

Définition d'un filtre Unix

Nombre de commandes Unix permettent de traiter un flux de données textuelles leur parvenant sur leur entrée standard, et restituent le résultat de leur opération sur la sortie standard : elles sont alors nommées *filtres*.

- 1) Au verso de la fiche vous trouverez un peu plus de détail concernant la commande « sort » créez un fichier, et testez les différentes options disponible (modifiez le fichier de sorte à voir les changements apportés par chaque option)

Il s'agit ici d'apprendre à utiliser une nouvelle commande et ses options dans le détail en la testant comme une boîte noire, on donne à la commande « sort » un fichier et on compare le résultat avec celui de sort + option, l'ordre et le type de caractères données dans le fichier peut influencer sur le résultat, il faut donc faire plusieurs essais en modifiant le fichier à chaque fois pour comprendre le fonctionnement de l'option.

-b : ignore les espaces en début de ligne : « ignore » ne veut pas dire « supprime », ce n'est pas le rôle de sort, cette commande tri les lignes du fichier selon un certain ordre, il faut donc essayer de faire changer l'ordre du tri avec et sans l'option -b.

Après plusieurs essais vous vous rendrez compte, non-pas que l'option ne marche pas mais qu'elle est déjà active dans le sort sans option, la raison est que la commande est influencée par un paramètre de configuration général du système (la variable d'environnement LC_CTYPE), c'est-à-dire qu'il ne s'agit pas d'un problème inhérent à la commande sort, mais nous pouvons retrouver ce genre d'anomalies ailleurs en fonction du paramétrage du système.

Pour comprendre la raison il peut être utile de recourir à la commande « info » qui donne des explications plus détaillées sur les commandes.

Une autre possibilité peut être que l'option soit faite pour fonctionner en combinaison d'une autre option.

-f : ignore la casse : la casse signifie la différenciation des majuscules et minuscules, ici la différence ne se verra pas entre deux lettres ou deux mots différents mais entre deux lettres ou mot similaires avec une casse différente. L'exemple doit être modifié en conséquence.

Cette option dépend également de LC_CTYPE et ne va avoir aucun effet utilisée seule.

-r : inverse l'ordre de tri : inverse l'ordre du tri, ce qui est très facile à voir.

-d tri sur les caractères alphanumériques (caractères, chiffres et espace) uniquement : cette option permet de ne pas prendre en considération la ponctuation par exemple.

-tx Le caractère x est considéré comme séparateur de champ : à l'exemple du fichier /etc/passwd expliqué dans la fiche précédente, les fichiers texte peuvent être utilisés pour contenir des données sous forme de tableau à condition de définir un caractère qui servira de séparateur entre les colonnes. L'option « -t » permet donc de définir le séparateur qu'utilisera sort pour identifier les champs et ainsi procéder au tri à partir d'un champ particulier, on désigne les champs sur lequel on veut effectuer le tri en ajoutant un + suivi de la position du champ désiré puis un - suivi de la position du dernier champ

Exemple : sort -t : +2 -3 NomDeFichier effectuera un tri à partir du 3^{ème} champ (+2) jusqu'au 4^{ème} champ (-3) car la numérotation des champs se fait à partir de zéro (0)

-n trie sur des chiffres : permet de prendre en considération tout le nombre et non-pas chiffre par chiffre comme s'il s'agissait de caractères alphabétiques.

-u supprime les lignes doublon : l'option supprime les lignes en double, celles-ci doivent être strictement identiques, il est intéressant ici de faire le test en ajoutant des espaces et de voir le comportement de la commande, vous verrez qu'ici l'espace est pris en considération même si celui-ci est en début de ligne, par contre si vous ajoutez l'option **-b** vous verrez que la ligne disparaît. Même chose pour les options **-f** et **-d**

- 2) Copiez le fichier `/etc/passwd` dans votre répertoire personnel et renommez la copie du nom de votre groupe et vérifiez la taille du fichier.

```
cp /etc/passwd Gx
```

```
ls -l Gx
```

Découpez le fichier en morceau (fichiers) de 100 octets chacun, et observez le résultat et la taille des fichiers.

```
split -b100 Gx
```

```
ls -l
```

vous remarquerez que le dernier fichier sera plus petit car il contiendra ce qui reste, à moins bien entendu que le fichier source ait une taille multiple de 100 octets.

