[Commandes Linux] Grep / Awk / Cut / Sed

Introduction

Cette page montre l'utilisation d'outils en ligne de commande qui permettent de travailler avec les chaînes de caractères.

Grep

Permet de trouver un motif dans un fichier.

Syntaxe globale

```
grep [options] motif [fichier...]
```

Voici les options les plus courantes :

```
-i : Ignorer la casse (recherche insensible à la casse).
-r : Rechercher récursivement dans les répertoires.
-l : Afficher uniquement les noms de fichiers contenant des correspondances.
-v : Inverser la recherche (afficher les lignes qui ne contiennent pas le motif).
-n : Afficher le numéro de ligne de chaque correspondance.
-c : Afficher le nombre total de correspondances trouvées.
-A num : Afficher num lignes après chaque correspondance.
-B num : Afficher num lignes avant chaque correspondance.
-C num : Afficher num lignes avant et après chaque correspondance.
-e motif : Rechercher plusieurs motifs.
-f fichier : Lire les motifs à partir d'un fichier.
--exclude=pattern : Exclure les fichiers correspondant au modèle donné.
--include=pattern : Inclure uniquement les fichiers correspondant au modèle donné.
--color=auto : Surligner les correspondances en couleur (par défaut).
```

```
--color=never : Désactiver la coloration.
```

Exemples :

- Rechercher un motif dans un fichier :

```
grep "motif" fichier.txt
```

- Rechercher un motif dans plusieurs fichiers récursivement :

```
grep -r "motif" répertoire/
```

- Rechercher un motif en ignorant la casse :

```
grep -i "motif" fichier.txt
```

- Afficher uniquement les noms de fichiers contenant des correspondances :

```
grep -l "motif" fichier1.txt fichier2.txt
```

- Inverser la recherche pour afficher les lignes sans correspondances :

```
grep -v "motif" fichier.txt
```

- Afficher le numéro de ligne de chaque correspondance :

```
grep -n "motif" fichier.txt
```

- Rechercher plusieurs motifs :

```
grep -e "motif1" -e "motif2" fichier.txt
```

- Utiliser un fichier contenant des motifs à rechercher :

```
grep -f motifs.txt fichier.txt
```

- Exclure certains fichiers de la recherche :

```
grep "motif" --exclude=*.log répertoire/
```

- Activer/désactiver la coloration des correspondances :

```
grep --color=always "motif" fichier.txt
grep --color=never "motif" fichier.txt
```

Awk

Il s'agit d'un outil de traitement de texte principalement utilisé pour manipuler et analyser des données tabulaires.

Syntaxe globale

```
awk 'pattern { action }' fichier
```

Il existe deux principales utilisation **Awk** :

- **Pattern** : Condition pour appliquer une action.
- **Action** : Commandes à exécuter si le pattern est vrai.

Exemples

- Afficher une colonne spécifique d'un fichier CSV :

  ```
  awk -F ',' '{print $2}' fichier.csv
  ```

- Calculer la somme d'une colonne :

  ```
  awk '{sum += $1} END {print sum}' fichier.txt
  ```

- Filtrer les lignes qui correspondent à un motif :

  ```
  awk '/motif/ {print}' fichier.txt
  ```

- Afficher les lignes avec plus de 3 champs :

  ```
  awk 'NF > 3 {print}' fichier.txt
  ```

- Utiliser des opérations mathématiques :

  ```
  awk '{total += $1 * $2} END {print total}' fichier.txt
  ```

- Remplacer du texte dans une colonne :

  ```
  awk '{gsub("ancien", "nouveau", $3); print}' fichier.txt
  ```

- Regrouper et compter les occurrences :

```
awk '{count[$1]++} END {for (item in count) print item, count[item]}' fichier.txt
```

- Utiliser des variables avec awk :

```
awk '{total += $1} END {average = total / NR; print "Moyenne :", average}'
fichier.txt
```

- Personnaliser le séparateur de champ (-F) :

```
awk -F ':' '{print $1}' fichier.txt
```

- Appliquer awk à plusieurs fichiers :

```
awk '{print FILENAME, $0}' fichier1.txt fichier2.txt
```

- Utiliser des conditions multiples :

```
awk '$1 > 50 && $2 == "A" {print}' fichier.txt
```

- Redirection de la sortie vers un fichier :

```
awk '{print $1, $2}' fichier.txt > sortie.txt
```

Cut

Cut st utilisée pour extraire des colonnes ou des champs de texte à partir de fichiers ou de données en entrée. Il est plus rapide et performant que awk et doit être utilisé, dans la mesure du possible, sur des grandes quantités de données.

