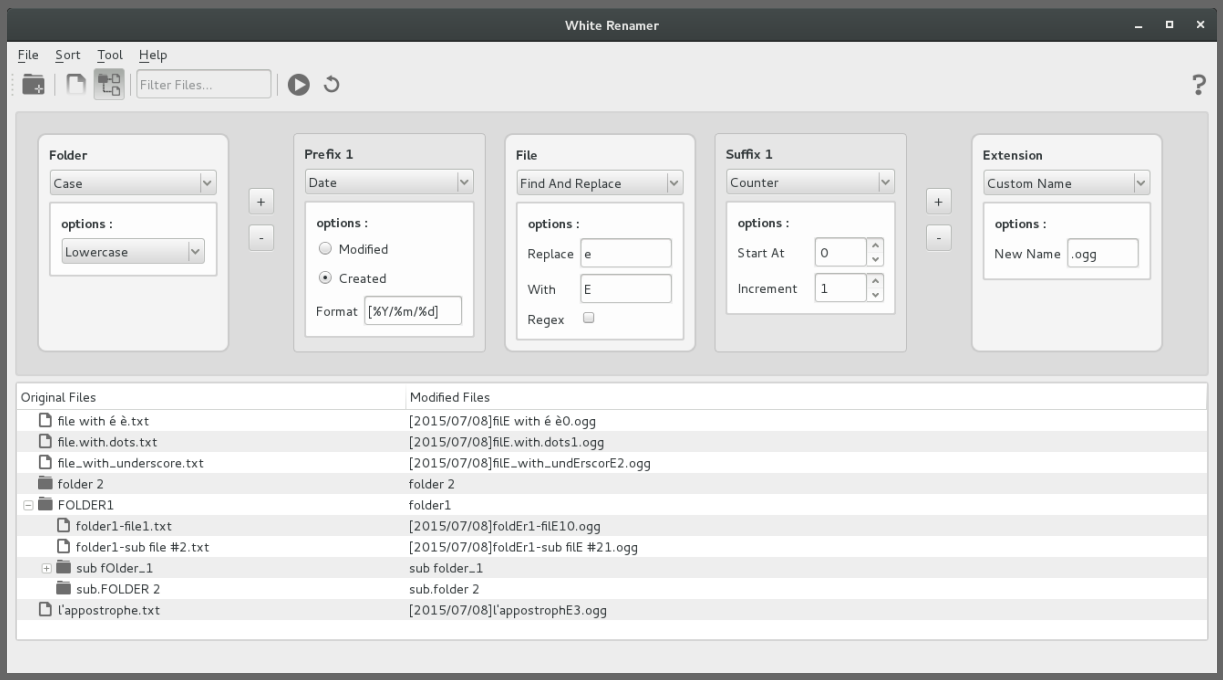
White Renamer

Rename your files, simply.



pierrecnalb@mailbox.org

1 Introduction

White Renamer is a software that ease the pain of renaming multiple files and folders by doing it in batch. The user can choose many actions to perform on the selected directory. The following sections describes what those actions are, and how to use it.

2 Description

White Renamer has a simple UI containing two distinct parts : the top part displays boxes representing the structure of the files.
The first box : *Folder* represents the changes made on the folders inside the selected directory.
The second box : *File* represents the changes made on the files inside the selected directory.
The third box : *Extension* represents the changes made on the extension of the files.
Some optional boxes can be inserted : *Prefix* and *Suffix* that give the possiblity to add prefix(es) or suffix(es) to the filename.

The bottom part displays a live preview of the changes made to the files/folders.

3 Step by step

1. Select which directory you want to process.
2. Select if you want to rename hidden directories.
3. Select if you want to rename subfolders.
4. Enter the desired pattern to rename only specific files.
5. Sort by name(default), size or date in normal or reverse order.
6. Add some prefixes and/or suffixes.
7. Select the desired actions to perform on the folders, files and extensions.
8. Press on Rename.

4 List of available actions

The following actions are available :

Original Name

(*see figure 1*)

No changes are made. The files or folders keep their original names.

Case

(*see figure 2*)

Three capitalization options are available :

- **Titlecase** : gives the possiblity to uppercase the first letter of each words.
  - *First Letter* : if checked, the first letter of the string will be capitalized.

