TP shell (3) : grep, sed. . . (suite)

EXERCICE 1 : commandes diverses.

Ecrire une commande `audit` qui regarde dans votre *home* les fichiers exécutables par tous les utilisateurs et les affiche (utilisez `find` et `grep`).

Ecrire une commande `vire-ligne-vide` permettant de lister le contenu d'un fichier passé en argument de la commande, sans les lignes vides et en les numérotant.

Ecrire une commande `grep-rec` qui recherche dans l'arborescence située sous le premier argument, toutes les lignes qui vérifient l'expression située en second argument, dans le fichier passé en troisième argument.
Par exemple :

```
grep-rec . "^#"  "*.c"
```

Indication : utiliser `xargs`.

EXERCICE 2 : affiche-session.

Ecrire un script `affiche-session` qui affiche la liste de tous les fichiers modifiés dans la journée.
Ecrivez une version avec `sed`.

EXERCICE 3 : affiche

1. Écrire une commande `affiche` qui liste les noms des fichiers dans le répertoire courant qui sont tels que :
   - ils possèdent le suffixe `.c`
   - ils contiennent la ligne :

     `#include <stdio.h>`

2. Améliorer la commande afin de vérifier que la ligne `#include <stdio.h>` n'est présente qu'une seule fois et exclusivement sur la première ligne du fichier.

EXERCICE 4 : replace

Ecrire une commande `replace` qui est un *sed* simplifié, à utiliser dans les cas simples. La syntaxe de cette commande est la suivante : `replace chaine1 chaine2 fich`. Elle a pour effet de remplacer toute occurence de `chaine1` par `chaine2` dans le fichier `fich`.

Faites en sorte que cette commande fonctionne aussi dans le cas d'une liste de fichiers (par exemple : `replace chaine1 chaine2 '*.txt'`).

EXERCICE 5 : Commande chsuff.

Dans la page *man bash* on trouve l'extrait suivant qui vous sera utile d'appliquer dans l'exercice qui suit (plutôt que d'utiliser *sed*) :

```
Given
    a=/a/b/c/d
    b=b.xxx

    csh             bash        result
    ---             ----        ------
    $a:h             ${a%/*}        /a/b/c
    $a:t             ${a##*/}       d
    $b:r             ${b%.*}        b
    $b:e             ${b##*.}       xxx
```

---

1. Écrire une commande `chsuff` à deux arguments, le premier est une chaîne de caractères représentant un suffixe ou le symbole `-`, le deuxième est un nom de fichier ou de répertoire.

   L'exécution de la commande :

   `chsuff chaîne nom`

   a pour effet : • de remplacer le suffixe de <`nom`> par <`chaîne`>, si <`nom`> comporte un suffixe,
   ou • de renommer l'objet par <`nom`>.<`chaîne`> sinon.

   L'exécution de la commande :

   `chsuff - nom`

   a pour effet de supprimer le suffixe de <`nom`>, s'il existe.

   Exemples d'utilisation :

`chsuff old prog.c`	renomme `prog.c` en `prog.old`	*(changement de suffixe)*
`chsuff xx prog`	renomme `prog` en `prog.xx`	*(ajout d'un suffixe)*
`chsuff - prog.old`	renomme `prog.old` en `prog`	*(suppression du suffixe)*
`chsuff - fichier`	ne doit rien faire...	

2. Reprendre la commande de façon à ce qu'elle s'applique à une liste de noms de fichiers ou de répertoires.

3. *Questions subsidiaires.*
   Quand on demande l'exécution de la commande :

   `chsuff - fich`

il est possible qu'un message d'erreur de la commande `mv` s'affiche :

```
mv:  fich and fich are identical
```

Modifier votre *script* pour faire écrire le message par votre *script* et non pas par la commande `mv`.

4. Que se passe-t'il quand vous demandez la suppression du suffixe d'un fichier commençant par le caractère point ? Adaptez votre *script* en conséquence.