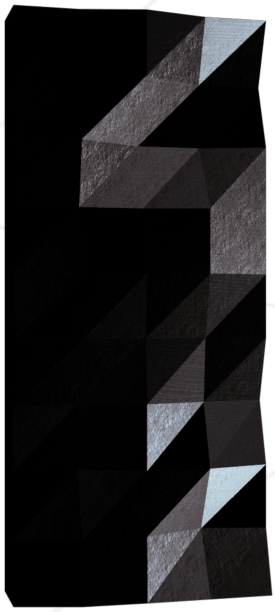

Get Next Line

Porque leer sobre un fd no es nada apasionante

Resumen: Este proyecto tiene como objetivo desarrollar una función que devuelva una línea que se termine con un salto de línea, leída desde un descriptor de archivo.

Índice general

Capítulo I

Reglas comunes

- Su proyecto debe estar programado respetando la Norma. Si tiene archivos o funciones extras, entrarán dentro de la verificación de la norma y, como haya algún error de norma, tendrá un 0 en el proyecto.
- Sus funciones no pueden pararse de forma inesperada (segmentation fault, bus error, double free, etc.) salvo en el caso de un comportamiento indefinido. Si esto ocurre, se considerará que su proyecto no es funcional y tendrá un 0 en el proyecto.
- Cualquier memoria reservada en el montón (heap) tendrá que ser liberada cuando sea necesario. No se tolerará ninguna fuga de memoria.
- Si el proyecto lo requiere, tendrá que entregar un Makefile que compilará sus códigos fuente para crear la salida solicitada, utilizando los flags `-Wall`, `-Wextra` y `-Werror`. Su Makefile no debe hacer relink.
- Si el proyecto requiere un Makefile, su Makefile debe incluir al menos las reglas `$(NAME)`, `all`, `clean`, `fclean` y `re`.
- Para entregar los extras, debe incluir en su Makefile una regla `bonus` que añadirá los headers, bibliotecas o funciones que no estén permitidos en la parte principal del proyecto. Los extras deben estar dentro de un archivo `_bonus.{c/h}`. Las evaluaciones de la parte obligatoria y de la parte extra se hacen por separado.
- Si el proyecto autoriza su `libft`, debe copiar sus códigos fuente y y su Makefile asociado en un directorio libft, dentro de la raíz. El Makefile de su proyecto debe compilar la biblioteca con la ayuda de su Makefile y después compilar el proyecto.
- Le recomendamos que cree programas de prueba para su proyecto, aunque ese trabajo **no será ni entregado ni evaluado**. Esto le dará la oportunidad de probar fácilmente su trabajo al igual que el de sus compañeros.
- Deberá entregar su trabajo en el git que se le ha asignado. Solo se evaluará el trabajo que se suba al git. Si Deepthought debe corregir su trabajo, lo hará al final de las evaluaciones por sus pares. Si surge un error durante la evaluación Deepthought, esta última se parará.

Capítulo II

Objetivos

Este proyecto no solo le va a permitir añadir una función muy práctica a su colección, sino que además le permitirá iniciarse a un nuevo elemento sorprendente de la programación en `C`: las variables estáticas.

Capítulo III

Parte Obligatoria - Get_next_line

Nombre de la función	
Prototipo	`int get_next_line(int fd, char **line);`
Ficheros a entregar	`get_next_line.c, get_next_line_utils.c, get_next_line.h`
Parámetros	`#1. El file descriptor sobre el que leer` `#2. El valor de lo que se ha leído`

Valor de retorno	`1: Se ha leído una línea`
	`0: Se ha terminado la lectura`
	`-1: Ha ocurrido un error`

Funciones externas autorizadas	`read, malloc, free`
Descripción	`Escriba un función que devuelva une línea leída desde un file descriptor, sin el salto de línea`

- Las llamadas sucesivas a su función `get_next_line` le deben permitir leer la totalidad del texto disponible sobre el file descriptor, línea a línea, hasta el EOF.
- No se permite el uso de la `libft` en este proyecto. Tiene que añadir el archivo `get_next_line_utils.c` que contendrá las funciones necesarias para que funcione su get_next_line.
- Asegúrese de que su función se comporta correctamente cuando lee desde un archivo o desde la entrada estándar.
- Su programa debe compilar con el flag `-D BUFFER_SIZE=xx` Este define debe utilizarse en sus llamadas de read de la get_next_line, para definir el tamaño del buffer. En la evaluación se modificará este valor y también lo hará la moulinette.
- Compilación: `gcc -Wall -Wextra -Werror -D BUFFER_SIZE=32 get_next_line.c get_next_line_utils.c`
- Su `read` DEBE utilizar el BUFFER_SIZE para leer desde un archivo o desde la stdin.

- En el archivo header `get_next_line.h`, debe al menos tener el prototipo de la función.

¿Si el valor de BUFFER_SIZE es 9999 sigue funcionando su función? ¿y si vale 1? ¿o 10000000? ¿Sabe por qué?

- Se considera que `get_next_line` presenta un comportamiento indeterminado si, entre dos llamadas, el file descriptor cambia de archivo mientras que todavía no se ha alcanzado el EOF del primer archivo.
- No, `lseek` no es una función permitida. Solo se puede leer el file descriptor una vez.
- Por último, se considera que `get_next_line` tiene un comportamiento indeterminado si leemos un archivo binario. No obstante, si lo desea, puede hacer que ese comportamiento se vuelva coherente.
- Se prohíben la variables globales.

El hecho de saber lo que es una variable estática es un buen comienzo. https://en.wikipedia.org/wiki/Static_variable

Capítulo IV

Parte Extra

El proyecto `get_next_line` deja poco margen para la imaginación, pero si ha completado toda la parte obligatoria, puede hacer la parte extra que le proponemos aquí. Para la entrega de la parte extra, tendrá que entregar los 3 archivos iniciales con `_bonus`. Estos archivos deben incluir todo el proyecto al igual que los ejercicios extra.

- Complete `get_next_line` con un sola variable estática.
- Complete `get_next_line` permitiéndole que gestione varios fd. Por ejemplo, si se puede acceder a los fd 3, 4 y 5 en modo lectura, entonces puede llamar a `get_next_line` una de cada 3 veces, después una de cada 4, después una de cada 5, después volverlo a llamar una de 3 veces, etc. Y eso, sin nunca perder el contenido leído en cada uno de los fd y sin mezclarlos.