Unix Tools and REGEX

Christophe Cérin

christophe.cerin@lipn.univ-paris13.fr

(version au 09/01/2011)

Objectif du polycopié

- Passer en revue des utilitaires « ligne de commande » qui vont permettre d'explorer des fichiers texte (typiquement des fichiers de configuration du système) pour les modifier -- ce sont souvent des outils pour l'administrateur… quand il n'y a pas d'interface graphique !

La commande tail

- Elle permet de lister les *p* dernières lignes d'un fichier Les principales options sont -r *num (num bytes sur la ligne)* OU -n *num* ; -r (display in reverse order -- attention avec le -r et -n)

La commande cut

- Elle permet de sélectionner une portion de texte sur une ligne
- On peut sélectionner du *i*-ème octet au *j*-ème octet ou en fonction d'un caractère dit délimiteur

La commande tr

- Permet de translater une chaine par une autre
- On peut spécifier des motifs au moyen des outils qui suivent
- tr -c "[:alpha:]" "\n" < file1 (remplace tout ce qui n'est pas alpha-numérique par un \n)
- tr "[:lower:]" "[:upper:]" < file1 (remplace toute minuscule par une majuscule)

La commande uniq

- Elle permet de ne garder qu'une fois une ligne d'un fichier qui en contient plusieurs consécutivement
- Utilité : supprimer des doublons (on trie les lignes (commande sort) puis on lance un uniq)

Outils sur la ligne de commande… un peu plus élaborés

- awk, gawk : pattern scanning and processing language
- sed : stream editor
- grep, egrep, fgrep : print lines matching a pattern. Egrep is the same as grep -E. Fgrep is the same as grep -F
- RAPPEL : CE SONT TOUS DES OUTILS DANS LE MONDE UNIX (parfois) portés dans le monde windows.

Outils sur la ligne de commande

- Awk et egrep utilise des DFA ; sed utilise un DFA comme “moteur” de mise en correspondance.
- Support des méta-caractères : \b \n \r \t ...
- Support des classes :
- [...] [^...] . \w \W
- [:*prop*:] : matches any char in the Posix character class
- Grouping, capturing : (PATTERN), |, *, + \{n, \} : match at least one time (sed, egrep).

Sed

- sed effectue une passe sur le fichier d'entrée ;
- Dans une fenêtre shell unix : man sed, apropos sed, info sed pour obtenir de l'aide
- Instruction de contrôle : saut à une étiquette si une subtitution a été réalisée : c'est presque un langage de programmation
- **Invocation** : via la ligne de commande avec des options (-r -s ...) ou VIA un fichier contenant un script (-f <nom_fichier>)

Example de programme Sed

```
#!/bin/sed -f
# increment one number stored in a file

# on supprime tout ce qui n'est pas un
digit
# en début de ligne
s/[^[0-9]*//

# replace all leading 9s by _  (any other
# char except digits, could be used)
:d
s/9\(_*\)$/_\1/
td

# incr last digit only.
# the first line adds a
# most-significant digit
# of 1 if we have to add a
# digit. The `tn' commands not
# necessary, but make the thing
# faster : rôle : si matching
# alors branch to etiq n

# On remplace par l'ancienne
# valeur + 1
s/^\(_*\)$/1\1/; tn
s/8\(_*\)$/9\1/; tn
s/7\(_*\)$/8\1/; tn
s/6\(_*\)$/7\1/; tn
s/5\(_*\)$/6\1/; tn
s/4\(_*\)$/5\1/; tn
s/3\(_*\)$/4\1/; tn
s/2\(_*\)$/3\1/; tn
s/1\(_*\)$/2\1/; tn
s/0\(_*\)$/1\1/; tn

:n
y/_/0/
wtiti
```

```
[christophe@localhost C#]$ more digit.txt
aaa    123456789
[christophe@localhost C#]$ cat digit.txt | sed -f digit.s
123456790
```

Les commandes de Sed

- Forme générale : [address[,address]]function [arguments]
- Les fonctions : man sed (elles sont nomreuses)
- Les adresses : soit un nombre qui fait ref à un num ligne dans le fichier d'entrée, soit un $ (dernière ligne), soit une regex… mais on ne s'en sert rarement et c'est plutôt compliqué !

Quelques commandes de sed

Command	Result
a\	Append text below current line.
c\	Change text in the current line with new text.
d	Delete text.
i\	Insert text above current line.
p	Print text.
r	Read a file.
s	Search and replace text.
w	Write to a file.

