Caractères spéciaux et expressions régulières

D. Bogdaniuk – P. Portejoie

Antoine Gicquel **07/10/2016**

Groupe 1A2

Les caractères spéciaux ou génériques (wildcard characters) permettent de désigner un ensemble d'objets et notamment un ensemble de noms de fichiers. Ils s'appliquent donc aux paramètres des commandes désignant des noms de fichiers.

Ils peuvent aussi désigner un ensemble de chaînes de caractères. On parle alors d'expressions régulières qui s'appliquent aux commandes d'édition (ex, vi, sed, ...) ou à des filtres (grep, egrep, awk, ...).

Partie 1 : caractères spéciaux pour la génération des noms de fichiers

- ***** désigne toutes les chaînes de caractères, y compris la chaîne vide
- **?** désigne un caractère quelconque
- **[…]** désigne un caractère quelconque appartenant à la liste. Deux caractères séparés par un tiret (**-**) définissent une liste de caractères rangés par ordre alphabétique, dont le premier élément est le premier caractère et le dernier élément le dernier caractère.
- **[^…]** désigne une liste de caractères à exclure
- **{…,…}** désigne une liste de chaînes de caractères

Il est possible d'annuler l'interprétation d'un caractère spécial en le faisant précéder de l'antislash (****). L'annulation de plusieurs caractères spéciaux se fait en les encadrant de l'apostrophe (**'**) ou de l'apostrophe double (**"**).

Remarque : les caractères spéciaux sont interprétés par le shell **avant** le lancement de la commande.

Exercice 1

Le répertoire courant contient les fichiers suivants (en TP vous devrez les créer) :

abc.s *a?c.svt* *a\?c.svt*
codage *codage.c* *fichier.txt*
3.1ab *texte*

Dans ces conditions indiquez, en cochant les cases correspondantes (X), les fichiers référés par les commandes données ci-après ; si aucun, donnez un exemple de nom qui serait référé par la commande :

	abc.s	*a?c.svt*	*a\?c.svt*	*codage*	*codage.c*	*fichier.txt*	*3.1ab*	*texte*
`ls *`	x	x	x	x	x	x	x	x
`ls *.*`	x	x	x		x	x	x	
`ls *.?`	x				x			
`ls ?.?`								
`ls f*.txt`						x		
`ls c*e`				x				
`ls *c*t`		x	x			x		
`ls a?c*`	x	x						
`ls a??c.*`			x					
`ls a\?c.*`		x						
`ls a\\\?c.*`			x					
`ls [ac]*`	x	x	x	x	x			
`ls [^ac]*`						x	x	x
`ls *[ag]*`	x	x	x	x	x		x	

Exercice 2

Dans le même contexte donnez la commande qui permet d'afficher les fichiers ayant les caractéristiques suivantes :

```
[e1600718@ens-iutva-0389 exos]$ ls -a
.  ..  3.1ab  abc.s  'a?c.svt'  'a\?c.svt'  codage  codage.c  fichier.txt  texte
```

- un nom commençant par `f` et finissant par `t`
  `ls f*t`

```
[e1600718@ens-iutva-0389 exos]$ ls -a
.  ..  3.1ab  abc.s  'a?c.svt'  'a\?c.svt'  codage  codage.c  fichier.txt  texte
```

- un nom commençant par `f` et finissant par `r` ou `t`
  `ls f*[rt]`

```
[e1600718@ens-iutva-0389 exos]$ ls f*[rt]
fichier.txt
```

- un nom commençant par `c` et ne finissant ni par `c`, ni par `e`
  `ls c*[1ce]`

```
[e1600718@ens-iutva-0389 exos]$ ls c*[1ce]
codage  codage.c
```

- un nom de 3 caractères
  ls ???

```
[e1600718@ens-iutva-0389 exos]$ ls ???
ls: impossible d'accéder à '???': No such file or directory
```

- un nom de 3 caractères commençant par a, b, ou c (donnez 2 solutions)
  ls [a-c]??

```
[e1600718@ens-iutva-0389 exos]$ ls [a-c]??
ls: impossible d'accéder à '[a-c]??': No such file or directory
```

- un nom avec un suffixe de 3 caractères
  ls *.???

```
[e1600718@ens-iutva-0389 exos]$ ls *.???
3.1ab  'a?c.svt'  'a\?c.svt'  fichier.txt
```

- un nom commençant par un chiffre
  ls [0-9]*

```
[e1600718@ens-iutva-0389 exos]$ ls [0-9]*
3.1ab
```

- un nom contenant la chaîne abc ou la chaîne def
  ls *[abc,def]*

```
[e1600718@ens-iutva-0389 exos]$ ls *[abc,def]*
3.1ab  abc.s  'a?c.svt'  'a\?c.svt'  codage  codage.c  fichier.txt  texte
```

