Table of Contents

Introduction

Once in a lifetime of a computer user is the need to rename multiple folders or files. Normally, the user will try to find all the locations of the folders or files to be renamed one by one starting from a starting folder *(i.e. target folder)*. Some sample use cases that multiple renaming is done by the user are the following:

- Copied a bunch of template folders and files and then renamed them accordingly.
- Downloaded a bunch of episode of a TV series and the user has his/her own file naming convention.
- A java developer is duplicating a big working package where the source files have embedded acronyms where he needs to rename those with a new the acronym.

Without a simple tool, these use cases are very time consuming to accomplish. This is the motivation that made this simple **Name Modifier** project to exist.

The Package

Name Modifier has three packages for different OS:

Package	Description
namemod-<version>.zip	Contains the Python source codes and documents packaged for Windows and **Python 3.4 is required**.
namemod-<version>.tar.gz	Contains the Python source codes and documents packaged for Linux and **Python 3.4 is required**.
namemod-<version>.win-bin.zip	Contains binary for Windows which doesn't need Python to be installed

These packages are normally downloadable in addresses below:

https://pypi.python.org/pypi/namemod

and

https://sourceforge.net/projects/namemodifier/files

If the user had Python 3.4 installed, he/she doesn't necessary needs to download this binary. The user can just use the pip module *(see Installing the module using pip)* for installation.

Installing the Package for Windows

The installation is as easy as downloading the namemod-<version>.win-bin.zip package and extracting it to wherever directory the user needs it to be. Moreover, it is recommended to include the installation directory to the users **PATH environment** variable.

Installing the module using pip

If the user had Python 3.4 installed, he/she can just use the pip module by running the command in Command 1.

Command 1. Using pip module for installation

```
C:\>python -m pip install namemod
```

Using the namemod command

Name Modifier can be used as easy as the other command line *(e.g. dir)* known to the user. The main command in this package is **namemod**. This command has many arguments that the user can opt to.

If the user opted to install namemod using pip module *(see Installing the module using pip)*, all the commands in the following sections can be carried out by using python with the following format found in Command 2:

Command 2. Using namemod with python

```
python -m namemod [optional arguments] <folder> <old_name_segment>
<new_name_segment>
```

The -h or --help argument

Normally the first thing to do is to know how to use the command. This can be accomplished quickly by the -h or --help argument. For example, execute the command found in Command 3.

Command 3. Using -h argument

```
C:\tmp\testfolder\>namemod -h
```

The output would be the one shown in Listing 1.

Listing 1. -h argument output

```
usage: namemod [-h] [-f] [-F] [-i] [-I] [-m {text,regex}] [-n] [-R]
               [-xr regex] [-xt text] [-Xr regex] [-Xt text] [-v]
               folder old_name_segment new_name_segment

Simplifying multiple files or folders renames.

positional arguments:
  folder                Specifies the target folder to process.
  old_name_segment      Old name segment to be renamed.
  new_name_segment      New name segment.

optional arguments:
  -h, --help            show this help message and exit
  -f, --file-only       Process only files.
  -F, --folder-only     Process only folders.
  -i, --interactive     Ask confirmation before renaming.
  -I, --include-target-folder
                        Include the target folder in renaming.
  -m {text,regex}, --mode {text,regex}
                        Default: text
  -n, --noop            Inhibit the actual renamig but display the intended
                        outcome.
  -R, --recursive       Recursive
  -xr regex, --exclude-file-regex regex
                        Exclude files by regex.
  -xt text, --exclude-file-text text
                        Exclude files by exact text match.
  -Xr regex, --exclude-folder-regex regex
                        Exclude folders by regex.
  -Xt text, --exclude-folder-text text
                        Exclude folders by exact text match.
  -v, --version         show program's version number and exit
```

The positional arguments

The simplest use case of namemod command is to use it with the required positional arguments. There are only three positional arguments namely: folder, old_name_segment and new_name_segment. So, why do we have three where normally we only need two for renaming, one for the old name and another one for the new name. The reason for this is that the namemod command is not just operating to the exact match of the old name but it can also operate in just the segment of the name. To make this possible the user needs to specify the target folder. This dictates namemod to only limit its processing to the target folder.

Figure 1. Specimen testfolder1 hierarchy

If the user had a folder structure likes the one specified by Figure 1 and the user wanted to change the name of filetest1.txt to filenew1.txt but not including the filetest1.txt in the subfolders. The user can run the command in Command 4.

Command 4. Basic use if the positional arguments

```
C:\tmp\testfolder>namemod testfolder1 filetest1.txt filenew1.txt
```

This command will output Listing 2 *(i.e. line numbers are not included in the actual output)*.

