Shell 1

Juliusz Chroboczek

21 septembre 2015

1 Caractères spéciaux du *shell*

Un *caractère spécial* est un caractère qui est interprété spécialement par le *shell*. Par exemple, l'espace est un caractère spécial, qui sert à séparer les paramètres de ligne de commande.

1.1 Expansion de motifs

Un paramètre de ligne de commande qui contient l'un des caractères « `*` » ou « `?` » est un *motif (pattern)* pour l'ensemble de tous les fichiers qui peuvent être obtenus en remplaçant « `*` » par une suite quelconque de caractères, et « `?` » par un seul caractère. Par exemple, le motif « `tp*` » représente tous les fichiers dont le nom commence par « `tp` », « `*.tex` » représente tous les fichiers dont le nom se termine par « `.tex` », et « `tp*.tex` » représente tous les fichiers dont le nom commence par « `tp` » et se termine par « `.tex` ».

Lorsqu'un tel motif apparaît dans une ligne de commande, il est remplacé par les noms de tous les fichiers présents sur le système de fichiers qui correspondent à ce motif. Ainsi, la commande `echo *.tex` pourra par exemple afficher

```
tp0.tex tp1.tex guide-unix.tex
```

et on pourra copier tous ces fichiers en exécutant une commande de la forme

```
cp *.tex ~/archives/
```

1.2 Citation

Parfois, il est nécessaire d'inhiber le comportement des caractères spéciaux, par exemple pour manipuler un fichier dont le nom contient un espace. Un caractère dont le comportement spécial est inhibé est dit *cité*.

Le *shell* supporte trois caractères de citation. Le caractère « \ » cite le caractère suivant, même s'il est lui-même un « \ ». Le caractère « `'` » cite tous les caractères qui suivent jusqu'au prochain « `'` ». Enfin, le caractère « `"` » cite certains caractères qui suivent jusqu'au prochain « `"` », mais pas d'autres ; il cite notamment « `*` » et « `?` », mais pas « \ » ou « `$` » (voir ci-desssous).

2 Permissions et privilèges

2.1 Utilisateurs et groupes

Chaque utilisateur d'un système Unix a un *nom d'utilisateur*, unique sur le système, et appartient à un *groupe*[1]. Par exemple, l'utilisateur `jch` des machines de l'UFR d'informatique appartient au groupe `ens`. À chaque nom d'utilisateur et groupe sont associés un certain nombre de *privilèges* ou *droits*, comme par exemple celui de modifier un fichier donné ou d'accéder à la *webcam*.

Il existe un compte distingué, appelé *root* ou *superuser*, qui a tous les privilèges possibles. Comme ce compte est très puissant, et permet très facilement de casser le système, on évite normalement de se connecter en étant *root* ; les commandes `su` et `sudo` permettent de devenir *root* de façon temporaire.

2.2 Permissions de fichier

Chaque fichier dans le système de fichiers appartient à un utilisateur, son *propriétaire*, et à un groupe, son *groupe propriétaire*. Il est aussi accompagné d'un ensemble de *permissions*, qui indiquent qui a le droit de le manipuler.

Les permissions d'un fichier consistent de neufs bits. Ce sont les droits
- de lecture,
- d'écriture, et
- d'exécution[2]

pour chacun du propriétaire, du goupe propriétaire et du reste du monde. Pour un répertoire, le droit d'exécution est remplacé par le droit de parcourir le répertoire (d'obtenir la liste des fichiers qu'il contient).

La commande `ls -l` affiche les permissions d'un fichier dans la première colonne :

```
-rw-r--r-- 1 jch jch 4689 Sep 11 19:03 tp2.tex
drwxr-xr-x 2 jch jch 4096 Sep 15 11:15 bioinfo
```

Le premier caractère est le type de fichier, « `-` » pour un fichier ordinaire, « `d` » pour un répertoire. Les trois caractères suivants indiquent les permissions du propriétaire, « `r` » ou « `-` » pour le droit de lecture, « `w` » ou « `-` » pour le droit d'écriture, et « `x` » ou « `-` » pour le droit d'exécution. Les deux groupes de trois caractères suivants indiquent les permissions pour le groupe et le reste du monde respectivement.

Changement de permissions On peut changer les permissions d'un fichier à l'aide de la commande `chmod`, par exemple

```
chmod go-r exam.tex
```

Le premier paramètre de cette commande commence d'un ou plusieurs parmi `u`, `g` et `o`, qui indique si c'est les permissions du propriétaire *(user)*, du groupe propriétaire *(group)* ou du reste

1. Ou plusieurs, sur les Unix modernes.
2. Vous remarquerez la différence avec Windows, où le droit d'exécution est codé dans le nom du fichier — un fichier sous Windows est exécutable si son nom se termine par « `.exe` », « `.com` » ou « `.bat` ».

du monde *(other)* dont il s'agit. Ces caractères sont suivis du caractère `+` ou `-`, indiquant s'il faut ajouter ou supprimer des permissions. Enfin, ce caractère est suivi d'un ou plusieurs parmi `r`, `w` et `x`, qui indiquent quelles sont les permissions à modifier.

La commande `chmod` permet aussi de spécifier exactement les permissions de fichier en spécifiant un entier de 9 bits (3 bits pour chaqu'un parmi `u`, `g` et `o`) noté en base 8 (octale). Par exemple, la commande

```
chmod 755 toto.sh
```

met les permissions du fichier `toto.sh` à `rwxr-xr-x`.

Seul le propriétaire d'un fichier (et l'utilisateur `root`) a le droit de changer les permissions d'un fichier.

Changement de propriétaire La commande `chown` permet de changer le propriétaire et le groupe propriétaire d'un fichier. Par exemple, après l'exécution de la commande suivante, le fichier `toto.c` appartient à l'utilisateur `jch` :

```
chown jch toto.c
```

tandis qu'après la commande suivante, il appartient à l'utilisateur `jch` et au groupe `ens` :

```
chown jch:ens toto.c
```

Seul l'utilisateur `root` a le droit de changer le propriétaire d'un fichier — il n'est pas possible de faire une « donation ». (Pourquoi ?)

3 *Scripts shell*

Jusqu'ici, nous avons utilisé le *shell* de façon interactive — chaque commande est exécutée dès que l'utilisateur la tape. Le *shell* est aussi capable d'exécuter une suite de commandes stockées dans un fichier.

Un *script shell* est un fichier texte exécutable dont la première ligne est « `#!/bin/sh` » et dont les lignes suivantes contiennent une suite de commandes. On exécute un tel *script* en indiquant son nom sur la ligne de commande du *shell*. Par exemple, je me sers du *script* suivant pour faire le ménage dans le répertoire courant :

```
#!/bin/sh
rm *~ a.out core
```

3.1 Paramètres de ligne de commande

```
#!/bin/sh
echo Premier paramètre : $1
echo Deuxième paramètre : $2
```

Tout comme une commande binaire, un *script* peut être invoqué avec des paramètres de ligne de commande. Si le *script* contient les notations `$1`, `$2`, `$3` etc., elles seront remplacées par les valeurs des paramètres.