rpl utilitaire de recherche/remplacement de texte

- Objet : Utilisation de rpl pour Rechercher et Remplacer de façon récursive en ligne de commande
- Niveau requis :
  débutant
- Commentaires : *Parce que un éditeur de texte pour un fichier, c'est bien, mais en ligne de commande pour 20 fichiers, c'est mieux*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  à-compléter
  - Création par bendia 03/05/2017
  - Testé par bendia le 3/05/2017
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Introduction

Rechercher et *Remplacer*, voilà une des fonction d'un éditeur de texte des plus utilisée. C'est super cool, je veux remplacer `Bob` par `Alice` dans mon super tuto, il n'y a qu'un fichier, facile. Mais voilà, mon tuto est découpé en 20 parties, chacune dans un fichier Il faut donc les ouvrir un par un, et lancer le remplacement.

`rpl` permet de faire ça en une ligne de commande, avec pas mal d'options pour affiner un peu, comme :

- le faire récursivement (en inspectant les sous-dossiers),
- tenir compte de la casse,
- ne remplacer que le mot entier,
- choisir l'extension du fichier
- et bien d'autres options encore...

Installation

```
apt-get install rpl
```

Utilisation

La syntaxe générale est la suivante

```
rpl [-LhiwbqvsRepfdt] [-xSUFFIX] ⟨vieille_chaine⟩ ⟨nouvelle_chaine⟩
⟨fichier_cible ...⟩
```

Par exemple, si l'on veut remplacer `Bob` par `Alice` dans les 2 premières parties de mon tuto fictif, après s'être placé dans le dossier contenant les fichiers à traiter :

```
rpl 'Bob' 'Alice' partie1.txt partie2.txt
```

On peut bien sûr utiliser les caractères joker, pour traiter tous les fichiers du dossier par exemple :

```
rpl 'Bob' 'Alice' *
```

La récursivité avec l'option R

Si les fichiers à visiter ne se situe pas dans le même dossier, on peut indiquer que tous les fichiers sont à visiter :

```
rpl -R 'Bob' 'Alice' *
```

La simulation

Pour voir le résultat sans rien toucher, l'option `-s` ou `–dry-run` permet de voir les fichiers qui seraient modifiés si besoin

La casse

Par défaut, la commande tient compte de la casse.
Pour passer outre ce comportement, utiliser l'option `-i`

L'extension des fichiers

Pour spécifier une extension, on ajoute l'option `-x`, qui peut être spécifiée plusieurs fois.
Si tu veux filtrer pour plusieurs extensions ça fonctionne genre pour ne remplacer que dans les fichier txt et html

```
rpl -R -x'txt' -x'html' 'Bob' 'Alice' *
```

Rechercher un mot entier

On peut s'assurer que le motif de recherche se limite à un mot entier avec l'option `-w`. Cela permet ainsi de remplacer `Bob` par `Alice`dans notre tuto mais d'éviter de se retrouver avec un truc qui fait mauvais genre dans votre devoir de Français 😉

```
Tire la chevillette, et la Aliceinette cherra
```

Un TP pour bien comprendre

Nous allons essayer d'illustrer l'utilisation de rpl et quelques-une de ses options en s'exercant sur quelques fichiers d'exemple

Mise en place

On va préparer un petit environnement d'exemple, avec des fichiers aynat différentes extensions, placés dans différents dossiers. Il suffit de télécharger ce fichier et l'executer (ou appliquer les commandes une par une) :-p

TPrpl.sh

```
#!/bin/sh
cd /tmp
mkdir -P test/dossier
cd test
echo "Bob tire la Bobinette" > f1.txt
cp f1.txt f2.html
cp f1.txt f3.tex
cp f* dossier/
```

TP

```
rpl "Bob" "Alice" f1.txt
```

```
Replacing "Bob" with "Alice" (case sensitive) (partial words matched)
.
A Total of 2 matches replaced in 1 file searched.
```

```
cat f1.txt
```

```
Alice tire la Aliceinette
```

```
cat f2.html
```

```
Bob tire la Bobinette
```

```
cat dossier/f*
```

```
Bob tire la Bobinette
Bob tire la Bobinette
Bob tire la Bobinette
```

Donc, on a touché uniquement à `f1.txt` (puisqu'on l'a explicitement donné en argument). LE fichier

f1.txt situé dans `/dossier` n'a pas été modifié. Et on voit que toutes les occurences de Bob on été remplacée, même dans un mot.

Remettons donc les choses comme à l'origine

```
rpl "Alice" "Bob" f1.txt
```

Pour que seul le mot complet Bob soit remplacé, on va utiliser l'option `-w`.

```
rpl -w "Bob" "Alice" f1.txt
```

```
Replacing "Bob" with "Alice" (case sensitive) (partial words matched)
.
A Total of 2 matches replaced in 1 file searched.
```

</code>

```
Replacing "Bob" with "Alice" (case sensitive) (whole words only)
.
A Total of 1 matches replaced in 1 file searched.
```

Et pour vérifier

```
cat f1.txt
```

```
Alice tire la Bobinette
```

Remplaçont à présent Bob par alice dans tous le dossier

```
rpl -w "Bob" "Alice" *
```

```
Replacing "Bob" with "Alice" (case sensitive) (whole words only)
...
A Total of 2 matches replaced in 3 files searched.
```

```
cat f*
```

```
Alice tire la Bobinette
Alice tire la Bobinette
Alice tire la Bobinette
```

```
cat dossier/f*
```

```
Bob tire la Bobinette
Bob tire la Bobinette
Bob tire la Bobinette
```

On voit donc que seul les fichiers du dossier `test` sont modifiés. Pour modifier l'ensemble des fichiers de tous les dossiers, on va utiliser l'option `-R`

Corrigeons donc ce *B* majuscule

```
rpl -R "Bobinette" "bobinette" *
```

Pour aller plus loin

N'hésitez pas à consulter la page de manuel pour voir toutes les options <code user>man rpl

[1)]

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