sed

- Objet : sed
- Niveau requis :
  débutant, avisé
- Commentaires : **sed** signifie *Stream EDitor*, autrement dit *éditeur de flux* et plus précisément *éditeur de flux orienté ligne*.
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  - Création par smolski le 24-04-2010
  - Testé par smolski le 08-10-2013
- Commentaires sur le forum : Lien vers le forum concernant ce tuto[1)]

Présentation

sed est souvent défini comme un éditeur de texte en ligne non-interactif.

1. Il lit les lignes d'un fichier une à une (ou provenant de l'entrée standard), sans traiter nécessairement tout le texte du fichier en cours (comme le font vim ou nano par exemple) ;
2. leur applique un certain nombre de commandes d'édition ;
3. et renvoie les lignes résultantes sur la sortie standard, sans modification du fichier traité.

sed est une évolution de l'éditeur **ed** lui-même précurseur de **vi**, la syntaxe n'est franchement pas très conviviale, mais il permet de réaliser des commandes complexes sur de gros fichiers.

La commande sed est une commande très riche, ne vous sont présentées ici que les fonctions les plus courantes, pour plus de détails faites un :

```
man sed
```

Syntaxe

La syntaxe de sed est la suivante:

```
sed <option> <commande ou programme sed> <fichier_a_traiter>
```

Tableau des principales options sed

Option	Commentaires
-n	Écrit seulement les lignes spécifiées (associé à l'option /p) sur la sortie standard
-e	Ajoute une commande à la liste des commandes que doit exécuter **sed** sur le fichier.
-f	Les commandes sont lues à partir d'un fichier préalablement rédigé.

Option	Commentaires
-i	Le fichier est édité sur place. Sinon, *le fichier n'est pas édité*, une copie de son contenu est éditée et renvoyée sur la sortie standard.
s/ (*Substitution*)	Commande de substitution (remplace les caractères définis par l'ensemble de la commande sed).
/g (*Global*)	Traite toutes les occurences définie (par défaut : seule la première occurence définie est traitée).
/p (*Print*)	Imprime (affiche) la ligne traitée (utile avec l'option -n)
/w <fichier>	Écrit la ligne dans le <fichier> specifié en plus de la sortie standard.
! (*point d'exclamation :* “!”)	Commande de négation (inversion des caractéristiques de la commande sed)
d (*Delete*)	Efface les lignes (au niveau de la sortie, le fichier d'origine n'est pas modifié)
a\<texte>	Écrit le <texte> après la ligne
i\<texte>	Écrit le <texte> avant la ligne
q (*Quit*)	Quitte la commande sed
= (*signe égal* : =)	Indique le numéro des lignes traitées
num (*Un chiffre*)	Indique le numéro de la ligne à traiter ou bien le numéro d'ordre de l'occurence à traiter
$	Indique la dernière ligne du fichier traité
\t	Indique une tabulation

Préparation aux exemples d'illustration de ce tuto :

Pour réaliser les exemples qui suivront sur votre pc, en session user, créez un fichier test1.txt ainsi :

fichier_test1.txt

```
cat >> ~/test1.txt <<EOF
toto et titi aiment les abricots.
toto préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
EOF
```

Voir : la commande cat

Option -i

ATTENTION ! L'option -i modifiera réellement les fichiers traités.

L'option `-i` est la fonction d'écriture de SED. Si elle n'est pas spécifiée, le fichier traité ne sera pas modifié.

Syntaxe :

```
sed -i <commande_sed> <fichier_à_traiter>
```

Option -e

`-e` permet de spécifier les commandes à appliquer sur le fichier.
Cette option est utile lorsque vous appliquez plusieurs commandes.
Afin d'éviter que le shell interprète certains caractères, il vaut mieux encadrer la commande avec des apostrophes simples ou doubles : `'` ou **“** .

-e renvoie le résultat sur l'entrée standard(écran), le fichier n'est pas modifié

Syntaxe :

```
sed -e <commande_sed> -e <commande_sed> <fichier_à_traiter>
```

Exemple :

La commande :

```
sed -e "s/[Tt][Oo]/ton/g" -e "s/abricots/myrtilles/g" test1.txt
```

Donne :

```
tonton et titi aiment les myrtilles.
tonton préfère les Fraises.
titi les cerises.
tonton et TITI sont des chipoteurs du jardin.
```

Où :

```
"s/[Tt][Oo]/ton/g"
```

est la commande `sed` *substituant* (**`s/`**) les chaînes de caractère :

```
TOTO
TOTo
toto
```

en :

```
tonton
```

sur la *globalité* du fichier, (*c'est à dire toutes les lignes*) (**`/g`**).

