

## Assignment 13 & 14: Indexing

### Complete assignment due: Wed., December 10th, 9:00 p.m.

Note: This is a double assignment, as defined in the course syllabus. The grade you receive for this assignment will be entered twice in the grade spreadsheet for both assignments 13 and 14. If your grade on this assignment is the lowest of your assignment grades, then the grade entered for assignment 13 will be dropped, but the grade entered for assignment 14 will still count.

Write a C program that creates an index of words and pages that appear in each document. The command line will be:

```
index [-1] dictionary-file [ book-file ...]
```

Note the option is the letter **1**, not the number one.

For each book-file, **index** will print an alphabetical list of words from the dictionary-file that appear in the book. For each word, print three lines:

- the word
- three blank spaces, "**count:** " and the number of times the word appears in the file
- three blank spaces, "**pages:** " and the number of each page on which the word appears at least once.

For example:

```
lectura-> index dictionary04.txt book04a.txt book04b.txt
```

```
File: book04a.txt
```

```
alpha:
```

```
count: 18
```

```
pages: 1
```

```
bravo:
```

```
count: 18
```

```
pages: 1
```

```
charlie:
```

```
count: 18
```

```
pages: 1
```

```
delta:
```

```
count: 18
```

```
pages: 1
```

```
echo:
```

```
count: 1
```

```
pages: 2
```

```
File: book04b.txt
```

```
alpha:
```

```
count: 8
```

```
pages: 1 3
```

```
charlie:
  count: 8
  pages: 1 3

echo:
  count: 8
  pages: 1 3

lectura->
```

## Linked Lists:

Use a set of 26 linked lists for the dictionary. Each linked list will contain the words that start with the same letter of the alphabet. The dictionary files that we will use will only contain lower-case letters.

Each word will need a linked list of the page numbers on which the word appears.

When processing the book files, use **strcasecmp** to determine if each word is, or is not, present in the dictionary.

## Library:

I will provide a compiled set of four C functions in a library named *liblexWrap.a*. I will also provide a header file named *lexWrap.h*. The header file contains the prototypes for the four functions. The functions provide an interface to a flex file. The flex file will find words within an input file.

- To use the input file, use **fopen** on the file name to create a **FILE \***. Call the function **setFile**, passing the **FILE \*** that you created.
- To get words from the file, repeatedly call the function **getWord**. **getWord** takes no arguments, and returns a **char \***. The **char \*** will point to a fixed (not dynamically allocated) character buffer that contains one word. Use **malloc** to create space for this string and **strcpy** it to the **malloc**'d space. **getWord** will return **NULL** when it encounters end-of-file.
- Call **getLineCount()** to get the line number on which the word was found. **getLineCount** takes no arguments and returns an **int**.
- When done with the input file, call the function **destroy**. **destroy** takes no arguments and returns nothing. It tells the flex code to de-allocate all the dynamic memory that the flex file created. You will then need to use **fclose** to close the file.

The compiled version of these functions are available in a library named **liblexWrap.a**. The library is located in **/home/cs352/fall08/assign13**. To compile your program with this function, you will add the following two items to the end of the compilation command:

```
-L/home/cs352/fall08/assign13 -llexWrap
```

The **-L** option specifies a non-standard directory where the compiler will search for libraries. The standard locations are */usr/lib* and */usr/local/lib*. The **-l** option specifies a non-standard library to be searched. (Note that is the letter **l** (“ell”), not the number **1**.) The library name will start with **lib** and end with **.a**. Thus, to use one of the math functions you would search the math library, which is named *libm.a*. You would add **-lm** when compiling. For this assignment, the library is named *liblexWrap.a*; thus, you will use **-llexWrap**.

The **-L** part can appear anywhere on the command-line, so long as it occurs before the **-l**. The **-l** part must appear *after* the **.c** files which need functions from that library. This requirement means that most command lines end up with all the **-l** options listed at the end of the line. Note it is possible to specify any number of libraries on the command line; there is no upper limit.

There is a fifth function, **setLexDebug**. In the provided *index.c*, the **-l** command-line option will invoke this function with a value of **1**. This will turn on debugging output in the provided *lexWrap* library. Every time that you call **getWord()**, the library will print both the word and the line number on which it was found.

### Include file:

There is a provided *lexWrap.h* file that contains the prototypes for the five functions: **setLexDebug**, **setFile**, **getWord**, **getLineCount**, and **destroy**. To include this in your code without having to put the full path name in the include statement, use the **-I** option when compiling. The line will then become:

```
-I/home/cs352/fall08/assign13
```

This goes on the compiler's command-line and needs to appear before any file that needs to include *lexWrap.h*. For example, if you have a C file named *dictionary.c* that needs to use one or more of the functions in the library, then to compile *dictionary.c* and produce a *dictionary.o* file, you would use:

```
gcc -Wall -g -I/home/cs352/fall08/assign13 -c dictionary.c
```

The include command in *dictionary.c* then becomes:

```
#include <lexWrap.h>
```

### Makefile:

You are to use multiple **.c** files in writing your programs, and at least one **.h** file. The **.h** file should contain constant and struct definitions, and function prototypes. It can contain additional items, as appropriate to your solution.

You are required to use the provided *liblexWrap.a* library. Your *Makefile* should get the library from the **~cs352/fall08/assign13** directory. The *lexWrap.h* file should be included in your **.c** files where needed. The *Makefile* should define the **~cs352/fall08/assign13** as the search location for the *lexWrap.h* file. Note that the *lexWrap.h* file does not count as the minimum required **.h** file specified in the first paragraph of this section.

Provide a *Makefile* that supports incremental compilation of your files. The *Makefile* will support the target: **make index**. This will produce an executable named **index**. The *Makefile* will also support the target: **make clean**. This will remove all **.o** files created by your program.

A sample *Makefile* is provided to help you get started.

### **Functions:**

You should use functions for this program. It should be possible for us to easily understand your program from its organization into functions and the comments you provide. Your organization does not have to match the one used in our solution. If you need help in figuring out a reasonable division among functions, please get in touch with us!

**Turnin:** Use **turnin** to turn in all the files for your program. This should include the **Makefile** and all the **.c** and **.h** files. The command is:

```
turnin 352assign13 Makefile ...
```

See the man page for the **turnin** program for details on what **turnin** can do and how you can confirm that your files were turned in.