Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2019 – 2020
DEV1 – ENVL

DEV1 – ENVL – Laboratoire d'environnement système

TD 6 – GNU/Linux (partie IV)

Table des matières

Dans votre répertoire `~/dev1`, créez un répertoire `td6`. Il contiendra tous les fichiers que vous allez créer aujourd'hui.

Version du document
21 septembre 2019

1 grep : recherche dans un fichier

On a déjà vu la commande `find` qui permet de rechercher un fichier en fonction de critères. La commande `grep` permet de faire une recherche **à l'intérieur** d'un fichier.

Grep

```
grep motif fichiers...
```

Dans son utilisation la plus simple, permet d'extraire de fichiers toutes les lignes qui contiennent un certain texte (appelé motif).

Penser aux jokers
Comme à chaque fois qu'une commande peut recevoir plusieurs noms de fichiers, on peut les indiquer explicitement et/ou utiliser les *jokers* pour en désigner plusieurs d'un coup.

Exemple 1

Où ai-je déjà utiliser Scanner ?

La commande `grep Scanner *.java` montre toutes les lignes contenant le mot `Scanner` dans tous les programmes Java du dossier courant.

Consultez la page de manuel pour voir comment paramétrer la commande.

2 Les filtres : présentation

La commande `grep` peut rechercher dans un fichier donné en paramètre mais aussi rechercher dans du texte donné en entrée.

Exemple 2

grep comme un filtre

```
cat Hello.java | grep Scanner
```

Avec cette écriture, le contenu du fichier `Hello.java` est envoyé à la commande `grep` qui n'affichera que les lignes contenant le mot `Scanner`.

On peut voir la commande comme un *filtre*.

Filtres - Définition

Les filtres sont des commandes qui lisent sur l'entrée standard et affichent sur la sortie standard une version réduite/modifiée de ce qui a été lu.

Il y a beaucoup d'autres commandes qui agissent comme des filtres. C'est à la base même de GNU/Linux : fournir des commandes qui font peu mais qui le font bien et les combiner (avec des tubes) pour obtenir un résultat conséquent [1].

1. C'est ce qu'on appelle le principe KISS, *Keep it simple, stupid !*

Exemple 3

Combiner les filtres

```
cat Hello.java | grep Scanner | grep System
```

Produit sur la sortie standard toutes les lignes du fichier `Hello.java` qui contiennent **à la fois** les mots `Scanner` et `System`.

3 Nourrir les filtres

Les filtres traitent les données reçues. Quelles peuvent être ces données ? N'importe quoi. Nous avons vu que ça peut être le contenu d'un fichier mais ça peut aussi être le résultat d'une commande quelconque. Afin d'enrichir nos exemples et exercices, voyons 2 commandes qui produisent des données.

Commandes pour nourrir les filtres

- ▷ `du nomDossier` : donne l'espace disque occupé par chaque sous-dossier du dossier indiqué.
- ▷ `who` : donne la liste des connexions à la machine.

4 sort : trier les lignes

La commande `sort` trie les lignes reçues.

Expérience 1

Trier les lignes

Expérimentons la commande `sort`.

- ✎ Placez-vous dans votre dossier `dev1`.
- ✎ `du .` affiche les dossiers et leur taille
- ✎ `du . | sort -n` affiche la même liste mais triée (numériquement sur la taille).

Consultez la page de manuel pour toutes les options.

Exercice 1

Trier le résultat de l'occupation disque

Reprenez l'exemple donné ci-dessus et voyez comment trier la liste des sous-dossiers et leur taille :

1. En ordre inverse de la taille (le plus gros d'abord).
2. Sur le nom du dossier.

5 uniq : enlever les doublons

La commande `uniq` ne laisse passer qu'une seule fois les lignes identiques **qui se suivent**.

Expérience 2

Enlever les doublons

- ✎ `cat | uniq`. La commande est en attente de texte et le reproduit tel quel.
- ✎ Faites-en l'expérience en entrant `salut` puis, à la ligne, `hello`.

- Entrez à nouveau `hello`. Il n'est **pas reproduit sur la sortie** ! La ligne précédente étant identique.
- Entrez à nouveau `salut` puis, à la ligne, `hello`. Ils sont reproduits car la ligne précédente est à chaque fois différente.
- Rappel : `CTRL-D` pour terminer l'expérience.

Exercice 2 **Les utilisations de Scanner**

Affichez toutes les lignes **différentes** contenant le mot `Scanner` dans tous vos fichiers JAVA du dossier `td4`.
Aide : Si vous devez enchainer plusieurs filtres, vous pouvez les tester étape par étape.

6 head / tail : les premières / dernières lignes

La commande `head` (respectivement `tail`) ne laisse passer que les premières (respectivement dernières) lignes reçues.

Expérience 3 **Les premières/dernières lignes**

`ls *.java | head -5` pour afficher le nom de 5 fichiers JAVA de votre dossier courant.