  - *And After* : represents the characters after which the letter will go uppercase.

  E.g : if First Letter is checked and "And After = _" the file *my_file* will become My_File

- **Uppercase** : The whole string will be converted to uppercase.
- **Lowercase** : The whole string will be converted to lowercase.

Custom Name

(*see figure 3*)

- **New Name** : changes the original name to the given new name.

Folder Name

(*see figure 4*)

Gives the name of the parent folder.

Find And Replace

(*see figure 5*)

- **Replace** : which characters should be replaced.
- **With** : with which characters the found token should be replaced.
- **Regex** : If checked, gives the ability to use regex. A list of commonly used regex is given below. For more information, please read the python re help page.

```
The special characters are:
       "."      Matches any character except a newline.
       "^"      Matches the start of the string.
       "$"      Matches the end of the string or just before the newline at
                the end of the string.
       "*"      Matches 0 or more (greedy) repetitions of the preceding RE.
                Greedy means that it will match as many repetitions as possible.
       "+"      Matches 1 or more (greedy) repetitions of the preceding RE.
       "?"      Matches 0 or 1 (greedy) of the preceding RE.
       *?,+?,?? Non-greedy versions of the previous three special characters.
       {m,n}    Matches from m to n repetitions of the preceding RE.
       {m,n}?   Non-greedy version of the above.
       "\\"     Either escapes special characters or signals a special sequence.
       []       Indicates a set of characters.
                A "^" as the first character indicates a complementing set.
       "|"      A|B, creates an RE that will match either A or B.
       (...)    Matches the RE inside the parentheses.
                The contents can be retrieved or matched later in the string.
       (?aiLmsux) Set the A, I, L, M, S, U, or X flag for the RE (see below).
       (?:...)  Non-grouping version of regular parentheses.
       (?P<name>...) The substring matched by the group is accessible by name.
       (?P=name)     Matches the text matched earlier by the group named name.
       (?#...)  A comment; ignored.
       (?=...)  Matches if ... matches next, but doesn't consume the string.
       (?!...)  Matches if ... doesn't match next.
       (?<=...) Matches if preceded by ... (must be fixed length).
       (?<!...) Matches if not preceded by ... (must be fixed length).
       (?(id/name)yes|no) Matches yes pattern if the group with id/name matched,
                          the (optional) no pattern otherwise.


The special sequences consist of "\\" and a character from the list
below.  If the ordinary character is not on the list, then the
resulting RE will match the second character.
    \number  Matches the contents of the group of the same number.
    \A       Matches only at the start of the string.
    \Z       Matches only at the end of the string.
    \b       Matches the empty string, but only at the start or end of a word.
```

```
\B      Matches the empty string, but not at the start or end of a word.
\d      Matches any decimal digit; equivalent to the set [0-9] in
        bytes patterns or string patterns with the ASCII flag.
        In string patterns without the ASCII flag, it will match the whole
        range of Unicode digits.
\D      Matches any non-digit character; equivalent to [^\d].
\s      Matches any whitespace character; equivalent to [ \t\n\r\f\v] in
        bytes patterns or string patterns with the ASCII flag.
        In string patterns without the ASCII flag, it will match the whole
        range of Unicode whitespace characters.
\S      Matches any non-whitespace character; equivalent to [^\s].
\w      Matches any alphanumeric character; equivalent to [a-zA-Z0-9_]
        in bytes patterns or string patterns with the ASCII flag.
        In string patterns without the ASCII flag, it will match the
        range of Unicode alphanumeric characters (letters plus digits
        plus underscore).
        With LOCALE, it will match the set [0-9_] plus characters defined
        as letters for the current locale.
\W      Matches the complement of \w.
\\      Matches a literal backslash.
```

Insert Characters

(*see figure 6*)

- **Insert** : what characters should be inserted.
- **At Position** : position, in the original name, of the characters to be inserted. 0 represents the beginning of the string.

Delete Characters

(*see figure 7*)

- **From** : starting position of the characters to remove.
- **To** : ending position of the characters to remove.

Counter

(*see figure 8*)

- **Start At** : starting number of the counter.
- **Increment** : increment number of the counter.

