Etude 13: Bug Squashing **Mathew Shields 2419874**

To the Junior Developer:

I've had a look at your code and noticed a few issues to be fixed. I have rewritten parts of it to make it more user-friendly and easier to follow. Here are some of the changes I made:

Methods

You had written several methods which I found weren't needed, or could be optimized. The methods I implemented get passed the array of client structs and the details to be found, and return the index of the client struct if there is a match. I also added in a print method to output the details of the found client. Lastly I added in emalloc() and freeAll() methods. The emalloc() method handles error checking when allocating memory, and the freeAll() frees any allocated memory. This method is very important for preventing memory loss, as all allocated memory must be free'd.

Main Method

I added in some error checking for when the program is run. This ensures that the user knows that 1 argument must be given to the program and that it must be a file. It also lets the user know if there is an issue reading from the file. I implemented a counter to count the number of clients in the client list, which is useful for iterating through all the clients.

Instead of using fscanf() to read in the data from the file, I opted to use fgets() and strtok(). fgets() allows you to set a limit for the amount of characters to be read in. strtok() is used to get each token from the string read in by fgets(). These tokens are stored in their own buffer variable, the length of which is used to allocate the memory to each of the buffer structs elements. The buffer variables are then copied into the buffer struct, which is then added to the array of all clients. The file is then closed after all it's data has been read.

When asking for input from the user, I again use fgets() and strtok(), and I opt to use characters for the switch method, rather than an ints. This gives the user a better idea of what each character is doing, as the character can represent the variable, e.g., f for first name. The input given is also checked for formatting errors, and the message explaining how the program works to the user is repeated if any occur.

Makefile

The makefile should contain options like -W -Wall -ansi -pedantic -g -O2 to help spot errors in your code that would not normally be picked up by the compiler. The >/dev/null should also be removed as this is outputting all the error messages into the void.

Other Advice or Small Issues

- I removed a repeated #include <stdio.h> and an unused #include <math.h>
- I spotted a typo in the emailAddress variable, be wary of this in the future as it can be difficult to deal with in large programs.
- You used magic numbers quite often in the code. These are numbers that don't really come from anywhere. These can be annoying to change so you are better off defining variables if necessary.
- Sometimes a '\n' new line character will have to be replaced with a '\0' null terminating character in order for strcmp() to work or to make sure the string is exactly how you expect it to be, rather than it containing a new line.
- When allocating memory for a variable containing a string, ensure you had 1 to the length of the string, to account for the '\0' null terminating character.
- In the future it may be easier for you and others to follow your code if you use variables and methods with more meaningful names. This way you know exactly what they are referring to.
- At the end of the program, I return EXIT_SUCCESS.

Remember, you are writing this program for users who may have no idea how it works, so it is important that it is clear how the program works. Any output should be easy to read with instructions on how input should be given. There should also be error handling for if the user provides bad or incorrect input.

I chose not to do it this way, but another way of implementing this program is by using binary search and a sorting algorithm to sort and then search the array of structs. This method would scale very well for large amounts of data.

All the best,
Mathew