Exercice 3 **Les gros dossiers**

Affichez le nom et la taille des 3 dossiers qui prennent le plus d'espace disque parmi tous vos sous-dossiers de `dev1`.

7 tr : transformer des caractères

La commande `tr` permet de faire des transformation du texte : remplacez certains caractères par d'autres, en supprimer ou simplifier plusieurs occurrences consécutives d'un caractère

Exemple 4 **Transformer des caractères**

Voici quelques utilisations possibles :

- ▷ `ls | tr "jv" "gf"` transforme tous les `j` en `g` et les `v` en `f`.
- ▷ `ls | tr -d "a"` supprime tous les `a`.
- ▷ `who | tr -s " "` remplace toute suite de plusieurs espaces par un seul. Comparez le résultat à celui produit par `who` tout seul.

Consultez le manuel pour bien comprendre les exemples et pouvoir aller plus loin.

Elle est particulièrement utile pour la commande qui suit.

8 cut : ne garder que certaines colonnes

La commande `cut` sert à ne garder que certaines colonnes (*champs*) parmi les lignes reçues.

Exemple 5 **Utilisation de la commande `cut`**

- ▷ `du . | cut -f 2` pour ne garder que les noms des dossiers (qui se trouvent dans le *champ/colonne* 2).
- ▷ `ls | cut -d "." -f 1` pour ne garder que les noms des fichiers sans leur extension. L'option `d` modifie le séparateur de champ qui est une tabulation par défaut.

Expérience 4 **Les heures de connexions**

Imaginons qu'on veuille les heures de connexions des utilisateurs actuels de `linux1` (et rien que cela).

- ✎ `who`. L'information demandée est en colonne 4 mais il y a beaucoup plus.
- ✎ On aurait envie d'écrire `who | cut -f 4`. Ca ne fonctionne pas (essayez !) car la commande affiche des espaces et pas des tabulations entre les éléments.
- ✎ Alors ce sera `who | cut -d " " -f 4`. Toujours pas ! Lorsque 2 espaces se suivent, la commande considère qu'il y a une colonne (vide) entre eux.
- ✎ La bonne commande est `who | tr -s " " | cut -d " " -f 4`.

Exercice 4 **Les machines de connexion**

Affichez la liste des machines à partir desquelles les utilisateurs de `linux1` sont connectés.

a) Plus facile : le nom complet ;

b) Plus difficile : le nom court (ex : `L301P01` sans le domaine).

9 wc : compter

La commande `wc` est un peu particulière. Elle compte le nombre de lignes/mots/caractères reçus.

Exemple 6 **Le nombre de connexions**

`who | wc` donne le nombre de connexions actuellement.

Exercice 5 **Clarifier le résultat**

L'exemple précédent affiche 3 nombres. Regardez comment afficher une et une seule fois le nombre de connexions.

10 Récapitulatif

Filtres

Voici les quelques filtres que nous avons appris à utiliser :

- ▷ `grep` : ne laisse passer que certaines lignes.
- ▷ `sort` : trie les lignes
- ▷ `uniq` : enlève les lignes en double
- ▷ `head` : ne laisse passer que les premières lignes
- ▷ `tail` : ne laisse passer que les dernières lignes
- ▷ `cut` : ne laisse passer que certaines colonnes
- ▷ `tr` : modifie certains (groupes de) caractères
- ▷ `wc` : compte les lignes, les mots et les caractères

Voici quelques exercices récapitulatifs qui vont vous demander d'utiliser la plupart des concepts vus.

Exemple 7 **Le nombre de personnes connectées différentes**

Vous avez déjà écrit une solution pour afficher le nombre de connexions. Modifiez-le pour affichez le nombre de personnes connectées. La différence ? Si une personne est connectée plusieurs fois, elle ne doit être comptée qu'une seule fois !

Exemple 8 **Le plus gros dossier**

Affichez le nom (et rien que cela) de votre sous-dossier de `dev1` qui occupe le plus d'espace sur le disque.

11 Conclusion

Notions importantes de ce TD

Voici les notions importantes que vous devez avoir assimilées à la fin de ce TD.

- ▷ Comprendre le concept de filtre.
- ▷ Savoir utiliser et enchainer les bons filtres pour répondre aux questions qu'on se pose.

Pour aller plus loin...

Ceux qui ont du temps peuvent aborder les exercices suivants.

Exercice 6 **Un mot ou un autre**

Vous avez vu comment utiliser la commande `grep` pour sélectionner les lignes qui contiennent `Scanner` **ET** `System`. Comment faire pour sélectionner celles qui contiennent `Scanner` **OU** `System`.

Exercice 7 **Un chiffre**

Comment sélectionner avec `grep` toutes les lignes qui contiennent un chiffre, quel qu'il soit.

Exercice 8 **La casse**

Affichez le contenu de votre dossier en remplaçant toutes les minuscules par des majuscules.