## Name

renamex − file rename tool

## SYNOPSIS

**renamex** *[options] files ...*

## DESCRIPTION

**Renamex** is a small and quick tool written in C to rename files. It can change, lowercase and uppercase the names of a batch of files. **Renamex** is powered by the *extended regular expression* for searching and substituting string patterns in the names.

## OPTIONS

**−l**, **−−lowercase**

Lowercase the specified file names.

**−u**, **−−uppercase**

Uppercase the specified file names.

**−p**,**−−prefix**

Add prefix to the file names.

**−x**,**−−suffix**

Add suffix to the file name. If there is an *extension name* , the **suffix** will be inserted before the *extension name* , otherwise it will be simply appended to the tail of the name.

**−R**, **−−recursive**

Perform on the specified files and all subdirectories.

**−t**, **−−test**

Test only mode. It won't change anything, just display the output of renaming.

**−f**, **−−file**

Load file names from a file.

**−v**, **−−verbose**

verbose display.

**−G**, **−−gui**

Start this program with a graphic user interface. The command line options and arguments will be passed to the GUI frontend.

**−s/PATTERN/STRING[/SW]**

Substitute PATTERN with STRING in the filenames. **renamex** will searching the *PATTERN* in the names. If the *PATTERN* matches, it will be replaced by the *STRING* and then use the system call to change the file name. **SW** is the combination of the following switches:

**g** replace all occurrences in the filename.

**i** ignore case when searching.

**b** backward searching and substituting. It does NOT support regular expression.

**e** search and replace in the file's extension name.

**r** claim the PATTERN is a regular expression.

**x** claim the PATTERN is an extended regular expression.

**1-9** replace 1 to 9 occurrences in the filename. Note that **0** is actually the same to the **g** switch.

## REGULAR EXPRESSION

This section about extended regular expression is digisted from the manpage of **fgrep(1).** See it for details.

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any metacharacter with special meaning may be quoted by preceding it with a backslash.

A list of characters enclosed by **[** and **]** matches any single character in that list; if the first character of the list is the caret ^ then it matches any character *not* in the list. For example, the regular expression **[0123456789]** matches any single digit. A range of ASCII characters may be specified by giving the first and last characters, separated by a hyphen. Finally, certain named classes of characters are predefined. Their names are self explanatory, and they are **[:alnum:]**, **[:alpha:]**, **[:cntrl:]**, **[:digit:]**, **[:graph:]**, **[:lower:]**, **[:print:]**, **[:punct:]**, **[:space:]**, **[:upper:]**, and **[:xdigit:].** For example, **[[:alnum:]]** means **[0-9A-Za-z]**, except the latter form is dependent upon the ASCII character encoding, whereas the former is portable. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket list.) Most metacharacters lose their special meaning inside lists. To include a literal **]** place it first in the list. Similarly, to include a literal ^ place it anywhere but first. Finally, to include a literal − place it last.

The period **.** matches any single character. The symbol **\w** is a synonym for **[[:alnum:]]** and **\W** is a synonym for **[^[:alnum]]**.

The caret ^ and the dollar sign **$** are metacharacters that respectively match the empty string at the beginning and end of a line. The symbols \< and \> respectively match the empty string at the beginning and end of a word. The symbol **\b** matches the empty string at the edge of a word, and **\B** matches the empty string provided it's *not* at the edge of a word.

A regular expression may be followed by one of several repetition operators:

**?** The preceding item is optional and matched at most once.

***** The preceding item will be matched zero or more times.

**+** The preceding item will be matched one or more times.

**{***n***}** The preceding item is matched exactly *n* times.

**{***n***,}** The preceding item is matched *n* or more times.

**{,***m***}** The preceding item is optional and is matched at most *m* times.

**{***n***,***m***}** The preceding item is matched at least *n* times, but not more than *m* times.

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated subexpressions.

Two regular expressions may be joined by the infix operator |; the resulting regular expression matches any string matching either subexpression.

Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole subexpression may be enclosed in parentheses to override these precedence rules.

The backreference *n*, where *n* is a single digit, matches the substring previously matched by the *n*th parenthesized subexpression of the regular expression.

In basic regular expressions the metacharacters **?**, **+**, **{**, |, **(**, and **)** lose their special meaning; instead use the

backslashed versions **\?**, **\+**, **\{**, **\|**, **\(**, and **\)**.

## SEE ALSO

mv(1), chown(1), regex(7), regex(3)

## COPYING



## BUGS

Please send bug reports to "Andy Xuming" <xuming@users.sourceforge.net>

## EXAMPLES

**renamex −l −R ***

To lowercase all files' names recursively.

**renamex −u −s/abc/xyz/gi *.c**

Substitute all *abc* substrings appeared in C sources files with *xyz* , ignoring the case, then uppercase the whole file name.

**renamex −v −s/.c/.cpp/e ***

Find all files which have the *'.c'* extension name in the current directory and change them to *'.cpp'* extension name. Print the verbose information.

**find . −name *.c > filename.lst**

**renamex −s/.c/.cpp/e −f filename.lst**

Find all files which have the *'.c'* extension name in the current directory and change them to *'.cpp'* extension name by the list file.

**renamex −s/abc/12345/bi ***

Read names from the *'filename.lst'* , find the last occurrence of *'abc'* and replace it with *'12345'* , ignoring the case.

**renamex −s/^[A−Z].*file/nofile/r ***

The target substring starts with a capital letter, and ends with string *'file'* . There are 0 or any numbers of characters between the capital letter and *'file'* . The substring, if encountered in filenames, will be replaced with *'nofile'*.

**renamex −s/^[A−Z].+file/nofile/xg ***

Similar to above, except it uses extended regular expression, such as the *'+'* metacharacter, and replaces all matching strings with *'nofile'*.

**renamex −t −s/^[A−Z].+file/nofile/xg ***

**Test**mode only. Simulate the rename process but no files would be actually changed.