- un nom ayant un a en première ou deuxième position
  ls {a,2a}*

```
[e1600718@ens-iutva-0389 exos]$ ls {a,2a}*
ls: impossible d'accéder à '2a*': No such file or directory
abc.s  'a?c.svt'  'a\?c.svt'
```

- un nom dont le suffixe commence par un chiffre
  ls *.[0-9]*

```
[e1600718@ens-iutva-0389 exos]$ ls *.[0-9]*
3.1ab
```

Partie 2 : caractères spéciaux associés aux expressions régulières de base

- **.** désigne un (et un seul) caractère quelconque
- ***** remplace zéro fois ou n fois le caractère qui le précède
- **\+** remplace 1 fois ou n fois le caractère qui le précède
- **\?** remplace 0 fois ou 1 fois le caractère qui le précède
- **\b** désigne la chaîne vide (en début ou en fin de ligne par exemple)
- **[...]** désigne un caractère quelconque appartenant à la liste donnée entre crochet
- **^** désigne le début de la ligne
- **$** désigne la fin de la ligne
- **[^...]** désigne une liste de caractères à exclure
- **\{m\}** désigne un nombre exact m d'occurrences d'un caractère
- **\{m,\}** désigne un nombre minimum m d'occurrences d'un caractère
- **\{m,n\}** désigne un nombre d'occurrences d'un caractère compris entre un minimum m et un maximum n.
- **\(…\)** désigne une expression régulière
- **\|** désigne une alternative

Remarquez l'anti-slash (\) devant certains caractères spéciaux, dû au fait que nous utiliserons les expressions régulières dites *de base* pour leur compatibilité ultérieure avec les scripts.

Exercice 3

La commande *grep (global regular expression printer)* lit les données à partir du canal d'entrée ou directement à partir d'un fichier, et affiche les lignes correspondant à l'expression régulière de base passée en paramètre.

Pour chacune des commandes suivantes, donnez un sens à la recherche effectuée (pour les plus complexes vous imaginerez des exemples) :

- `grep Cesar Timbres.txt`
  `- cherche Cesar`
- `grep '[Cc]esar' Timbres.txt`
  `- cherche Cesar ou cesar`
- `grep '[A-Z]' Timbres.txt`
  `- cherche une lettre en majuscule sur une ligne`
- `grep '[^A-Z]' Timbres.txt`
  `- cherche une lettre en minuscule sur une ligne`

- `grep '^[A-[e1600718@ens-iutva-0389 TP4]$ grep '[A-Z]' Timbres.txtZ][A-Z]$' Timbres.txt`

  - cherche une ligne composée uniquement de deux lettres en majuscule

- `grep '^[^A-Z]' Timbres.txt`

  - cherche une ligne qui commence par une lettre minuscule

- `grep '\\' Timbres.txt`

  - cherche des antislashs

- `grep '\(a\{2\}b\)\+' Timbres.txt`

  - cherche une ligne contenant la suite de caractère aab au moins 1 fois

- `grep '\(a\{2,3\}b\|ab\{2\}\)\+' Timbres.txt`

  - cherche une ligne contenant la suite de caractère aaab ou aab au moins une fois ou abb au moins une fois.

Exercice 4

Commencez à réfléchir à l'exercice 4 qui vous sera demandé en TP (cf fin de cet énoncé)

En TP :

Partie 1 : caractères spéciaux pour la génération des noms de fichiers

Créez un répertoire TP4, puis créez-y les fichiers nécessaires aux tests exigés pour les exercices 1 et 2 (cf TD).

Exercice 1

Complétez le tableau en testant chaque commande ; donnez une copie des résultats.

Exercice 2

Dans le même contexte testez chaque commande (pour les commandes ne donnant aucun résultat vous devrez créer un fichier correspondant) ; donnez une copie des résultats.

Partie 2 : caractères spéciaux associés aux Expressions Régulières de base

Vous trouverez dans le forum prof (cf TPs précédents) les fichiers *Timbres.txt* et *essai* nécessaires aux exercices. Copiez-les dans votre environnement et visualisez-les pour avoir une idée de ce qu'ils contiennent.

En ligne de commande, vous pourrez tester vos ER sous leur forme étendue (pas de nécessité de \) avec *egrep* avant de les rédiger et les tester sous forme d'ER de base.

