Buscar y remplazar

2015-12-01

[TOC]

Introducción:

Vamos a repasar algunos métodos para buscar y remplazar una palabra o expresion regular. Las opciones que vamos a considerar son las siguientes:

- Sea recursivo
- Permite usar expresiones regulares

Herramientas nativas

Las siguientes herramientas son nativas de linux y posiblemente estén instaladas en todas las distribuciones de linux.

Search and replace con egrep

Muestra los ficheros que contienen el patrón a buscar con `egrep` y con `sed` los remplaza.

- Recursivo:
- Regex: *(Con matices, sed no acepta caracteres de búsqueda literal como ¸ pero si [0-9]+)*

```
#!/bin/bash
buscar="uno?"
remplazar="UNO"
ruta="/tmp/kk"
egrep -rl $buscar $ruta | xargs sed -i -E "s/$buscar/$remplazar/g"
```

Search and replace con find

Busco los ficheros filtrándolos con `find`, después con `sed` remplazo.

Funcionalidad: - Recursivo: - Regex: *(Con matices, sed no acepta caracteres de búsqueda literal como ¸ pero si [0-9]+)*

```
#!/bin/bash
buscar="\d"
remplazar="uno"
ruta="/tmp/kk"
filtro='*'
find $ruta -type f -name "$filtro" | xargs sed -i -E "s/$buscar/$remplazar/g"
```

Herramientas externas

Search and replace con Python

Script hecho en python para buscar y remplazar en un arbol de directorios.

Funcionalidad: - Recursivo: - Regex:

Descargar script

```
#!/usr/bin/env python
# Fuente: http://code.activestate.com/recipes/580653-recursive-find-replace-in-files-using-regex/
```

```
import os
import fnmatch
import sys
import shutil
import re
import argparse


def find_replace(cfg):
    search_pattern = re.compile(cfg.search_regex)
    for path, dirs, files in os.walk(os.path.abspath(cfg.dir)):
        for filename in fnmatch.filter(files, cfg.glob):
            pardir = os.path.normpath(os.path.join(path, '../../../others/content'))
            pardir = os.path.split(pardir)[-1]
            print
            '[%s]' % pardir,
            filepath = os.path.join(path, filename)

            # backup orig file
            if cfg.create_backup:
                backup_path = filepath + '.bak'

                while os.path.exists(backup_path):
                    backup_path += '.bak'
                print('DBG: creating backup', backup_path)
                shutil.copyfile(filepath, backup_path)

            with open(filepath) as f:
                data = f.read()

            all_matches = search_pattern.findall(data)

            if all_matches:

                one_match = all_matches[0]

                print('Found {} matches in file {}'.format(len(all_matches), filename))

                if not cfg.dryrun:
                    with open(filepath, "w") as f:
                        print('DBG: replacing in file', filepath)
                        # s = s.replace(search_pattern, replacement)
                        new_data = search_pattern.sub(cfg.replace_regex, data)
                        f.write(new_data)
                else:
                    for line in data.split('\n'):
                        if one_match in line:
                            print("    OLD: {}".format(line.strip()))
                            print("    NEW: {}".format(search_pattern.sub(cfg.replace_regex, line).strip()))

            else:
                print('File {} does not contain search regex "{}"'.format(filename, cfg.search_regex))


if __name__ == '__main__':
    ejemplo = './search_replace.py -d "../" -s "tags:" -r "Tags:" -g "*.md"'
    parser = argparse.ArgumentParser(
        description='DESCRIPCION:\n   Buscar y reemplazar recursivamente desde la carpeta dada usando expresi
        formatter_class=argparse.RawDescriptionHelpFormatter,
```

```
        epilog='''USO:\n   {0} -d "directorio/" -s "busca" -r "remplaza" -g "*.md"'''.format(
            os.path.basename(sys.argv[0])))

    parser.add_argument('--dir', '-d',
                        help='Carpeta a buscar, por defecto la carpeta actual',
                        default='.')

    parser.add_argument('--search-regex', '-s',
                        help='busca regex',
                        required=True)

    parser.add_argument('--replace-regex', '-r',
                        help='remplaza regex',
                        required=True)

    parser.add_argument('--glob', '-g',
                        help='patron de filtro para expecificar ficheros a remplazar, ej: *.md',
                        default="*.*")

    parser.add_argument('--dryrun', '-dr',
                        action='store_true',
                        help="No modificar nada, solo mostrar por pantalla",
                        default=False)

    parser.add_argument('--create-backup', '-b',
                        action='store_true',
                        help='Create backup files',
                        default=False)

    config = parser.parse_args(sys.argv[1:])

    find_replace(config)
```

Search and replace con Perl

Parámetros de perl: - -e significa ejecutar la siguiente línea de código. - -i significa editar en el lugar - -w escribe advertencias - -p en el archivo de entrada, imprimiendo cada línea después de que se le aplique el script.

Funcionalidad: - Recursivo: - Regex:

En Fedora no me funciona

```
#!/bin/bash
buscar="v"
remplazar="12_"
ruta="/tmp/kk"
perl -pi -we "s/$buscar/$remplazar/g;" $(grep -rl "$ruta")
```

Tabla resumen

Metodo	Recursivo	Regex
egrep		*
find		*
python		
perl		

* sed no acepta caracteres de búsqueda literal como ¸ pero si [0-9]+)