TP 12 - Expressions regulières

1 Outils du Shell GNU

1 - Une Debian classique fournit deux outils de recherche de texte, `grep` et `egrep`, ce dernier utilisant par défaut les Regex "ERE". A l'aide de la commande `which`, donner la commande qui va donner la localisation de ces deux outils, et le résultat :

Commande : which grep && which egrep

localisation : `grep` : /usr/bin/grep `egrep` : /usr/bin/egrep

2 - En utilisant une substitution de commande utilisant la commande `file`, donner la commande qui va donner la nature de ces deux fichiers, ainsi que le résultat :

Commande : file /usr/bin/{commande}

type : `grep` : ELF 64-bit LSB pie executable `egrep` : OSIX shell script, ASCII text executable

3 - En utilisant une substitution de commande utilisant `du --bytes`, donner la commande qui va donner la taille de ces deux fichiers, ainsi que leur taille :

Commande : du -h /usr/bin/{commande}

taille : `grep` : 184K /usr/bin/grep `egrep` : 4,0K /usr/bin/egrep

4 - Pour ce dernier, l'afficher avec `cat`. Que pouvez-vous dire ? je comprends rien cmd=${0##*/}

2 Mise en pratique

Pour les exercices suivants, vous pourrez faire des essais préliminaires avec les outils en ligne donnés dans le cours, mais on demande une validation avec `grep` avec l'option `-P` (utilisation du dialecte "PCRE").

Vous trouverez ici : `https://github.com/skramm/but3_rt/tree/main/regex` les fichiers de données à utiliser.

Vous saisirez en ligne dans le questionnaire présent sur la page du cours sur Universitice les regex que vous proposez.

2.1 Nombres

Proposer une regex qui va matcher sur les chaines suivantes :
`3,14529 -255,34 128 1,9e10 123.340,00`

en évitant de matcher sur les suivantes :
`720p 384$ 248.22€`

Vous utiliserez ce fichier pour la validation :
`https://github.com/skramm/but3_rt/blob/main/regex/nombres.txt`

2.2 Noms de fichiers

Match : `img0912.jpg` `updated_img0912.png` `favicon.gif`

No match : `documentation.html` `.bash_profile` `img0912.jpg.tmp` `access.lock` `workspace.doc`

grep -P "^[[:alpha:]].*\.jpg$|png$|gif$" nom_fichiers.txt

2.3 Numéros de téléphone (Américains)

Dans un champ de texte, les utilisateurs doivent saisir des numéros de téléphone américains. Par exemple :
`415-555-1234` `650-555-2345` `(416)555-3456` `202 555 4567` `4035555678` `1 416 555 9292`

```
grep -P "^1?-?[( ]?\d{3}-?[ -)]?\d{3}-?[ ]?\d{4}$" no_tel_USA.txt
```

Vous utiliserez ce fichier pour la validation :
`https://github.com/skramm/but3_rt/blob/main/regex/no_tel_USA.txt`

2.4 Adresses IPv4

Matcher une adresse IP n'est pas si simple qu'il n'y parait : Il s'agit de 3 groupes de 3 chiffres, mais qui doivent respecter la contrainte de ne pas dépasser 255.

Commencer par écrire une Regex qui matche sur une valeur entre 1 et 255 avec un point derrière. Vous utiliserez ce fichier pour la validation :
`https://github.com/skramm/but3_rt/blob/main/regex/adresses_ipv4_1.txt`

```
grep -P "(^[1-9]\d?|^2[0-4]\d|^25[0-5]|^1\d\d)\.$" adresses_ipv4_1.txt
```

Puis l'étendre pour avoir la regex complète :

```

```

Vous utiliserez ce fichier pour la validation :
`https://github.com/skramm/but3_rt/blob/main/regex/adresses_ipv4_2.txt`

3 Analyse d'un fichier système

Le fichier `/etc/services` liste les services réseau et le port qui leur est assigné. Examiner ce fichier (**sans le modifier ! ! !**) et donner les commandes suivantes :

1 - Donner le nombre de services utilisant un port TCP :

→ Nbe=

2 - En prenant à la place du fichier réel le fichier ici :
`https://github.com/skramm/but3_rt/blob/main/regex/services.txt`
Donner le nombre de services utilisant un port UDP et avec un numéro de port inférieur à 1023.

→ Nbe=

4 Validation d'entrée utilisateur

Reprendre le code du TP4 : web dynamique Python/Flask dockerisé (on pourra laisser de coté la BDD MySql)

Ajouter le "endpoint" `/newuser/` à votre code Python/Flask. Faites en sorte que ceci produise une page web à partir d'un template Jinja contenant un simple formulaire web permettant de saisir un identifiant et un bouton d'envoi (page type d'une inscription à un service)

Puis, ajouter le code qui va récupérer la saisie utilisateur et valider au moyen d'une regex ce qui a été saisi. Il faudra utiliser le package Python "re" pour gérer les expressions régulières.

(doc ici : `https://docs.python.org/3/library/re.html`)

Pour la regex, deux approches sont envisageables :

- soit on fait une regex globale, qui vérifie que la chaine saisie respecte tous les critères
- soit on fait plusieurs regex qui vont tester **un** des critères, et on les teste successivement (plus simple et plus robuste).

On demande que les identifiants respectent les critères suivants :

- Au moins 6 caractères
- Au moins 1 chiffre
- Au moins 1 majuscule et 1 minuscule
- Au moins 1 caractère parmi les 5 suivants : `#%{}@`

Afficher un message indiquant si la saisie respecte ces critères ou pas.

Amélioration : indiquer sur la page quel est le critère que la saisie utilisateur ne respecte pas.