Syntaxe globale

```
cut [OPTIONS] [FICHIERS]
```

Options courantes

```
-c LISTE : Extraire des caractères (par numéro de colonne).
-f LISTE : Extraire des champs (délimités par un séparateur).
-d CARACTERE : Spécifier le séparateur de champ (par défaut : tabulation).
-s : Ignorer les lignes sans délimiteur, ne pas les afficher.

Liste :
```

```
- Une liste de numéros de colonnes (pour l'option -c) ou de numéros de champs (pour l'option -f), séparés par des virgules.
- Les plages de numéros de colonnes ou de champs sont spécifiées avec un tiret (par exemple, 1-3).
```

Exemples

- Extraire une colonne spécifique d'un fichier CSV (séparé par des virgules) :

```
cut -d ',' -f 2 fichier.csv
```

- Extraire une colonne spécifique d'un fichier TSV (séparé par des tabulations) :

```
cut -f 3 fichier.tsv
```

- Extraire plusieurs colonnes d'un fichier CSV :

```
cut -d ',' -f 1,3,5 fichier.csv
```

- Extraire une plage de caractères d'une ligne de texte :

```
echo "ABCDEFG" | cut -c 2-4
```

- Ignorer les lignes sans délimiteur :

```
cut -d ',' -f 2,3 -s fichier.csv
```

- Extraire une colonne spécifique d'un fichier avec des espaces comme séparateur :

```
cut -d ' ' -f 2 fichier.txt
```

- Extraire plusieurs plages de caractères d'une ligne de texte :

```
echo "12345ABC67890" | cut -c 1-5,7-9
```

- Extraire tous les champs d'un fichier de configuration (séparés par des espaces ou des tabulations) :

```
cut -d ' ' -f - fichier.conf
```

Sed

C'est un outil puissant de traitement de texte en ligne de commande.

Syntaxe globale

```
sed [OPTIONS] 'commande' [FICHIERS]
```

Options courantes

```
-e 'script' : Ajouter un script de commande à exécuter.
-i[SUFFIXE] : Modifier le fichier en place (crée une sauvegarde avec un SUFFIXE si spécifié).
-n : Désactiver la sortie automatique, nécessite l'utilisation de p (impression explicite).
-r : Activer les expressions régulières étendues (compatibles avec les regex modernes).
```

Commandes courantes

```
- s/regex/remplacement/g : Remplacer les occurrences de regex par remplacement (g pour global, sinon seulement la première occurrence).
- p : Imprimer la ligne courante.
- d : Supprimer la ligne courante.
- 1,10s/regex/remplacement/g : Appliquer la commande de remplacement uniquement aux lignes 1 à 10.
- /motif/cmd : Appliquer la commande cmd aux lignes contenant le motif.
- !cmd : Appliquer la commande cmd à toutes les lignes sauf celles spécifiées.
- a\ : Ajouter une ligne après la ligne courante.
- i\ : Insérer une ligne avant la ligne courante.
- c\ : Remplacer la ligne courante par une nouvelle ligne.
```

Exemples

- Remplacer toutes les occurrences d'un mot dans un fichier :

  ```
  sed 's/motif/remplacement/g' fichier.txt
  ```

- Supprimer les lignes vides d'un fichier :

  ```
  sed '/^$/d' fichier.txt
  ```

- Remplacer une ligne spécifique par du texte dans un fichier :

```
sed '7c\Nouveau texte' fichier.txt
```

- Ajouter du texte à la fin de chaque ligne :

```
sed 's/$/ Texte à ajouter/' fichier.txt
```

- Imprimer uniquement les lignes contenant un motif :

```
sed -n '/motif/p' fichier.txt
```

- Supprimer la dernière ligne d'un fichier :

```
sed '$d' fichier.txt
```

- Modifier un fichier en place avec sauvegarde :

```
sed -i.bak 's/motif/remplacement/g' fichier.txt
```

- Utiliser des expressions régulières étendues avec -r :

```
sed -r 's/(motif1|motif2)/remplacement/g' fichier.txt
```

- Ajouter une ligne avant chaque ligne contenant un motif :

```
sed '/motif/i\Nouvelle ligne' fichier.txt
```

- Supprimer toutes les lignes sauf celles contenant un motif :

```
sed -n '/motif/p; /motif2/p' fichier.txt
```

Revision #5
Created 2023-09-13 08:55:38 UTC by Elieroc
Updated 2023-09-13 13:06:12 UTC by Elieroc