Un exemple d’utilisation des expressions régulières

La problématique

Lorsqu’on fait une sauvegarde sur un disque ou une partition NTFS, ou lorsqu’on synchronise des données dans un parc hybride comprenant des PC avec Windows (donc NTFS), Linux ou Mac, certains caractères dans les noms de dossier ou de fichier peuvent empêcher la sauvegarde ou la synchronisation des données.

Voici les contraintes à respecter pour éviter un blocage :

- interdiction des caractères ci-après
< > * ? " : | (| est le 3ème caractère de la touche 6)
- les dossiers ne doivent pas se terminer par .. ou ...
- pas d’espaces en fin de nom
- pas de point en fin de nom (c’est rare mais ça arrive)

Corriger manuellement les noms de dossier ou fichier peut être fastidieux. Aussi j’ai cherché à élaborer un script qui automatiserait les corrections. Une recherche de solution initiale sur internet renvoie surtout à la commande find, avec des exemples qui se rapprochaient du besoin, mais sans explication, impossible de les adapter.

J’ai commandé chez Eyrolles le manuel suivant https://www.eyrolles.com/Informatique/Livre/les-expressions-regulieres-par-l-exemple-9782914010658/ et là j’ai pu commencer à comprendre quelque chose.

Donc d’abord la commande find

find -name ‘*caractère ou chaîne de caractères à rechercher*’

find renvoie tous les noms qui contiennent le caractère ou la chaîne de caractères à rechercher. La recherche commence à partir du répertoire courant.
* signifie peu importe ce qu’il y a avant ou après.

Après il faut dire ce qu’on l’on fait de ce caractère ou de cette chaîne de caractères, donc pour chaque détection on va passer une nouvelle commande. Ce sera de la façon suivante, avec exec :

find -name ‘*caractère ou chaîne de caractères à rechercher*’ -exec

Cette 2ème commande à passer sera construite à partir d’expressions régulières. A ce sujet, un extrait du manuel, un petit livre, plus petit qu’un livre de poche.
Page 3 :
« Les expressions régulières sont des outils indispensables dès que l’on travaille sur des chaînes de caractères contenant des informations. Au moyen d’une notation compacte et efficace, elles permettent de détecter facilement des schémas dans une chaîne et d’en extraire ce dont on a besoin. »

A la lumière du manuel, voici un exemple commenté, celui du repérage de la présence du caractère ‘?’ et de son remplacement.

find -iname '*?*' -exec rename -n 's/[?]{1}//g' {} \;

find est la commande de recherche
-iname permet de chercher sans tenir compte de la casse (majuscule ou minuscule peu importe)
‘* ?*’ signifie « je cherche un point d’interrogation peu importe ce qu’il y a avant ou après »
-exec signifie qu’on va exécuter ce qui va suivre
rename consiste bien sûr à renommer

-n permet de simuler l'effet de rename sans qu'il y ait exécution. Par contre on est informé de ce que ça donnerait si on le faisait pour de vrai. Si on veut exécuter pour de vrai, on enlève -n
Le bloc 's/[?]//g' comprend 4 critères séparés par des /
s signifie Subtitution
Entre crochets, ce qu'on va remplacer, donc ?
Le 3ème critère est vide car on remplace ? par rien
g signifie Global, c'est à dire sur l'ensemble de la chaîne de caractères. En gros, si on détecte un ? dans un nom, on supprime tous les ? susceptibles d'être présents dans le nom.
La chaîne de caractères à traiter est ce qui est retourné par la commande find, elle est représentée par {}
Enfin, la ligne de commande doit se terminer par \; C'est comme ça.

Les autres caractères indésirables :

find -iname '*:*' -exec rename 's/[.]//g' {} \;
find -iname '*>*' -exec rename 's/[>]//g' {} \;
find -iname '*<*' -exec rename 's/[<]//g' {} \;
find -iname '*|*' -exec rename 's/[|]//g' {} \;
find -iname '*"*' -exec rename 's/["]//g' {} \;
On recommence dans les mêmes conditions avec les caractères : | > < et "

find -iname '***' -exec rename 's/[*]//g' {} \;
Dans '***' la première étoile et la 3ème signifient « peu importe ce qu'il y a, rien ou quelque chose, au début ou à la fin. »
La deuxième étoile représente le caractère d'imprimerie *. Et non une chaîne de caractères quelconque. Pour éviter la confusion, on place avant \ (l'anti-slash) dit « caractère d'échappement ».

Supprimer … ou .. à la fin d'un nom de dossier :
find -type d -iname '*...' -exec rename 's/[.]{3}//g' {} \;
find -type d -iname '*..' -exec rename 's/[.]{2}//g' {} \;
-type d signifie que l'on va ne traiter que les dossiers
'*…' signifient qu'on cherche s'il y a … à la fin du nom du dossier, peu importe ce qu'il y a avant.
'*…' idem
[.]{3} signifie 3 points consécutifs
[.]{2} idem

Remplacement d'un point en fin de nom de dossier ou de fichier :
find -iname '*.' -exec mv {} {}_renamed \;
On ne peut pas faire comme dans les autres cas, car . tout seul signifie le répertoire courant. Le système l'utilise, on ne peut pas le modifier. On est donc obligé de faire autrement. On utilise mv qui veut aussi dire renommer. On renomme les dossiers et fichiers qui se terminent par point, en collant _renamed Comme ça, le problème est réglé. Pour la question de . tout seul, il y aura un message d'erreur :
mv: impossible de déplacer '.' vers '._renamed': Périphérique ou ressource occupé.
Mais pas grave.

find -iname '* ' -exec rename 's/[ ]//g' {} \;
La dernière ligne consiste à supprimer les espaces pour les dossiers et fichiers dont le nom se termine par espace. Il faut mettre cette ligne en dernier, car si lors des détections et remplacement précédents, on peut avoir laisser un dossier ou un fichier se terminant pas un espace.

Le script final :

find -iname '*:*' -exec rename 's/[:]//g' {} \;
find -iname '*?*' -exec rename 's/[?]//g' {} \;
find -iname '*|*' -exec rename 's/[|]//g' {} \;

```
find -iname '*"*' -exec rename 's/["]//g' {} \;
find -iname '*>*' -exec rename  's/[>]//g' {} \;
find -iname '*<*' -exec rename  's/[<]//g' {} \;
find -iname '*\**' -exec rename  's/[*]//g' {} \;
find -type d -iname '*...' -exec rename  's/[.]{3}//g' {} \;
find -type d -iname '*..' -exec rename  's/[.]{2}//g' {} \;
find  -iname '*.' -exec mv {} {}_renamed \;
find  -iname '* ' -exec rename  's/[ ]//g' {} \;
```

que l’on peut lancer de la façon suivante :
sh   le_nom_du_script.sh

Test du script avec les dossiers et fichiers ci-après : https://cloud.oisux.org/s/dX49QMLnZbR6XeP
Les extraire dans un dossier, y placer le script et l’exécuter.
Puis copier les fichiers et dossiers modifiés sur une clé ou un disque ou une partition NTFS.
Comparer avec la copie faite sans avoir exécuté le script.

J’espère qu’il n’y a pas d’erreur ou d’approximation dans ce que j’ai écrit.
Il y a certainement moyen de faire mieux mais ça m’a bien intéressé.