

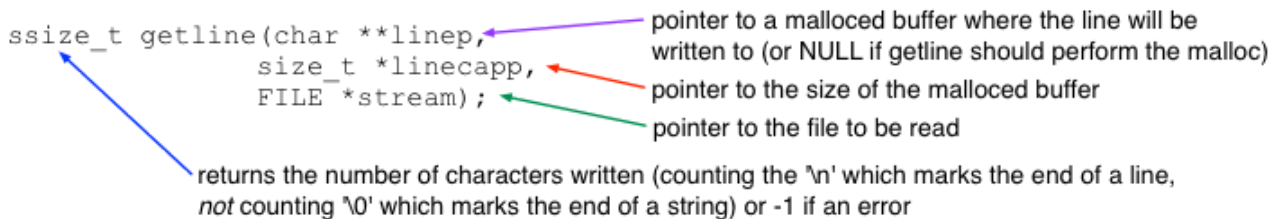
getline

getline

You have already seen `fgets`, which lets you read a string into a buffer that you have preallocated, specifying the maximum size for the string to read (i.e., how much space is in the buffer). However, how could you write code that would read a string of any length? Before this chapter, we have not had the tools to think about such a function—it clearly requires dynamic allocation as we would need that function to allocate memory which lasts after the function returns. Now, we can learn about **getline**.

getline is a C function available in the C Standard I/O Library (**#include <stdio.h>**). Its signature (shown below) looks a bit intimidating, but it is worth understanding.

```
ssize_t getline(char **linep,
                size_t *linecapp,
                FILE *stream);
```



pointer to a malloced buffer where the line will be written to (or NULL if getline should perform the malloc)

pointer to the size of the malloced buffer

pointer to the file to be read

returns the number of characters written (counting the '\n' which marks the end of a line, not counting '\0' which marks the end of a string) or -1 if an error

`getline` reads a single line from the file specified in its third argument, **stream**. It does this by reading characters from the file repeatedly until it sees the character '\n', which indicates the end of a line. As it reads each character, it copies the characters into a buffer in memory. After reading the newline character, `getline` places a '\0' character, which indicates the end of the string.

So far, this behavior sounds much like **fgets**; however, the difference is in the fact that `getline` allocates space for the string as needed. The difference arises from the fact that `getline` uses **malloc** and **realloc** to allocate/enlarge the buffer. The **linep** parameter points at a pointer. If ***linep** is NULL, `getline` mallocs a new buffer. If ***linep** is not NULL, `getline` uses that buffer (which is ***n** bytes long) to start with. If the initial buffer is not long enough, `getline` reallocs the buffer as needed. Whenever `getline` mallocs or reallocs the buffer, it updates ***n** to reflect the number of bytes allocated. It also updates ***linep** to point at the new buffer. When the `getline` function returns, ***linep** is a pointer to the string read from the file.

The `getline` function returns -1 on an error (including end of file), and the number of bytes read (not counting the null terminator byte) on success. Note that the return type is `ssize_t`, which stands for "signed `size_t`"—that is, the signed integer type which is the same number of bytes as `size_t` (so it can return -1).

The `getline` function can be used in two ways. First, the user can provide `getline` with a pointer to a malloced buffer ***linep**, whose size is pointed to by **linecapp**. If the line being read is larger than ***linecapp**, `getline` will perform a `realloc` for the user and modify the values of ***linep** (to point to the newly reallocated region) and ***linecapp** (to be the size of the newly re-allocated buffer) accordingly. Second, the user can provide `getline` no buffer, indicated by **linep** containing a pointer to `NULL` (**linep** cannot be `NULL`, but ***linep** can be `NULL`). In this case, `getline` will perform a `malloc` for the user and modify the values of ***linep** (to point to the newly malloced region) and ***linecapp** (to be the size of the newly allocated buffer) accordingly. These two ways can be used together—*i.e.*, one can write a loop where no buffer is provided the first time, and the same buffer is reused on subsequent iterations.

The next video shows an example of using `getline` to read lines from a file and print them out. The final video shows a slightly larger example, which combines `getline` and `realloc` to read all the lines from a file into an array. The example then sorts them (using **qsort**, which you saw in Course 3), prints out the results, and frees the memory appropriately.



Completed