Essayez de lire quelques-uns des blocs

Vous verrez que les lignes sont coupées en plein milieu et que le nouveau prompt apparaît sur la dernière ligne de résultat

Supprimez les fragments (fichiers) d'une seule commande

```
rm x*
```

Redécoupez à nouveau le fichier en morceaux de 3 ligne chacun, observez le résultat et la taille des fichiers

```
split -l3 Gx
```

 les fichiers ont cette fois-ci des tailles différentes car les lignes n'ont pas la même longueur

Essayez d'en lire quelques-uns, observez le contenu et supprimez-les de nouveau.

Les lignes ne sont pas entrecoupées, cette fois-ci, le dernier fichier va contenir les lignes restantes.

- 3) Créez un fichier « `pass1` » contenant les lignes 6,7,8,9,10 et 11 du fichier `passwd` à l'aide des commandes `head` et `tail`

```
head -11 < Gx | tail -6 > pass1
```

- 4) Remplacez le séparateur « `:` » du fichier « `pass1` » par le symbole de votre choix.

```
tr ':' '*' < pass1
```

Remplacez tous les nombres par des caractères « `z` »

```
tr '0-9' 'z' < pass1 ou alors tr '0123456789' 'z' < pass1 ou tr [:digit:] 'z' < pass1
```

Mettez tout le texte en majuscule

```
tr 'a-z' 'A-Z' < pass1 ou alors tr [:lower:] [:upper:] < pass1
```

Refaites les trois dernières opérations dans un pipeline.

```
tr ':' '*' < pass1 | tr '0-9' 'z' | tr 'a-z' 'A-Z'
```

- 5) Testez la commande « `wc` » avec et sans ses différentes options.

```
wc pass1
```

```
wc -l pass1
```

```
wc -w pass1
```

```
wc -c pass1
```

vous remarquerez que la ligne entière est considérée comme un seul mot s'il n'y a pas d'espaces.

Vous remarquerez également en faisant le test avec un fichier plus petit que le retour chariot (saut de ligne) est également comptabilisé comme un caractère (vous pouvez également

déduire cela en observant la taille du fichier en octets qui sera égale au nombre de caractères car un caractère occupera exactement un octet)

- 6) A partir du fichier « pass1 », testez la commande « cut » en extrayant les 10 premiers caractères de chaque ligne.

`cut -c 1-10 pass1`

Extrayez les logins (**champ 1**) et les UUIDs (**champ 3**) du fichier pass1 et enregistrez-les dans un fichier « pass2 »

`cut -d : -f 1,3 pass1 > pass2` ,il faut préciser quel est le séparateur avec l'option -d

- 7) Créez le fichier « pass3 » semblable à « pass2 » mais avec deux lignes en moins.

`head -4 pass2 > pass3`

Extrayez les logins du fichier pass3 et enregistrez-les dans un fichier « pass4 »

`cut -d : -f 1 pass3 > pass4`

Extrayez les UUIDs du fichier pass3 (**champs n°2 dans le fichier pass3**) et enregistrez-les dans un fichier « pass5 »

`cut -d : -f 2 pass3 > pass5`

Fusionnez les fichiers « pass4 » et « pass5 » dans le fichier « pass6 » de sorte à le rendre similaire à « pass3 » (avec le même séparateur)

`paste -d : pass4 pass5 > pass6` , il faut définir à nouveau le séparateur avec -d

- 8) Affichez les lignes communes aux deux fichiers pass2 et pass6

`comm pass2 pass6`

la commande sépare le contenu du fichier en trois colonnes (lignes similaire, lignes propres au premier fichier, lignes propres au second fichier)

Comparez ces deux fichiers (**cmp**) puis affichez leurs différences (**diff**), faites également des essais avec d'autres fichier

`cmp pass2 pass6` , donne l'emplacement de la première différence entre les deux fichiers

`diff pass2 pass6` , vous indique de quel manière rendre les deux fichiers similaires (« d » pour delete, « a » pour add , « c » pour change)