Filtres

1 - Rappel sur les tubes

Un tube est la mise en relation du canal de sortie (STDOUT) d'une commande avec le canal d'entrée (STDIN) d'une autre commande.

Les tubes peuvent être mis bout à bout sans limite.

La commande à gauche d'un tube alimente en continu la commande à droite qui, elle-même, peut être la partie gauche d'un autre tube, et ainsi de suite.

La première commande alimente tout le système.

La dernière commande produit le résultat final souhaité.

La machine à compter les glaçons

Supposons que nous disposions des éléments suivants :

- robi : un robinet qui fournit une certaine quantité d'eau.
- freeze : une machine à faire des glaçons qui a une arrivée d'eau et une sortie pour les glaçons.
- count : une machine qui compte les objets qui entrent dedans sur un tapis roulant.

Voici maintenant la machine à compter les glaçons :

robi | freeze | count

Chaque “machine” fait son travail en transformant ce qui rentre en autre chose qui en sort pour passer à la machine suivante.

Peut-être que robi accepte un paramètre pour donner la quantité d'eau à fournir ? On aurait par exemple :

robi 2L | freeze | count

qui nous donne le nombre de glaçons qu'on peut produire avec 2L d'eau.

Les tubes fonctionnent exactement de la même chose avec les commandes.

Peut-être qu'il existe une autre machine, qui s'appelle pack, qui met des objets par paquet de N dans des sacs.

Elle reçoit les objets sur un tapis d'entrée et ressort les sacs sur un tapis de sortie. Ainsi :

robi 2L | freeze | pack 20 | count

donne maintenant le nombre de sacs de 20 glaçons qu'on peut produire avec 2L d'eau. Les possibilités sont infinies.

2 - Filtres

Les filtres sont des commandes qu'on appelle des utilitaires.

Ils font de petits traitements rudimentaires qui peuvent être très facilement combinés entre eux. Exactement comme nos machines évoquées précédemment (freeze, pack, count).

Plutôt qu'avoir des commandes qui font des tâches complexes mais qui peuvent aussi ne pas convenir à tous les besoins des utilisateurs, il est plus simple et efficace d'avoir des commandes très basiques mais qui peuvent être combinées entre elles avec des tubes pour faire très exactement ce que l'utilisateur attend.

Eviter le “trop spécifique”, un exemple

Avoir une commande qui recherche des noms de famille dans un fichier pour en extraire les prénoms et finir par les afficher en MAJUSCULES, c'est un besoin très spécifique et ponctuel.

Comment fait-on si on veut maintenant extraire le code postal de la ville de naissance et afficher juste le numéro de département ?

On a besoin d'une tout autre commande, encore très spécifique et ponctuelle.

Découper le problème

La solution est de découper le problème en petites tâches “élémentaires” qui répondent à des besoins simples et récurrents.

Des besoins simples et récurrents comme :

- Trouver des occurrences d'une chaîne dans un fichier.
- Extraire un champ ou une partie de texte dans une ligne.
- Transformer un texte en minuscules, MAJUSCULES, Capitales.

Ces tâches élémentaires sont des filtres qui peuvent être combinés à l'aide de tubes.

Rôles

Les filtres peuvent servir à travailler sur du texte (ce sera notre principal usage en TP/TD) :

- Extraire des informations.
- Compter des occurrences d'un texte.
- Transformer un texte (min, MAJ, Cap).
- Convertir des caractères.
- Trier des lignes.
- Sélectionner du texte suivant des critères.

Les filtres peuvent aussi travailler sur des binaires.
Quelques exemples :

- Convertir des images.
- Transformer des images.
- Extraire des informations EXIF d'images.
- Lire des fichiers compressés.
- Extraire des fichiers d'archives.

Outils programmables

Certains filtres sont de véritables outils programmables avec leur langage, plus ou moins évolué. Exemple :

- Awk
- Sed
- Find

D'autres proposent des paramètres très puissants :

- Grep et les expressions régulières.

3 - Types de fichiers

Les filtres que nous allons étudier travaillent généralement sur des :

- fichiers textes “au kilomètre”.
- fichiers textes structurés.
- résultats de commandes.

Il est plutôt commun de travailler sur des fichiers textes structurés et très souvent sur des résultats de commandes.

Fichiers textes structurés

Un fichier texte structuré contient généralement :

- Des lignes de texte séparées par des “Retours à la ligne” (\n).
  On les appelle aussi des enregistrements.
- Sur chaque ligne, des entités soit séparées par un caractère spécial : TAB, pipe (|), virgule, deux points, etc., soit placées à des positions fixes.
  On les appelle aussi des champs.

Exemples de fichiers structurés sous Linux :

- /etc/passwd : fichier des utilisateurs. 1 ligne par utilisateur, champs séparés par des “:”.
- /etc/crontab : liste des tâches automatiques et autonomes exécutées sans présence de l’utilisateur (batchs). Champs à positions fixes.

Autre exemple classique : fichiers CSV (Comma Separated Values) - Séparateur commun “;”, même si “Comma” signifie “Virgule” !

4 - Syntaxe

Les filtres s'utilisent très souvent avec les tubes.
La syntaxe est donc évidemment :

commande | filtre
commande | filtre > fichier

On peut aussi les utiliser seuls avec des redirections :

filtre < fichier
filtre < fic_source > fic_resultat

5 - Types de filtres

Comptage - wc

Commande wc : word count
Syntaxe : wc [option]

Compte :

- -w : des mots.
- -l : des lignes.
- -c : des caractères.

