SED : "The Stream EDitor"

source : http://www.commentcamarche.net/faq/9536-sed-introduction-a-sed-part-i

Première partie

Ce document est une introduction à la pratique et à l'utilisation de l'éditeur de flux “SED”.

Il essaie de couvrir certaines fonctionnalités assez méconnues, pour ne pas dire “quasi inconnues”, qui font de “SED” un outil indispensable dans la boîte à outils de tout Linuxien désireux de se rompre aux maniements et aux arcanes du traitement de fichiers via une console et un shell.

Présentation

Sed signifie “Stream EDitor” autrement dit “éditeur de flux”, et plus précisément “éditeur de flux orienté ligne”. De par sa conception et son mode de fonctionnement, Sed est un éditeur non-interactif.

Tout comme l'éditeur “ed” -dont il est issu et que l'on trouve toujours dans les distributions actuelles- et contrairement aux autres éditeurs tels que vi, emacs, Nedit, Xedit, etc., qui eux fonctionnent sur une page complète de texte affiché à l'écran, Sed agit sur une seule ligne à la fois.

À ses débuts l'éditeur “ed” s'est vu doté d'une commande travaillant sur son flux d'entrée standard plutôt que sur un fichier, capable d'afficher toutes les lignes contenant une expression régulière. Cette commande dont la syntaxe s'écrit sous la forme “g/re/p” (global/regular expression/print) donnera naissance à l'utilitaire “grep”. Quelques temps après une nouvelle implémentation d'une version de “ed” vit le jour, travaillant uniquement sur le flux d'entrée standard tout en tirant ses instructions d'un fichier de scripts. Cette version fut baptisée Stream EDitor, plus connue sous le nom de “Sed”.

L'éditeur de flux Sed lit les lignes d'un ou plusieurs fichiers depuis l'entrée standard, enchaine des commandes lues elles aussi depuis l'entrée standard sous forme d'expressions (commandes d'édition) ou depuis un fichier texte (script), et écrit le résultat du traitement sur la sortie standard.

On pourrait résumer le mécanisme de fonctionnement de Sed de cette façon :

- lecture d'une ligne sur le flux d'entrée (une ligne étant délimitée par un caractère de saut de ligne);
- traitement de la ligne en fonctions des diverses commandes lues;
- affichage (ou non) du résultat sur la sortie standard (écran);
- passage à la ligne suivante.

Notons que pour sélectionner la ou les ligne(s) sur la(les)quelle(s) elles doivent opérer, les commandes acceptent des numéros de lignes, des intervalles, ou encore des expressions régulières (notées RE ou regex).

Introduction

Sed prend ses instructions (commandes) depuis la ligne de commandes ou depuis un fichier (script) et applique chaque instruction, dans l'ordre de leur apparition, à chaque ligne en entrée. Une fois que chaque instruction a été appliquée à la 1ère ligne d'entrée, la ligne est affichée (ou non, selon ses besoins) sur la sortie standard (l'écran, ou redirigée dans un fichier) et Sed procède alors à la lecture et au traitement de la ligne suivante et ainsi de suite jusqu'à la fin du fichier d'entrée (à moins qu'il ne rencontre une instruction de sortie explicite). Ce mécanisme est appelé “cycle”. On entend par cycle le traitement des données présentes dans l'espace de travail par l'ensemble des commandes qui composent le script. Par défaut un cycle correspond à :

- L'ajout dans l'espace de travail d'une ligne en entrée (une ligne étant délimitée par le caractère fin de ligne (\n))
- Normalement l'espace de travail doit être vide, à moins qu'une commande “D” ait achevé le cycle précédent (auquel cas un nouveau cycle repartira avec les données restantes dans l'espace de travail).
- Sed appliquera alors les commandes (issues d'un script ou depuis la ligne de commande) concernant les données présentes dans l'espace de travail, séquentiellement, et une fois arrivé en fin de script, copiera les données traitées vers la sortie standard, sauf indication contraire avec l'option “-n” et effacera l'espace de travail. Toutes les données envoyées vers la sortie standard ou un fichier, sont suivies par un caractère de fin de ligne (\n).
- Chargement d'une nouvelle ligne ou sortie si la fin du fichier est atteinte.

Essayons d'illustrer à l'aide d'un organigramme le fonctionnement de Sed à travers un script tout simple qui efface les lignes vides d'un fichier et celles ne comportant qu'un seul caractère “dièse” (#) en début de ligne. Pour ce faire voici un exemple de fichier comportant pour la circonstance quelques lignes vides, quelques dièses seuls dont un en retrait et non pas en début de ligne et deux lignes avec plusieurs dièses à la suite :.