Exercice 3

Testez chaque commande et vérifiez la conformité des résultats ; donnez-en une copie.

```
[e1600718@ens-iutva-0389 TP4]$ grep Cesar Timbres.txt
Cesar apparait sur de nombreux timbres
[e1600718@ens-iutva-0389 TP4]$ grep '[Cc]esar' Timbres.txt
Cesar apparait sur de nombreux timbres
cesar apparait sur de nombreux timbres
[e1600718@ens-iutva-0389 TP4]$ grep '[A-Z]' Timbres.txt
César apparait sur de nombreux timbres
Cesar apparait sur de nombreux timbres
CESAR APPARAIT SUR DE NOMBREUX TIMBRES
\César apparait sur de nombreux timbres
\\César apparait sur de nombreux timbres
A
AA
AB
AAB
ABB
ABAB
[e1600718@ens-iutva-0389 TP4]$ grep '[^A-Z]' Timbres.txt
César apparait sur de nombreux timbres
Cesar apparait sur de nombreux timbres
cesar apparait sur de nombreux timbres
césar apparait sur de nombreux timbres
CESAR APPARAIT SUR DE NOMBREUX TIMBRES
\César apparait sur de nombreux timbres
\\César apparait sur de nombreux timbres
a
aa
ab
aab
abb
abab
```

```
[e1600718@ens-iutva-0389 TP4]$ grep '^[A-Z][A-Z]$' Timbres.txt
AA
AB
[e1600718@ens-iutva-0389 TP4]$ grep '^[^A-Z]' Timbres.txt
cesar apparait sur de nombreux timbres
césar apparait sur de nombreux timbres
\César apparait sur de nombreux timbres
\\César apparait sur de nombreux timbres
a
aa
ab
aab
abb
abab
[e1600718@ens-iutva-0389 TP4]$ grep '\\' Timbres.txt
\César apparait sur de nombreux timbres
\\César apparait sur de nombreux timbres
[e1600718@ens-iutva-0389 TP4]$ grep '\(a\{2\}b\)\+' Timbres.txt
aab
[e1600718@ens-iutva-0389 TP4]$ grep '\(a\{2,3\}b\|ab\{2\}\)\+' Timbres.txt
aab
abb
```

Les commandes s'exécutent comme prévu.

Exercice 4

Donnez les commandes qui permettent d'afficher les lignes du fichier `essai` possédant les caractéristiques demandées ci-après ou répondez aux questions ; vérifiez la conformité des résultats et donnez-en une copie :

- contient le mot "un" en majuscule

```
[e1600718@ens-iutva-0027 TP4]$ grep un essai
un     un    un
un UN Un
UN un
cela se termine par un .
Enfin une phrase bien construite.
Cette fois la phrase se termine par un point.
cette phrase ne commence pas par une majuscule.
Cette phrase bien construite finit par un point.
Cette phrase mal construite finit cependant par un point .
```

- contient le mot "un" en majuscule ou en minuscule
  - que se passe-t-il avec l'écriture suivante : *`grep '[Uu][Nn]' essai`*

```
[e1600718@ens-iutva-0027 TP4]$ grep  '[Uu][Nn]'  essai
un     un    un
un UN Un
UN un
cela se termine par un .
Enfin une phrase bien construite.
Une moins         bien   construite    car trop d'espaces.
Cette fois la phrase se termine par un point.
cette phrase ne commence pas par une majuscule.
Cette phrase bien construite finit par un point.
Cette phrase mal construite finit cependant par un point .
```

  - l'option -i serait-elle appropriée (cf cours ou man) ?

D'après man : -i, --ignore-case = Ignorer les distinctions de casse à la fois dans les MOTIF et dans les fichiers d'entrée. (-i est spécifiée par POSIX.)

Donc l'option serai appropriée car elle passe outre la distinction minuscule / majuscule :

```
[e1600718@ens-iutva-0027 TP4]$ grep -i  un essai
un     un    un
un UN Un
UN un
cela se termine par un .
Enfin une phrase bien construite.
Une moins         bien   construite    car trop d'espaces.
Cette fois la phrase se termine par un point.
cette phrase ne commence pas par une majuscule.
Cette phrase bien construite finit par un point.
Cette phrase mal construite finit cependant par un point .
```

- contient au moins un mot composé de 2 n consécutifs (donnez 3 solutions : une sans quantificateur, deux avec un quantificateur)

```
[e1600718@ens-iutva-0027 TP4]$ grep nn essai
nnon     nni    nnu
nnnonn       nnnu
[e1600718@ens-iutva-0027 TP4]$ grep 'n\{2\}' essai
nnon     nni    nnu
nnnonn       nnnu
```

- ne contient que des caractères en majuscules

```
[e1600718@ens-iutva-0027 TP4]$ grep '^[A-Z]*$' essai
OUIQUEDESMAJUSCULES
OUI
```