Date

(*see figure 9*)

- **Modified** : date of the last modification.
- **Created** : date of the creation.
- **Format** : date format specification. Commonly used formats are :

```
%Y  Year with century as a decimal number.
%m  Month as a decimal number [01,12].
%d  Day of the month as a decimal number [01,31].
%H  Hour (24-hour clock) as a decimal number [00,23].
%M  Minute as a decimal number [00,.8].
%S  Second as a decimal number [00,61].
%z  Time zone offset from UTC.
%a  Locale's abbreviated weekday name.
%A  Locale's full weekday name.
%b  Locale's abbreviated month name.
%B  Locale's full month name.
%c  Locale's appropriate date and time representation.
%I  Hour (12-hour clock) as a decimal number [01,12].
%p  Locale's equivalent of either AM or PM.
```

5 Examples

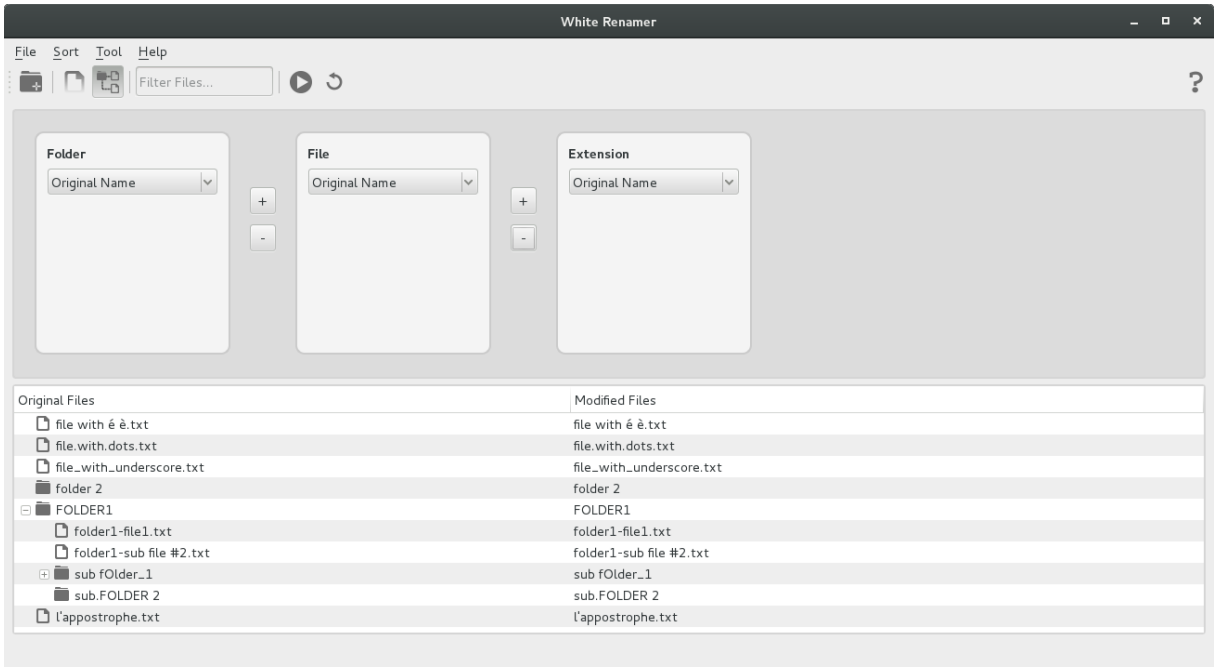


Figure 1: Original Name option example.

