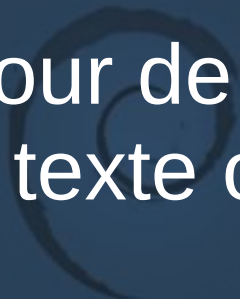


Chapitre 5 : Les filtres Unix

Commandes utiles pour de petits traitements sur des données de type texte organisées en tableau

The Debian logo, which is a stylized white spiral on a dark blue background, is positioned behind the text.

Debian

Le principe général

production
de données

filtre1

filtre2

TUBE1

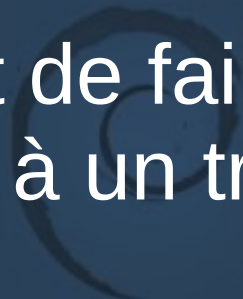
TUBE2

résultats



5.1 – Filtres et tubes

Les tubes permettent de faire travailler **plusieurs filtres** pour arriver à un traitement complexe

The Debian logo, which is a stylized white spiral on a dark blue background, is positioned behind the text.

Debian

Ordre des traitements

- Dans un tube, chaque commande traite des données venant de sa gauche et fournit les résultats à sa droite
 - Analogue à la composition de fonctions en maths



Rôles de ces filtres

- Extraire des informations dans des *textes*
 - compter les occurrences d'une information
 - comparer différentes informations
 - faire des classements, des sélections...
- Pour aller plus loin
 - il existe des outils programmables : sed, awk, perl...
- Traitement d'images avec la librairie netpbm
 - changer la taille, les couleurs...

Fichiers gérés par les filtres

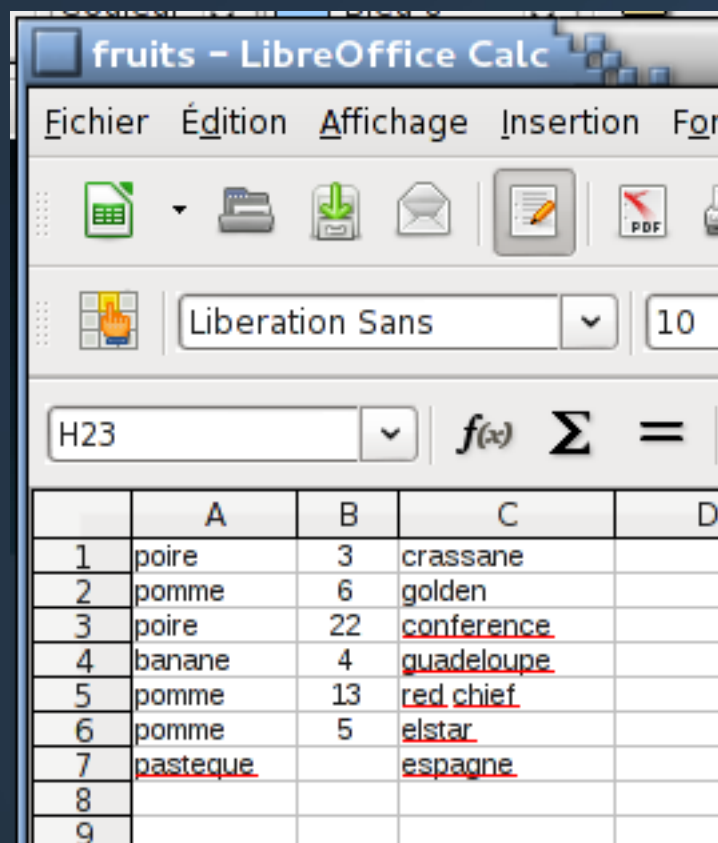
- Les **filtres** qu'on va voir travaillent sur des *textes* (lisibles, modifiables très facilement) :
 - rarement de vrais fichiers textes
 - souvent les **résultats de commandes Unix**
 - `ls`, `ps`, `find`, `cat`...
 - des **fichiers de configuration** du système :
 - `/etc/passwd` `/etc/crontab`
 - des fichiers CSV = **données structurées en lignes** :
 - **1 ligne = 1 entité (cf OMGL)**
 - **les colonnes = les attributs de l'entité**

