Introduction au Système d'Exploitation Unix/Linux

Utilitaire de fichier Unix : sed

B. Jacob

IC2

26 septembre 2017

Plan

Plan

Qu'est ce que c'est ?

`sed` est un programme qui

- applique des transformations sur des fichiers textes
- possède leurs propres commandes d'édition
  $\rightarrow$ est plus complexe à utiliser que les commandes déjà vues
- est un filtre
- est disponible sur pratiquement tous les SE disposant de commandes en ligne.

Résumé

Pour manipuler des fichiers textes on peut le faire par

l'utilitaires `sed`

→ outil puissant mais complexe à utiliser

les fonctions *Search and Replace* d'un éditeur de texte

→ ce sont des fonctions faciles à utiliser mais fastidieuses si nombre d'opérations ↗

Dans quel cas l'utiliser ?

Analogie au découpage de planches :

- pour une planche : utilisation d'une scie à main
  → mode d'emploi simple
  → temps de traitement court
  → on ne passe pas plus de temps à utiliser l'outil que d'apprendre à s'en servir
  (⇝ fonctions *Search and Replace* d'un éditeur de texte)

- pour un lot de planches : utilisation d'une scie circulaire de menuisier
  → mode d'emploi plus complexe
  → temps de traitement long mais automatique
  → gain de temps par rapport au découpage du lot de planches avec une scie à main (⇝ utilitaire `sed`)

Plan

Présentation

- sed = *StreamEDitor*
- format :

`sed [-n] [-e` *`script`*`] [`*`fichier`*`]`

ou

`sed [-n] [-f` *`fichier_script`*`] [`*`fichier`*`]`

- c'est un filtre, donc :
  - Prend ses données (des lignes)
    - dans l'entrée standard (clavier, par défaut) ou
    - dans un fichier (si `[`*`fichier`*`]` positionné)
  - Affiche ses résultats sur
    - la sortie standard (écran, par défaut) ou
    - ne les affiche pas (si option `[-n]` positionnée)

Présentation

`sed [-n] [-e script] [fichier]`

ou

`sed [-n] [-f fichier_script] [fichier]`

`sed` modifie les lignes à partir d'un `script`

- `script` = commandes d'édition
  - en ligne si `-e script` (1 commande par -e)
  - contenues dans un fichier si `-f fichier_script`

Commandes d'édition

`[addresse [,adresse]] fonction [arguments]`

- sélectionne les lignes selon les `adresses`
- leur applique une `fonction` de sed avec ses `arguments`

Adresses de sed

- vide $\rightarrow$ toutes les lignes sont sélectionnées
- *n* $\rightarrow$ la ligne de numéro *n* dans chaque fichier
- \$ $\rightarrow$ seulement la dernière ligne de chaque fichier
- *n1*,*n2* $\rightarrow$ n^{o}de lignes entre *n1* et *n2*
- `/expression reguliere/` $\rightarrow$ définit un contexte d'adresse

Contexte d'Adresses

- décrit le contexte dans lequel doivent être les lignes sélectionnées.
- définit par une `/expression reguliere/`
  - `sed` supporte les expressions régulières étendues (voir cours sur les Regexp)
  - + `\n` (NEWLINE)

Exemples adresses sed

Transformation des lignes de tous les fichiers commençant par `toto`

```
sed -e "fonction [arguments]" toto.*
```
→ transformation sur toutes les lignes des fichiers

```
sed -e "12 fonction [arguments]" toto.*
```
→ transformation sur la 12^{e} ligne de chaque fichier

```
sed -e "12,24 fonction [arguments]" toto.*
```
→ transformation sur sur les lignes 12 à 24 de chaque fichier

```
sed -e "$ fonction [arguments]" toto.*
```
→ transformation sur la dernière ligne de chaque fichier

```
sed -e "/ ^[eE].*z$/ fonction [arguments]" toto.*
```
→ transformation sur les lignes commençant par e ou E et se terminant par z sur tous les fichiers

Fonctions de sed

- Il existe beaucoup de fonctions (Commandes d'édition)
  → `man sed`
- 1 fonction = 1 caractère

Parmis les plus utilisées :

