Sed et les expressions rationnelles

- Objet : apprendre à utiliser les E.R avec sed
- Commentaire : détail de la documentation de sed relative au E.R.
- Niveau requis :
  avisé

Contexte d'utilisation des ER

On utilise les expressions rationnelles simples ou étendues (*ERE*) avec sed

- pour les adressages par l'encadrement, simple et double.
- avec la commande s :
  - `s/motif/substitut/[options_de_s]` où “motif” peut être une *ER* ou une *ERE*
  - qui utilise aussi des ER pour ses adressages.

Dans ce qui suit, seule la commande s servira pour illustrer d'exemples l'utilisation des *ER* et *ERE*.
Pour l'utilisation globale de sed voir sed par la méthode structuraliste.

Les caractères utilisés pour créer des regexp avec sed

info sed

```
`-r'
`--regexp-extended'
  Use extended regular expressions rather than basic regular
  expressions.  Extended regexps are those that `egrep' accepts;
  they can be clearer because they usually have less backslashes,
  but are a GNU extension and hence scripts that use them are not
  portable.  *Note Extended regular expressions: Extended regexps.
```

Sed utilise une version particulière des expressions régulières.

- **Sans l'option `-r` :**

→ **utilisation des expressions régulières de basiques**,
c'est-à-dire, celles qui utilisent les métacaractères simples (POSIX) qui sont en communs à tous les programmes utilisant les expressions régulières.
En voici un rappel : **.** ; *****, **^**, **$**, **[ ]**,**\<** , **\>**

Attention au caractère * :

1. comme métacaractère :
   * → zéro ou plusieurs occurrence(s) de n'importe quel caractère
2. expression régulière :
   * → zéro ou plusieurs fois l'élément précédent (b*BABA → BABA, bBABA,

bbBABA, bbbBABA, etc.)

3. expression régulière :
   .* → zéro ou plusieurs occurrence(s) de n'importe quel caractère

→ **utilisation des classes** :

> Attention il y a une petite ambiguïté !
> Les classes peuvent effectivement être utilisées avec sed sans l'option `-r` bien que la documentation de sed renvoie à celle grep -E (ou egrep) où elles font partie des regexp étendues.

`[[:alnum:]]`	Alpha-numerique [a-z A-Z 0-9]
`[[:alpha:]]`	Alphabetic [a-z A-Z]
`[[:blank:]]`	Espaces ou tabulations
`[[:cntrl:]]`	Caractères de contrôle
`[[:digit:]]`	Nombres [0-9]
`[[:graph:]]`	Tous les caractères visibles (à l'exclusion des espaces)
`[[:lower:]]`	Lettres minuscules [a-z]
`[[:print:]]`	Caractères imprimables (tous caractères sauf ceux de contrôle)
`[[:punct:]]`	Les caractères de ponctuation
`[[:space:]]`	Les espaces
`[[:upper:]]`	Les lettres majuscules [A-Z]
`[[:xdigit:]]`	Chiffres hexadécimaux [0-9 a-f A-F]

- **Avec `-r` :**

→ On peut utiliser tous les caractères vus précédemment.
L'ajout de l'option -r ne change pas leur signification.

→ On peut utiliser les caractères POSIX étendus :

`|` ; `+` ; `?` ; `( )` ;`{ }` ; `[ - ]`

Auxquels il s'ajoute les caractères non-portables : \< et \>
-\< : cible le début d'un mot[1] : \< doit figurer devant la sous-chaîne désirée
-\> : cible la fin d'un mot : \> doit figurer en fin de la sous-chaîne désirée

> Voici la liste complète et testée des caractères nécessitant l'option `-r` de sed :
>
> **`|` ; `+` ; `?` ; `( )` ;`{ }` ; `[ - ]` ; `\<` ; `\>`**

→ Il s'y ajoute quelqu'uns des raccourcis ci-dessous.
Il s'agit là de quelques unes des extensions GNU[2] qui sont pas portables, mais qui fonctionnent avec sed `-r` :

`\f` : Produit ou correspond à un saut
`\n` : Produit ou correspond à un retour à la ligne
`\t` : Produit ou correspond à un onglet horizontal
`\v` : Produit ou correspond à une tabulation verticale
`\w` : Synonyme de `[[:alnum:]]` → correspond à un mot.

\W : Synonyme de `[^[:alnum]]` → ce qui autre qu'un mot.
\b : Correspond à une chaîne vide (blanc) à l'extrémité d'un mot[3)]

Référence : http://sunsite.ualberta.ca/Documentation/Gnu/sed-3.02/html_chapter/sed_3.html

Pour utiliser les REGEXP, il faut avant tout maîtriser les syntaxes de substitution, et l'adressage.

