CMPE230

20.05.2020

EGE CAN KAYA - 2018400018

CMPE230 SYSTEMS PROGRAMMING HOMEWORK 2

Problem description: In this assignment we were asked to write a Python program which would traverse the directories given as command line arguments and look for duplicate files and directories and print such files in blocks as the output. We were also asked to implement optional command line arguments.

Problem solution: In order to solve this problem, I first defined three classes: File, Directory and Block.

- The File class: Holds the name, full path, and the size of a system file.
- The Directory class: Holds the name, full path of a system directory. Then calculates its size using Python's built-in walk function in the os module, by recursively summing the size of each of its subdirectories and each file it contains.
- The Block class: The output of the program is the full paths of all the duplicate files/directories in blocks of text. The Block class holds these blocks of text and also the common size of all the duplicate files/directories, which is used if the -s flag is given.

Then I made use of one of Python's built-in modules, namely, argparse to define what kind of an input is accepted. I defined the -f and -d flags to be mutually exclusive and also the -c, -n and -cn flags to be mutually exclusive optional flags. The -s flag is also optional. Finally an input of one or more directories may or may not be given.

The program then parses the given input and sets the default flags if needed. (-f and -c if none given in their respective mutual exclusion groups). If no directories are given, the current working directory is chosen as the input. Also, if any of the directories are given as relative paths, they are replaced with their absolute paths counterparts.

The program then proceeds to place every subdirectory and subfile of the given directories into respective lists. It instantiates appropriate File and Directory objects before doing so. These lists will then be used to iterate over and get the hash value of each subfile and subdirectory.

If the -f flag is present, only the elements of the list holding the Files will be hashed. The process of hashing uses sha256 as a base, but what is actually hashed depends on the flags given:

- If the -c flag is given, the file is read and its contents are hashed.
- If the -n flag is given, the name of the file is hashed.
- If the -cn flag is given, the name of the file is concatenated to the end of the contents of the file, then the whole string is hashed.

Regardless, the hash value and the file itself are mapped into a defaultdict data structure: hashed_elems, basically a hash map.

If the -d flag is present, a special function named hash_directory directs the process of hashing:

- If the directory is empty, it gets the sha256 hash of an empty string.
- Otherwise, the hashes of the subfiles and the subdirectories of the directory are first calculated. Then they are sorted in alphabetical order and concatenated to get a hash-0 value. The hash of the directory is the sha256 hash of this hash-0 value.

Then, the rest of the process is carried out in the same manner as the -f flag.

In the final portion of the program, the defaultdict hashed_elems is traversed to see if any of the hash values correspond to multiple files or directories. In such a case, the full path of these files or directories are first sorted in alphabetical manner, then the blocks are sorted between themselves, alphabetically by default or by their size if the -s flag is present. Finally, the blocks of text are printed to the output stream.