- `a` (append) ajoute du texte
- `c` (change) remplace la ligne
- `d` (delete) efface la ligne
- `w` *fichier* (write) écrit la ligne dans *fichier*

Exemple fonction ajout

- Nom : a
- Argument : ligne à insérer

Avec commande en ligne (option -e)

```
$ cat test_sed.txt
aaaaaa
bbbbbb
cccccc

$ sed -e "1 a \\
zzzzzz" test_sed.txt

aaaaaa
zzzzzz
bbbbbb
cccccc
```

Exemple fonction ajout

Avec un fichier script (option -f)

$ cat test_sed.txt	$ cat script.sed
aaaaaa	1 a \
bbbbbb	zzzzzz
cccccc	

```
$ sed -f script.sed test_sed.txt
```

```
aaaaaa
zzzzzz
bbbbbb
cccccc
```

Exemple fonction change

- Nom : c
- Argument : ligne qui remplace

$ cat test_sed.txt aaaaaa bbbbbb cccccc	$ cat script.sed 3 c \ zzzzzz

```
$ sed -f script.sed test_sed.txt
```

```
aaaaaa
bbbbbb
zzzzzz
```

Exemple fonction delete

- Nom : d
- Argument : rien

$ cat test_sed.txt aaaaaa bbbbbb cccccc	$ cat script.sed 2 d

```
$ sed -f script.sed test_sed.txt

aaaaaa
cccccc
```

Exemple fonction write

- Nom : w
- Argument : nom du fichier où l'on écrit

```
$ cat test_sed.txt
aaaaaa
bbbbbb
cccccc
```

```
$ cat script.sed
1 w toto.txt
```

```
$ sed -f script.sed test_sed.txt
```

```
aaaaaa
bbbbbb
cccccc
```

```
$ cat toto.txt
aaaaaa
```

Fonction substitute

C'est **la** plus utilisée

- `Nom` : `s`
- `Argument` : `/reg-exp/remplacement/flags`
- `reg-exp` expression régulière de sed
  - Stockage `\( reg-exp \)`
  - Rappel `\1 \2 \3 ...`
- `flags` :
  - `g` (global) (faire toutes les substitutions de la ligne)
  - `n` avec $n \in [1 - 512]$ remplace seulement la n^{ieme} occurence
  - `p` (print) si ok sort la ligne sur la sortie standard
  - `w` *fichier* (write) si ok écrit la ligne dans *fichier*

Exemples simples

```
$ cat f
aaa bbb aaa bbb aaa
aaa ccc ddd
```

`$ sed -e "s/aaa/AAA/" f` → substitution 1re occurence

```
AAA bbb aaa bbb aaa
AAA ccc ddd
```

`$ sed -e "s/aaa/AAA/g" f` → substitution toutes les occurences

```
AAA bbb AAA bbb AAA
AAA ccc ddd
```

`$ sed -e "s/aaa/AAA/3" f` → substitution 3e occurence

```
aaa bbb aaa bbb AAA
aaa ccc ddd
```

Exemples simples

```
$ cat f
aaa bbb aaa bbb aaa
aaa ccc ddd
```

```
$ sed -e "s/ccc/CCC/w toto.txt" f
```

→ substitution + écriture dans le fichier toto.txt

aaa bbb aaa bbb aaa aaa ZZZ ddd	$ cat toto.txt aaa ZZZ ddd

Exemples simples

```
$ cat f
aaa bbb aaa bbb aaa
aaa ccc ddd
```

```
$ sed -e "s/aaa/AAA/p" f
```

→ affichage contenu de f + affichage ligne substituée (bof)

```
aaa bbb aaa bbb aaa
aaa ZZZ ddd
aaa ZZZ ddd
```