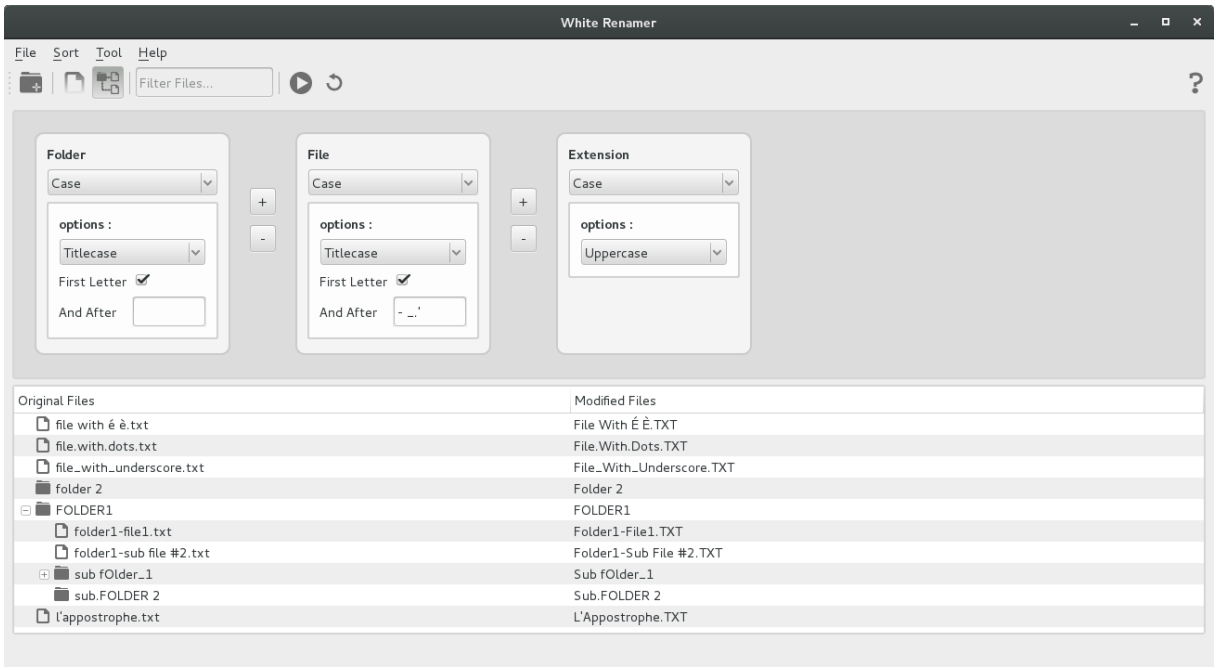


Figure 2: Case option example.

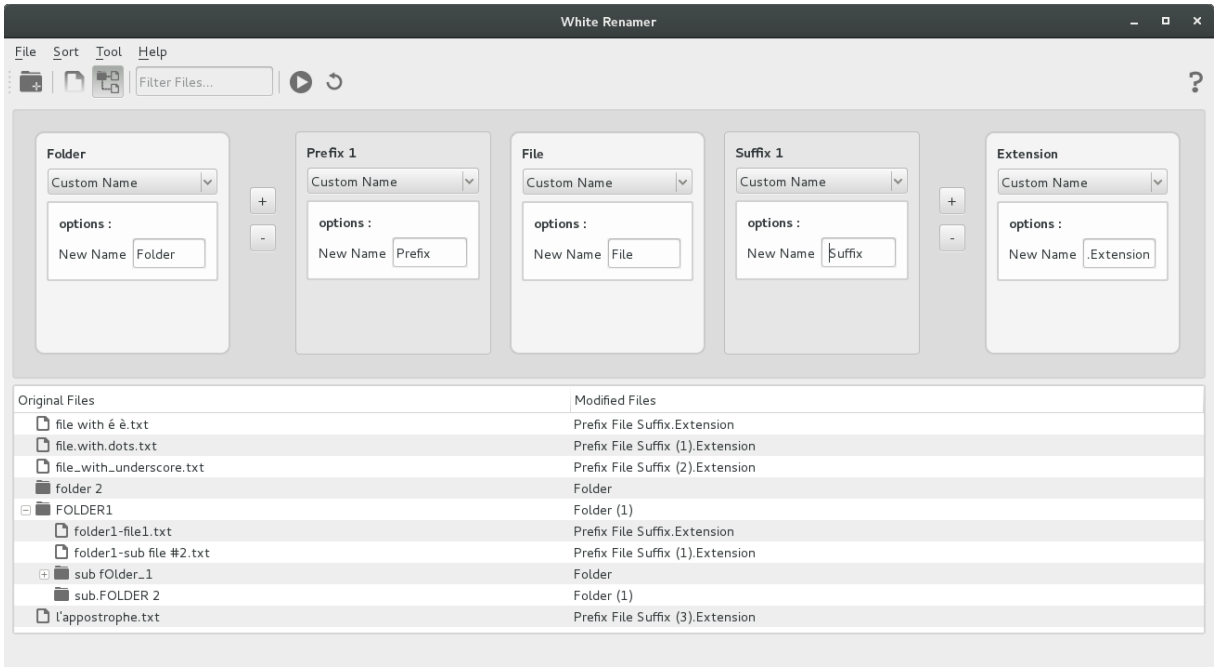


Figure 3: Custom Name option example.

