Document Chaos

Hiring Problem Task

Procedure

Welcome dear Senior Programmer!
We are Black Forest IT GmbH, a German software company with a special focus on high-quality software development.

We are pleased that you are conducting our application process today.

Please note:

1. You have as much time as you need.
2. You do not have to complete the task if you do not want to. You can stop the exercise at any time.
3. Please track your time on this exercise and hand in your time logs as a CSV file.
4. Please track your code versions using git and enclose the .git folder with the code files you will hand in.
5. Feel free to choose an appropriate answer to any questions that may arise during this exercise that this document cannot answer.

We wish you a lot of success and look forward to a possible cooperation!

The Task

Peter needs a little tool, that helps him organise his huge amount of files.

"Wouldn't it be great, if I had a little tool" he starts to think, "that would rename my files using the schemes I already use.". Elaborating his idea further he mumbles "If I download a new Amazon bill for instance, it should fire up, ask me some predefined questions and rename the file accordingly. That would actually be amazing!".

Peter decides to pursue his idea, allocates a budget and tasks you with the programming part. His software designer has created a more or less formal software description.

Software Description

1. The name of the tool shall be `ffret`.

2. The tool shall offer a CLI with the following options and arguments:

```
1. CALL: ./ffret [<options>]

  Options:
    -c <CONFIGFILE>,--config=<CONFIGFILE>    The path of the config file.
                                             Default value is: $HOME
    -p <PPATH>,--plugins=<PPATH>             The path to the plugin folder.
    -h,--help                                Show this help.
```

3. When started the tool shall read the configuration file at CONFIGFILE.
4. Afterwards the tool shall try to match each file or folder in WORKDIR to a rule in the configuration file, that is find a rule where the RULE_MATCHER_SECTION evaluates to true. If no rule matches it shall ignore the file or folder that did not match.
5. The tool shall rename each file or folder that matched a rule by evaluating the FILENAME_SECTION of the matching rule and using the return value as the new filename.

The configuration file

The configuration file consists of one or more folder blocks. An example configuration file with one folder block looks like:

.ffret.txt

```
1. Downloads Folder (~/Downloads:recursive):
2.   - Amazon Invoice (amazon/amazon_invoice):[invoice_date(yymmdd) |
date_created(yymmdd)]_[title | length(3, 'back')].[ext | lowercase]
3.   - Amazon Comment (!amazon/amazon_invoice && freddy/amazon):
[date_created(yymmdd)]_[title | length(3)].[ext | lowercase]
4.   - Photos (me/photos): [date_shot(yymmdd) |
date_created(yymmdd)]_[sequence_number(3)].[ext | lowercase]
```

Line 1 is the start of a new folder block. A folder block consists of three sections:

```
{FOLDER_BLOCK_NAME} ({FOLDER_BLOCK_PATHS}[;recursive])
{RULE_SECTION}
```

1. FOLDER_BLOCK_NAME: The folder block name is any alphanumerical string at the beginning of the line. The folder block name shall be trimmed (remove left and right whitespace).
2. FOLDER_BLOCK_PATHS: A comma separated list of folder paths, that define all the places, that `ffret` shall use as a WORKDIR. If the optional `;recursive` string exists, than ffret shall add each subfolder of any WORKDIR to the list of FOLDER_BLOCK_PATHS after renaming them properly.
3. RULE_SECTION: Each line of the rule section defines a new rule. A rule must be preceded by two spaces, a dash and another space (" - "). A rule itself consists of three sections:

   ```
   {RULE_NAME_SECTION} ({RULE_MATCHER_SECTION}): {FILENAME_SECTION}
   ```

   3.1. RULE_NAME_SECTION: See FOLDER_BLOCK_NAME above.

   3.2. RULE_MATCHER_SECTION: The rule matcher section is a boolean expression. built by using round brackets, `plugin namespaces`, `not` (operator), `and` (operator) and `or` (operator) where the order in this text defines the precedence (so brackets have the highest precedence).
   In order to evaluate the boolean expression ffret evaluates the 'matches' plugin of the PLUGIN NAMESPACES defined in the expression. 'matches' plugin must return a boolean value. E.g:

   In line three of `.ffret.txt` above the boolean expression means "TRUE if `amazon/amazon_invoice/matches` returns FALSE and `freddy/amazon/matches` returns TRUE. FALSE otherwise".
   `amazon/amazon_invoice` and `freddy/amazon` are PLUGIN NAMESPACES of the matches plugin of `ffret`. See PLUGIN CALLS `section below.`

   3.3. FILENAME_SECTION: The filename section defines the final filename by using a pattern of plugin calls (See PLUGIN CALLS section) in brackets and direct strings. A pattern of plugin calls is a bar-separated ('|') list of plugin calls in brackets. E.g:

   ```
   [title | length(3, back)]
   ```

   Is a call to 'core/title' and than a call to 'core/length(3, 'back')'.

3.4.PLUGIN CALLS: A plugin call is made of:

[{PLUGIN_NAMESPACE}/]{PLUGIN_NAME}({PLUGIN_ARGS})

3.4.1.OPTIONAL, PLUGIN_NAMESPACE: The namespace of the plugin to call. If no PLUGIN_NAMESPACE is provided the value of the last plugin call shall be assumed. If there was no previous plugin call, than the default value of 'core' shall be assumed. Each namespace hierarchy has to be seperated by a slash. So 'my/new/namespace' references a three levels deep nested namespace.

3.4.2.PLUGIN_NAME: The name of the plugin that exists in the PLUGIN_NAMESPACE.

3.4.3.PLUGIN_ARGS: A colon seprated list of strings or numbers that shall be passed as arguments to the plugin by `ffret`. Strings shall be enclosed in single quotes. A colon INSIDE a string should not break the string into two arguments.

Plugins

`ffret` calls plugins by resolving their PLUGIN NAMESPACE into subfolders (of `ffrets` plugin folder) and a filename. E.g. the namespace 'my/amazon' is resolved to '<`ffrets` plugin folder>/my/amazon.php' file (e.g. a plugin written in PHP).
A PLUGIN NAME refers to a method or function inside that plugin file.

A plugin method or function receives three arguments from `ffret`:
1. A path to the file or folder that `ffret` is currently working on.
2. A list of arguments as defined in the configuration file.
3. An optional return value of another PLUGIN CALL that directly precedes this call in a pattern of plugin calls (See FILENAME_SECTION).

Your task

Your task is to develop `ffret` using any technology you want (Python, Javascript, PHP, C++, etc.) and also a default plugin named 'core' wich has methods to access all information that the system call 'stats' offers (see the `struct stat` here https://linux.die.net/man/2/stat).