L'option `-e` n'est pas nécessaire quand vous n'avez qu'une seule commande à faire exécuter à sed.

Astuce

En cas de confusion avec un chemin, genre `/machin/truc/chouette` après l'option les séparateurs `/` peuvent être remplacés par des caractères neutres, comme la virgule ou le dièse. Par exemple avec le chemin précédent :

```
sed -e 's#/machin/truc/chouette#/machin/truc/wouap#'
```

Évitant ainsi des confusions dans l'interprétation de la commande.

Merci à **enicar** *sur le chan df !*

Dans la même idée d'interprétation confuse, voir sur le forum :

- https://debian-facile.org/viewtopic.php?pid=342480#p342480

Option -f

L'option -f permet d'utiliser un fichier texte contenant les commandes que `sed` devra appliquer.
Ce fichier est donc un script qui sera interprété par le programme `sed` d'où le nom utilisé dans ce tuto : <script_sed>

Syntaxe :

```
sed -f <script_sed> <fichier_à_traiter>
```

Pour que les fichiers traités le soient réellement pensez à utiliser l'option `-i`. Sinon, rien ne sera réalisé réellement.

Exemple :

Créer un fichier `monScriptSed` qui contiendra toutes les commandes que `sed` devra exécuter :

monScriptSed

```
cat >> ~/monScriptSed <<EOF
s/[Tt][Oo]/ton/g
s/[Tt][Ii]/ti/g
s/Fraises/fraises/
EOF
```

Application de la commande `sed` avec l'option `-f` invitant à appliquer les commandes contenues dans le fichier *monScriptSed* :

```
sed -f monScriptSed test1.txt
```

Donne :

```
tonton et titi aiment les abricots.
tonton préfère les fraises.
titi les cerises.
tonton et titi sont des chipoteurs du jardin.
```

Option -n

Vous disposez de l'option "-n" qui supprime la sortie standard par défaut.
sed va écrire uniquement les lignes concernées par le traitement (sinon il écrit tout, même les lignes non traitées).
Utile avec l'option "p " qui permettra elle de visualiser le traitement du fichier.

Syntaxe :

```
sed -n <commande_sed>/p <fichier_à_traiter>
```

Exemple :

```
sed -n s/Fraises/fraises/ test1.txt
```

Bien que traité, rien n'apparait.

Avec la fonction p :

```
sed -n s/Fraises/fraises/p test1.txt
```

Donne :

```
toto préfère les fraises.
```

Seule la ligne traitée est affichée.

La fonction de substitution s

La fonction de substitution "s" permet de changer la première ou toutes les occurences d'une chaîne par une autre.

Syntaxe :

```
sed <option> "s/ <commande_sed>" <fichier_à_traiter>
```

Pour que les fichiers traités le soient réellement pensez à utiliser l'option `-i`. Sinon, rien ne sera réalisé réellement.

Exemples :

```
sed "s/toto/TOTO/" test1.txt
```

va changer la première occurence de la chaîne *toto* par *TOTO* (uniquement la première chaîne *toto* rencontrée dans le texte), et affichera ceci :

```
TOTO et titi aiment les abricots.
toto préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

```
sed "2s/toto/TOTO/" test1.txt
```

va changer la seconde occurence de la chaîne *toto* par **TOTO** (uniquement la seconde chaîne *toto* rencontrée dans le texte)

```
toto et titi aiment les abricots.
TOTO préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

```
sed "s/toto/TOTO/g" test1.txt
```

g = global
g va changer toutes les occurences de la chaîne *toto* par *TOTO* (toutes les chaînes *toto* rencontrées seront changées)

Nous pouvons également choisir de ne changer qu'une occurence en la situant dans le texte.
Dans le fichier exemple ci-dessous :

texte_exemple

```
toto et titi sont sur un bato
```

La commande :

```
sed -e "s/to/teau/3" texte_exemple
```

n'agira que sur la *troisième occurence* “to” et affichera la rectification :

texte_exemple

```
toto et titi sont sur un bateau
```

Cool, non ? 😎

```
sed "s/toto/TOTO/p" test1.txt
```

en cas de remplacement la ligne concernée est affichée sur la sortie standard (uniquement en cas de substitution)

```
sed "s/toto/TOTO/w résultat" test1.txt
```

en cas de substitution la ligne en entrée est inscrite dans un fichier `résultat`

```
sed "s/toto/TOTO/w résultat1.txt" test1.txt
```

Donne :

```
TOTO et titi aiment les abricots.
TOTO préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

Lecture du fichier résultat1.txt créé :

```
less résultat1.txt
```

Donne :