Le fichier :

```
#
#
 #
##
# Ceci est un commentaire

# En voici un autre
#

#
#
###
# Et un autre


#

#
# Et un dernier pour la route
```

```
#
```

Le script en lui même est relativement simple. Le voici sur une seule ligne :

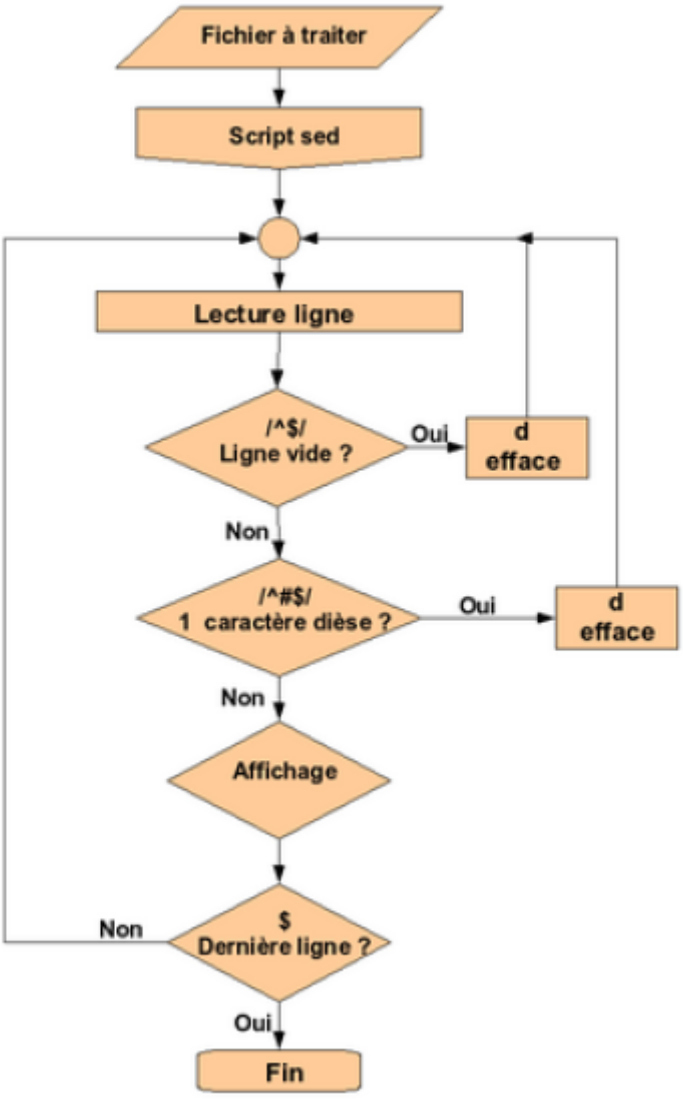
```
sed -e '/^$/d;/^#$/d'
```

et dans un fichier script :

```
#! /bin/sed -f

/^$/d           # on efface les lignes vides
/^#$/d  # on efface les lignes ne comportant qu'un seul caractère "dièse"
                    #+ se trouvant en début de ligne et rien d'autre
derrière
```

Et l'organigramme :



.

Syntaxe

Syntaxe générale

Syntaxe d'une commande

Adressage

Les options (paramètres)

Les commandes

Les commandes basiques 1

flags

Les commandes basiques 2

Les commandes avancées

Les commandes multi-lignes

Les mémoires tampons

Étiquettes

Branchement inconditionnel

Branchement conditionnel

Deuxième partie

Les délimiteurs

Délimiteur de commande

Délimiteur de motif

Le métacaractère &

Les sous-expressions et références arrières

Les sous-expressions

Les références arrières

Expression régulière précédente

La négation

Le regroupement de commandes

Le remplacement de variables

Les expressions régulières

Les caractères d'échappement

Les extras

Les classes de caractères

Les différentes versions

Unix

Windows

Debuggers

Quand ne dois-je pas utiliser Sed ?

Limites connues des différentes versions

Les références

Livres

Les liens

Débutants et initiés

Gurus

IRC

Troisième partie

Les exemples

Substitutions

nième occurence

Substituer les fins de lignes par un espace

Afficher un intervalle entre 2 motifs sans les motifs

Gourmandise des expressions régulières

La commande "n"

Inverser 2 lignes

Effacement d'une ligne et insertion plus loin

Dissocier les commentaires des commandes

Affichage conditionné

Émulation de grep

Exemple 1

Exemple 2 - Commande "x"

Exemple 3 - Commandes "h" "H" "x"

Étiquettes, boucles et mémoires tampons

Supprimer deux lignes précédents un motif donné

Effacer les n dernières lignes

Émulation de "tac" (inverser les lignes d'un fichier)

Exemple de branchement inconditionnel

Exemple de branchement conditionnel (t)

Autre exemple de branchement conditionnel

Exemple de branchement conditionnel (T)

Substitution avec tampons

Décommenter les directives d'un fichier

Conversion de caractères

Mise en forme de texte 1

Avec boucle conditionnée

Avec mémoire tampon

Mise en forme de texte 2

Méthode avec boucle

= Méthode avec mémoires tampons =

La commande "c"

Les fichiers de références pour les exemples

fich.txt

fich2.txt

fich3.txt

adresses.txt

signature.txt

prog.sed

Discussions en rapport sur le forum