

CMSC 212

Project #2

Due 02/27/2007 8:00 PM

Background

This project will give you practice in command line arguments, both file and standard I/O, managing structures, and parsing strings.

The Game of MadLibs

You can find out more about how to play the game in general from <http://us.penguinroup.com/static/packages/us/yreaders/madlibs/fun.html>. This is a game that is fun and helps children learn about the different parts of speech (nouns, verbs, numbers, adjectives and adverbs). A paragraph is provided with some words removed. In place of the word – the paragraph indicates which of the above listed parts of speech should be inserted there. One player does not see the paragraph, but is asked to supply the missing words strictly based on the part of speech they qualify as. The other player then reads the paragraph with the words inserted. They are sometimes quite humorous because not every noun fits equally well into every sentence.

Your Implementation of the Game

Your game will get the Mad Libs Paragraph from a file; the file has the words to make the story with numbers to indicate where words from the other player need to be inserted. The name of the file to be used is passed to your program as a command line argument. The program will make one pass through that file recording what types of words are needed. It will then prompt for a single word stating the name for the part of speech needed. It will use standard output for this and then read the word typed using standard input. It will then make a second pass through the file named as the argument to print the entire paragraph with the words inserted (printing the funny story).

The parts of speech that will be needed to complete the Mad Lib will be represented by numbers (no other numeric digits will be in the file). Each part of speech has its own number (1=noun, 2=verb, 3=number, 4=adjective and 5=adverb). If any other numeric digit appears in the file, you should end the program with a message that says “Invalid number code” before any standard input is requested.

The Mad Lib data file will be an ASCII text file, and no line should be longer than 80 characters (including the new line character that ends the line). If there is a line longer than 80 characters in the file, the program should end with a message that says “Invalid line length” before any standard input is requested. The Mad Lib data file will be processed in two passes – the first one to find what words are needed (completed before any standard input is requested) and the second to print the story (after all standard input is completed).

When making the first pass of the Mad Lib data file, you will fill an array of structures with enough information so you know what words (of what types) will be needed in what order. Each element of this array will use the structure shown in the following text box. During the first pass, you will fill in the “code” portion of that element of the array with the part of speech needed, the “word” should be left as an empty string until the list of words needed are requested in the next step.

```
#define WMAX 20

typedef char Word[WMAX+1];
typedef struct{
    int code;
    Word inword;
}WTy;
```

There should not be more than 20 words requested in any one Mad Libs data file. If there are more than 20 requested in one Mad Libs file, the program should give the error message “Too many words needed” and end immediately without requesting any standard input.

Once you have completed the first pass of the Mad Lib file, you will use standard output and input to request and get each of the words needed (in the same order they are required in the Mad Lib file). Simply display the name for the part of speech requested followed by a colon, and then read the one line that is entered by the user. The line typed by the user should not be over 20 characters long. If the line given by the user is over 20 characters long, the program should end printing the message “Invalid word length” before any other input is requested. If there are any non-alphanumeric characters on the line of input they should be removed. For example, if the person enters “ 2 thousand” or “2-thousand” or “ 2 t h o u s a n d”, it should be stored and later printed as “2thousand” instead.

When printing the Mad Libs result to standard output, all lines should be exactly as they were in the input file except for the replacement of the corresponding word for the valid numeric digit. This means that some lines could extend over 80 characters in the output file (since multiple single digit numbers could have been replaced by words). The punctuation and white space that appeared in the original Mad Libs input file should also be printed. Only the numeric digits which represented the words to be inserted should be omitted. Those numeric digits will be replaced by the word typed into standard input (only with the modification described above).

The name of the Mad Libs file should be given as the one and only one argument. If there is no argument, the program should end with the error message “No arguments”. If there is more than one argument, the program should end with the error message “Too many arguments”. If the argument given is not a file that is readable by the user running the program, the program should end with the error message “Invalid argument”.

Additional Information

You may use any functions in the standard `<string.h>` library. The functions that we believe you will want to use from this library will be covered during discussion sections. You may use others, but they are not necessary.

You can not store the complete file (it must be processed in two passes) because you have no indication of how many lines will be in the Mad Libs file.

You may assume that the input file will not be changed between when you read it in the first pass and when you read it the second pass.

Your program should be entirely contained in the file `madlibs.c` which is included as an almost empty file with the files you will check out. The `madlibs.h`, a Makefile and the file to allow submit to work correctly will be provided. For this project, you should only modify the `madlibs.c` file in the version you submit. Because we are trying to make sure you are comfortable with parameter passing in C and because we do not want to tell you exactly what functions you should implement for this project – there is one other restriction in your design. No function (including the main) should be over 20 lines of code (since end of lines are not required in C – this means 20 statements where each one would take one line in good style). This project also has the purpose of giving you a chance to practice the C-string functions. In order to do this, try to use the functions who are already defined to manipulate strings as much as possible rather than using character by character traversal and a lot of extra loops. Especially cases line I/O and copying of strings should be done using functions rather than character by character.

You can obtain the files from the 212files directory link you created during the lab session from the file named `p2.tar.gz`. You will then need to uncompress and `untar` as you have done for previous exercises. You will also use the submit application as you have for previous exercises applying the `.submit` file in the directory created by the `untarring`.