Données structurées en ligne

- Certains fichiers sont organisés en **enregistrements** (*records*) de **champs** (*fields*)
 - chaque ligne décrit une entité : personne, chose...
 - Une entité est définie par différents champs.
ex : personne = (nom, prénom, n°sécu, adresse...)
- Les champs sont séparés par un caractère particulier, absent des données :
 - champ1[®]champ2[®]champ3[®]...
 - Souvent, c'est le caractère **:** qui sert de séparateur
ex: /etc/passwd décrit les comptes Unix

Fichiers CSV

- Ce genre de fichiers est connu des tableurs, c'est le format *CSV comma separated values*



	A	B	C	D
1	poire	3	crassane	
2	pomme	6	golden	
3	poire	22	<u>conference</u>	
4	banane	4	<u>guadeloupe</u>	
5	pomme	13	<u>red chief</u>	
6	pomme	5	<u>elstar</u>	
7	<u>pasteque</u>		<u>espagne</u>	
8				
9				

```
poire:3:crassane  
pomme:6:golden  
poire:22:conference  
banane:4:guadeloupe  
pomme:13:red chief  
pomme:5:elstar  
pasteque::espagne
```


Mode d'emploi général

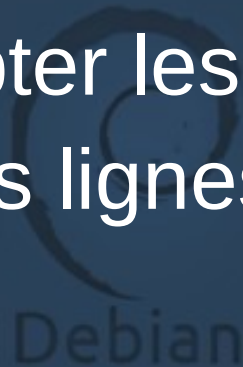
- On utilise un tube plus ou moins complexe :
`commande | filtre1 | filtre2 | filtre3...`
 - La commande envoie le texte à traiter dans les filtres
 - les filtres l'altèrent chacun leur tour (**attention ordre**)
- On peut aussi utiliser des redirections :
`filtre1 < fichE > fichtmp`
`filtre2 < fichtmp > fichS`

Catégories de filtres

- Comptage
 - des lignes, mots, caractères
 - des exemplaires des mêmes données
- Sélection d'une partie du texte
 - le début, la fin
 - des colonnes, des champs
 - certaines lignes contenant des chaînes précises
- Classement des lignes
- Divers
 - transformation des caractères, mise en page...

5.2 – Comptage

Compter les lignes
Compter les lignes identiques

The Debian logo, featuring a stylized spiral and the word "Debian" below it, is faintly visible in the background.

Debian

Comptage simple

- Commande `wc [-lwc]`
 - option `-l` : affiche le nombre de lignes
 - option `-w` : affiche le nombre de mots (!)
 - option `-c` : affiche le nombre de caractères du texte
- Exemples :
 - `ls | wc -l`
 - `grep pomme < fruits | wc -w`

Debian

Comptage des lignes identiques

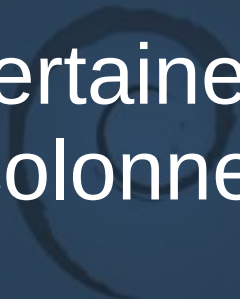
- Commande **uniq**
 - sans option : enlève les lignes successives identiques
 - option -c : compte les lignes successives identiques

- Exemple, fichier fruits :
 - `uniq < fruits`
 - `uniq -c < fruits`
 - `sort < fruits | uniq -c`

```
poire
pomme
poire
banane
banane
pomme
pomme
```

5.3 – Sélections

Garder ou enlever certaines lignes ou certaines colonnes

The Debian logo, which is a stylized spiral, is positioned behind the text 'Garder ou enlever certaines lignes ou certaines colonnes'.

