https://vader-fr.fr/spip.php?article165

Scripts bash Linux de mise en minuscule des extensions.

- Linux & Logiciels Libres -

Date de mise en ligne : dimanche 6 décembre 2015

Sous Linux, les extensions des fichiers n'ont pas d'importance, du moins pour le système.

En revanche, l'utilisateur peut être agacé de voir des fichiers avec des extensions en majuscules, générés le plus souvent par Windows.

Par exemple, les scanneurs auto-alimentés ne fonctionnant pas toujours très bien avec Linux [1], il faudra passer par une machine Windows pour effectuer le scan des documents, ce qui génèrera des fichiers avec extensions en majuscule.

Pour enlever ou remplacer une partie du nom d'un fichier, les mêmes principes seront utilisés.

Ces fichiers seront traités en sortie d'un `find`, qui permettra de débusquer les fautifs.
La commande `find` sera de la forme :

`find` *répertoire* `-type f -regex ".*\.[A-Z]+$"`

- on cherche à partir de *répertoire*
- des fichiers de type fichier simple
- et dont le chemin répond à l'expression régulière :
  - `.*` n'importe quel caractère (`.`) en n'importe quelle quantité (`*`)
  - suivi d'un point, protégé par un `\` afin que le point ne soit pas interprété comme "n'importe quel caractère".
  - suivi d'une lettre entre "A" et "Z", donc en majuscule, et ce répétée au moins une fois (`+`)
  - et ce juste avant la fin de la ligne (`$`), et donc du chemin complet du fichier

Cette expression régulière va donc sélectionner tout ce qui finit par un point, suivi d'une ou plusieurs lettres majuscules.
Les extensions avec mélange de majuscules et minuscules ou chiffres ne sont pas gérées.

Il "suffirait" pour cela de remplacer `[A-Z]+` par `[a-z0-9]*[A-Z]+[a-z0-9]*`, autrement dit :

- ce qui commence éventuellement par une ou plusieurs lettres minuscules ou chiffres,
- contient au moins une lettre majuscule
- et termine éventuellement par une ou plusieurs lettres minuscules ou chiffres.

Afin de transformer ces extensions majuscules en minuscules, il existe plusieurs méthodes, plus ou moins lisibles :

Avec awk et tr

à la fin de la commande `find` citée plus haut, on ajoute le paramètre `-exec` *commande* `{} \;`, permettant d'exécuter la *commande* sur le chemin complet du fichier renvoyé par `find` : `{}`

La commande correspondra à un script de ce genre :

```
#!/bin/bash
nom="$1";
debutnom="";
```

Déclaration de l'interpréteur et initialisation de quelques variables.
`$1` est le premier argument fourni au script, i.e le chemin complet du fichier. Pouvant contenir des espaces ou caractères spéciaux, il est protégé par des `"`

```
nchamps=$(echo "$1" | awk -F \. '{ print NF }');
```

`awk` permet de séparer les champs d'un texte en fonction d'un délimiteur, défini `-F` comme étant le `.`, caractère spécial protégé par un `\`.
On exécute ensuite la commande d'affichage print, de la variable `awk` NF, qui contient le nombre de champs ainsi obtenus.
Le résultat de cette commande `$(...)` est stocké dans ma variable nchamps.

```
i=1;
while [ $i -lt $nchamps ] ;
do
```

tant que la variable i est strictement inférieure (lesser than, `-lt`) au nombre de champs,

```
 champ=$(echo "$1" | awk -F \. '{ print $'$i' }');
 if [ $i == 1 ] ;
 then
 debutnom=$champ;
 else

 debutnom=$debutnom.$champ;
 fi
 i=$(expr $i + 1);
done
```

On récupère le i-ème champ du texte.
Si i est supérieur à 1, alors on remet le délimiteur (.) entre les champs.

cela permet de traiter les fichiers avec de multiples extensions ou contenant des "." dans leur nom, mais ne corrigera que la dernière chaîne.

```
extension=$(echo "$1" | awk -F \. '{ print $'$nchamps' }');
```

On récupère le champ n°"nchamps", donc le dernier.
il est également possible d'écrire : `extension=$(echo "$1" | awk -F \. '{ print $NF }');`, NF étant remplacé par le nombre de champs, le champ n°NF sera donc le dernier.

```
nvext=$(echo "$extension" | tr "[:upper:]" "[:lower:]");
nvnom="$debutnom.$nvext";
mv "$1" "$nvnom";
```

La commande `tr` est utilisée pour transformer chaque caractère majuscule en minuscule. Plutôt que lister les caractères de A à Z (ABCDEF....), on utilise la notation `[:upper:]`.
Cette notation n'est pas utilisable pour toutes les commandes gérant les expressions régulières.

Enfin, la commande `mv` renommera le fichier.

Ce script `awk` est assez long, mais a le mérite d'être assez facilement lisible.

Avec sed

le même script, avec la commande `sed`, sera remplacé par ;

```
#!/bin/bash
nom="$1";
nvnom=$(echo "$nom" | sed -r 's/(.*\.)([[:upper:]]+)$/\1\L\2/');
mv "$1" "$nvnom";
```

Explications de la commande `sed` :

- on utilise les expressions régulières étendues `-r`
- et on effectue `'...'` sur la chaîne envoyée en entrée standard (par le `|`) :
  - une substitution `s/`*chaine à remplacer*`/`*chaîne de remplacement*`/`
  - de la chaîne numéro 1 `(`...)/ Les parenthèses permettent de stocker la partie de chaîne correspondante dans une variable "1"
    - contenant "n'importe quel caractère (`.`) dans n'importe quelle quantité (`*`) suivi d'un point, caracère spécial protégé par un `\`
  - suivi d'une chaîne numéro 2 - mêmes remarques pour les parenthèses
    - contenant des majuscules (`[[:upper:]]`) au moins une fois (`+`)
  - le tout juste avant la fin de la ligne (`$`)
- la chaîne de remplacement est :
  - la chaîne numéro 1 `\1` récupérée depuis la partie gauche de l'expression
  - puis on met en minuscule `\L` la chaîne numéro 2 `\2` récupérée depuis la partie gauche de l'expression

là encore, on ne gère pas les extensions avec mélange de majuscules et minuscules ou chiffres, et on ne corrigera que la dernière chaîne.

C'est déjà plus court et rapide, mais nettement moins lisible.

Avec for et sed

Tout ceci peut être effectué en une seule ligne de commande.
La commande `find` ne remplaçant qu'une seule fois `{}` par le chemin complet du fichier trouvé, et ce chemin étant requis 2 fois, on utilisera une boucle for afin d'externaliser le traitement.

```
for fichier in $(find répertoire -type f -regex ".*\.[A-Z]+$"); do mv "$fichier" "$(echo "$fichier" | sed -r 's/(.*\.)([[:upper:]]+)$/\1\L\2/')" ; done
```

- Pour chaque fichier que renvoie la commande `find` (expliquée plus haut)
- on renomme avec `mv` le fichier par la valeur renvoyée par le `sed` expliqué juste avant.

Et comme expliqué précédemment, on ne gère pas les extensions avec mélange de majuscules et minuscules ou chiffres, et on ne corrigera que la dernière chaîne.

C'est court, rapide, mais très peu lisible.
De plus, cette dernière notation ne gère pas les espaces dans le nom du fichier.

[1] les scanneurs pourvus d'une alimentation externe ne posent pas de problème en revanche