LES CODES FANTASTIQUES : QUELQUES EXPANSIONS POSIX

Serge GUELTON
[Ingénieur ex-penseur]

Continuons cette série sur les codes fantastiques avec une implémentation POSIX de dirname assez élégante.

Lors d'un `git grep` dans le code source de LLVM, j'ai remarqué ce petit extrait de code qui m'a rappelé un détail du shell POSIX qui m'a toujours amusé. Mais commençons par l'extrait en question :

```
DIR=${0%%`basename $0`}
```

Comme le suggère le nom de variable, ce fragment permet d'extraire la partie répertoire du chemin contenu dans `${0}`. Pour cela, il utilise la commande `basename` qui va faire l'opération inverse — extraire le dernier composant d'un chemin — puis supprimer le résultat de cette commande de `${0}` en vertu du séparateur `%%`. Cette syntaxe est normalisée et respectée par tout shell POSIX, ce qui peut justifier de l'utiliser à la place de la fonction intrinsèque `dirname` fournie, entre autres, dans `bash`.

On va regarder deux catégories de séparateurs qui peuvent influer sur l'expansion d'une variable. La première, évoquée dans notre exemple, permet de supprimer un suffixe ou un préfixe du résultat d'une expansion.

```
${a#*/}
```

s'évalue en `${a}`, puis supprime le plus petit préfixe de `${a}` qui satisfait l'expression `*/`. La version où l'on double le caractère, `${a##*/}` procède de même, mais supprime le plus long préfixe.

```
${a%/*}
```

procède de même, mais en agissant sur le suffixe, `${a%%/*}` permettant donc de supprimer le plus long préfixe.

En combinant les deux syntaxes, on peut s'amuser à écrire une variante de `basename $path .py` sous la forme de `${${path##*/}%.py}` qui n'a cependant guère d'autre intérêt que d'amuser la galerie :-).

Une seconde catégorie de séparateur permet de remplacer le résultat de l'expansion d'une variable par une autre valeur en fonction de la valeur de la variable évaluée. Un exemple :

```
${a:-default}
```

s'évalue en `${a}` si la variable `a` est positionnée et si `${a}` n'est pas vide. Dans le cas contraire, cette expansion de `a` s'évalue en `default`. Cette syntaxe est commode pour donner une valeur par défaut à un paramètre de commande, par exemple :

```
output_file=${1:-/dev/stdin}
```

Il existe plusieurs variantes de cette syntaxe, utilisant les séparateurs `-=`, `:=`,`:+` et `-`, `=`, `+`. Elles diffèrent suivant leur façon de considérer les chaînes vides et les variables non positionnées.

On notera enfin la syntaxe :

```
${a:?error}
```

qui affiche `error` sur la sortie d'erreur si la variable `a` n'est pas positionnée ou si elle s'évalue en une chaîne vide. ■