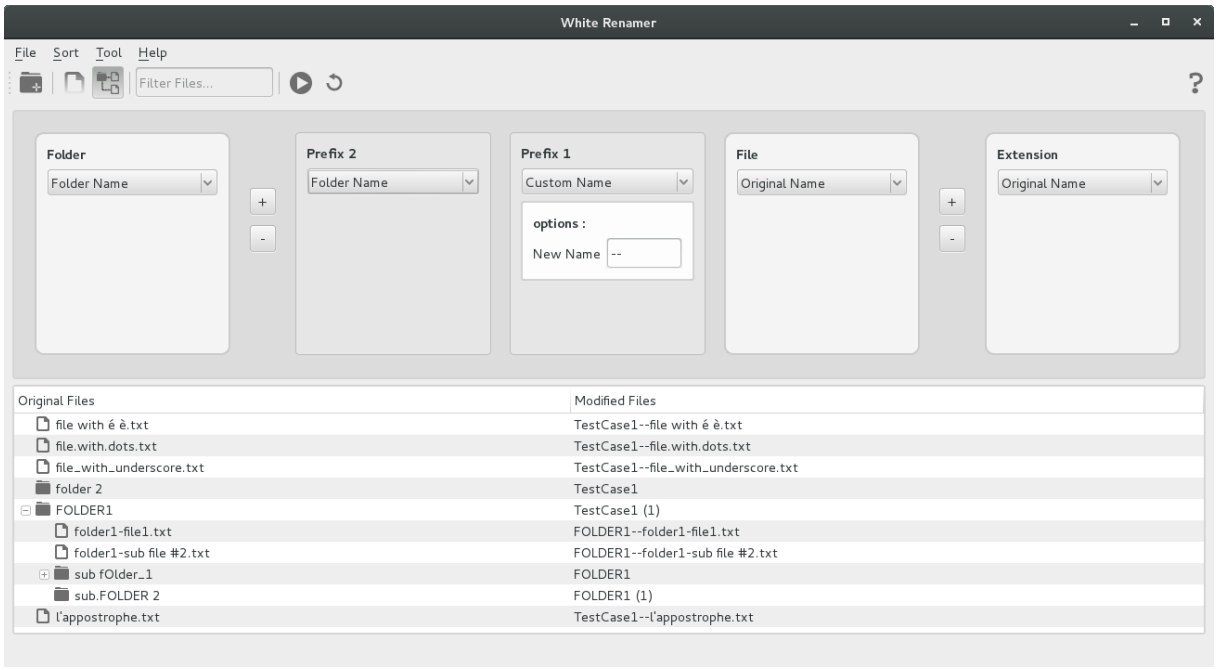


Figure 4: Folder Name option example.