Listing 2. Output of Command 4

```
1  Renaming testfolder1\filetest1.txt to testfolder1\filenew1.txt
2  Renamed
3  Finished
```

Since the user just used the basic use of the namemod command the default behaviour are the following:

- The command is only processing file *(i.e. like using -f or --file argument)*.
- The argument old-name-segment is operating in exact match *(i.e. like using -m text)*.
- The command is only the processing at the level of the folder positional argument and doesn't go down the subfolders.

The typical output that the user should expect is like the one in Listing 2. Based on the listing, namemod is telling to user what it is doing and if it succeeded with it. Here is the breakdown of the meaning of the output.

- Line 1, namemod it is trying to do a rename.
- Line 2, the rename is successful.
- Line 3, notifies the user that it is finished.

The -F or --folder argument

The argument that tells namemod that the user wants to do renaming of folders instead of files is -F or --folder. If the user still wanted to use the folder structure in Figure 1 and also wanted to rename testfolder11 to newfolder11 inside the C:\tmp\testfolder\testfolder1. The user can use the command in Command 5.

Command 5. Using -F argument

```
C:\tmp\testfolder>namemod -F testfolder1 testfolder11 newfolder11
```

Now, if the folder has an argument in namemod, is there any equivalent to a file? Yes, the user can use -f or --file to explicitly tell namemod that it must do file renaming. But this is not necessary since this is the default behaviour.

> ***Note:*** *-F and -f arguments cannot be used together in namemod.*

The -I or --include-target-folder

The supplementary argument to -F argument is -I or --include-target-folder *(i.e. this doesn't do anything with -f argument)* argument. Using -I or --include-target-folder means that namemod will include the **folder positional argument** *(see*

The positional arguments *section)* in the renaming activity if it qualifies the **old_name_segment positional argument** *(see*

The positional arguments *section)* depending on the mode used *(see The -m or --mode argument section)*.

The -m or --mode argument

There are two modes of processing files or folders that can be used with namemod. One is text mode which is the default and the other is regex mode. Because of this, the argument -m or --mode requires an option *(see Table 1)* when being used.

Table 1. Options of the mode argument.

Option	Description
text	If -m has this option, namemod will do exact matching.
regex	If -m has this option, namemod will do segment matching based on regular expression found in the old_name_segment positional argument. Regex is not a topic in this user guide but if the user wanted to know more about this please check http://www.regular-expressions.info.

Still using the directory structure in Figure 1, if the user wanted to rename all the files in folder C:\tmp\testfolder\testfolder1\testfolder1 to make all the **test segments** to **new segments** *(e.g. file**test**1.txt to file**new**1.txt)*. The user can issue the command in Command 6.

Command 6. Using -m regex argument

```
C:\tmp\testfolder>namemod -m regex testfolder1\testfolder1 test new
```

The output is similar to Listing 3.

Listing 3. Output of Command 6

```
Renaming testfolder1\testfolder1\filetest1.txt to testfolder1\testfolder1\filenew1.txt
Renamed
Renaming testfolder1\testfolder1\filetest2.txt to testfolder1\testfolder1\filenew2.txt
Renamed
Finished
```

The new folder structure would be the one shown in Figure 2.

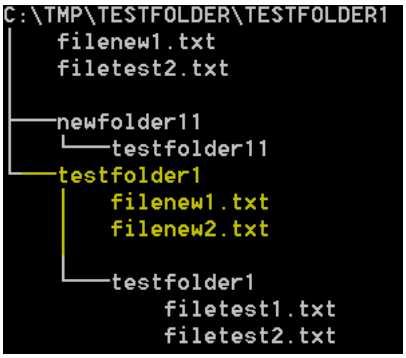

Figure 2. Folder structure after executing Command 6

Additionally, the argument -m or --mode is only affecting how namemod will treat the old_name_segment positional argument.

The -n or –noop argument

The user has the option to see first what the namemod will do before deciding to actually do it. This is the job of the -n or --noop argument. If the user ran Command 4 *(i.e. but renaming filetest2.txt to filenew2.txt)* with this argument *(see Command 7)*, the user will see the output like the one in Listing 4 but it is not actually done.

Command 7. Using -n argument

```
C:\tmp\testfolder>namemod -n testfolder1 filetest1.txt filenew1.txt
```

Listing 4. Output of Command 7

```
[FILE]  Rename testfolder1\filetest2.txt -> testfolder1\filenew2.txt
Finished
```

The output in Listing 4 is telling the user that if he/she executes Command 7 without -n argument it will rename the file specified in the output. This is also true for folder. However, instead of beginning the line with [FILE] it will then be [FOLDER].

