Indexer

CS214 Project 3

Contains :

- indexer.c
- tokenizer.c , tokenizer.h
- path.c , path.h
- sorting-tokens.c , sorting-tokens.h
- hash.c , hash.h
- Makefile

Command line:

```
make

./index  <inverted-index file name> <directory or file name>

make clean
```

Indexer.c contains main() and creates two structures, a hash table of size HASH_SIZE (or 100) and a linked list. The hashtable is used to map tokens and the linked list is used to store new tokens in alphabetical order. Note: the linked list structure 'Node' is defined in sorting-tokens.h

tokenizer.c has predefined separators, "!#$%&` ()*+'-./:;<=>?@[]^_{|}~,\n\r\f\a\\\?\'\"\t\v\b" that it uses to tokenize a string.

Path.c has functions that handle directory or file names given by the user. Index_file() is used to tokenize and give the tokens as arguments for the function add_ Token(), which also takes a hashtable, a linked list, and the file name in which the token was found.

Hash.c is where some of the grunt work is done. The running time of function add_Token(x) , x a token, is O(n). Add_Token() handles new and old tokens. It handles new tokens by creating a new Token object with which to map. It also handles old tokens by calling the update(), which finds the token, increments frequency, removes(), and inserts().

The running time of finding a token in the hashtable is O(n), n is the number of tokens with the same key.

The write_file() is called in main after all the files have been parsed and mapped. If x different tokens were mapped than the linked list created in main() should have x 'Nodes' in alphabetical order. Write_file() traverses the linked list and for each 'Node', which contains a token name, it calls the find_token() defined in hash.h. find_token() essentially returns the "address" of yet another list of length y. At most, y can be the number of all the files parsed.