==========

Ajouter
 à la fin de chaque ligne d'un fichier texte

C.Turrier – 18 juin 2023

==========

On peut avoir de placer la balise
 à la fin de chaque ligne d'un fichier texte. Cela peut être utile pour formater un texte pour une utilisation dans des pages HTML. Par exemple :

```
\documentclass[14pt,twoside, openright]{extbook}
%=====================================================
% PAQUETS PRINCIPAUX
%=====================================================
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
...
```

devient

```
\documentclass[14pt,twoside, openright]{extbook}<br/>
%=====================================================<br/>
% PAQUETS PRINCIPAUX<br/>
%=====================================================<br/>
\usepackage[T1]{fontenc}<br/>
\usepackage[utf8]{inputenc}<br/>
...
```

Le programme **abr.c** suivant, écrit en langage C, réalise ce travail automatiquement, ce qui est bien pratique lorsque le nombre de lignes à traiter est important.

Pour compiler ce programme avec GCC, afin d'obtenir le fichier exécutable **abr**, il suffit de saisir la commande suivante depuis le terminal ouvert dans le répertoire où se trouve **abr.c** :

```
gcc -Wall abr.c -o abr
```

Il suffit ensuite de placer l'exécutable **abr** dans le répertoire où se trouve le fichier **in.txt** dont les lignes sont à compléter avec "
".

Pour exécuter le programme, il suffit alors de saisir la commande suivante :

```
./abr
```

Le fichier **out.txt** est alors créé automatiquement et contient les lignes complétées avec "
".

Code source du programme

```
/************************************************
Ajoute <br/> - Auteur C.Turrier - 18 juin 2023
source: abr.c
compilation: gcc -Wall abr.c -o abr
exécution: ./abr
Ce programme ajoute la balise html <br/> à la fin de chaque ligne
d'un fichier texte in.txt
il crée un fichier résultat out.txt à partir d'un fichier in.txt
ligne1 -> ligne1<br/>
ligne2 -> ligne2<br/>
etc...
************************************************/
#include <stdio.h>
#include <string.h>
int main(int argc, char* argv[])
{
FILE* pout = fopen("out.txt", "w+");
FILE* pin = fopen("in.txt", "r");
char line[128];
char newline[128];
int size;
int i;
while (fgets(line, sizeof(line), pin))
{
size=strlen(line);
for(i=0;i< size-1;i++)
{
newline[i]=line[i];
}
newline[size-1]='<'; newline[size]='b'; newline[size+1]='r';
newline[size+2]='/'; newline[size+3]='>'; newline[size+4]='\n';
newline[size+5]='\0';
fputs(newline,pout);
}
fclose(pin);
fclose(pout);
return 0;
}
```

Rappel

En langage C, si on déclare char s[128]; puis qu'on écrit l'instruction strcpy(s, "abc");
on obtient les résultats suivants :

- s[0] = 'a'
- s[1] = 'b'
- s[2] = 'c'
- s[3] = '\0' (le caractère nul de fin de chaîne)

Dans ce cas, strlen(s) renverra la valeur 3, car il compte tous les caractères précédant le caractère nul de fin de chaîne.

Si on a un caractère de fin de ligne '0A' à la fin de la chaîne s, on a
s[0] = 'a'
s[1] = 'b'
s[2] = 'c'
s[3] = '\n' (le de fin de ligne)
s[4] = '\0' (le caractère nul de fin de chaîne)

Dans ce cas, strlen(s) renverra la valeur 4, car il compte tous les caractères précédant le caractère nul de fin de chaîne.

Fonctionnement du programme

Ce programme en langage C effectue les opérations suivantes :

1. Les bibliothèques standard <stdio.h> et <string.h> sont incluses.
2. La fonction main est déclarée avec les arguments argc et argv[].
3. Deux pointeurs de fichiers, pout et pin, sont déclarés pour ouvrir les fichiers "out.txt" en mode écriture avec création ("w+") et "in.txt" en mode lecture ("r"), respectivement.
4. Deux tableaux de caractères line et newline de taille 128 sont déclarés pour stocker les lignes lues depuis le fichier d'entrée et les lignes modifiées pour être écrites dans le fichier de sortie.
5. La variable size est déclarée pour stocker la taille de la ligne lue depuis le fichier d'entrée.
6. La variable i est déclarée pour une utilisation ultérieure dans des boucles.
7. Une boucle while est utilisée pour lire chaque ligne du fichier d'entrée à l'aide de la fonction fgets(). La boucle continue jusqu'à ce qu'il n'y ait plus de lignes à lire.
8. À l'intérieur de la boucle, la variable size est mise à jour avec la longueur de la ligne lue depuis le fichier d'entrée à l'aide de la fonction strlen().
9. Une boucle for itère sur chaque caractère de la ligne lue depuis le fichier d'entrée jusqu'au caractère précédant le caractère de fin de ligne ('\n'). Les caractères de line sont copiés un par un dans newline.
10. Les caractères '<', 'b', 'r', '/', '>' sont ajoutés successivement à la fin de newline pour créer une balise HTML
.

11. Un caractère de nouvelle ligne ('\n') est ajouté après la balise
.
12. Un caractère nul de fin de chaîne ('\0') est ajouté à la fin de newline pour marquer la fin de la chaîne de caractères.
13. La fonction fputs() est utilisée pour écrire la ligne modifiée newline dans le fichier de sortie.
14. Après avoir parcouru toutes les lignes du fichier d'entrée, les fichiers pin et pout sont fermés à l'aide des fonctions fclose().
15. La fonction main se termine avec la valeur de retour 0.

En résumé, ce programme lit chaque ligne du fichier d'entrée, ajoute une balise HTML
 à la fin de chaque ligne et écrit les lignes modifiées dans un fichier de sortie.

Remarques

Structure de ligne de in.txt en entrée

```
e[0]; e[1];....e[size-1]='/n'; e[size]='\0'
```

Structure de ligne de out.txt en sortie

```
s[0] à s[size-2] identiques à e[0] à e[size-2]
s[size-1]='<'; s[size]='b'; s[size+1]='r';
s[size+2]='/'; s[size+3]='>'; s[size+4]='\n';
s[size+5]='\0';
```

Le caractère '\n' en langage C (c'est à dire OA en hexadécimal) représente le saut de ligne ;
Le caractère '\0' en langage C représente la fin de ligne