```
TOTO et titi aiment les abricots.
TOTO préfère les Fraises.
```

sed et les regexp

La fonction de substitution peut aussi être utilisée avec une **expression rationnelle**.
Voir : regexp

Pour que les fichiers traités le soient réellement pensez à utiliser l'option `-i`. Sinon,

rien ne sera réalisé réellement.

Exemples :

```
sed "s/[Ff]raises/FRAISES/g" test1.txt
```

substitue toutes les chaînes *Fraises* ou *fraises* par FRAISES

```
sed "s/^/#/" test1.txt
```

Donne :

```
#toto et titi aiment les abricots.
#toto préfère les Fraises.
#titi les cerises.
#TOTo et TITI sont des chipoteurs du jardin.
```

Ainsi, nous avons commenté l'ensemble des lignes du fichier test1.txt.

```
sed "3s/^/#/" test1.txt
```

Donne :

```
toto et titi aiment les abricots.
toto préfère les Fraises.
#titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

Ainsi, nous avons commenté uniquement la 3ème ligne du fichier *test1.txt*.

```
sed "/cerises/s/^/#/" test1.txt
```

Donne :

```
toto et titi aiment les abricots.
toto préfère les Fraises.
#titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

Ainsi, nous avons commenté uniquement la ligne du fichier *test1.txt* contenant le motif : **cerises**.

La fonction de suppression d

La fonction de suppression d supprime les lignes comprises dans un intervalle donné.

Syntaxe :

```
sed <commande_sed>d <fichier_à_traiter>
```

Pour que les fichiers traités le soient réellement pensez à utiliser l'option `-i`. Sinon, rien ne sera réalisé réellement.

Exemple :

```
sed "2,4d" test1.txt
```

Cette commande va supprimer les lignes 2 à 4 du <fichier>.

On peut utiliser les expressions régulières:

```
sed "/toto/d" test1.txt
```

Cette commande supprime les lignes contenant la chaîne `toto`.

Si, à l'inverse, on ne veut pas effacer les lignes contenant la chaîne toto (toutes les autres sont supprimées), on tapera:

```
sed "/toto/!d" test1.txt
```

Les fonctions p, l, et =

La commande p

La commande ”p“ (print) affiche la ligne sélectionnée sur la sortie standard. Elle invalide l'option ”-n“.
*Voir option **n** précédente.*

La commande l (la lettre L en minuscule)

La commande "`l`" (list) affiche la ligne sélectionnée sur la sortie standard avec en plus les caractères de contrôles en clair avec leur code ASCII (deux chiffres en octal).

```
sed l test1.txt
```

Donne :

```
toto et titi aiment les abricots.$
toto et titi aiment les abricots.
toto pr\303\251f\303\250re les Fraises.$
toto préfère les Fraises.
titi les cerises.$
```

La commande =

La commande "=" donne le numéro de la ligne sélectionnée sur la sortie standard.

```
sed = test1.txt
```

Donne :

```
1
toto et titi aiment les abricots.
2
toto préfère les Fraises.
3
titi les cerises.
4
TOTo et TITI sont des chipoteurs du jardin.
```

```
sed "/toto/=" test1.txt
```

Donne : Cette commande va afficher le numéro de chaque ligne contenant la chaîne `toto` ainsi :

```
1
toto et titi aiment les abricots.
2
toto préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

Remarque

Ces trois commandes sont utiles pour le débogage, quand vous mettez au point vos

commandes sed.

Les fonctions q, r et w

La fonction q

La fonction ”q“ (quit) va interrompre l'exécution de **sed**, la ligne en cours de traitement est affichée sur la sortie standard (uniquement si l'option ”-n“ n'a pas été utilisée).

```
sed "/cerises/q" test1.txt
```

Donne :

```
toto et titi aiment les abricots.
toto préfère les Fraises.
titi les cerises.
```

La fonction r

La fonction ”r“ (read) lit le contenu d'un fichier et écrit le contenu sur la sortie standard.

```
sed r test1.txt
```

Donne :

```
toto et titi aiment les abricots.
toto préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

La fonction w <fichier>

La fonction ”w“ (write) écrit la ligne sélectionnée dans un fichier.
Par exemple :

```
sed "/^toto/w resultat2.txt" test1.txt
```

Cette commande va écrire dans le fichier `resultat2.txt` toutes les lignes du fichier *test1.txt* commençant par la chaîne *toto*.

Les fonctions a et i

La fonction i

La fonction "i" (insert) va placer un texte avant la ligne sélectionnée.

Syntaxe :

```
 i\le texte
```

Exemple :

```
sed 1i\ Début test1.txt
```

Donne

```
Début
toto et titi aiment les abricots.
toto préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
```

La fonction a

La fonction "a" (append) va placer un texte après la ligne sélectionnée.

Syntaxe :

