8.4.2. La función `getline`

La librería GNU ofrece la función no estándar `getline` que hace sencilla la lectura de líneas:

```
#include <stdio.h>
ssize_t getline(char **lineptr, size_t *n, FILE *stream);
```

Esta función lee una línea entera de `stream`, almacenando el texto (incluyendo el carácter de nueva línea y el de terminación) en un buffer y almacenando la dirección del buffer en `*lineptr`. Antes de llamar a `getline`, tienes que colocar en `*lineptr` la dirección de un buffer de `*n` bytes de largo usando `malloc`. Si este buffer no es lo suficientemente grande como para albergar la frase leída, `getline` hace el buffer más grande usando `realloc`, actualizando la nueva dirección en `*lineptr` y el tamaño en `*n`. Si inicializas `*lineptr` a `null` y `*n` a cero, `getline` hace ese `malloc` por tí.

Si todo ha ido bien, `*lineptr` es un `char *` que apunta al texto que se ha leido. La función devuelve el número de caracteres leídos aunque sin contar el carácter '\0' de terminación (sí con el de nueva línea); si ha habido algún error o se alcanza final de fichero, devuelve **-1**.

Nota

Siempre que tengas que leer una línea de texto del teclado, hazlo pues con `getline`; te evitarás cualquier problema posterior.

El siguiente programa muestra cómo usar `getline` para leer una línea de texto desde teclado de manera segura. Intenta teclear más de 10 caracteres, verás que `getline` lo gestiona correctamente, independientemente del número de caracteres que teclees.

```
1  #include <stdio.h>
2  #define TAM_MAXIMO 10
3
4  int main(void)
5  {
6    ssize_t bytes_leidos;
7    size_t numero_bytes;
8    //ssize_t y size_t son sinónimos de unsigned int
9    char *cadena;
10
11   puts("Por favor, introduce una línea de texto:\n");
12   /* Puedes pasar a getline los argumentos inicializados: */
13   //numero_bytes = TAM_MAXIMO;
14   //cadena = (char *) malloc (numero_bytes + 1);
15   //bytes_leidos = getline(&cadena, &numero_bytes, stdin);
16
17   /*O bien, más sencillo, poner el número a 0 y la cadena a NULL, para que él mismo te haga
18   la reserva necesaria*/
19   numero_bytes = 0;
20   cadena = NULL;
21   bytes_leidos = getline(&cadena, &numero_bytes, stdin);
```

```
22
23    if (bytes_leidos == -1)
24    {
25      puts("Error.");
26    }
27    else
28    {
29      puts("La línea es:");
30      puts(cadena);
31    }
32    free(cadena);
33
34    return 0;
35  }
```

Comprueba con estas preguntas si has entendido este documento.

1. Indica la afirmación INCORRECTA respecto a la función `getline`:

   ○ Devuelve el número de caracteres leídos, incluyendo el caracter '\n' de fin de línea, pero no cuenta el '\0' de final de cadena.

   ○ Si el buffer de memoria que recibe como parámetro de entrada no tiene espacio suficiente, devuelve un error.

   ◉ Si el puntero al buffer de memoria que se le pasa para guardar el resultado es `NULL`, reserva memoria internamente con un `malloc`.

De nuevo Soluciones

8.4. Funciones de entrada para leer strings de manera segura

8.5. Bibliografía de apoyo