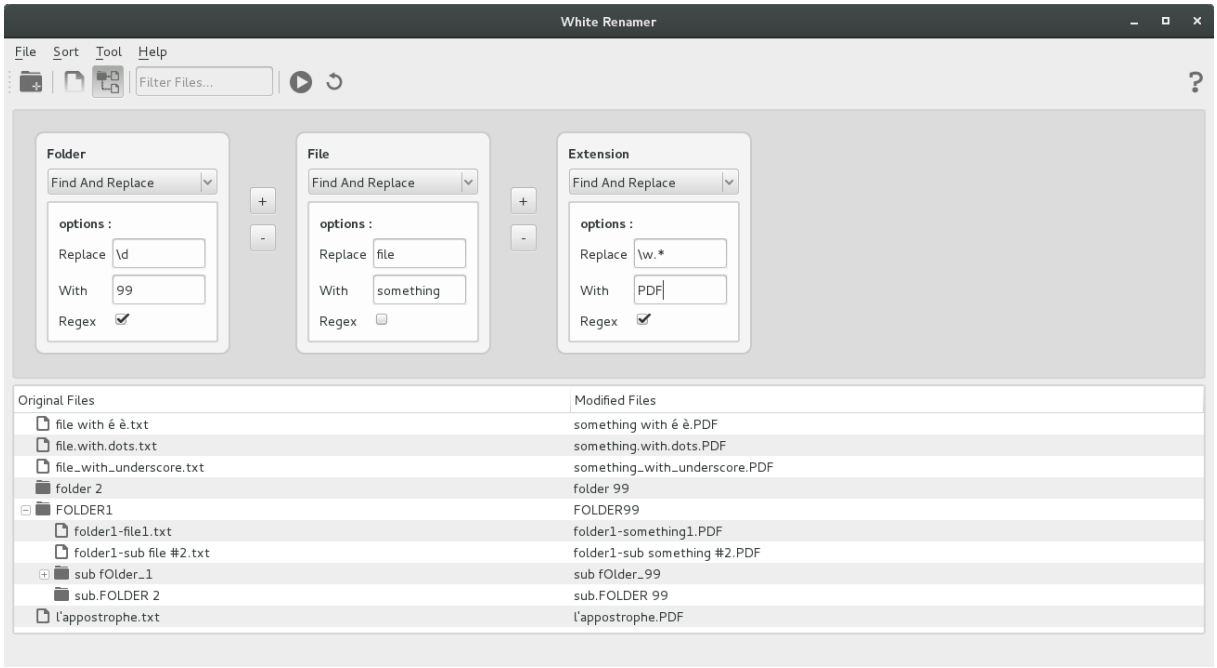


Figure 5: Find And Replace option example.

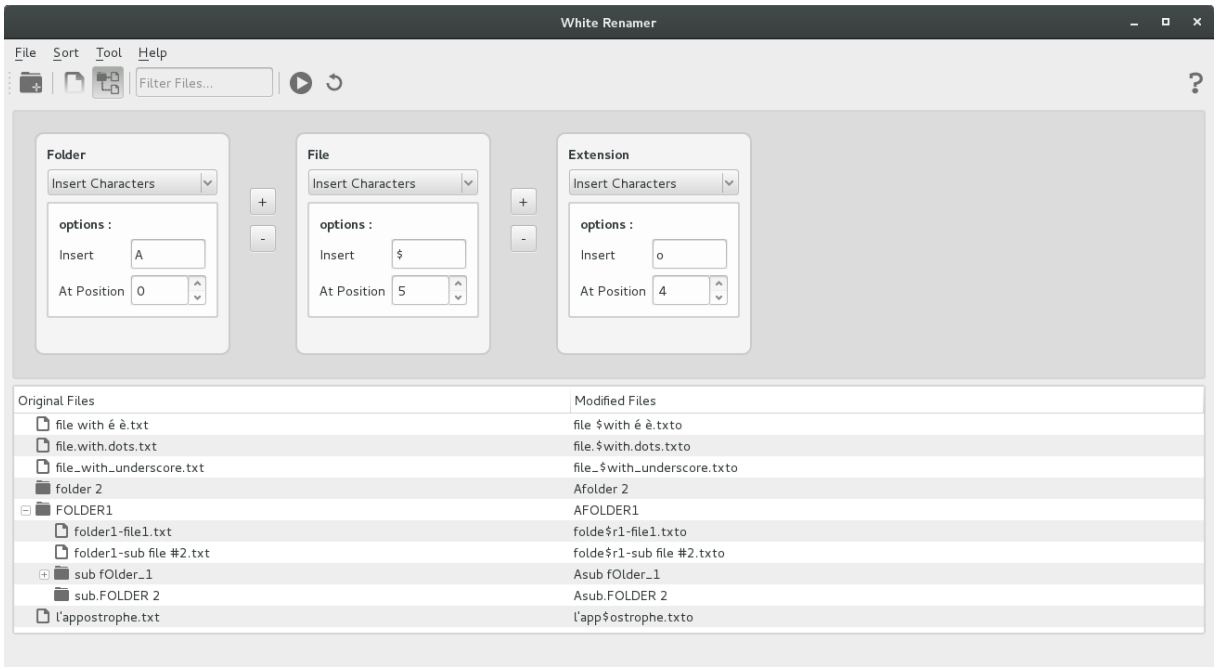


Figure 6: Insert Characters option example.

