A1: Command-Line Interaction

Rename Utility

Build a command-line application that can rename files.

```
(c) 2021 Jeff Avery. Revised: April 13, 2021.
Usage: rename [-option argument1 argument2 ...]

Options:
-f|file [filename]          :: file(s) to change.
-p|prefix [string]          :: rename [filename] so that it starts with [string].
-s|suffix [string]          :: rename [filename] so that it ends with [string].
-r|replace [str1] [str2]    :: rename [filename] by replacing all instances of [str1] with [str2].
-h|help                     :: print out this help and exit the program.
```

The program is invoked by 'rename' and the required options ('rename -help' will display the above). To operate successfully, it requires a filename, and at least one option describing an action to take on that file.

Command-Line Options

Options are a list of strings provided on the command-line that tell the program how to execute. Options always start with a dash ("-") followed by a unique letter or keyword. The following options are supported:

Option	Flag	Alt flag	Values	Description
Filename	-f	-flag	filename	Specify the filename of the file to rename. Multiple filenames may be provided if they are space-separated.
Prefix	-p	-prefix	string	Rename a file by prepending a string to the filename.
Suffix	-s	-suffix	string	Rename a file by appending a string to the filename.
Replace	-r	-replace	string1 string2	Rename a file by replacing string1 with string2 in the filename.
Help	-h	-help		Display the help, including your copyright and usage details, and exit. Specifying -help will cause other options to be ignored.

Multiple options can be provided, and they should be applied from left to right in the order specified. e.g. "rename -prefix a -replace abc def" would apply the prefix before replacing strings. The exception to this is `-help'. If '-help' is provided as an option, all other arguments will be ignored, the help text will be displayed and the program will exit.

Options often require additional values, which must be immediately specified after the option (e.g. -prefix requires a string to operate).

- "-help" is the only option that does not require any additional values to be provided.
- Values cannot start with a dash ("-"), since that symbol is used to designate options.

Values may consist of:

- a string, or
- @date which should be replaced by your program with the current date as a string (MM-DD-YYYY format), or
- @time which should be replaced by your program with the current time as a string (HH-MM-SS format).

Filenames

The `file' option can support multiple arguments, representing multiple filenames. If more than a single filename is passed, it should be assumed that the operations are performed on all files.

A valid filename argument can consist of:

- a string representing a single file, or
- multiple strings representing multiple files, separated by spaces,
- multiple files specified using using an asterisk (*) or question mark (?) as wildcards.

Note on wildcards: Under the bash shell on Unix, the wildcards will be expanded by the shell, and the filenames will be passed as separate strings to your program automatically. You MUST test using bash on Unix, since Windows (and bash under Windows) does not behave the same way.

If no path is qualified for your filename, your program should assume that the files exist in the current directory.

Examples

Valid examples (given the existence of files named `f1' and `f2'):

- java rename -prefix a -file f1 —> af1

- java rename -prefix a -suffix b -file f1 —> af1b
- java rename -prefix a -suffix b -file f? —> af1b af2b
- java rename -prefix y y _ -file f1 —> yy_f1
- java rename -prefix @date _ -file f1 —> 08-26-2019_f1
- java rename -replace f abcd -file f1 —> abcd1
- java rename -prefix a -prefix b -file f1 —> baf1
- java rename -suffix a b -file f1 —> f1ab

Invalid examples:

- java rename f1 // no options provided
- java rename -prefix a // no filename provided
- java rename -prefix a -suffix b f1 // no filename provided (`f1' interpreted as argument)
- java rename -prefix a -suffix b -file f3 // file f3 does not exist

User Feedback

If a rename is successful, you should echo the operations that are performed so that the user has immediate feedback (e.g. "renaming `file1' to `file2'). In the case of multiple files, you would repeat for each file affected. [note: there are marks for aesthetics, so make this as flashy as you want!]

In the case of an error, you should provide a specific error message to inform the user and suggest a course of action. Displaying the online help is NOT sufficient for full marks: you must provide a specific, actionable message to the user.

A suggested list includes the following errors. You are expected to identify and address others for full marks.

- No filename provided
- Filename provided, but no option specified
- Filename provided but file does not exist
- Invalid option specified
- Valid option, but required value is missing
- Valid option, value is provided, but unrecognized

If you are renaming more than one file, and one of the file operations fail, it is acceptable to process some of the files, and report the files that didn't process [note: this is an exception to general behaviour, simply because it's difficult to predict if a rename will fail ahead of time].

Technical Requirements

- You must use Java 15 to compile this code.
- You may use classes from the JDK, and samples provided in the public repo. **You cannot use any other source code without explicit permission from the instructor. Failing to abide by this policy will be considered an Academic Integrity violation, with associated penalties.**

Submission

You must submit the following:

- your source code for this assignment,
- a `makefile' that supports building your source code (`make build'), and
- a `readme.md' file that includes details required to help the TA grade your assignment, and
- a bash script that allows your program to be invoked as described above (i.e. the user should NOT need to type "java rename" to invoke your program).

The directory structure for your assignment should look similar to thisthis:

```
a1/
├── Rename.java
├── makefile
├── readme.md
└──rename
```

Your readme file needs to include, at a minimum, your name, student number, the version of Java that you used (from `java -version'), and your operating system. If the TA needs to know anything else to run your assignment (e.g. undocumented hotkeys), include them in this file. For example:

```
Jeff Avery
12345678 j2avery
openjdk version "15.0.2" 2021-01-19
macOS 11.2.3 (MacBook Pro 2019)
```

Assessment

Your submission will be assessed roughly as follows:

5%
Complete submission including script, `makefile' and `readme'. Code compiles and runs.

20%
Correctly parse and identify options and arguments.
20%
Provide feeback on incorrect options and arguments.
15%
Chain multiple operations in the correct order.
30%
Rename files correctly using the operations above.
10%
Support for wildcards and multiple arguments.

Versions

April 12, 2021. 1.0 Initial release.

CS 349 User Interfaces (Spring 2021)

Cheriton School of Computer Science

University of Waterloo