Exemple 1 (plus simple)

On affiche les lignes d'un fichier et en double les lignes contenant un motif :

```
> sed  '/erors/p' example
This is the first line of an example text.
It is a text with erors.
It is a text with erors.
Lots of erors.
Lots of erors.
So much erors, all these erors are making me sick.
So much erors, all these erors are making me sick.
This is a line not containing any errors.
This is the last line.
```

Exemple 1 (plus simple) : suite

On affiche les lignes d'un fichier contenant un motif :

```
> sed -n '/erors/p' example
It is a text with erors.
Lots of erors.
So much erors, all these erors are making me sick.
```

Exemple 2

On détruit les lignes contenant un motif :

```
> sed '/erors/d' example
This is the first line of an example text.
This is a line not containing any errors.
This is the last line.
```

Exemple 3

On affiche sauf les lignes 2 à 4

```
> sed '2,4d' example
This is the first line of an example text.
This is a line not containing any errors.
This is the last line.
```

On n'affiche pas de la ligne 2 à la fin du fichier :

```
> sed '2,$d' example
```

On paramétrise ?

```
> sed '2,${var1}d' example
```

NON : c'est pas possible !!!

Exemple 4

On affiche d'une ligne contenant un motif a
une ligne contenant un autre motif

```
> sed -n '/a text/,/This/p' example
It is a text with erors.
Lots of erors.
So much erors, all these erors are making me sick.
This is a line not containing any errors.
```

Exemple 5 : substitution

On remplace uniquement la première occurence

```
> sed 's/erors/errors/' example
This is the first line of an example text.
It is a text with errors.
Lots of errors.
So much errors, all these erors are making me sick.
This is a line not containing any errors.
This is the last line.
```

Pour substituer de manière globale : g

```
> sed 's/erors/errors/g' example
```

Ou alors faire une boucle avec une etiquette

Comme dans le tout premier exemple !

Pour se familiariser…

- Scripts avec des commandes sed dans les répertoires /etc/rc.d /etc/init.d selon votre distribution
- Consulter les… et décoder les.

```
Sur mon portage Powerbook G4
> grep sed /etc/rc.boot
netboot=$(/usr/sbin/sysctl kern.netboot | /usr/bin/sed -e 's/^[^0-9]*//')
```

Pour décoder, il peut vous manquer de l'information sur les expressions régulières.

Egrep Tool

- Recherche d'un motif dans un fichier texte, binaire
- Options de la ligne de commande : -i (Ignore case distinctions in both the PATTERN and the input files) ; -m NUM (Stop reading a file after NUM matching lines), -r (Read all files under each directory, recursively) ;

Egrep : examples

```
$ egrep '.a.' digit.txt
  aaa    123456789
$ egrep 'aa*' digit.txt
  aaa    123456789
$ egrep 'aaaa*' digit.txt
  aaa    123456789
$ egrep 'aaaaa*' digit.txt
$ egrep '[[:space:]]' digit.txt
  aaa    123456789
$ egrep '^[[:space:]]' digit.txt
  aaa    123456789
$ egrep '^[^[:space:]]' digit.txt
$ egrep '^[^[:digit:]]' digit.txt
  aaa    123456789
$ egrep '9$' digit.txt
  aaa    123456789
```

La sémantique des symboles dépend du contexte (voir l'interprétation de * et celle de ^)

```
egrep '[[:blank:]]*[[a]]*[[:blank:]]*[[:digit:]]*' digit.txt
  aaa    123456789
```

Sur le web

- http://www.die.net/doc/linux/abs-guide/index.html

gawk

- GNU Project's implementation of the AWK programming language – command line instruction;
- Utilisation : soit en passant des instructions soit en demandant à interpréter un fichier 'script'
- Un script est une séries de `/pattern/{action}`
- Le code `action` est appliqué à chaque ligne qui matche le motif `pattern`
- Faire un `man awk` pour de l'aide en ligne

Fonctions disponibles

- match(text,pattern) : if pattern mathes in text, return the position in text where the match starts. Positionnement de RSTART et RLENGTH
- gsub(pattern,replacement,text) : substitutes each mathch of pattern in text with replacement
- sub(pattern,replacement,text) : substitutes first match only

Example script awk

http://www.novia.net/~phridge/programming/awk/

- On lance la commande w et si l'utilisateur (qui se trouve listé dans la colonne 1) est christoph, alors on affiche les colones 1, 2 et 3, ce qui donne :

```
w | awk '$1 == "christop" { print $1, $2, $3 }'
christop vc/1 08:50
christop pty/s0 08:50
christop pty/s1 08:50
christop pty/s2 08:59
christop pty/s3 09:20
```

```
[christophe@localhost pact]$ w
 11:10:41 up  2:21,  5 users,  load average: 0,00, 0,00, 0,00
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
christop vc/1      08:50    2:20m  0.10s  0.00s /bin/usr/X11R6/bin/startx
christop pty/s0    08:50   12444days  0.00s  0.16s kdeinit: kwrited
christop pty/s1    08:50   48.00s  0.53s  0.00s /bin/bash
christop pty/s2    08:59   41:50   2:32   0.01s man
christop pty/s3    09:20    0.00s  0.03s  0.00s w
```