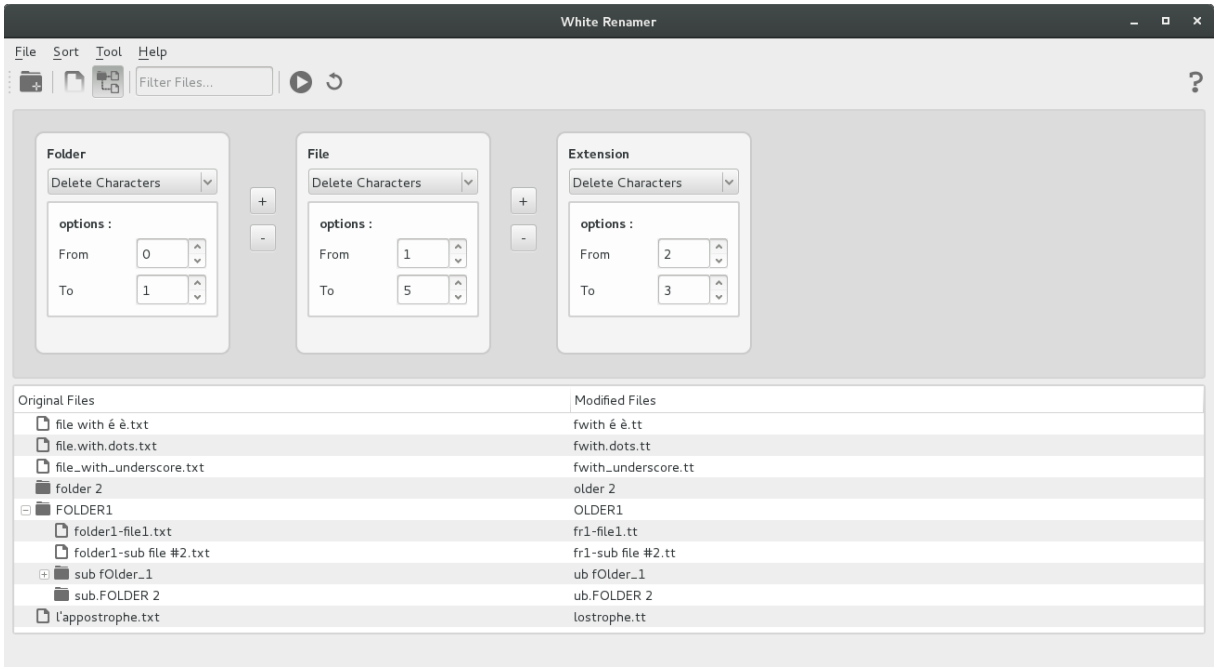


Figure 7: Delete Characters option example.