Debian

Suppression de colonnes

- Commande `colrm col1 [col2]`

supprime les caractères de chaque ligne compris entre les deux numéros de colonne

- col1, col2 : n° des caractères (1 à 80...)
- si col2 absent => suppression jusqu'à fin de ligne
- Exemples :
 - `ls -l | colrm 10 47`
 - `who | colrm 9`

Sélection de champs dans un fichier CSV

- Commande `cut -d'®' -fnos des champs`

conserve les champs dont les numéros sont indiqués dans l'option `-f` (liste ou intervalle)

- Exemple, fichier fruits

– trois champs : type quantité variété, séparés par :

– `cut -d':' -f1 < fruits`

– `cut -d':' -f2,3 < fruits`

```
poire:3:crassane
pomme:6:golden
poire:22:conference
banane:4:guadeloupe
pomme:13:red chief
pomme:5:elstar
pasteque::espagne
abricot:54:abricot
```


Sélection de lignes par leur position

- Commande **head -n Nbre**
garde les Nbre premières lignes
- Commande **tail -n Nbre**
garde les Nbre dernières lignes
- Exemples :
 - `ls -S | head -n 5`
 - `head -n 4 < fruits | tail -n 2`
 - `cut -d: -f1 < fruits | sort | uniq -c | sort -n | tail -n 1`

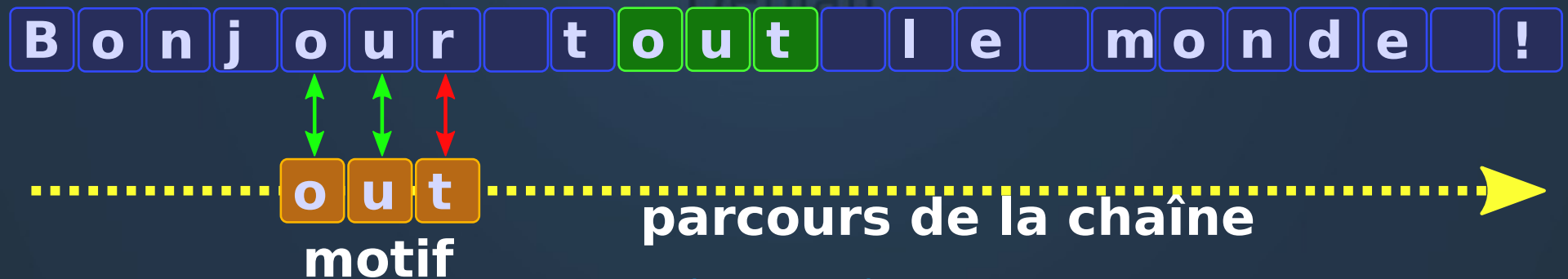
Sélection de lignes par leur contenu

`egrep options 'expression régulière'`

- affiche les lignes qui contiennent l'expression
- l'expression peut tout simplement être un mot
- l'expression peut contenir des caractères spéciaux appelés jokers permettant une recherche plus complexe : * + { } \ . ^ \$ [] () |
 - on l'appelle **expression régulière**, ou **expression rationnelle** (*regular expression, regexp*) ou **motif** (*pattern*) car elle définit la règle que doivent vérifier les données
 - on utilise les expressions régulières pour filtrer, vérifier des données