Sed et les regexp simples

Explications

le caractère ^

Le caractère ^ n'est spécial qu'en début d'E.R. ou immédiatement à gauche dans une chaîne encadrée par des '[]' (voir : “les crochets”).

- Exemple 1 : supprimer la ligne comprenant le caractère “#”, du fichier :

```
cat >> ~/sed.txt <<EOF
> abc
> #ABC
> EOF
```

```
sed '/^#/ d' sed.txt
```

```
abc
```

- Exemple 2 : supprimer le caractère # seulement du fichier sed.txt

```
sed -e '/^#/s/#ABC/ABC/' sed.txt
</code
<code>
abc
ABC
```

- Exemple 3 : ne pas supprimer le ligne qui commence par #

```
sed '/^#/ !d' ~/sed.txt
```

```
#
```

Le caractère $

Le caractère $ représente la fin de la ligne. *(Le caractères $ n'est spécial qu'en fin d'E.R ou d'un ensemble d'E.R.)*

- Exemple : supprimer la ligne finissant par “C”

```
sed -e 's/.*C$//' -e '/^$/ d' sed.txt
```

```
abc
```

L'étoile (*)

Une E.R. d'un seul caractère suivie d'un * est une E.R. qui recherche zéro occurrence ou plusieurs de ce caractère.
`c*123` → 123 ou `c123` ou `cc123` ou `ccc123`, etc. L'expression régulière précédente, doit être un caractère ordinaire, un caractère spécial précédé par `\`, `a.`, une expression rationnelle groupés[4], ou une expression entre crochets.

- S'il y a plusieurs choix, alors la chaîne de gauche la plus longue est choisie.
- La E.R `.*` représente zéro ou plusieurs occurrences de n'importe quel caractère.

Les crochets

La chaîne non-vide encadrée par `[ ]` est considérée comme une `E.R.` d'un seul caractère pouvant avoir n'importe quelle valeur définie entre ces `[]`.

```
echo "1 B 2 b à bB" | sed 's/[Bb]/gaga/g'
```

```
1 gaga 2 gaga à gagagaga
```

À l'inverse, la E.R représentée par `[^…]` est une chaîne composée de tous caractères à l'exclusion de ceux encadrés.
(le caractère ^ ne spécifie cette fonctionnalité d'exclusion que s'il se trouve en première position, immédiatement après le [ )

> À l'exclusion de tous caractères ! Y compris l'espace :

```
echo "1 B 2 b à bB" | sed 's/[^Bb]/gaga/g'
```

```
gagagagaBgagagagagagabgagagagagagabB
```

Une combinaison de plusieurs E.R. d'un seul caractère est une E.R. qui correspond à l'ensemble de ce que chaque E.R. désigne.
Par exemple `a[bB]` recherche ab ou aB.

Les caractères `.` , `*` , `[` et `\` sont toujours spéciaux sauf s'ils sont encadrés entre `[]`.

- `[A-Za-z-]` → Tout caractère alpha ou le `-`.
- `[]a-z]` → Tout caractère alpha ou le `]`.

le point (.)

Un point `.` est une E.R. d'un seul caractère qui correspond à n'importe quel caractère sauf le saut de ligne.

- Exemple 1 :

```
echo "a B 2 ! * & : ?" | sed 's/./Gaga/g'
```

```
GagaGagaGagaGagaGagaGagaGagaGagaGagaGagaGagaGagaGagaGagaGagaGaga
```

- Exemple 2, point<espace> :

```
echo "C " | sed 's/. /Coucou/'
```

```
Coucou
```

Sed -r : les caractères étendues

Rappel :

`+ ; ? ; | ; ( ) ; [ ; { }`

le caractère (+)

- substituer une ou plusieurs fois “z” par OK

```
echo "abc azbc azzbc azzzbc azzzzbc" | sed -r 's/az+bc/OK/g'
abc OK OK OK OK
```

Le caractère (?)