Exemple de script awk

- Ecriture de fonctions : on remplace les négatifs du fic foo.txt par leur val. absolue

```
[christophe@localhost C#]$ cat foo.txt | awk '
> {
>     for (i = 1; i <= NF; i = i + 1)
>     {
>         if ($i < 0)
>         {
>             $i = -$i
>         }
>     }
>     print
> }
> '
1 2 3 4 5 6 7
8 9 7 4 5 7 8
```

[christophe@localhost C#]$ cat foo.txt
1 2 3 -4 5 6 -7
8 9 7 4 -5 -7 8

Exemple de script awk

- On lance la commande w et on affiche les lignes ou le champs PCPU est compris entre 0.00 et 0.09
- Expliquer en quoi la solution suivante n'est pas satisfaisante

```
{
    if (/[0-9][0-9]s[[:blank:]]+0\.0[0-9]s/)
    {
        print
    }
}

[christophe@localhost pact]$ w | awk -f foo.awk
```

Exemple de programme awk

```
[christophe@localhost pact]$ w | awk -f foo.awk
christop vc/1      08:50    3:27m  0.10s  0.00s /bin/sh
christop pty/s1    08:50   50:15   0.53s  0.00s /bin/bash
christop pty/s3    09:20    0.00s  0.62s  0.01s w

[christophe@localhost pact]$ w
 12:17:56 up  3:28,  5 users,  load average: 0,00, 0,00, 0,00
USER     TTY         LOGIN@   IDLE   JCPU   PCPU WHAT
christop vc/1      08:50    3:27m  0.10s  0.00s /bin/sh
christop pty/s0    08:50   12444days  0.00s  0.19s kwrited
christop pty/s1    08:50   50:21   0.53s  0.00s /bin/bash
christop pty/s2    08:59   51:06   3:07   0.06s /bin/bash
christop pty/s3    09:20    0.00s  0.62s  0.01s w
```

Exemple programme AWK

- On affiche les lignes du fichier foo1.txt qui contiennent des réels

```
BEGIN {
        sign = "[+-]?"
        decimal = "[0-9]+[.]?[0-9]*"
        fraction = "[.][0-9]+"
        exponent = "([eE]" sign "[0-9]+)?"
        number = "^" sign "(" decimal "|" fraction ")" exponent "$"
     }
{
    if ($0 ~ number)
    {
        print
    }
}
```

```
[christophe@localhost C#]$ awk -f foo1.awk
  foo1.txt
+1.10
10.123E+12
-123
0.0


[christophe@localhost C#]$ more foo1.txt
+1.10
10.123E+12
-123
titi
0.0
```

Extraire/remplacer

- ***Le fichier foo2.txt contient des citations entre “ et “. On veut les numéroter.***

[christophe@localhost C#]$ gawk -f foo2.awk foo2.txt

"... `and what is the use of a book', thought Alice,
`without pictures or conversations?'."

"... if I'd a knowed what a trouble it was to make a book
I wouldn't a tackled it and ain't agoing to no more." [2]

"...`enseigner c'est transférer de l'information
depuis les notes du professeur aux notes des étudiants
sans jamais passer par le cerveau des étudiants." [3]

Extraire/remplacer

```
/\.\"$/{ count++;
         sub($0, $0 " [" count "]") ;
         print $0 ; next }
        { print $0 }
```

- Il y a un problème avec le caractère ? car la première citation n'est pas numérotée. Expliquer ce qui peut bien se passer.

Exemple (perl et les regex)

On fait un matching de 2 mots (contenus dans \1 et \2) puis on fait un nouveau matching avec \2\1 ou \1
Le mot qui match est donc de la forme xx ou xyyx

```
#!/usr/bin/perl

open (F,"/usr/share/dict/words");
while(<F>){
  if(/^(\w+)(\w+)?(?(2)\2\1|\1)$/) { print($1,
"--",$2, "--",$_); }
}
close(F);
```

Exemple

```
[christophe@localhost C#]$ perl
perl2.perl
beri----beriberi
b--o--boob
boo----booboo
co----coco
d--e--deed
ma----mama
mur----murmur
n--o--noon
pa----papa
p--e--peep
s--e--sees
```

Conclusion

- De nombreux outils «ligne de commande Unix» permettent la gestion des expressions régulières (REGEX) ;
- La présentation des REGEX en contexte est bien entendu fonction du langage (sed, perl...) mais la syntaxe reste la même quelque soit le langage ;
- Dans les langages de programmation, nottament ceux pour le Web, le filtrage d'une information ne se fait plus avec la confrontation d'un motif avec texte mais par l'utilisation de normes XML c.à.d par la description de langages à balises.