TP 2 - *Pipes* et `grep`

Dans le cadre de cette séance, vous devrez créer des commandes qui réalisent des tâches d'apparence complexe. Pourtant, ces commandes ne font que combiner plusieurs programmes à l'aide du mécanisme de *pipes* vu en cours. Essayez donc de décomposer les tâches à effectuer en morceaux plus simples pour trouver quelles commandes utiliser, comment les combiner et dans quel ordre.

Utilisation basique des filtres et des *pipes*

Consultez la section sur les filtres dans les notes de cours! Nous n'aurons pas besoin des expressions régulières avant l'exercice 5.

Exercice 1 (Un *pipe*)**.** Utilisez les filtres appropriés et les *pipes* pour répondre aux questions suivantes:

1. affichez les noms des groupes triés alphabétiquement du fichier `/etc/group` (il s'agit de la première colonne);
2. affichez les UIDs triés dans l'ordre croissant du fichier `/etc/passwd` (il s'agit de la troisième colonne);
3. uniquement la vingt-cinquième ligne d'un fichier;
4. les lignes 25 à 50 d'un fichier;
5. une ligne au hasard d'un fichier.
6. afficher uniquement les numéros des lignes de `/etc/apt/sources.list` où l'on trouve le mot `deb`;

Exercice 2 (`xargs`)**.** Certains programmes, par exemple `cat`, ne lisent pas les entrées sur `STDIN`, et demandent des paramètres à la place. Ainsi, si l'on veut afficher le contenu de tous les fichiers dont le nom commence par un `A`, ceci ne fonctionnera pas:

```
1 $ ls A* | cat
```

car `cat` ne reçoit pas d'entrées. On doit utiliser le programme `xargs` comme suit:

```
1 $ ls A* | xargs cat
```

Utilisez `xargs` et les *pipes* pour répondre aux questions suivantes:

1. en utilisant les données de `dataset01.zip` du TP 1, affichez la première ligne de chaque fichier dont le nom se termine par un `5`, en examinant ces fichiers dans l'ordre alphabétique inverse. Le résultat attendu est:

```
==> IE7L6PIDK5 <==
KOZX6A2R4NVOINHYAQ64LHAGC62GZKJXYP7AX5RVNOR3MDDRB2IRCIIQA7IXY34XC2RI4PTGOBGK

==> E7O56VJ4S5 <==
3FHQEJN7QL7OCGMETMW5Z7SJQ4WFS4WMEA4UQBDBKIRYFNWBWCQQGBHCMUGPRCHHDVS6D36QNN52

==> CPKIQZTYQ5 <==
RUDG4MVSIAGWR3KYUYHU2DTEVCGNPDXOXR3O3AMLWRQ7EIRD7SFFNEW4P6BUCDEHOZIST7YQCTCL
```

2. afficher avec sa taille le fichier .jpeg le plus gros parmi *tous* les fichiers du système.

Applications des *pipes*

Exercice 3. Écrivez une commande qui donne la liste de tous les paquets contenant un fichier dont le nom contient une chaîne donnée. On veut que les résultats soient triés alphabétiquement, numérotés, et sans doublons. Par exemple:

```
$ commande beamer.sty
     1        texlive-latex-extra
     2        texlive-pictures
     3        texlive-publishers
     4        texlive-xetex
```

Attention: on ne vous demande pas (encore) de créer une fonction. Une solution au format `proc1 beamer.sty | proc2 | ...` est suffisante.

`grep`: utilisation basique

Récupérez le fichier `dataset02.zip` sur la page du cours à l'aide de la commande suivante:

```
$ wget https://igm.univ-mlv.fr/~alabarre/teaching/shnu/linux/dataset02.zip
```

puis extrayez-le avec la commande `unzip`.

Exercice 4 (La comédie humaine)**.** Écrivez les commandes permettant de répondre aux questions suivantes:

1. dans quels fichiers de *La comédie humaine* le personnage Girodet apparaît-il?
2. modifiez votre réponse pour n'afficher que les noms des fichiers où Girodet apparaît, sans doublons;
3. combien de fois Girodet apparaît-il dans chaque volume?
4. affichez toutes les lignes qui contiennent Girodet, mais pas le mot "que";

`grep` avec expressions régulières

Rappel: pour que `grep` comprenne que l'on utilise des expressions régulières, il faut utiliser l'option `-E`.