- ne contient aucun caractère en majuscule (sans utiliser d'option de grep)

```
[e1600718@ens-iutva-0027 TP4]$ grep '^[^A-Z]*$' essai
un     un    un
nnon     nni    nnu
nnnonn       nnnu
ni * oui ni * non
au moins sept fois la lettrrrre r plus r et rr et rrr
à  (moins  que)
cela se termine par un .
la fin ou les fins
cette phrase ne commence pas par une majuscule.
```

- utilisez l'option -v de grep avec la commande précédente, observez et commentez

```
[e1600718@ens-iutva-0027 TP4]$ grep -v '^[^A-Z]*$' essai
un UN Un
UN un
OUIQUEDESMAJUSCULES
NON PAS QUE DES MAJUSCULES IL Y A DES ESPACES
OUI
La la fin ou les LES fins
Enfin une phrase bien construite.
Une moins         bien   construite    car trop d'espaces.
Cette fois la phrase se termine par un point.
Cette phrase bien construite finit par un point.
Cette phrase mal construite finit cependant par un point .
```

-v, --invert-match : Inverser le sens de la correspondance pour sélectionner les lignes qui ne correspondent pas. (-v est spécifiée par POSIX.)

L'option -v inverse le sens de l'expression régulière donc ici, cela affiche les lignes ne contenant pas uniquement des minuscules c'est à dire les phrase avec un moins une majuscule.

- contient au moins un caractère `*`

```
[e1600718@ens-iutva-0027 TP4]$ grep '*\{1,\}' essai
ni * oui ni * non
```

- contient au moins une parenthèse `(`

```
[e1600718@ens-iutva-0027 TP4]$ grep '(\{1,\}' essai
à  (moins  que)
```

- est terminée par un point

```
[e1600718@ens-iutva-0027 TP4]$ grep '\.$' essai
cela se termine par un .
Enfin une phrase bien construite.
Une moins         bien   construite    car trop d'espaces.
Cette fois la phrase se termine par un point.
cette phrase ne commence pas par une majuscule.
Cette phrase bien construite finit par un point.
Cette phrase mal construite finit cependant par un point .
```

- testez la commande suivante : `grep '\(r\)\{3,\}' essai`. Concluez

```
[e1600718@ens-iutva-0027 TP4]$ grep '\(r\)\{3,\}' essai
au moins sept fois la lettrrrre r plus r et rr et rrr
```

Recherche une expression d'au moins 3 'r' de façon consécutive.

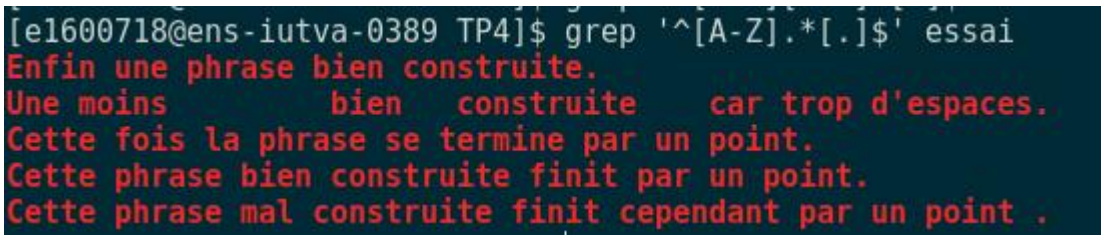
- contient au moins 7 fois (pas forcément de façon consécutive) la lettre `r`

```
[e1600718@ens-iutva-0389 TP4]$ grep '\(.*r.*\)\{7,\}' essai
au moins sept fois la lettrrrre r plus r et rr et rrr
```

- contient au moins 2 fois le mot "`la`" ou "`les`" avec la première lettre en minuscule ou majuscule (donnez 2 solutions)

```
[e1600718@ens-iutva-0389 TP4]$ grep '[Ll]\(a\|es\)' essai
au moins sept fois la lettrrrre r plus r et rr et rrr
cela se termine par un .
la fin ou les fins
La la fin ou les LES fins
Cette fois la phrase se termine par un point.
[e1600718@ens-iutva-0389 TP4]$ grep 'la\|La\|les\|Les' essai
au moins sept fois la lettrrrre r plus r et rr et rrr
cela se termine par un .
la fin ou les fins
La la fin ou les LES fins
Cette fois la phrase se termine par un point.
```

- est une phrase bien formée (on entend par *« bien formée »* une phrase dont le premier mot commence par une majuscule et dont le dernier finit par un point, les autres mots sont séparés par un ou plusieurs espaces)



- question subsidiaire (fonction du temps et de vos capacités) : contient au moins 6 mots (on considérera ici qu'un mot est une suite de caractères autres qu'un espace précédée ou suivie d'un ou plusieurs espaces)