Project 4: Indexer

Jimmy Le[1], Carlos Dominguez[2]
Rutgers University
[1]jl1415@rutgers.edu
[2]cmd363@rutgers.edu

Description

Our implementation of the Indexer consists of three components working in conjunction. The entries of tokens returned by our indexer are stored in memory as a massive linked list, which is kept sorted and alphabetized.

The Indexer runs in $O(n^2)$ time–it runs through the input from the user linearly, iterates over the linked list stored in memory to ensure that new entries are placed in the proper place, and then writes each entry to file.

Tokenizer

Finds the discrete tokens in a given file and stores them into a linked list. Our improved tokenizer file has been stripped of features unnecessary for the indexer, such as the C keyword detection system and the token 'type' identifier. Much of the algorithm had to be rewritten in order to accommodate these changes, but the end result is a modular, minimalistic library suitable for use in larger programs like our indexer.

We have some new functionality in our tokenizer, such as converting characters into their hexadecimal/octodecimal equivalents, and checking whether or not a given token is an octodecimal string.

index.h

This file contains the function definitions required to call the Tokenizer on a given file or directory. It is also responsible for placing new entries in alphabetical order.

evalFile()

Called upon every file that is found in the input directories, this function uses the Tokenizer to tokenize a file and then stores every valid token in a linked list. When finding a token for the first time, an entry in the list is initialized with the token, the filename it is found in, and the number of times it is found in the file (initially 1).

If this token is found again, the entry is instead updated, and is done so in two cases:

1. If the token is found again in the same file, the number of occurrences is incremented
2. If the token is found again in a different file, a new sub-entry is created, with the same token but with this current filename and the number of occurrences in this file

writeFile()

Taking the entire linked list of tokens created by running evalFile(), writeFile() writes the entire list into file.

index.c

Processes the input from the user specifying where the list of tokens should be saved to, as well as the directory or file to tokenize. It then calls the functions found in index.h on this input, which is able to recursively tokenize all files in any subdirectories in an inputted directory.

After the files have been tokenized and stored into memory, writeFile() is called, saving the entire output to local storage. The monolithic linked list of entries is then freed.