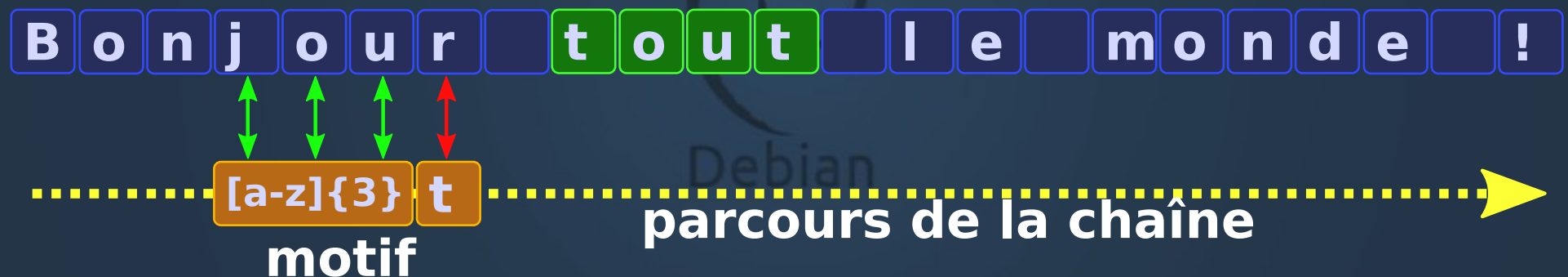
Pattern matching

- La commande egrep compare l'expression régulière (ou *motif*) à chaque ligne du fichier :
 - Elle met en correspondance l'expression avec la ligne et regarde caractère par caractère si c'est la même chose
 - Tout caractère est significatif (espaces...)
 - Ex : on cherche « out »



Pattern matching avec jokers

- L'expression régulière peut contenir des jokers, donc il y a une forme de raisonnement lors de la mise en correspondance
 - Ex : on cherche 3 lettres minuscules suivies d'un t



Utilité de egrep

- Permet de rechercher ou vérifier des données :
 - Ex : repérer des numéros de d'immatriculation de voitures : 2 lettres, 3 chiffres, 2 lettres espacés ou non
 $[A-Z]\{2\} \ ?[0-9]\{3\} \ ?[A-Z]\{2\}$
 - $[A-Z]$ signifie une lettre majuscule, $\{2\}$ signifie répétée 2 fois
 - Ex : vérifier des adresses mails
 $[a-z]^+(\.[a-z]^+)?@[a-z0-9.]^+$
 - $+$ signifie répété un certain nombre de fois, $?$: optionnel

Options de egrep

- Rajouter **-v** pour inverser la recherche : les lignes affichées ne contiennent pas l'expression
- Autres options :
 - Rajouter **-i** pour confondre majuscules et minuscules
 - Rajouter **-o** pour n'afficher que la partie qui correspond à l'expression régulière
 - Rajouter **--color** pour afficher en couleurs la partie qui correspond à l'expression régulière
 - Rajouter **-n** pour afficher les n° de ligne

Jokers de egrep

- Cinq types principaux de jokers :
 - **Correspondance** : le joker spécifie un ensemble de possibilités pour le prochain caractère du texte
 - **Répétition** : le joker indique qu'il y a une suite de caractères similaires
 - **Position** : le joker indique où se trouve l'expression dans la ligne de texte
 - **Alternatives** : les jokers indiquent un choix
 - **Grouperments et références**
 - **Divers**

Jokers divers

- Pour annuler le rôle spécial d'un caractère :

`\J` est le simple caractère `J`

- Ex : chercher les lignes qui contiennent des `*`

```
egrep '\*' texte
```



Jokers de correspondance

- Correspondance de caractères
 - un caractère normal correspond à ce caractère
 - `.` correspond à 1 caractère quelconque
 - `[liste]` correspond à 1 caractère de la liste
 - `[^liste]` correspond à 1 caractère hors liste
 - `[c1-c2]` correspond à 1 car. de l'intervalle
- Exemples d'expressions régulières
 - `bon[js]o[iu]r`
 - `02 9[9687] [0-9][0-9] [0-9][0-9] [0-9][0-9]`
 - `p...e:1[2-4]:`

Jokers de répétition

- Pour spécifier une séquence du même joker **J** :
 - **J*** un nombre quelconque de fois **J**, y compris 0
 - **J?** une fois **J** ou aucune
 - **J+** au moins une fois **J**
 - **J{n}** exactement n fois **J**
 - **J{n1,n2}** entre n1 et n2 fois **J**
- Exemples dans le fichier fruits :
 - **[a-z]+ : [0-9]* : [a-z]+**
 - **: [0-9][0-9]+ :**
 - **: [12][0-9]{12} :**

Jokers de position

- Ils spécifient où doit être le motif :
 - en début de ligne : `^motif`
 - en fin de ligne : `motif$`
- Exemples sur le fichier des fruits :
 - `^po`
 - `i[a-z]?e$`



Jokers d'alternatives

- Pour un choix entre plusieurs motifs :

(ceci | cela | . . .)

correspond soit à *ceci*, soit à *cela*, etc.

