Lab #3: readline

Overview

The C standard library does not provide a function to effectively read an entire "line" of arbitrary length. There is the `fgets` function, but it copies characters into a fixed buffer[1].

For this lab you will write a function to read a line of arbitrary length from a file stream. You will use this function in various assignments this quarter, so it behooves you to thoroughly test it now.

More specifically, your function should take a single argument of type **FILE *** and return a C-style string (**char ***). Your function will repeatedly read from the file stream (you may use `fgetc` or `getc`, or, if you would really like to, `fgets`) until either a newline or `EOF` is encountered. Once a newline or `EOF` is encountered, the "line" is returned (if there is no "line" in progress when `EOF` is reached, then return `NULL`). You will also likely find it convenient to remove the newline character, if present, from the "line" before returning it.

As your function reads, it will build a single string that holds all characters read. This will be done by initially allocating an array and then, once the array is filled, "growing" the array to hold new characters to be read (look into the `realloc` function).

You will find that the number of elements by which you "grow" the array will greatly affect performance; do not, for example, grow the array by only a single character at a time.

Demonstration

To demonstrate this function, write a small program that repeatedly reads, using your function, lines of arbitrary length (until `EOF`) and prints them back out. Be sure to properly `free` the memory used for a "line" when you are done with it.

You should use `valgrind` on this program to check your memory management.

[1] There is also `gets`, but it lacks basic array bounds protections.