⇒ Intérêt de l'option [-n]

```
$ sed -n -e "s/aaa/AAA/p" f
```

→ seulement affichage ligne substituée

```
aaa ZZZ ddd
```

Exemple avec stockage et rappel des champs

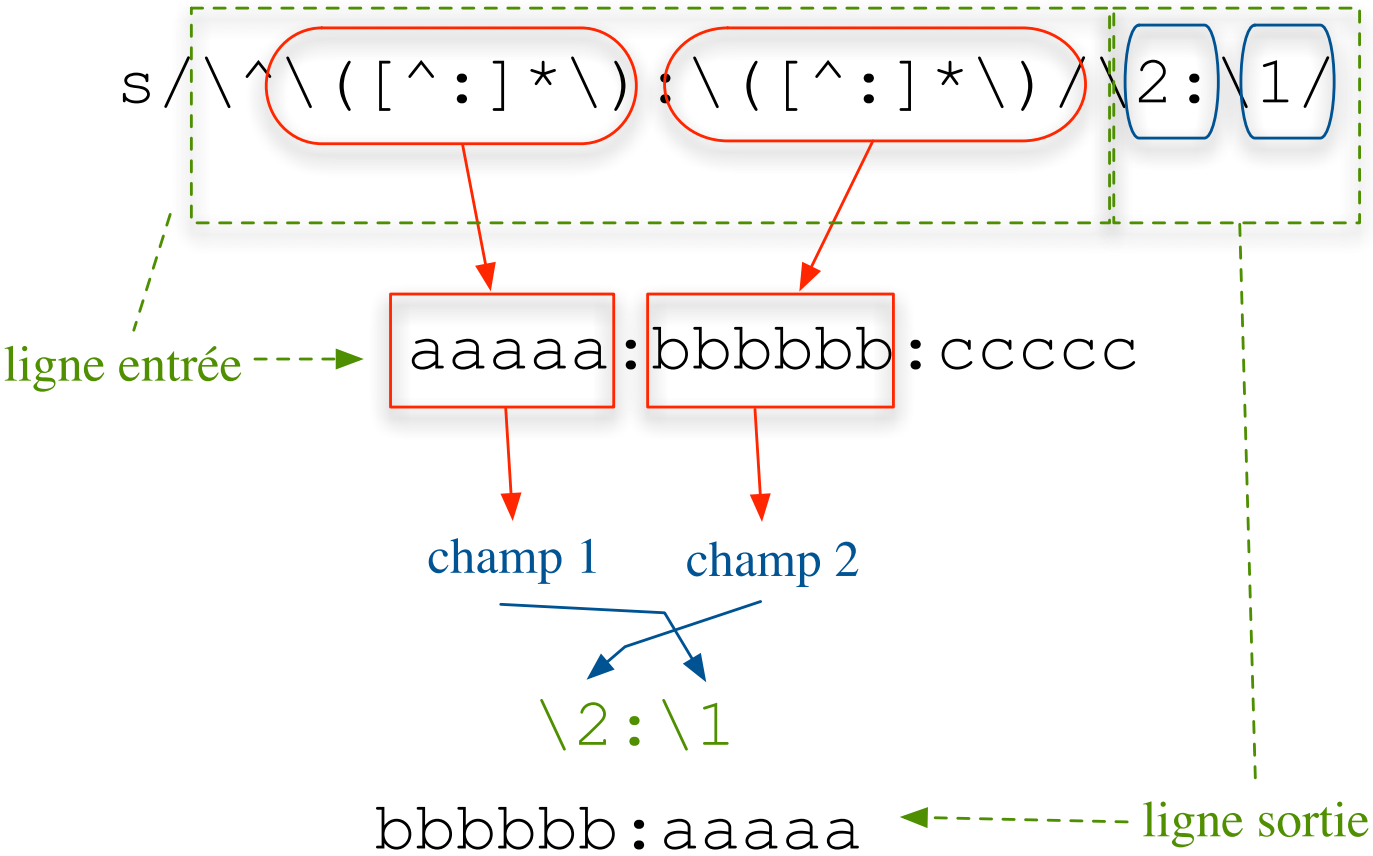
Fonction d'inversion des 2 premiers champs

```
$ cat f
aaaaa:bbbbbb:ccccc
ddddd:ee:ff
ggg:hhhhh:iiii

$sed -e 's/\^\([^:]*\):\([^:]*\)/\2:\1/' f
bbbbbb:aaaaa:ccccc
ee:ddddd:ff
hhhhh:ggg:iiii
```

Exemples avec stockage et du rappel des champs

Synopsis



Exercices

Exercices sur sed