SCR.1.2 TP 12 ⊥ :

Les expressions régulières

Les commandes `grep` - `sed`

I. Pour tester ses expressions régulières, on pourra utiliser les programmes `regexp.c` pour la version basique des expressions régulières , et `extregexp.c` pour la version étendue.

Le premier argument à donner à l'exécutable est une expression régulière, et le second argument est une chaîne de caractères. Le programme affiche alors si la chaîne contient un motif qui est en correspondance avec l'expression. Les délimiteurs ' ' (quote) empêchent les caractères spéciaux internes d'être interprétés par le shell :

```
$ ./regexp 'aaa' zzzzzaaaaaaaaccccccc
Correspondance !
$ ./extregexp 'a{3,}' bbbbbaaaaaccccccc
Correspondance !
$ ./extregexp 'a{6,}' "zzzzzaaaaaccccc"
Pas de correspondance.
$ ./regexp '\baaa' zzzzzaaaaaaaccccc
Pas de correspondance.
$ ./regexp '\baaa' "zzzzz aaaaaccccc"
Correspondance !
```

II.

↪ La commande `grep` recherche les lignes du fichier en entrée qui contiennent une correspondance avec un motif donné sous la forme d'une expression régulière.

```
grep [OPTIONS] PATTERN [FILE...]
```

La commande `grep` travaille sur l'entrée standard si aucun nom de fichier n'est spécifié.

L'option `--color` de `grep` est vivement recommandée. Elle permet d'afficher en couleurs la chaîne de caractères qui est en correspondance avec l'expression régulière `PATTERN`. On peut définir, dans son `.bashrc`, la commande `grep --color` comme un alias de `grep`, comme on a fait pour certaines autres commandes, il y a quelques TP de cela.
Par défaut, la commande `grep` comprend la version basique des expressions régulières. Pour la version étendue, il faut utiliser `grep -E` ou `egrep`.

Les options suivantes de `grep` peuvent s'avérer utiles dans le cadre de ce TP :

`-E` `-i` `-v` `-w` `-c` `-n` `-L` `-l` `-r`

↪ `sed` (Stream Editor) est un éditeur de texte non intéractif. Les fichiers édités ne sont pas modifiés. Leur contenu est simplement utilisé pour construire un flux sur la sortie standard. Consulter `man sed` ou `info sed`.

Voici quelques formes simples de lignes de commandes avec `sed` :

```
sed [line_number]basic-command[\text] [input-file]
ou
sed [/motif/]basic-command[\text] [input-file]
ou
sed s/motif/replacement/[option/] [input-file]
```

Voici une description pour la première forme de la ligne de commande :

`basic-command` est appliquée sur la ligne numéro `line_number` des données en entrée. On réfère à la dernière ligne par le symbole `$`. Si `line_number` est omis, `basic-command` sera appliquée sur chaque ligne. Lorsqu'il faut entrer un texte `text` à `basic-command`, on le lui fournit juste à la suite du symbole `\`

Afin d'éviter que des caractères spéciaux pour le shell ne soient interprétés par le shell, on entoure les commandes données à `sed` par `' '` (quote).

Ce TP utilise essentiellement les formes suivantes de lignes de commandes avec `sed` :

```
sed [/REGEXP/]basic-command[\text] [input-file]
```
et
```
sed s/REGEXP/replacement/[FLAGS] [input-file]
```

où `REGEXP` est une expression régulière, et `FLAGS` est constitué d'un ou plusieurs indicateurs pour la commande `'s'` de `sed`. Voici une liste de tels indicateurs. Pour consulter la liste complète, on peut faire :

`info sed`, puis `sed scripts --> The "s" Command`.

```
The ‘s’ command can be followed by zero or more of the following FLAGS:

‘g’
     Apply the replacement to _all_ matches to the REGEXP, not just the first.

‘NUMBER’
     Only replace the NUMBERth match of the REGEXP.

‘p’
     If the substitution was made, then print the new pattern space.

‘w FILE-NAME’
     If the substitution was made, then write out the result to the named file.
```

III. Seules les commandes `grep, egrep, sed,` peuvent être utilisées. **On n'utilisera aucun pipeline**. Les expressions régulières proposées **doivent fonctionner sur n'importe quel fichier où on chercherait la même chose**.

