Attention à respecter les instructions suivantes :

1. vous **devez** utiliser l'outil `make` pour compiler vos programmes. Une introduction à cet outil est disponible sur moodle ;
2. chaque programme doit être composé **d'au moins** deux fichiers `.c` (plus les éventuels `.h`). Un fichier contenant le `main` et un ou plusieurs fichiers contenant le reste du TP ;
3. chaque fichier **doit** compiler avec les options `-Wall -Wextra -Werror`.

Si une de ces condition n'est pas vérifiée, le TP ne sera pas validé par votre enseignant.

1 But du TP

Utilisation des appels système « entrées/sorties bas niveau ». Écrire une commande `wordCount` semblable à la commande `wc`.

Rappel. La commande `wc` compte les caractères, mots et lignes d'un ou plusieurs fichiers. La forme générale de la commande est :

```
wc [-cwl] [fichier]*
```

les éléments entre crochets sont optionnels. La commande sans options affiche, pour chaque fichier cité, le nombre de caractères, mots et lignes du fichier. Si aucun nom de fichier n'est mentionné, le fichier utilisé est l'entrée standard. Les options permettent de restreindre l'affichage :

- `-c` pour afficher le nombre de caractères ;
- `-w` pour afficher le nombre de mots ;
- `-l` pour afficher le nombre de lignes.

Exemple. `wc -l -c toto` affiche le nombre de caractères et le nombre de lignes du fichier `toto` (attention, les commandes `wc -lc toto`, `wc -cl toto` et `wc -c -l toto` ont le même effet).

Exemples. L'exécution de la commande doit donner un résultat semblable aux exemples suivants :

```
bash$ wordCount -l toto essai
lignes : 30 --> toto
lignes : 20 --> essai

lignes : 50 --> total
bash$ wordCount -l -w toto
mots : 97 lignes : 30 --> toto
bash$ wordCount toto
caracteres : 317 mots : 97 lignes : 30 --> toto
```

Essayer la commande `wc` avec quelques exemples pour constater son comportement en cas d'erreurs ou selon les utilisations. La seule différence entre `wordCount` et `wc` est que dans `wordCount` les options sont forcément avant les noms de fichier.

2 Réalisation de la commande

L'écriture de la commande comporte deux parties, il s'agit d'une part de lire un fichier en comptant le nombre de caractères, mots et lignes et d'autre part de traiter les paramètres de la ligne de commande de manière à effectuer le traitement demandé.

2.1 Traitement d'un fichier

Écrire une fonction

```
int traiter (int f, int *car, int *mot, int *lig);
```

qui calcule les nombres de caractères, mots et lignes contenus dans le fichier (ouvert en lecture) de descripteur `f`. Cette fonction devra utiliser les primitives « bas niveau » vues en cours. Pour éviter les accès répétés aux fichiers, il est souhaitable d'utiliser un « buffer » pour traiter les caractères par blocs (nous choisirons ici un buffer de 80 caractères). La fonction doit donc lire un bloc de caractères dans le fichier puis le traiter (compter le nombre de caractères, mots et

lignes) et passer au bloc suivant. Attention, un mot ou une ligne peuvent être à cheval sur plusieurs blocs ! La fonction place respectivement les nombres de caractères, mots et lignes dans les paramètres `car`, `mot` et `lig` et retourne `0` en cas de succès et `1` en cas d'erreur.

> Le fichier d'en-tête `ctype.h` contient les déclarations de fonctions permettant de tester les caractères comme `isupper`, `islower`, … et en particulier :
>
> ```
> int isspace (int c);
> ```
>
> qui retourne une valeur non nulle si le caractère testé est un caractère d'« espacement » (espace, saut de page, fin de ligne, retour chariot, tabulation, tabulation verticale).

Écrire une fonction `main` très simple permettant de tester votre fonction.

2.2 Les paramètres de la commande

Il s'agit à présent de traiter les paramètres de la ligne de commande. Écrire la fonction `main` de sorte qu'elle traite les options et produise l'affichage souhaité. Dans tous les cas d'erreur de syntaxe, une fonction `erreur` doit être appelée. Celle-ci affichera la syntaxe de la commande et provoquera l'arrêt du programme. Exemple :

```
bash$ wordCount -l toto essai
lignes : 22 --> toto
lignes : 3 --> essai

lignes : 25 --> total
bash$ wordCount -l -w toto
mots : 2 lignes : 1 --> toto
bash$ wordCount toto
caracteres : 75 mots : 23 lignes : 2 --> toto
bash$ wordCount
aa zzz dfl
caracteres : 12 mots : 3 lignes : 1 --> stdin
```

Indications. La syntaxe de la commande impose que toutes les options précèdent tous les noms de fichier, il est donc possible de parcourir toutes les options puis de traiter tous les fichiers « à la volée ».