Le filtres simples

1 Introduction

Dans les systèmes de type Unix, tout est considéré comme fichier. Il y a plusieurs types de fichier mais ceux qui nous intéressent, ce sont les fichiers contenant du texte ou à partir desquels on peut extraire du texte. Nous nous intéressons aussi aux sorties d'une commande qui produit du texte.

Quand on traite des fichiers, toutes les informations ne sont pas toujours pertinentes à tout moment. Il est donc important de pouvoir isoler les informations pertinentes, ou d'être en mesure d'organiser le fichier d'une manière plus accessible, ou encore d'être capable d'obtenir des informations liées au contenu du fichier. Supposons que vous ayez un fichier de plusieurs centaines de milliers de lignes, en extraire une information pertinente peut s'avérer complexe. Des commandes du shell permettent de faire ce travail simplement et rapidement. Cette activité s'intéresse à ces commandes-là. Nous allons en effet voir ici comment extraire des informations des fichiers textes, et nous allons pour cela utiliser des filtres simples.
Un filtre est un moyen, comme son nom l'indique, de simplifier, séparer, épurer ou clarifier un flux de données. Un flux de données est souvent un texte ou une chaîne de caractères.

2 Découpage d'un fichier

Comme nous l'avons déjà vu, les commandes **head** et **tail** permettent de n'afficher que les premières ou dernières lignes d'un fichier ou de l'entrée standard. Les syntaxes des deux commandes sont les suivantes :

```
$ head [options] <fichier>

$ tail [options] <fichier>
```

exemple les 15 premières lignes du fichier syslog

```
$ head -n 15 /var/log/syslog
```

L'utilisation d'une commande en tant que filtre se fait le plus souvent de la manière suivante (pour le même résultat) :

```
$ cat /var/log/syslog | head -n 15
```

De façon similaire, la commande suivante affiche les 15 dernières lignes du fichier */var/log/syslog* :

```
$ cat /var/log/syslog | tail -n 15
```

3 Modification de l’affichage d’un fichier à l’écran

La commande ***sort*** permet d’afficher la sortie standard ou le contenu d’un fichier en triant les lignes. On peut choisir les colonnes qui seront les clefs du tri. Cette commande ne supprime pas d’information : en sortie toutes les colonnes seront présentes et le contenu de chaque ligne sera inchangé.

Utilisée sans option, la commande ***sort*** trie le fichier donné en argument dans l’ordre alphabétique.

Prenons comme exemple le fichier suivant nommé tri.txt :

```
$ cat tri.txt
21 Béatrice 1 AA
3 Belle 2 BB
11 Alice 2 CC
```

```
$ sort tri.txt
11 Alice 2 CC
21 Béatrice 1 AA
3 Belle 2 BB
```

Notons ici que la commande précédente pourrait s’écrire : *cat tri.txt* | *sort.*
Le tri réalisé ci-dessus se fait par ordre alphabétique. Comme vous le savez, dans l’ordre alphabétique 1 et 2 sont avant 3, c’est pourquoi ici la ligne qui commence par 3 se retrouve en dernière position. Pour changer ça et effectuer un tri numérique il faut utiliser l’option ***-n***.

```
$ cat tri.txt | sort -n
3 Belle 2 BB
11 Alice 2 CC
21 Béatrice 1 AA
```

On peut aussi trier par ordre alphabétique sur la deuxième colonne, qui dans notre exemple correspond à la colonne des prénoms. Dans la commande sort on spécifie le numéro de la colonne avec l’option ***-k***.

```
$ cat tri.txt | sort -k 2
11 Alice 2 CC
3 Belle 2 BB
21 Béatrice 1 AA
```

L’ordre des deux dernières lignes s’explique par le fait que dans l’ordre alphabétique de l’ordinateur e est avant é. On peut combiner les options -n et -k si on veut trier sur la troisième colonne de manière numérique.

```
$ cat tri.txt | sort -nk 3
21 Béatrice 1 AA
11 Alice 2 CC
3 Belle 2 BB
```

Les autres options usuelles de la commande ***sort*** sont ***-t*** pour changer le séparateur de colonne, et ***-r*** pour inverser l’ordre. Pour plus d’options, vous pouvez consulter la page de manuel de sort.

4 Extraction d'information

La commande ***cut*** permet d'extraire des colonnes d'un fichier. Par exemple la commande **cut -c1-2 tri.txt** extrait les deux premiers caractères du fichier tri.txt,
alors que
cut -f2,5 tri.txt extrait les seconde et cinquième colonne.
Faites attention au fait que le séparateur par défaut est la tabulation,ce qui est différent du caractère espace. Le séparateur peut être modifié avec l'utilisation de l'option **-d**.

