisée **individuellement !** L'échange de code relatif à cet exercice noté est **strictement interdit** ! Cela inclut la diffusion de code sur le forum.

   Le code rendu doit être le résultat de **votre propre production**. Le plagiat de code, de quelque façon que de soit et quelle qu'en soit la source sera considéré comme de la tricherie (c'est, en plus, illégal et passible de poursuites pénales).

   En cas de tricherie, vous recevrez la note «NA» pour l'entièreté de la branche et serez de plus dénoncé(e) et sanctionné(e) suivant l'ordonnance sur la discipline.
2. Vous devez donc également garder le code produit pour cet exercice dans un endroit à l'accès *strictement* personnel.
3. Les deux fichiers à fournir pour cet exercice sont `filtre.sh` et `protege.sh` (respectez strictement ces noms). Ils ne devront plus être modifiés après la date et heure limite de rendu.
4. Veillez à rendre du code anonyme : **pas** de nom, ni numéro SCIPER, ni autre identification personnelle !
5. Utilisez le site Moodle du cours pour rendre vos exercices.

   Vous avez **jusqu'au 17 mars 23:59** (heure du site Moodle du cours faisant foi) pour soumettre votre rendu.
6. Lisez attentivement et *complètement* la question de façon à ne faire que, mais tout, ce qui vous est demandé.

   Si la donnée ne vous paraît pas clair, ou si vous avez un doute, demandez des précisions à l'un des assistants.

EXERCICE (Shell) :

Dans cet exercice, nous vous demandons de fournir deux scripts Shell `filtre.sh` et `protege.sh` (respectez strictement ces noms).

Ces scripts devront :
– être le plus robuste possible et se terminer avec un message d'erreur si quelque chose d'imprévu se produit ;
– éviter toute duplication de code ;
– contenir chacun au moins une fonction ;
– traiter également les sous-répertoires.

suite au dos ☞

`filtre.sh`

Vous êtes l'administrateur système d'un petit fournisseur d'accès à Internet. Périodiquement, vous recevez sous forme de documents textes, des listes d'adresses email qui ont été identifiées comme étant source de courriers indésirables et de virus par un organisme spécialisé. Ces fichiers textes contiennent une adresse email par ligne et sont tous stockés dans un même répertoire (ou ses sous-répertoires).

Vous souhaitez écrire un shell-script `filtre.sh` qui prend comme paramètre le chemin vers le répertoire contenant les fichiers d'adresses indésirables, ainsi qu'un nom de domaine Internet.

Exemple d'appel : `filtre.sh ./listes/bad_addresses epfl.ch`

Le but de ce shell-script est de renvoyer une liste de tous les noms d'utilisateurs présents dans ces fichiers pour le nom de domaine spécifié. Cette liste doit être triée par ordre alphabétique et ne pas contenir de doublons.

Il faudra par ailleurs veiller à ce que le script soit le plus robuste possible, par exemple en cas de mauvais arguments, de noms de fichier avec des espaces, ou de sous-répertoires présents dans le répertoire de référence.

On vous demande de plus à ce que la commande opère de façon récursive sur les sous-répertoires: tous les répertoires partie de l'arbre dont la racine est le premier paramètre doivent être pris en compte.

Supposez par exemple que le répertoire `bad_addresses` contient les fichiers et sous-répertoires suivants :

```
./bad_addresses           : file1.txt  dir1/  dir2/
./bad_addresses/dir1      : file2.txt
./bad_addresses/dir2      : file3.txt dir3/
./bad_addresses/dir2/dir3 : file4.txt
```

La commande

```
filtre.sh ./bad_addresses epfl.ch
```

cherchera dans les fichiers `file1.txt`, `file2.txt`, `file3.txt` et `file4.txt` des adresses qui appartiennent au domaine `epfl.ch`, par exemple `machin.truc@epfl.ch`.

Notes :

- Pour rechercher des mots dans un fichier, utilisez `grep` (`man grep` pour plus de détails), mais vous n'avez pas le droit d'utiliser l'option `-r`.
- Dans une adresse email, vous pouvez utiliser la commande « `cut -d '@' -f1` » pour séparer le nom de l'adresse (`man cut` pour plus de détails).
- Pour trier par ordre alphabétique, vous pouvez utiliser la commande « `sort` »; et pour éliminer les doublons, vous pouvez utiliser la commande « `uniq` ».

Exemple de déroulement (les lignes `##>` indiquent des *messages* (sorties) du programme)

Si le répertoire `~/bad_addresses/` contient deux fichiers contenant les listes d'adresses suivantes :

fichier 1	fichier 2
toto@epfl.ch	ducoude@unil.ch
titi@unil.ch	totobis@epfl.ch
tata@unil.ch	despieds@unil.ch
tutu@epfl.ch	dugenou@epfl.ch
toto@epfl.ch	ducoude@unil.ch
	delatete@epfl.ch

Votre script, appelé avec comme paramètres le nom de répertoire `~/bad_addresses` et le domaine `epfl.ch`, devra renvoyer les résultats suivants :

```
./filtre.sh ~/bad_addresses epfl.ch
##> delatete
##> dugenou
##> toto
##> totobis
##> tutu
```

`protege.sh`

Vous souhaitez maintenant écrire un script permettant de configurer les droits d'accès au contenu stocké sur votre serveur Web. Chaque répertoire à protéger contient un fichier `.htaccess` qui spécifie les droits d'accès pour chaque utilisateur. Ce fichier contient entre autres des lignes du type
`Require user hebus`
qui autorisent, par exemple, l'accès à l'utilisateur « hebus ».

Écrivez un script Shell `protege.sh` qui prend en paramètre le nom d'un répertoire à protéger et une liste (non vide) d'utilisateurs. Exemple d'appel :

`protege.sh /var/www/secret hebus cixi`

Ce script doit si nécessaire créer le fichier `.htaccess` dans le répertoire spécifié et y ajouter les autorisations d'accès pour les utilisateurs en question. Le fichier `.htaccess` doit contenir au maximum une entrée par utilisateur.

Si le répertoire donné comme paramètre contient des sous-répertoires, ils doivent également être traités de façon récursive : les utilisateurs en question doivent être ajoutés à un fichier `.htaccess` en chaque sous-répertoire.

Exemple de déroulement Supposez par exemple que le fichier `/var/www/secret/.htaccess` contienne déjà les lignes suivantes :

```
Require user cixi
Require user thanos
```

La commande

```
./protege.sh/var/www/secret hebus cixi
```

changera le contenu du fichier `/var/www/secret/.htaccess` en:

```
Require user cixi
Require user thanos
Require user hebus
```

Notez bien qu'elle n'a pas ajouté à nouveau de ligne pour `cixi`, vu qu'elle existait déjà (**Note :** pensez à nouveau à `grep`).

Par ailleurs, la commande précédente ajoutera les droit à `hebus` et `cixi` dans tous les sous-répertoires de `/var/www/secret`, et ceci récursivement (sous-répertoires des sous-répertoires, etc.).