GET\_NEXT\_LINE

* Utiliser stdin pour BUFFER\_SIZE
* Int main(argc, argv[]) avec argv = fichier à lire
* Prototype : int get\_next\_line(int fd, char \*\*line)

Fd: file descriptor sur lequel lire

Line = valeur de ce qui a été lu

* Descripteur de fichier = int
* Read(fd, buffer\_size)
* size\_t = guaranteed to be large enough to represent all possible indices into the array.

Size\_t fread (void \* PTR, size\_t size, size\_t nmemb, file \* stream );

Fread () is used to read data from a file stream. The stream parameter is an opened file pointer. the PTR parameter points to the data space to be read. The number of characters read is determined by the size \* nmemb parameter. Fread () returns the number of nmemb actually read. If this value is smaller than the nmemb parameter, it indicates that the file may be read at the end or an error occurs. In this case, feof () must be used () or ferror () to determine what happens.

PROGRAMME :

* function pour compter le nombre de caractères dans chaque phrase avant le retour à la ligne ou la fin du texte ou E0F

While (str[] = ! ‘\n’ || str[] = ! ‘\0’)

n++;

* function pour lire le contenu de la phrase 🡪 read
* utiliser static int pour pouvoir appeler la fonction plusieurs fois et ne pas toujours faire la même ligne
* il faut lire n caractères après n caractères. Donc quand le programme arrive à \n, il peut y avoir des caractères en trop qui ne seront donc pas gérés si pas static. Il faut alors créer une variable char \*rest
* Je me place a l'endroit indiqué dans le fichier

Tant que je ne tombe pas sur 'EOF' ou '\0' :

   je lis le mot suivant dans le fichier, je le stock dans ma variable

fin tant que

* faire un strsplit a chaque \n
* realloc est toute simple à implémenter:

1) allouer (malloc) une zone de la taille demandée, retourner NULL en cas d'échec  
2) copier l'ancienne zone dans la nouvelle  
3) libérer (free) l'ancienne zone  
4) retourner l'adresse de la nouvelle zone.

Bonjour. \n

Comment ca va. \n

Coucou. OEF

Tout dépend la taille du buffer

Strncmp

QUESTIONS :

* faut-il faire apparaitre le texte sur la console ou doit-on seulement retourner 0, 1 ou -1 ? Comment alors savoir si le programme a bien lu le texte ?
* comment utiliser dans le programme et notamment le read le BUFFER\_SIZE donné dans le gcc -D ?

#ifndef BUFFER\_SIZE

# define BUFFER\_SIZE

Si fin du texte arrive avant le buffer, il faut garder en mémoire le reste du buffer.

Get\_next\_line se relance après chaque \n. 🡪 on va donc devoir allouer de la mémoire à chaque fois qu’on utilise GNL 🡪 utilité de pouvoir réallouer la mémoire ?

Read = n’over-read pas : une fois que des caractères sont lues, il ne les relie plus si on rappelle la fonction.

Fonctions à utiliser :

* count
* split ?

Questions :

* Que se passe-t-il si on a + de caractères à lire que la taille du BUFFER ? -1, 0 ou 1 ?