

Les métacaractères, ou globs, ou encore patterns

- Objet : Les métacaractères, ou globs, ou encore patterns
- Niveau requis :
[débutant](#), [avisé](#)
- Commentaires : 
- Débutant, à savoir : [Utiliser GNU/Linux en ligne de commande, tout commence là !](#) 😊
- Suivi :
[à-tester](#)
 - Création par  [MaTTuX_](#) le 17/01/2009
 - Testé par le
- Commentaires sur le forum : [C'est ici](#)¹⁾

Nota : Contributeurs, les  sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Présentation

Les shells de type bourne (comme **Bash**, **Zsh**, **Ksh**, **Dash**...) supportent une syntaxe très puissante : les **globs** encore appelés **patterns**

(*glob: "global command"*)²⁾

Les globs les plus basiques sont :

L'astérisque (*) :

L'astérisque signifie "n'importe quelle chaîne de n'importe quels caractères"

```
echo *
```

affiche le nom de tous les fichiers du répertoire.

Exemples

```
for f in *; do echo "$f"; done
```

Même chose, mais en faisant une boucle. Il sera affiché un fichier par ligne

```
mv /repertoire/*.pdf /autre/part
```

déplacer tout les fichiers pdf de "/repertoire" et les placer dans "/autre/part"

On notera que les fichiers dont le nom commencent par un point ne sont pas affectés par * .
A votre avis, comment peut on faire pour les afficher aussi ? 😊

Le point d'interrogation (?) :

Un point d'interrogation signifie "un caractère quelconque et un seul".

Exemples

Imaginons que nous nous retrouvions avec les fichiers :

```
ls
```

Affiche :

[retour de la commande](#)

```
toto    titi    tata    test    tintin  milou
```

Nous voulons n'afficher que les fichiers dont les noms font quatre lettres de longueur :

```
ls ????
```

Affiche :

[retour de la commande](#)

```
tata    test    titi    toto
```

Nous aimerions maintenant afficher les fichiers toto titi et tata, mais ne pas toucher aux autres.

Nous pouvons aisément faire, par exemple :

```
ls t?t?
```

Affiche :

[retour de la commande](#)

```
tata    titi    toto
```

Simple, non ? 😊

Les crochets ([]) :

Nous savons nous servir de *, de ?, très bien, mais il ne sont parfois pas assez complets pour exprimer des besoins précis. D'où l'introduction des **expressions entre crochet** (POSIX parle de bracket expression). Une expression entre crochets peut avoir plusieurs formes, la plus simple est la liste.

La liste [abcde] peut correspondre à n'importe quel lettre entre crochets : **a** ou **b** ou **c** ou **d** ou **e**.

Donc, [hv]elo peut correspondre à la chaîne de caractères he`l`o tout comme à ve`l`o.

Exemples

Poursuivons avec nos fichiers.

Nous aimerions quelque chose d'encore plus précis qu'avec t?t?.

Pour n'afficher que toto et titi nous allons utiliser les crochets.

La liste des fichiers :

```
ls
```

Affiche :

[retour de la commande](#)

```
tata    titi    toto    test    tintin  milou
```

La commande avec les crochets :

```
ls t[oi]t[oi]
```

Affiche :

[retour de la commande](#)

```
titi    toto
```

Génial ! Mais, voyons ce qu'il se passe lorsqu'un fichier au nom inattendu montre le bout de son nez :

```
touch tito
```

Vous savez qui c'est Tito, n'est ce pas ? 😊

```
ls
```

Affiche :

[retour de la commande](#)

```
tata test titi tito toto tintin milou
```

```
ls t[oi]t[oi]
```

Affiche :

[retour de la commande](#)

```
titi tito toto
```

C'est déjà intéressant, mais il y a mieux :

la possibilité d'utiliser des groupes de lettres dans les expressions entre crochets :

```
ls t[a-z]t[a-z]
```

[retour de la commande](#)

```
tata titi tito toto
```

Remarque

Voilà, vous avez tout en main pour comprendre les **globs** basiques.

Pour plus d'informations, n'oubliez pas le manuel :

```
man nom-du-programme
```

exemple :

```
man mkdir
```

Affiche :

[retour de la commande](#)

```
NAME
    mkdir - make directories
```

SYNOPSIS

```
mkdir [OPTION]... DIRECTORY...
```

DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

Avoir le man en français

Non recommandé :

```
apt-get install manpages-fr
```

Plus tard nous verrons quelques concepts de **globs** plus avancés.

Liens

- [regexp ou Regular Expression ou Expression Régulière](#)

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

2)

https://en.wikipedia.org/wiki/Glob_%28programming%29

From:

<http://debian-facile.org/> - **Documentation - Wiki**

Permanent link:

<http://debian-facile.org/doc/programmation:shell:les-metacaracteres>



Last update: **03/03/2017 15:26**