- substituer zéro ou une fois “z” par OK

```
echo "abc azbc azzbc azzzbc azzzzbc" | sed -r 's/az?bc/OK/g'
OK OK azzbc azzzbc azzzzbc
```

Le ou (|)

- substituer ab ou cd par OK

```
echo "abef cdef efgh abcd" | sed -r 's/ab|cd/OK/g'
OKef OKef efgh OKOK
```

- substituer ab ou cd par OK et effacer le reste

```
echo "lflfabef cdef efgh ftyabcdopm" | sed -r\
 's/^.*[ab|cd]*/OK/g'
OK
```

Les parenthèses

```
echo "a b c d e f" | sed -r 's/(b|e)/X/g'
```

```
a X c d X f
```

Parenthèses et inversion de champs

Avec la commande s (substitution), on peut de définir plusieurs E.R mises entre parenthèses.
Cela permet de faire une inversion en se servant du séparateur de chaque champ à matcher.

- Exemple :

```
echo "coucou:toto" | sed -r 's/^(.*):(.*)/\2:\1/'
```

```
toto:coucou
```

/^\(.*\)\1$/ : correspond à une ligne contenant au moins deux fois la même sous-chaîne de caractères

Les accolades { et }

Un seul caractère suivie par {m}, {m,}, ou {m,n} est une E.R. qui correspond au caractère précédant { s'il est présent, exactement m fois (m); au moins m fois (m,); entre “m” et “n” fois (m,n).
'm' et 'n' doivent être des entiers positifs ou nuls inférieurs à 256.

- substituer seulement “aaa, aaaa et aaaaa” par OK

```
echo "a aa aaa aaaa aaaaaa aaaaaaaa aaaaaaa" | sed -r\
 's/a{3,5}/OK/g'
a aa OK OK OKa OKOK OKaa
```

- substituer trois “a” et plus par OK

```
echo "a aa aaa aaaa aaaaaa aaaaaaaa" | sed -r 's/a{3,}/OK/g'
```

```
a aa OK OK OK OK
```

Traitement des mots

- le caractère \<

Le caractère \< oblige l'E.R. à correspondre avec le début d'un mot (mot = chaîne de caractères chiffres, lettres ou '_').
\< doit figurer devant la sous-chaîne désirée.

```
echo "info pour tous informatique" | sed -r 's/\<info/OK/g'
```

```
OK pour tous OKrmatique
```

- le caractère \>

Le caractère \> oblige l'E.R. à correspondre à la fin d'un mot.
\> doit figurer en fin de la sous-chaîne désirée.

```
echo "métrique cosmétique" | sed -r 's/métique\>/OK/g'
```

```
métrique cosOK
```

Les sous-chaînes : associer ( ), [ ], { }

- substituer “Hello” par OK

```
echo "coucou Hello Yep" | sed -r 's/[A-Z]{1}[a-z]{4}/OK/g'
```

```
coucou OK Yep
```

- substituer “ABC-123 abc-123” par OK

```
echo "ABC-123 abc-123" | sed -r 's/[A-Z]{3}-[[:digit:]]{3}\
 [[a-z]{3}-[[:digit:]]{3}/OK/'
OK
```

SHELL : \

→ Les longues commandes peuvent être sectionnées avec :

```
\<retour à la ligne>[<espace>]
```

- substituer “ab efab ef” par OK

```
echo "123ab efab ef123" | sed -r 's/(ab[[:blank:]]ef){2}/OK/'
```

```
1230K123
```

- substituer “123ab efab ef123” par trois “OK”

```
echo "123ab efab ef123" | sed -r -e 's/(ab[[:blank:]]ef){2}/OK/g'\
 -e 's/[[:digit:]]{3}/OK/g'
OKOKOK
```

- substituer jusqu'à trois “a” par OK

Attention !
La E.R `{,m}` n'est pas utilisée avec sed.
Les chaînes comportant plus de trois “a” contiennent aussi moins de trois “a” !

```
echo "a aa aaa aaaa aaaaaa aaaaaaaa" | sed -r 's/a{,3}/OK/g'
OK OK OK OKOK OKOK OKOKOK
```

```
echo "a aa aaa aaaa aaaaaa aaaaaaaa" | sed -r\
 's/(a{1}[[:blank:]]|a{2}[[:blank:]])/OK/g'
OKOKaOKaaOKaaaaOKaaaaaaaa
```

```
echo "a aa aaa aaaa aaaaaa aaaaaaaa" | sed -r\
 -e 's/(a{1}[[:blank:]])/OK/' -e 's/(a{2}[[:blank:]])/OK/'
OKOKaaa aaaa aaaaaa aaaaaaaa
```

1)

Attention : Un mot est une chaîne de caractères faite de chiffres, lettres ou de _ et séparée par tout autre caractère qu'un chiffre, une lettre ou le _.

2)

Escapes

3)

`info\b` →“info pour tous” et `\binfo` → “informatique”

4)

voir les parenthèses