Les fichiers sur lesquels on appliquera les commandes se trouvent dans `DIR/`

1. Afficher les lignes du fichier `sysctl.conf` qui contiennent `ipv4` ou `ipv6` entre deux symboles "`.`"
2. Afficher les lignes du fichier `sysctl.conf` qui ne commencent pas par le symbole "`#`" On proposera deux solutions ; une avec l'option spécifique offerte dans `grep` ; l'autre sans cette option. Vérifier qu'on a bien le même résultat.
3. Afficher les lignes du fichier `ca-certificates.conf` qui contiennent des noms de certificats avec la chaîne de caractères `root` entre deux symboles "`_`". Les lettres dans `root` peuvent être en minuscule ou en majuscule. Trouver l'option de `grep` correspondante.
4. Afficher les lignes du fichier `ca-certificates.conf` qui contiennent des noms de certificats avec la chaîne de caractères `Class` entre deux symboles "`_`", et suivie d'un chiffre.
5. Avec la commande `grep`, afficher le nom du fichier du répertoire `DIR/` où se trouve `KEYMAP`. On ne veut que le nom du fichier qui contient `KEYMAP`. On ne veut pas l'affichage des lignes qui contiennent `KEYMAP`.
6. Afficher les lignes du fichier `Xorg.0.log` qui contiennent (au moins) une expression entre parenthèses.

7. Le fichier `krb5.conf` contient des noms de machines d'une certaine forme (par exemple, `lithium.lcs.mit.edu` ou `mit.edu.`
   Le nombre de fois où le symbole `.` (dot) apparaît dans la composition du nom est quelconque. Afficher les lignes de ce fichier qui contiennent des noms de cette forme.
8. Afficher les lignes du fichier `syslog` qui contiennent (au moins) une chaîne de caractères formée de 4 nombres séparés par le symbole `.` (dot) (comme : `172.16.1.34`).
9. Afficher les lignes du fichier `sensors.conf` qui contiennent des expressions sous forme de produits où l'opérateur est entouré de caractères `<space>` comme `3.3 * 0.95` ou `5 * 0.95`
10. Le fichier de configuration de *Name Service Cache* s'appelle `nscd.conf`. Les lignes non commentées (dont le premier caractère autre que `<space>` n'est pas "`#`") permettent de configurer des services. Par exemple, on a :

    ```
    enable-cache        passwd      yes
    ```

    Afficher toutes les lignes non commentées de ce fichier, dont la colonne du milieu contient soit `passwd`, soit `services`.
11. Afficher toutes les lignes du fichier `man_db.conf` qui contiennent (au moins) deux chemins absolus (par exemple : `/bin /usr/share/man`), qui ne se suivent pas forcément.
12. Sans utiliser un éditeur de texte intéractif, sans modifier le fichier `syslog`, supprimer toute ligne dont le dernier caractère autre que `<space>` est `.` (dot), en plaçant le résultat de la transformation dans le fichier `sub-syslog`.
13. Sans utiliser un éditeur de texte intéractif, sans modifier le fichier `auth.log`, supprimer toute occurrence de la chaîne de caractères `RP2-13`, et écrire le résultat de la transformation correspondante de `auth.log` dans `sub-auth.log`.
14. Sans utiliser un éditeur de texte intéractif, sans modifier le fichier `pacman.conf`, commenter (ajouter un symbole "`#`" au début de) toutes les lignes non commentées qui commencent par `Include =` en mettant le résultat dans le fichier `pacman-v1.conf`.
15. Sans utiliser un éditeur de texte intéractif, sans modifier le fichier `sysctl.conf`, créer un fichier `sysctl-v1.conf` qui contient les mêmes lignes que le fichier `sysctl.conf`, mais dans lequel on supprime le premier symbole "`#`" de chaque ligne qui contient soit `ipv4`, soit `ipv6`, entre deux symboles "`.`"
16. Sans utiliser un éditeur de texte intéractif, sans modifier le fichier `daemon.log`, on veut remplacer dans chaque ligne de `daemon.log` qui se termine par `eth0.0'`, cette dernière occurrence de `eth0.0` par `eth1.0`. On veut enregistrer uniquement les lignes ainsi transformées dans le fichier `sub-daemon.log`.