Compter des occurrences peut se faire très simplement en extrayant les lignes contenant ce qu'on cherche et en passant le résultat à un filtre de comptage :

- ls -l *.c | wc -l : compte le nb de fichiers de type .c
- grep Dupont < etu_2018 | wc -l : compte les Dupont dans la liste des étudiants 2018.

Tri - sort

Commande sort
Syntaxe : sort [option]

Permet de trier des lignes de texte suivant des critères en fonction des options utilisés.

sort est une commande dont la fonction est simple mais dont les options sont multiples et très complètes.

Usages communs :

- sort -t<sep> -k<champ> : permet de définir le caractère séparateur de champ (-t) et le numéro du champ (-k, pour key) servant à trier.
- sort -r : tri en sens inverse (reverse).
- sort -n : tri numérique, par défaut c'est alphabétique. (Exemple : 2 < 10 en numérique, pas en alphabétique).

Extraction sans doublons - uniq

Commande uniq
Syntaxe : uniq [-c]

Fonctionne uniquement sur un fichier trié, ou au moins ayant les lignes en doubles placées consécutivement.

Liste les lignes de la source en affichant une seule occurrence s'il y en a plusieurs consécutives.

Exemples avec le fichier villes :

- uniq < villes
- sort < ville | uniq

L'utilisation de la commande sort permet de trier le fichier pour être sûr de grouper les villes identiques et d'obtenir ainsi le résultat attendu.

Fichier villes :

- Lannion
- Rennes
- Quimper
- Rennes
- Rennes
- Brest

uniq propose une option de comptage (-c) :

- uniq -c < villes
- sort < ville | uniq -c

Le nombre d'occurrences s'affiche devant chaque ligne.

Suppression de colonnes - colrm

Commande colrm
Syntaxe : colrm col1 [col2]

Permet de supprimer des portions de texte sur chaque ligne, entre le caractère placé en colonne col1 et celui placé en colonne col2, inclus. La 1ère colonne est en 1.

Si col2 pas précisé : suppression jusqu'à la fin de la ligne.

Exemples :

- ls -l | colrm 1 11 : Supprime les blocs rwx du début de ligne.
- date | colrm 11 : Date en ne gardant que le Jour de la semaine + le mois + le jour du mois.

Extraction de colonnes - cut

Commande cut
Syntaxe : cut -d<sep> -f<champs>

Permet d'extraire certains champs de chaque ligne de texte en excluant tout le reste.

-d : indique le séparateur de champs.
-f : indique le ou les champs à extraire. 1er champ = 1

Exemples :

- cut -d':' -f1 < fichier : extraction du 1er champ (séparateur :).
- cut -d',' -f3,6 < fichier : extraction des champs 3 et 6 (séparateur ,).
- cut -d' ' -f2-5 < fichier : extraction des champs 2, 3, 4 et 5 (séparateur espace).

Extraction des premières lignes - head

Commande head
Syntaxe : head [-<n>]

Permet d'extraire des lignes de la tête (head) d'une source de texte.

-n : Pour indiquer le nombre, sinon, par défaut c'est 10.

Extraction des dernière lignes - tail

Commande tail
Syntaxe : tail [-<n>]

Permet d'extraire des lignes de la fin, de la queue (tail) d'une source de texte.

-n : Pour indiquer le nombre, sinon, par défaut c'est 10.

Les commandes head et tail sont très utiles quand un grand nombre de lignes sont traitées et qu'on ne s'intéresse qu'à quelques unes d'entre elles.

Par exemple, si la commande fait un tri par taille ou par date, et qu'on ne s'intéresse qu'aux valeurs les plus grandes ou les plus récentes. Exemple :

- ls -lS /bin | head -3 : les 3 fichiers les plus gros (Size).
- ls -ltr | tail -5 : les 5 fichiers les plus récents (-t : tri par "time", -r : sens "reverse").

Transformation - tr

Commande tr (Origine : translate)
Syntaxe : tr <quoi> <par_quoi>

Très simple et pratique, cette commande convertit une chaîne en remplaçant les caractères trouvés dans <quoi> par les caractères situés à la même position dans <par_quoi>.

Exemple :

- tr aeiouy AEIOUY < fichier : transforme les voyelles minuscules d'un texte en des MAJUSCULES. Le reste du texte est inchangé.

Exemple avec des jokers "à la Shell" :

- tr '[a-z]' '[A-Z]' < fichier : convertit tout le fichier en MAJUSCULES.

Les ' sont très importants et nécessaires, car ces jokers sont traités par tr et non pas par le Shell.

Autres syntaxes :

- tr -s <caractere> : Ne conserve qu'une seule occurrence de <caractere> si plusieurs apparaissent consécutivement. Exemple :
  tr -s ' ' < texte : supprime les espaces multiples consécutifs.
- tr -d <caractere> : Supprime toutes les occurrences de <caractere>.

Quelques raccourcis :

- [:upper:] : toutes les MAJUSCULES.
- [:lower:] : toutes les minuscules.
- [:digit:] : tous les chiffres.
- [:punct:] : tous les caractères de ponctuation.
- [:blank:] : tous les “blancs” (espaces, tabulations)

Exemple :

- tr [:lower:] [:upper:] : convertit les min en MAJ.

https://commons.wikimedia.org/wiki/File:Thats_all_folks.svg