CmpE 230 Spring 2017

Project II

Mustafa Enes ÇAKIR **2013400105**

Problem

I think the hardest part of the project is hash of directories. You have to calculate hash of everything in directory. Other parts are straightforward. Traverses directories, hash everything, keep them with paths and process paths that have same hashes.

My Code

I wrote all of my codes in one file imperatively. My file has 2 main section: `# Main #` and `# Methods #`. I divided my main flow to functions. **Detailed explanations of my methods are written in my code via comments**.

I will briefly mention the architecture of my algorithm. I called my methods that manages a general flow of my code at the `# Main #` part that at the end of the file. First of all, I get arguments from command line and parse them with `parse_arguments()`. End of my arguments list might have a list of pattern and directories to traverse. I checks first element of the `extras` list. If it has `"` as a first character, it is a pattern. Remove quotes and return it as `regex`. Rest of the `extras` list is directories. If the first character of path is `/` it's a absolute path don't do anything. If it's not, add current directory path to relative path. If no path is given, add current directory to `dirlist` global variable. I traverse `dirlist` with `hash_all()` method and calculate hash of everything that match with criterias. It keeps hashes associated with paths in `{"path":"hash"}` dictionary structure. After the traversing all directories I reverse my `hash_of_paths` to `hash_dict` with `reverse_path_hash_dict()` method. `hash_dict` keeps hashes like `{"hash": [list of paths,]}` dictionary. So if the `list of paths` has more than one

elements, it's duplicated. Print or execute command given command. But it's has a tricky part. Files and directory names might have spaces and quotes. We need to handle it. I make safe to use path with `replace_escapes(path:str)` method.

```
## METHODS ##
parse_arguments() -> list
handle_extra_arguments() -> str, list
hash_all() -> list
hash_file(path:str) -> str
hash_dir(path:str) -> str
reverse_path_hash_dict() -> dict
print_paths()
execute_commands()
replace_escapes(path:str) -> str
```