Chapitre III

Partie obligatoire

Function name	`get_next_line`
Prototype	`char *get_next_line(int fd);`
Fichiers de rendu	`get_next_line.h, get_next_line.c, get_next_line_utils.c`
Paramètres	`fd : le descripteur de fichier depuis lequel lire`
Valeur de retour	`Le contenu de la ligne lue : comportement correct NULL : rien d'autre à lire ou une erreur s'est produite`
Fonctions externes autorisées	`read, malloc, free`
Description	`Écrire une fonction qui retourne une ligne lue depuis un descripteur de fichier`

- Des appels successifs à votre fonction `get_next_line()` doivent vous permettre de lire l'intégralité du fichier texte référencé par le descripteur de fichier, **une ligne à la fois**.
- Votre fonction doit retourner la ligne qui vient d'être lue.
  S'il n'y a plus rien à lire, ou en cas d'erreur, elle doit retourner `NULL`.
- Assurez-vous que votre fonction se comporte correctement qu'elle lise un fichier ou qu'elle lise sur l'entrée standard.
- **Important :** Vous devez toujours retourner la ligne qui a été lue suivie du `\n` la terminant, sauf dans le cas où vous avez atteint la fin du fichier et que ce dernier ne se termine pas par un `\n`.
- Le fichier d'en-tête `get_next_line.h` doit contenir au minimum le prototype de la fonction.
- Le fichier `get_next_line_utils.c` vous servira à ajouter des fonctions supplémentaires nécessaires à la réalisation de votre `get_next_line()`.

Savoir ce qu'est une variable statique est un bon point de départ.

- Votre programme doit compiler avec l'option : `-D BUFFER_SIZE=n`
  Cette macro définie à l'invocation du compilateur servira à spécifier la taille du *buffer* lors de vos appels à `read()` dans votre fonction `get_next_line()`.
  Cette valeur sera modifiée lors de la peer-evaluation et par la Moulinette dans le but de tester votre rendu.

We must be able to compile this project with and without the -D BUFFER_SIZE flag in addition to the usual flags. You can choose the default value of your choice.

- Votre programme sera donc compilé de la manière suivante (exemple ci-dessous avec une taille de *buffer* de `42`) :
  `cc -Wall -Wextra -Werror -D BUFFER_SIZE=42 <files>.c`

- Nous considérons que `get_next_line()` a un comportement indeterminé si, entre deux appels, le fichier pointé par le descripteur de fichier a été modifié, alors que le premier fichier n'a pas été lu en entier.

- Nous considérons aussi que `get_next_line()` a un comportement indeterminé en cas de lecture d'un fichier binaire. Cependant, si vous le souhaitez, vous pouvez rendre ce comportement cohérent.

Votre fonction marche-t-elle encore si la valeur de BUFFER_SIZE est de 9999? Ou de 1 ? Ou encore de 10 000 000 ? Savez-vous pourquoi ?

Votre programme doit lire le moins possible à chaque appel à get_next_line(). Si vous rencontrez une nouvelle ligne, vous devez retourner la ligne précédente venant d'être lue.
Ne lisez pas l'intégralité du fichier pour ensuite traiter chaque ligne.

Ce qui n'est pas autorisé

- La `libft` n'est pas autorisée pour ce projet.
- La fonction `lseek()` est interdite.
- Les variables globales sont interdites.