Chris Fretz
Professor Parashar
Systems Programming
10/20/14

Indexer

- System Overview
  - Project uses following data structures:
    - Index Node
      - Keeps track of the count for a specific token/filename pair.
    - Hash
      - Basic hashtable that hashes linked lists of index nodes against tokens.
    - Hash Node
      - Node used for storing data inside of hash. Allows for resolution of hash-collisions through chaining.
    - Splitter
      - Tokenizer that allows a given file to be tokenized through the use of the split_next helper function.
  - Algorithm Overview:
    - Indexer allocates a single hash to store all data for current indexing operation.
    - Indexer iterates across all specified files, inserting and updating information in the hash as it goes.
    - After finishing all files, indexer iterates across the keys contained within the hash, and generates output.
- Runtime Analysis
  - For each file that must be indexed, the process can be broken down as such:
    - File is opened and passed to the tokenizer (declared in splitter.h), O(1).
    - Indexer loops until the entire file has been tokenized (calling function split_next O(1) to get each token), passing each individual token into the hash (consists of a get operation O(1), and an insert operation O(n)), and moves onto the next one.
    - Indexer iterates across all tokens contained within the hash, generating output for each file that contains the current token. Number of operations can be estimated by taking the number of unique tokens existing across all files and multiplying it by the average number of files mentioning any given unique token.
  - In general, the upper bound runtime of my program should be something like O(f * t) where f is the number of files, and t is the average number of unique tokens in each file.
- Memory Usage
  - Memory usage of each structure
    - index_node - 20 bytes.
      - Filename string, takes up 8 bytes locally, and usage in the heap is determined by the size of the filename.
      - Count integer, 4 bytes
      - Next pointer, 8 bytes.
    - hash_node - 24 bytes
      - Key string, takes up 8 bytes locally, and usage in the head is determined by the size of the key.
      - Data index node, 8 bytes locally, sizeof(index_node) bytes in the heap.
      - Next pointer, 8 bytes.

Chris Fretz
Professor Parashar
Systems Programming
10/20/14

    - hash - 16 bytes
        - hash_node array, takes up 8 bytes locally, and usage in the heap is determined by the current number of elements contained in the hash.
        - Count integer, 4 bytes.
        - Size integer, 4 bytes.
    - splitter - sizeof(FILE) + 16 bytes.
        - File struct, 8 bytes locally, sizeof(FILE) in the heap.
        - Line string, 8 bytes locally, size of the current line in the heap.
        - Remaining string, 8 bytes locally, points to same string as line string.
- In general, memory usage is based primarily on the number of files specified and the number of unique tokens in each file.