- Attention, la barre verticale n'est pas un tube
- Exemples :
 - `egrep '(rouge|jaune):' < fruits`
 - `egrep 'po(mm|ir)e' < fruits`

Groupement et référence

- Soit un motif contenant un (sous-motif)
 - Ex : Monsieur ([A-Z][a-z]+)
- Le sous-motif entre les () est appelée **groupe** et peut être réutilisé plus loin dans le motif avec la notation **\n°**, le n° étant celui du groupe
 - Ex : on cherche les lignes telles que le nom du fruit = le nom de sa variété

```
egrep '^([a-z]+):[0-9]*:\1$' fruits
```

\1 représente le 1er groupe, ici, c'est le nom du fruit

Retour sur les jokers du bash


- Ne pas mélanger egrep et bash (shell) :
 - **egrep** : les jokers spécifient les lignes à chercher

```
ls | egrep '^proj.*\.[co]$\nls | egrep '^.*t.*t.*$\nls | egrep '^projet\\.c$'
```
 - **bash** : les jokers spécifient les fichiers en paramètres

```
ls proj*.[co]\nls *t*t*\nls projet?.c
```
- bash a ses jokers : * ? [liste]
- egrep en a d'autres : .* [liste] |

5.4 – classements (tris)

Changer l'ordre des lignes

The Debian logo, which is a stylized spiral, is positioned behind the text 'Changer l'ordre des lignes'.

Debian

Classement des lignes

- Commande **sort**

met les lignes dans l'ordre lexicographique croissant

- Options à connaître :

-t'®' -k Nb classe selon le Nb^e champ

-r ordre décroissant

-n ce sont des nombres ($2 < 10$)

-s si possible garder l'ordre d'entrée (tris successifs)

Mélange des lignes

- Pour faire le travail « opposé » à sort, il existe la commande shuf

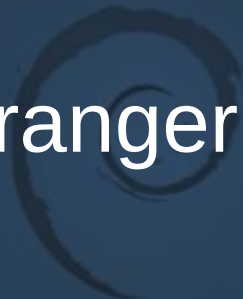
Elle range les lignes dans un ordre aléatoire et différent à chaque fois (*shuffle*)

Nb : cette commande n'est pas à connaître par cœur

Debian

5.5 – Inclassable

Transformer les caractères, numéroté les lignes...

The Debian logo, which is a stylized white spiral on a dark blue background, is positioned behind the text.

Debian

Afficher page par page

- La commande **more** affiche page par page le texte qu'on lui fournit sur son entrée
 - touche **q** pour arrêter (*quit*)
 - touche **espace** pour descendre d'une page
 - touche **b** pour remonter d'une page (*back*)
 - touche **/** pour chercher un motif
 - touche **n** pour chercher le suivant (*next*)
 - touche **N** pour chercher le précédent

Numérotation des lignes

- La commande **cat -n** numérote les lignes qu'on lui fournit sur son entrée
- Elle permet donc de capturer la n-ième ligne de données, exemple pour la 3e :

```
... | cat -n | egrep '^ *3 '
```

remarquer la présence de certains espaces !

- Utiliser **colrm 1 8** pour retirer les numéros

Transformation de caractères

- Commande **tr -s 'caractère'**

réduit les séquences du caractère à un seul

- **tr -s ' '** enlève les espaces en trop

- Commande **tr -d 'caractère'**

supprime toutes les occurrences du caractère

- Commande **tr 'table1' 'table2'**

transforme les caractères entrants : ceux de table1 deviennent leurs correspondants dans table2

- **tr '[a-z]' '[A-Z]'** met en majuscules

Commande tr, suite

- Pour simplifier l'écriture de certaines listes, il y a des symboles prédéfinis
 - [:upper:] lettres majuscules, accents compris
 - [:lower:] lettres minuscules, accents compris
 - [:blank:] espaces et tabulations
- On peut donc plus facilement convertir :
 - `tr '[:upper:]' '[:lower:]'`
 - `tr -s '[:blank:]' ' '`