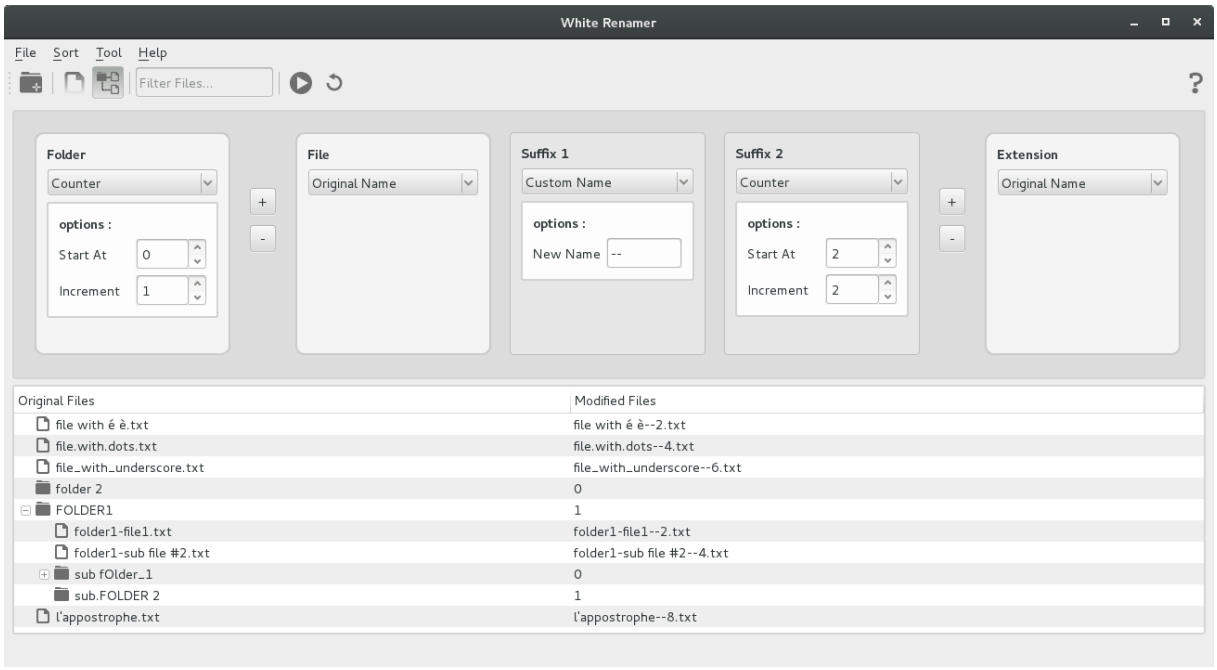


Figure 8: Counter option example.

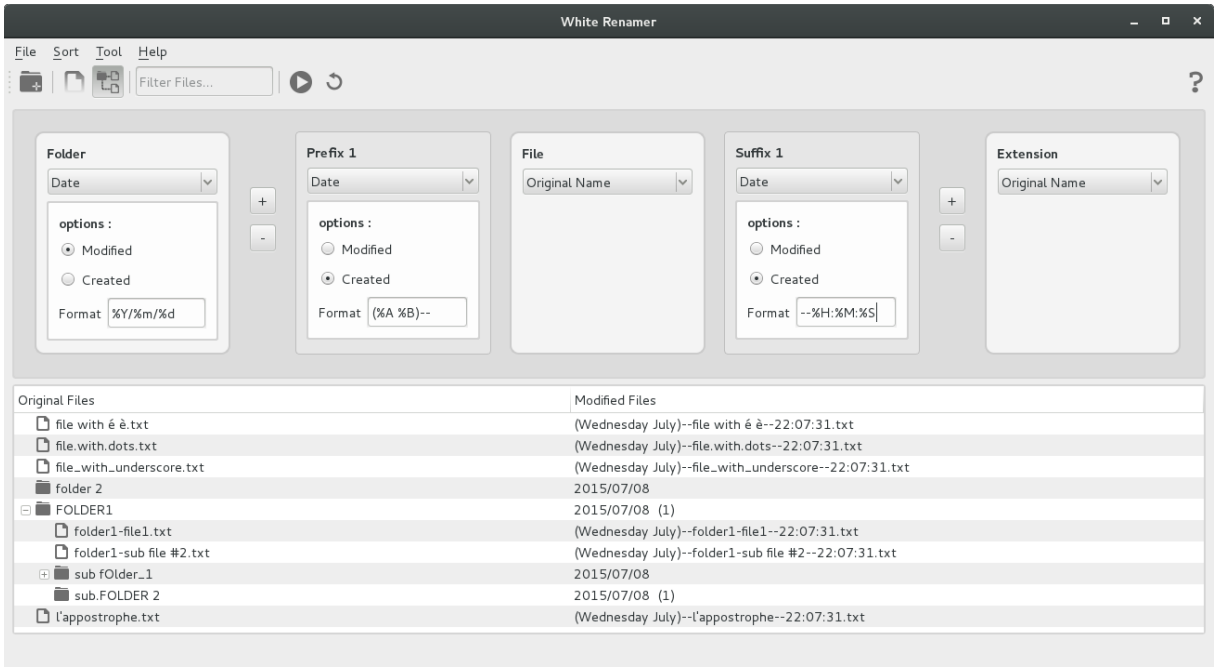


Figure 9: Date option example.