



Get Next Line

Leer una línea de un fd es tedioso

Resumen: El objetivo de este proyecto es permitirte programar una función que devuelva una línea (terminada con un salto de línea), leída de un fd.

Índice general

I.	Objetivos	2
II.	Instrucciones generales	3
III.	Parte obligatoria - Get_next_line	4
IV.	Parte bonus	6

Capítulo I

Objetivos

Este proyecto no solo te permitirá añadir una función bastante práctica a tu colección; también te permitirá aprender el increíble concepto de las variables estáticas en C.

Capítulo II

Instrucciones generales

- Tu proyecto debe estar escrito siguiendo la Norma. Si tienes archivos o funciones adicionales, estas están incluidas en la verificación de la Norma y tendrás un 0 si hay algún error de norma dentro.
- Tus funciones no deben terminar de forma inesperada (segfault, bus error, double free, etc) ni tener comportamientos indefinidos. Si esto pasa tu proyecto será considerado no funcional y recibirás un 0 durante la evaluación.
- Toda la memoria alocada en heap deberá liberarse adecuadamente cuando sea necesario. No se permitirán leaks de memoria.
- Si el subject lo requiere, deberás entregar un **Makefile** que compilará tus archivos fuente al output requerido con las flags **-Wall**, **-Werror** y **-Wextra**, por supuesto tu **Makefile** no debe hacer relink.
- Tu **Makefile** debe contener al menos las normas **\$(NAME)**, **all**, **clean**, **fclean** y **re**.
- Para entregar los bonus de tu proyecto, deberás incluir una regla **bonus** en tu **Makefile**, en la que añadirás todos los headers, librerías o funciones que estén prohibidas en la parte principal del proyecto. Los bonus deben estar en archivos distintos **_bonus.{c/h}**. La parte obligatoria y los bonus se evalúan por separado.
- Si tu proyecto permite el uso de la **libft**, deberás copiar su fuente y sus **Makefile** asociados en un directorio **libft** con su correspondiente **Makefile**. El **Makefile** de tu proyecto debe compilar primero la librería utilizando su **Makefile**, y después compilar el proyecto.
- Te recomendamos crear programas de prueba para tu proyecto, aunque este trabajo **no será entregado ni evaluado**. Te dará la oportunidad de verificar que tu programa funciona correctamente durante tu evaluación y la de otros compañeros. Y sí, tienes permitido utilizar estas pruebas durante tu evaluación o la de otros compañeros.
- Entrega tu trabajo a tu repositorio **Git** asignado. Solo el trabajo de tu repositorio **Git** será evaluado. Si Deepthought evalúa tu trabajo, lo hará después de tus compañeros. Si se encuentra un error durante la evaluación de Deepthought, la evaluación terminará.

Capítulo III

Parte obligatoria - Get_next_line

Nombre de función	<code>get_next_line</code>
Prototipo	<code>int get_next_line(int fd, char **line);</code>
Archivos a entregar	<code>get_next_line.c</code> , <code>get_next_line_utils.c</code> , <code>get_next_line.h</code>
Parámetros	#1. fd del que leer #2. El valor de lo que se ha leído
Valor devuelto	1 : Una línea se ha leído 0 : Se ha llegado a EOF -1 : Ha ocurrido un error
Funciones autorizadas	<code>read</code> , <code>malloc</code> , <code>free</code>
Descripción	Escribe una función que devuelva la línea leída de un fd, sin el salto de línea.

- Llamar a tu función `get_next_line` en un bucle te permitirá leer el contenido de un file descriptor línea a línea hasta el EOF.
- Asegúrate de que tu función se comporta adecuadamente cuando lea de un archivo y cuando lea de stdin.
- `libft` no se permite para este proyecto. Debes añadir un archivo `get_next_line_utils.c` que tendrá las funciones necesarias para que tu `get_next_line` funcione.
- Tu programa debe compilar con la flag `-D BUFFER_SIZE=xx`. Esta flag se utilizará para determinar el tamaño del buffer de las lecturas de tu `get_next_line`. Este parámetro será modificado por tus evaluadores y por Moulinette.
- La compilación se realizará de la siguiente forma: `gcc -Wall -Werror -Wextra -D BUFFER_SIZE=32 get_next_line.c get_next_line_utils.c`
- Tu `read` deberá utilizar el `BUFFER_SIZE` definido durante la compilación para leer de un archivo o del stdin.
- En el header `get_next_line.h` deberás tener como mínimo el prototipo de la función `get_next_line`.



¿Funciona correctamente tu `get_next_line` si el `BUFFER_SIZE` es 9999?
¿Y si es 1? ¿Qué tal con 10000000? ¿Sabes por qué?



Deberás intentar leer lo menos posible cada vez que se llame a `get_next_line`. Si encuentras un salto de línea, deberás devolver la línea actual. No leas el archivo entero y luego proceses cada línea.



No entregues tus proyectos sin probarlos. Hay muchísimos casos que debes validar, búscalos todos.
Intenta leer de un archivo, de una redirección, del `stdin`. ¿Qué hace tu programa si le mandas un salto de línea desde el `stdout`? ¿Qué hace con `ctrl-D`?

- Consideramos que `get_next_line` tiene un comportamiento indefinido si, entre llamadas, el mismo fd cambia de archivo antes de llegar al EOF en el primer archivo.
- `lseek` no se permite. Debe leerse el archivo una única vez.
- Finalmente, consideramos que `get_next_line` tiene un comportamiento indefinido si se lee un binario. Sin embargo, y si así lo deseas, puedes dar coherencia a este comportamiento.



Una buen punto de partida es conocer lo que es una [variable estática](#).

Capítulo IV

Parte bonus

El proyecto `get_next_line` es bastante directo y no deja mucho margen a los bonus, pero seguramente tengas un montón de imaginación. Si has clavado la parte obligatoria y la tienes PERFECTA, puedes ir a por los bonus sin problema. De nuevo, tus bonus no se tendrán en cuenta si la parte obligatoria no está PERFECTA.

Entrega los tres archivos iniciales con `_bonus` para esta parte.

- Entrega `get_next_line` con una sola variable estática.
- Sé capaz de gestionar múltiples fd con tu `get_next_line`. Es decir, si tienes tres fd disponibles para lectura (por ejemplo: 3, 4 y 5), debes poder utilizar `get_next_line` una vez sobre el fd 3, otra vez sobre el fd 4, y otra vez sobre el fd 5 de forma intermitente. Y sí, no debe perder el hilo de lectura de cada uno de los fd.