CSc 352 (Summer03) Homework 4

Start: 06/30/03
Due: 07/14/03
Turnin ID: cs352_assg4

Turn in file list: **my_wc.c, my_grep.c**

0. Background Reading

- fscan( )
- fgets( )
- fgetc( )
- fopen( )
- fclose( )
- lstat( )
- getopt( )
- regcomp( )
- regex( )
- tolower( )
- isspace( )
- strcat( )
- strdup( )
- memset( )
- realloc( )

1. my_wc (Word Count)

Write your own version of "wc" utility tool. The command line of **my_wc** is quite similar to **wc**.

Usage: my_wc [-lwcL] [FILES]

You need to handle very flexible command line options, including:

-c Count bytes.
-l Count lines.
-w Count words delimited by white space characters or new line characters.
-L The number of characters of the longest line of the file.

Options in the command line may be written very flexible. For example, the following commands are all correct and they are all actually equal to each other.

my_wc –wcL text1 text2
my_wc –wc –L text1 text2
my_wc –L –w text1 –c text2

When no **–w**, **-c**, **-l** or **–L** are specified, by default, for each file, the number of lines, words, characters and filename are printed out in order in a single row of the output. If any combination of the above 4 options are specified, the correspondent information are always printed out in the order: number of lines, number of words, number of characters and filename.

When no filename is specified in the command line, **my_wc** should read the text to be processed from the **stdin** until **EOF** is received.

If the command line syntax is wrong, you need to print out the usage of **my_wc** (listed above).

The maximum number of filenames allowed to appear in a single command line is **100**. If more than that, you should print the error message "Too many files\n" to the **stderr** and exit the program with return value 1.

If any file specified in the command line is not a regular file (e.g. a directory, etc.), you need to print the error message "%s: Not regular file\n" (where %s is the filename) to the **stderr** and continue to process the next file. But the return value of the program should be 1 instead of 0.

If any file specified in the command line can not be found, you need to print the error message "%s: No such file or directory\n" (where %s is the filename) to the **stderr** and continue to process the next file. But the return value of the program should be 1 instead of 0.

2. my_grep (Search Tool)

Write your own version of "**grep**" utility tool. The command line of **my_grep** is quite similar to **grep**.

Usage: my_grep [-chilnw] reg_expr [FILES]

You need to handle ver flexible command line options, including:

-c Print only a count of the lines that contain the pattern.
-h Prevents the name of the file containing the matching line from being appended to that line. Used when searching multiple files.
-i Ignore upper/lower case distinction during comparisons.
-l Print only the names of files with matching lines, separated by NEWLINE characters. Does not repeat the names of files when the

pattern is found more than once.

-n Precede each line by its line number in the file (first line is 1).

-w Search for the expression as a word.

The flexible style of the command line options are similar to that of **my_wc** explained above.

When specifying different command line options, the result output may be different. But information should always be printed in the following order: filename, number of lines matched, line number and the line of text matching the regular expression.

When no filename is specified in the command line, **my_grep** should read the text to be processed from the **stdin** until **EOF** is received.

If the command syntax is wrong, you need to print out the usage of **my_grep** (listed above).

The maximum number of filenames allowed to appear in a single command line is **100**. If more than that, you should print the error message "Too many files\n" to the **stderr** and exit the program with return value 1.

If any file specified in the command line is not a regular file (e.g. a directory, etc.), you need to print the error message "%s: Not regular file\n" (where %s is the filename) to the **stderr** and continue to process the next file. But the return value of the program should be 1 instead of 0.

If any file specified in the command line can not be found, you need to print the error message "%s: No such file or directory\n" (where %s is the filename) to the **stderr** and continue to process the next file. But the return value of the program should be 1 instead of 0.

If no regular expression is specified, print out the error message "Regular expression missing\n" to the **stderr**; if the regular expression is wrong, print out the error message "Wrong regular expression\n" to the **stderr**. In both cases, exit the program with return value 1.

If there is not a single line (of all the files or the text from the **stdin**) that matches the regular expression, the return value of the program should be 1 instead of 0.

Provided executables can be found in the class home directory **/home/cs352/summer03/assignments/hw4**