Considérons le fichierextr.txt suivant :

```
$ cat extr.txt
11 Alice 2 ; CC
21 Béatrice 1 ; AA
3 Belle 2 ; BB:
```

Pour extraire la deuxième colonne contenant les prénoms, on utilise la commande suivante. Attention, il faut redéfinir le séparateur de colonne pour que ce soit une espace et non une tabulation avec l'option **-d " "**. L'option **-f 2** permet de sélectionner la seconde colonne.

```
$ cut -d " " -f 2 extr.txt
Alice
Béatrice
Belle
```

Notez à nouveau que la commande précédente, comme toutes les commandes de filtres, peut aussi s'écrire :

cat extr.txt | cut -d " " -f 2

Pour extraire les deuxième et cinquième colonnes on utilise la commande suivante :

```
$ cat extr.txt | cut -d " " -f 2,5
Alice CC
Béatrice AA
Belle BB
```

Pour extraire les trois premières colonnes, on peut utiliser la commande précédente en y listant toutes les colonnes ou, dans le cas de notre fichier, changer le séparateur en ";" et extraire la première colonne en fonction de ce séparateur.

```
$cat extr.txt | cut -d ";" -f 1
11 Alice 2
21 Béatrice 1
3 Belle 2
```

Il est aussi possible de spécifier un ensemble de colonnes. Par exemple,

avec l'option **-f 1-3**, on peut extraire les colonnes 1 à 3 (soit les colonnes 1,2 et 3). On peut combiner la sélection des colonnes avec l'écriture

-f 1-3,5,7 qui sélectionnera les colonnes 1,2,3,5,7. Pour sélectionner la colonne 3 et toutes celles qui la suivent on peut utiliser l'écriture **-f 3-**.

5 Assemblage

1 La commande cat

La commande **cat** a déjà été vue. Elle permet de concaténer plusieurs fichiers les uns à la suite des autres. Les fichiers concaténés par la commande cat sont affichés sur la sortie standard. Supposons que nous ayons deux fichiers low.txt et high.txt. Le contenu de chaque fichier est le suivant :

```
$cat low.txt
1
2
3
$cat high.txt
4
5
6

$cat low.txt high.txt
1
2
3
4
5
6
```

6 Modification du contenu

Aucune des commandes précédentes ne supprime ni ne modifie une information spécifique. Dans cette section nous allons voir deux commandes simples permettant de modifier le contenu d'un fichier ou de l'entrée standard.

1 la commande uniq

La commande **uniq** permet de supprimer des lignes adjacentes identiques dans un fichier. Supposons que nous ayons le fichier suivant, appelé double.txt:

```
$catdouble.txt
AAAA
AAAA
BBBB
CCCC
CCCC
DDDD
AAAA
```

Le résultat de la commande **uniq** sur ce fichier sera le suivant :

```
$ uniq double.txt
AAAA
BBBB
CCCC
DDDD
AAAA
```

Les options les plus communes de la commande **uniq** sont **-d**, qui affichera seulement les lignes ayant des doublons adjacents, et **-u**, qui affichera seulement les lignes n'ayant pas de doublons adjacents.

```
$ uniq -d double.txt
AAAA
CCCC
```

```
$ uniq -u double.txt
BBBB
DDDD
AAAA
```

2 La commande tr

La commande `tr` (translate) remplace une liste de caractères par une autre. Supposons que l'on souhaite changer les majuscules en minuscules en utilisant la commande `tr`. Nous avons le fichier`slogan.txt` suivant :

```
$cat slogan.txt
Live free or Die UNIX
```

Pour changer toutes les majuscules en minuscules, on utilise :

```
$ tr "[A-Z]" "[a-z]" < slogan.txt
live free or die unix
```

Dans cette commande, **"[A-Z]"** correspond à toutes les lettres majuscules et **"[a-z]"** correspond à toutes les lettres minuscules. Dans ce cas "A" sera remplacé par "a", "B" par "b" etc.

3 Meta-information : wc

La commande **wc** (Word Count) est un peu particulière pour cette section car elle ne permet pas de transformer, d'extraire ou de mixer des informations d'un fichier texte ou de l'entrée standard. La commande `wc` fournit des méta-informations sur un fichier ou sur l'entrée standard. **wc** affiche le nombre de lignes, mots et octets contenus dans les fichiers dont les noms sont donnés en argument.

```
$ wc slogan4.txt
4 20 88 slogan4.txt
```

La commande **wc** s'utilise aussi comme filtre avec la syntaxe suivante, qui produit le même résultat :

```
$catslogan4.txt | wc
4 20 88 slogan4.txt
```