Project 4 Writeup CS214

David DeSimone

November 10, 2014

1 Data Structures

As we are dealing with lists of strings that are variable in size, we implement a **linked list** for string storage. To do this, we declare a type str_arr_s, which is a linked list for types str_link_s, which are a simple wrapper for a string.

To handle the indexer, we use a linked-list of terms, with each item in the linked-list itself pointing to another linked list which resolves the files that it belongs to. While insertion/retrieval is slower due to this choice, I have chosen this out of concern for memory constraints.

2 Design and Exception Handling

The search program is built into several functions. In main, we loop over input for input in the form so ¡terms¿ or sa ¡terms¿. Once we recognize such an input, we call our peel() function to tokenize the input terms, and insert them into our linked list data structures. From here, we call so() or sa() on our linked list data structure to return a list of relevant files. This list is printed by our print function. At any time, the user can enter "q" (without quotes) to exit the program.

Exception handling includes: Non-termination on invalid/NULL user input, accomplished by various strcmps and NULL checks. Reporting of an invalid file format, accomplished by various logic checks in the parsing of the file.

3 Memory Space Usage

The program makes the expectation that it can load the entire file(s) into memory at run time. If n is the number of input words, and m is the number of files, our worst case memory space usage would be O(nm), as each word could be contained in every file. The searcher only adds linked-lists that are bounded by this number, and thus they do not add to the asymptotic running time.

4 File Format

Files are in the format
¡list¿ word
file freq file freq file freq...

5 Make File Usage

"make" Makes the search utility
"make indexer" Makes the indexer utility
"make clean" Removes files