The -i or --interactive argument

If the user's old_name_segment will match multiple files or folders and he/she wants to manually allow what files or folders must be renamed, the -i or --interactive argument will make namemod to ask the user before actually performing any rename. Using Figure 2 as in the example, if the user wanted to return back the rename made by Command 6 and allow him/her to decide. The user can execute the command in Command 8.

Command 8. Using -i argument

```
C:\tmp\testfolder>namemod -i -m regex testfolder1\testfolder1 new test
```

And the expected output would be the similar to the one found in Listing 5 *(i.e. the line numbers are not part of the actual output but they exist only for analysis.)*.

Listing 5. Output of Command 8

```
1  Are you sure you want to rename testfolder1\testfolder1\filenew1.txt to
   testfolder1\testfolder1\filetest1.txt? [Y/n]
2  Skipping: testfolder1\testfolder1\filenew1.txt
3  Are you sure you want to rename testfolder1\testfolder1\filenew2.txt to
   testfolder1\testfolder1\filetest2.txt? [Y/n]y
4  Renaming testfolder1\testfolder1\filenew2.txt to
   testfolder1\testfolder1\filetest2.txt
5  Renamed
6  Finished
```

Analysis of Listing 5

Let's analyse the output in Listing 5. If we compare the output from Listing 2 the lines that the user is not familiar with are the lines 1, 2 and 3. These lines are the result of using -i argument. Lines 2 and 3 are asking the user for confirmation. The user can respond by type y or n for yes and no respectively. For ease, the user has the option to just press enter for no response and only type y for all the files or folders that he/she wants to be renamed. Lastly, line 2 is the message if the user responded no to the question.

The exclusion arguments

A group of arguments exists in instructing namemod to automatically exclude some files or folders in its renaming activity *(i.e. the renaming activity here will affect multiple files)*. This group is known as exclusion arguments. The user will recognize these arguments by checking their prefix. Normally for the file exclusion that argument is starting with x and for the folder it starts with X. And for the expanded argument *(i.e. the argument starting with --)*, it is prefixed with exclude. Moreover, these arguments are not dependent to -m *(see The -m or --mode argument)* argument.

The -xt or --exclude-file-text argument

If the user wanted to exclude a specific file based on exact match, he/she can use the argument -xt followed by the exact filename.

The -xr or --exclude-file-regex argument

If the user wanted to exclude some files based on regex, he/she can use the argument -xr followed by the regex.

The -Xt or --exclude-folder-text argument

If the user wanted to exclude a specific folder based on exact match, he/she can use the argument -Xt followed by the exact folder.

The -Xr or --exclude-folder-regex argument

If the user wanted to exclude some folders based on regex, he/she can use the argument -Xr followed by the regex.

The -R or --recursive argument

The user has the option to instruct namemod to include subfolder in renaming. This can be done by using the -R or --recursive argument. Using Figure 3 as the folder structure, the user can user -R to rename all the test segments to new segments *(e.g. file**test**2.txt to file**new**2.txt)* of all files. This can be done by running the command specified in Command 9.

Figure 3. The updated folder structure

Command 9. Using -R argument

```
C:\tmp\testfolder>namemod -R -m regex testfolder1 test new
```

After running the command, the new folder structure should look like the one found in Figure 4.

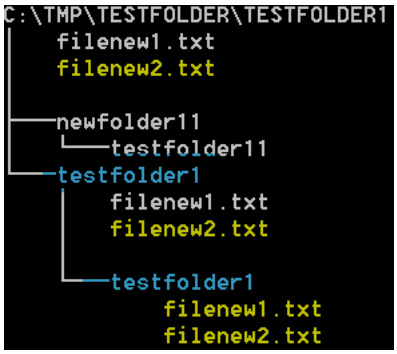

Figure 4. Resultant folder structure after running Command 9

If the user checked Figure 4 carefully, the rename is not limited to the level of C:\tmp\testfolder\testfolder1. But instead the rename reaches the subfolders. This is the effect of -R argument. Highlighted in blue are subfolders that have their contents modified by namemod. And the one highlighted in yellow are the files renamed. To validate the user can compare Figure 3 and Figure 4.

The -v or --version argument

The simplest of the namemod argument is -v or –version which its sole purpose is just to display its version.

Where to report any issues or improvements?

The user can create a ticket for any issues discovered or wish for improvements to the https://sourceforge.net/p/namemodifier/tickets address.