Lire un fichier texte ligne par ligne

par Patrick Gonord

Date de publication : 1-07-08

Dernière mise à jour :

Ce module permet la lecture d'un fichier texte ligne par ligne. Il réalise l'allocation dynamique nécessaire au stockage de la ligne.
La taille d'une ligne est uniquement limitée par la mémoire disponible.

I - Présentation

Une des difficultés rencontrées lors de la lecture des lignes d'un fichier est de prévoir la longueur maximale que peut avoir une ligne pour dimensionner correctement le buffer de réception. Cette fonction réalise les allocations nécessaires.
Ses caractéristiques principales sont :

- Chaque ligne est terminée par '\n' ou par la fin du fichier.
- La fonction réalise l'allocation du buffer nécessaire au stockage de la ligne.
- Le caractère de fin de ligne n'est pas stocké dans le buffer.
- Elle renvoie le nombre de caractères placés dans le buffer (sans compter le '\0' de fin de chaîne).
- Elle indique les erreurs d'allocations et la fin de fichier.

Le prototype de cette fonction est :

Prototype
```
long LireLigne(FILE * f, char ** buff, long dim);
```

- *f* est un pointeur sur un fichier texte déjà correctement ouvert.
- *buff* est l'adresse d'un pointeur sur char qui recevra l'adresse du buffer créé par la fonction et contenant la ligne lue.
  - Si à l'appel de la fonction **buff* est différent de NULL, il est considéré comme contenant l'adresse d'un buffer précédemment créé par la fonction et la fonction procède à la libération de cet espace mémoire avant de lire la ligne suivante.
  Lors du premier appel, *buff doit alors être mis à NULL
  Si le fichier n'est pas entièrement lu, il est de la responsabilité du programmeur de libérer la mémoire contenant la dernière ligne lue.
  - En sortie de la fonction, **buff* contiendra l'adresse du buffer contenant la ligne lue.
  **buff==NULL* indique la fin du fichier ou une erreur d'allocation mémoire.
- *dim* est le nombre de bytes alloués initialement par la fonction pour le buffer devant contenir la ligne. Le buffer sera redimensionné en fonction de la longueur de la ligne. Si *dim* est inférieur à 2, la dimension initiale sera prise comme la valeur de la macro *DIM_DEFAULT*
- En retour :
  - la fonction renvoie le nombre de caractères placés dans le buffer (le caractère '\0' n'est pas compté).
  - Si la ligne est vide, la valeur de retour est 0 et le buffer ne contient que '\0'.
  - En cas de fin de fichier, la valeur de retour est 0 et *buff est placé à NULL.
  - En cas d'erreur d'allocation, la valeur de retour est -1
  Pour résumer :

Valeur de retour	-1	0	>0
*buff == NULL	Erreur d'allocation	Fin de fichier	X
*buff != NULL	X	Ligne vide	Nombre de caractères

II - Code source

Le fichier d'en tête (LireLigne.h):

```
#ifndef LireLigne_H
#define LireLigne_H
#include <stdio.h>
#define DIM_DEFAULT 32
long LireLigne(FILE * f, char ** buff, long dim);
/*
 - Lecture d'une ligne sur un fichier texte. la ligne peut être terminée par
   une fin de ligne ou la fin de fichier.
   La fonction alloue la quantité suffisante de mémoire pour stocker la ligne
```

```
    et le zéro terminal. Le caractère de fin de ligne, si il existe, n'est pas stocké.

    - f    : Fichier texte ouvert sur lequel est lue la ligne
    - buff : Adresse du pointeur dans lequel sera placée l'adresse de la chaîne obtenue.
             Le pointeur doit contenir l'adresse de la chaîne obtenue lors de l'appel
             précédent à la fonction (pour libération de la mémoire allouée) ou NULL
             si il s'agit du premier appel.
             Le pointeur est mis à NULL et la mémoire libérée si
                 - la mémoire disponible est insuffisante
                 - la fin du fichier est atteinte.
    - dim  : Nombre de bytes alloués initialement pour stocker la ligne.
             Si dim >2 , une valeur par défaut (DIM_DEFAULT) est utilisée.
             La taille allouée sera augmentée d'un facteur 2 tant que nécessaire
             puis ramenée finalement à la taille  de la ligne.

    - Renvoie le nombre de caractères de la chaîne. le zéro terminal n'est pas compté
             En cas de fin de fichier renvoie 0 (et *Buff == NULL)
             En cas d'erreur d'allocation mémoire renvoie -1 (et *Buff == NULL)
             Pour une ligne vide renvoie 0 (et *Buff != NULL pointe sur '\0')
*/
#endif
```

Le fichier source (LireLigne.c):

```
#include <string.h>
#include <stdlib.h>
#include "LireLigne.h"
/*---------------------------------------------------------------------------*/
static long Cherchecar(char* buffer, long offset, int car)
{
    char * pos = strchr(buffer+offset,car);
    return  pos != NULL ? pos - buffer : -1;
}
/*---------------------------------------------------------------------------*/
static char * RedimAlloc( char *buffer, size_t dim)
{
    char * q = realloc(buffer,dim);
    if (q == NULL) free(buffer);
    return q;
}
/*---------------------------------------------------------------------------*/
long LireLigne(FILE * f, char ** buff, long dim)
{
    long eoln = -1;
    long zero = -1;
    char *notEOF;
    int AllocError;
    long offset = 0 ;
    free(*buff);
    if (dim<2) dim = DIM_DEFAULT;
    *buff = malloc((size_t)dim);
    AllocError = *buff == NULL;
    if (!AllocError)
    {
        notEOF= fgets(*buff,dim,f);
        if (!notEOF)
        {
            free(*buff);
            *buff = NULL;
        }
        while (notEOF && eoln < 0 && *buff != NULL)
        {
            eoln = Cherchecar(*buff,offset,'\n');
            if (eoln >=0) (*buff)[eoln] = '\0';
            else
            {
                zero = Cherchecar(*buff,offset,'\0');
                if (zero == dim-1)
                {
                    offset = dim-1;
```

```
                    dim *=2;
                    *buff = RedimAlloc(*buff,(size_t)dim);
                    AllocError = *buff == NULL;
                }
                if (!AllocError)
                    if (!(notEOF = fgets(*buff+offset,offset+2,f))) eoln = zero;
            }
        }
        if (*buff!= NULL) *buff = RedimAlloc(*buff,(size_t)(eoln+1));
    }
    return *buff != NULL ? eoln
                         : AllocError ? -1 : 0;
}
/*---------------------------------------------------------------------------*/
```

III - Code exemple

Voici un exemple d'utilisation de ce code :

```
#include <stdlib.h>
#include <stdio.h>
#include "LireLigne.h"
/*---------------------------------------------------------------------------*/
int main(void)
{
    char* buff = NULL;
    long nb;
    FILE * f;
    f = fopen("LireLigne.txt","r");
    if (f!=NULL)
    {
        do
        {
            nb = LireLigne(f,&buff,5);
            if (buff!= NULL) printf("<%s> nb = %ld\n", buff,nb);
        }
        while (buff!= NULL);
        printf(nb <0 ? "Erreur d'allocation\n" : "Fin de fichier\n");
    }
   else printf("\n fichier non trouve");
   return EXIT_SUCCESS;
}
```