Exercice 5. Filtrez la sortie de `ls -l` avec `grep` pour afficher uniquement les fichiers du répertoire courant satisfaisant les critères suivants:

1. le fichier est un répertoire;
2. le fichier n'est **pas** exécutable par les autres utilisateurs (aucune contrainte sur le propriétaire ni le groupe);

   (résultats attendus pour le contenu de `dataset01.zip`: `3OSTHVDVO3`, `57VHGBS66F`, `CJ6JWHPWTA`, `E7O56VJ4S5`, `FMKFQV4WVG`, `RWPDBNP2XD`, `WITZKK6GMV`)
3. le fichier est exécutable par tout le monde;

   (résultats attendus pour le contenu de `dataset01.zip`: `6OYZHVGCHJ`, `IE7L6PIDK5`, `IZ3M7TRPOY`, `NT3FTC27L2`, `SFHOWXM5LU`)
4. modifiez vos réponses aux questions précédentes pour n'obtenir que les noms des fichiers dans le résultat. Vous aurez besoin de filtrer au préalable la sortie avec `tr -s ' '`, qui permet de remplacer les séries d'espaces consécutifs par un seul espace. Par exemple:

```
$ echo "bonjour    tout le     monde" | tr -s ' '
bonjour tout le monde
```

Exercice 6. Comme on l'a vu, `apt-file search motif` renvoie tous les fichiers contenant `motif`, avec les noms des paquets qui les contiennent. Ceci n'est pas très utile pour trouver des *programmes*: en effet, la recherche du programme `compress` à l'aide de `apt-file search compress` renvoie plus de 9 000 résultats... Combinez cette commande avec `grep` pour n'obtenir que les fichiers dont le nom est *exactement* `motif`. Par exemple, pour `compress`, votre commande devra donner le résultat suivant:

```
ncompress: /usr/bin/compress
scrollz: /usr/share/scrollz/help/compress
```

Expressions régulières et groupes

Nous allons écrire des expressions régulières que nous voudrons tester sur des données réelles. Pour ce faire, nous aurons besoin de `curl`, un outil qui nous permettra de récupérer le contenu d'une page web (installez-le si vous ne l'avez pas). Pour afficher le code HTML d'une page web dans le terminal, on utilisera:

```
$ curl -s ADRESSE
```

Rappels:

1. utilisez `https://regexr.com/` pour vous aider à trouver les solutions;
2. il faudra utiliser `pcregrep` au lieu de `grep` (voir les notes de cours).

Exercice 7. Écrivez une expression régulière pour chacun des éléments suivants, et vérifiez vos réponses sur les pages données. Vos réponses ne doivent pas fonctionner sur n'importe quelle page web.

1. les adresses e-mail; le résultat attendu sur `https://canonical.com/contact-us` est:

   ```
   pr@canonical.com
   legal@canonical.com
   ```

   (ce n'est pas grave si chaque adresse apparaît deux fois)

2. les numéros de téléphone au format international; le résultat attendu sur `https://canonical.com/contact-us` est:

   ```
   +1 737 2040291
   +33 184889319
   +49 615 127 46816
   +81 3 6205 3075
   +34 932 201131
   +44 203 656 5291
   +44 203 656 5291
   ```

   (il est normal que la dernière ligne apparaisse deux fois)

3. les prénoms, triés par ordre alphabétique et sans doublon, des hommes parmi les députés français, dont la liste complète est disponible ici:

   `https://www2.assemblee-nationale.fr/deputes/liste/alphabetique`

   Attention: les lettres en français peuvent être accentuées, donc l'intervalle habituel `[a-z]` n'exprimera pas toutes les minuscules autorisées. Utilisez donc:

   - `[A-ZÀ-ÜÇ]` pour obtenir toutes les majuscules accentuées et le Ç;
   - `[a-zà-üç]` pour obtenir toutes les minuscules accentuées et le ç.

   De plus, on doit aussi autoriser:

   - les traits d'union (`-`) dans le prénom et le nom;
   - les espaces dans le nom;
   - les apostrophes (`'`) dans le nom.