```
 a\le texte
```

Exemple :

```
sed 4a\ Fin test1.txt
```

Donne :

```
toto et titi aiment les abricots.
toto préfère les Fraises.
titi les cerises.
TOTo et TITI sont des chipoteurs du jardin.
Fin
```

Vous pouvez appeler la fonction i ou a dans un fichier de commande sed.

Exemple :
Soit un fichier *monScriptSed* rédigé ainsi :

```
cat >> ~/monScriptSed <<EOF
1i\\
début du traitement
s/[Tt][Oo]/ton/g
```

```
s/Fraises/fraises/
\$a  \\
fin du traitement\\
EOF
```

On exécute la commande en tapant :

```
sed -f monScriptSed test1.txt
```

Résultat :

```
début du traitement
tonton et titi aiment les abricots.
tonton préfère les fraises.
titi les cerises.
tonton et TITI sont des chipoteurs du jardin.
fin du traitement
```

les commandes rédigées dans le fichier *monScriptSed* ont pour effet d'inscrire avant la première ligne (1i) le texte :

```
 début de traitement
```

et après la dernière ligne ($a) le texte :

```
 fin du traitement
```

de notre fichier. test1.txt

Pour rendre effective cette commande programmée, écrivez : `-i` ainsi :

```
sed -i -f monScriptSed test1.txt
```

sed et les sous-chaînes

La commande:

```
 sed -e "s/\([0-9][0-9]*\)/aa\1aa/" <fichier>
```

La sous-expression (sous-chaîne) \([0-9][0-9]*\) désigne un ou plusieurs chiffres, chacun sera entouré des caractères aa.
La chaîne :

```
 to2to
```

deviendra :

```
toaa2aato.
```

sed et le point

La commande:

```
sed -e 's/motif: .*/motif: nouveau/' <fichier>
```

Remplacera **motif:** par **motif: nouveau** dans le <fichier>

Merci à **cthulu** *Hop !*

Sed et recherche de motif non-gourmande

- Création par David5647 31/08/2020
- Commentaires sur le forum : https://debian-facile.org/viewtopic.php?pid=341973#p341973

Comportement par défaut

Par défaut, lors de la recherche d'une expression régulière, la correspondance retournée par sed est la plus longue chaîne de caractères possible.

Soit le texte suivant:

```
Une 1ère phrase. Une deuxième phrase. Une troisième phrase. Une quatrième phrase!
```

Extraire la première phrase c'est lister tout les caractères jusqu'au premier point, soit l'expression régulière :

```
.*/.
```

Mais une telle recherche renvoie la plus longue chaîne possible:

```
echo "Une 1ère phrase. Une deuxième phrase. Une troisième phrase. Une quatrième phrase!" | sed -r 's/(.*\.).*/\1/'
```

```
Une 1ère phrase. Une deuxième phrase. Une troisième phrase.
```

Recherche non-gourmande

Pour effectuer une recherche plus restrictive et trouver la plus petite chaîne correspondante, il suffit simplement de rechercher tout les caractères sauf le point. Soit l'expression régulière suivante:

```
[^.]*\.
```

On obtient alors :

```
echo "Une 1ère phrase. Une deuxième phrase. Une troisième phrase. Une
quatrième phrase!" | sed -r 's/([^.]*\.).*/\1/'
```

```
Une 1ère phrase.
```

Aller plus loin

En répétant la motif plusieurs fois, on peut alors extraire toutes les correspondances. Soit l'expression régulière:

```
([^.]*\.){$i}
```

avec i le nombre de répétitions

Soit dans une boucle:

```
for i in 1 2 3; do
   echo "Une 1ère phrase. Une deuxième phrase. Une troisième phrase. Une
quatrième phrase!" | sed -r "s/([^.]*\.){$i}.*/\1/" ;
done
```

```
Une 1ère phrase.
 Une deuxième phrase.
 Une troisième phrase.
```

Voilà, c'est bête, mais j'ai cherché longtemps :P

Lien :

- http://www.funix.org/fr/unix/expr-sed.htm#haut
- http://www.shellunix.com/sed.html

Voir aussi ssed aux fonctions plus étendues :

- https://debian-facile.org/utilisateurs:phlinux:tutos:omegat-merge-de-la-source-avec-la-traduction

Remerciements

Tous mes remerciements à **Malekal_morte** et **captnfab** *pour leurs sciences et patiences et à l'équipe redoutable d'*__adrien__*,* **appzer0** *et* **morphalus** *sur le salon toujours souriant du chan* **#slackware-fr**. 😀

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !