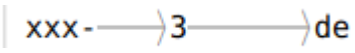
PROJET ALGO2

Shared Words

KAMILOGLU Emre
DELCOURT Louis

I - Introduction

Le but du projet est de réussir à afficher les mots partagés entre plusieurs fichiers avec l'utilisation des modules hashtable et holdall fournis qui étaient obligatoires. Exemple :



le mot "de" apparaît 3 fois en tout dans les fichiers 1, 2 et 3

L'élément "xxx-" permet d'identifier dans quels fichiers le mot "de" apparaît. Il y a sans doute une multitude de façon de récupérer un élément comme ça, mais nous on est parti sur l'idée de créer une chaîne de caractères composée de "x" et de "-" qui va être mise à jour pendant l'exécution du programme. On a décidé de la nommer "fword" (pour file word), c'est comme ça qu'on l'appellera partout.

On a utilisé plusieurs méthodes pour gérer ce fword, au début on a créé notre propre module mais on a eu du mal à le mettre en place. Et une fois fait, c'était pas assez optimal.

C'est pourquoi au final on a créé une structure directement dans le main pour réussir à gérer ce fword : val qui comporte deux pointeur, l'un vers la chaîne de caractère définissant le fword, l'autre vers le tableau count pour compter les mots. C'était au final la manière la plus facile de gérer les mots et leur fword rapidement et simplement.

```
struct val {
    char *fword;
    char *cptr;
};
```

On a utilisé le pointeur valptr du la structure hashtable qui permet d'avoir accès à val. Comme ça depuis la table de hachage, on a directement accès à toutes les informations nécessaires pour chaque mot.

II - Algorithme

Pseudo code

DEBUT :

Tant que argument lu **alors**
 gérer argument selon options ou fichiers.
fin tant que

Si nombre fichier < 2 OU nombre fichier > 2 **alors**
 -> Renvoie Erreur
fin si

Initialiser table de hachage et fourre tout.

Pour chaque fichier lu **faire**
 Si mot lu **alors**
  Effectuer une option dessus.
  rechercher sa présence dans la table de hachage.
  Si est présent **alors**
   mettre à jour les occurrences et la présence dans les fichiers.
  fin si
  Sinon
   ajouter le nouveau mot.
   ->Dans table de hachage(présence dans le fichier et occurrence à 1).
   ->Dans fourre tout.
  fin sinon
 fin si
fin Pour
-> trier selon les exigences
-> afficher

Au début du main il y a beaucoup d'initialisation, notamment pour gérer les options. Chaque option a sa propre variable qui permet de faire ou de ne pas faire certaines actions.

Vient ensuite la lecture des arguments, c'est ici qu'on a va prendre en compte les options et modifier les variables liées à chacune d'elles, ou stocker les fichiers (s'ils sont lisibles évidemment).

Une fois tout cela fait, on passe aux choses sérieuses: on initialise la table de hachage ***ht*** et le fourre-tout ***has*** dans lesquels on va insérer chaque nouveau mot qu'on va lire dans les fichiers donnés en argument s'ils sont valides. Par exemple, la taille maximale d'un mot est la variable ***prefix*** qui est de base à 63 caractères, mais on peut modifier cette limite avec l'option ***-i X*** avec X le nombre de caractères voulu.

Si un mot est nouveau, on l'ajoute dans la table de hachage en initialisant les valeurs du fword à la chaîne de caractère composée uniquement de "-" sauf à l'indice du numéro de fichier actuel qui sera "x" et le compteur à 1 évidemment . On ajoute également le mot dans le fourre-tout.

Si un mot est déjà présent, on va mettre à jour le fword en changeant l'indice du fichier actuel en "x" et augmenter le compteur de 1.

On va continuer à lire les fichiers jusqu'au dernier mot du dernier fichier. Une fois tous ces mots insérés on va pouvoir trier le fourre-tout obtenu pour déjà se débarrasser des mots apparaissant que dans un seul fichier, puis ensuite trier le reste des mots selon d'autres modalités. *Voir la prochaine partie dédiée uniquement au tri*

On arrive sur la fin du programme, il nous reste plus qu'à afficher ce nouveau fourre-tout pour obtenir la liste des mots partagés entre les fichiers mis en argument.
Pour obtenir ce résultat on a évidemment utilisé plusieurs fonctions auxiliaire comme par exemple la fonction ***lecture***, qui permet de prendre chaque caractère 1 par 1 pour en faire un mot et le renvoyer.
Il ne manque plus qu'à libérer toutes les allocations faites.

Voici des exemples d'exécutions:

On exécute sur 4 fichiers a.txt, b.txt, c.txt, d.txt

```
emre@emre-MS-7B17:~/Bureau/Licence/L2 v2/ALGO2/PROJET/main$ ./ws a.txt b.txt c.txt d.txt
xxxx    5       UTOUS
xxxx    4       TOUS
xx-x    3       trois
-xxx    3       troiss
x--x    2       bdeux
-x-x    2       deux
```

On peut voir les différents mots partagés, à quels fichiers est-ce qu'ils appartiennent et leur occurrences

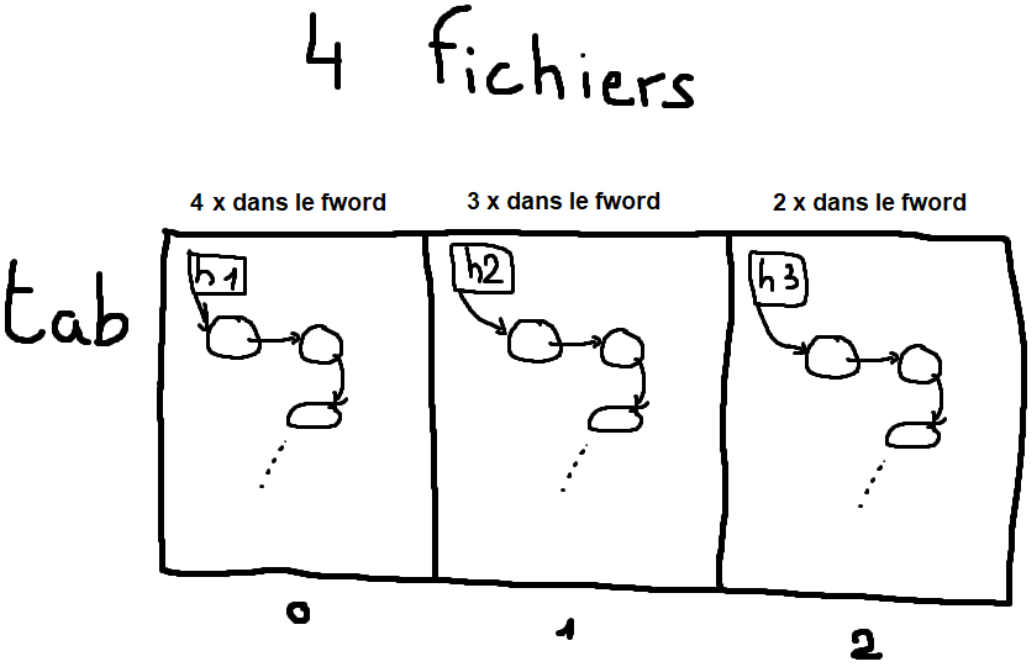
On peut ensuite s'amuser avec différentes options :

```
emre@emre-MS-7B17:~/Bureau/Licence/L2 v2/ALGO2/PROJET/main$ ./ws a.txt b.txt c.txt d.txt -t 4 -u
xxxx    5       UTOUS
xxxx    4       TOUS
xx-x    3       TROIS
-xxx    3       TROISS
```

l'option ***-t 4*** permet de n'afficher que les 4 premiers mots
l'option ***-u*** permet de tout mettre en majuscule

III - Tri du fourre-tout

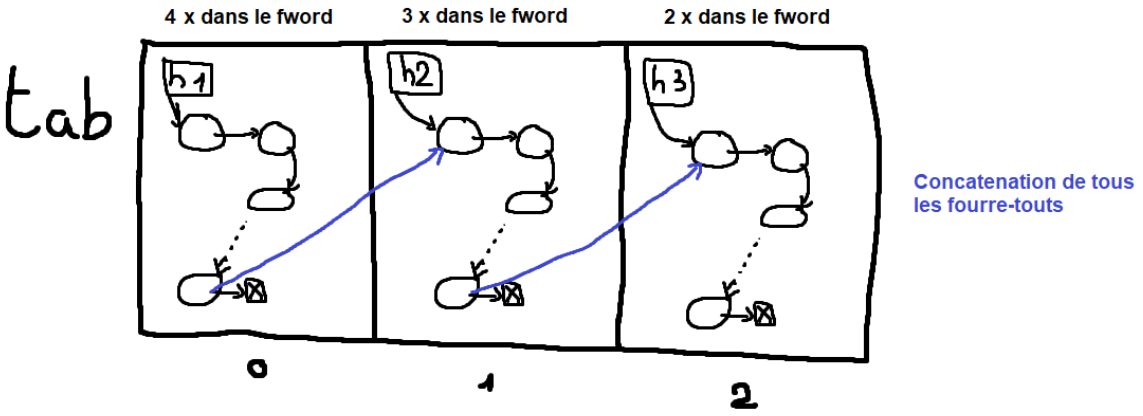
Pour trier tout ce beau monde, on s'y est pris d'une manière un peu spéciale : on a créé un tableau (***tab***) de fourre-tout de taille (nombre de fichiers total) - 1. A partir du dernier indice de ce tableau, on peut retrouver un pointeur vers le fourre-tout des mots ayant 2 x dans le fword, puis à reculons 3 x dans le fword, puis 4 x dans le fword s'il y a quatre fichiers en tout ect…



On va ensuite aller chercher les mots du fourre-tout originel passé en argument, puis les insérer dans le bon fourre-tout créé juste avant. Si le nombre de x dans le fword est inférieur à 2, c'est que le mot en question n'est pas partagé par plusieurs fichiers donc on va le mettre dans aucun fourre-tout et passer directement au prochain.

Il y a trois conditions de tri, premièrement dans l'ordre décroissant du nombre de x dans le fword, dans l'ordre décroissant du nombre d'occurrence, puis enfin dans l'ordre lexicographique.

Après l'étape précédente, la première condition est respectée. Pour les suivantes on fait un tri rapide sur les fourre-touts du tableau pour trier par rapport aux occurrences et au sens de compar. On aura donc (nombre de fichier) - 1 fourre-touts triés correctement, il nous reste qu'à concaténer ces fourre-touts nous voilà avec le fourre-tout originel trié comme on le voulais.



La suite va dépendre des options et de leur état. L'option ***-t X*** permet d'afficher ***X*** mots et elle vaut de base 10 (0 permet d'afficher tous les mots), donc il faut garder que les ***X*** premiers mot du fourre-tout.

Nous voilà avec le fourre-tout qu'on voulait, qui comporte le nombre de mots voulut

IV - Difficultés rencontrés

La plus grosse difficulté qu'on a eu était valgrind. On a eu la mauvaise idée de tout coder et de ne s'occuper qu'à la fin de la désallocation des éléments alloués (chaîne de caractères, fourre-touts etc…) surtout dans la fonction ***holdall_sort***. C'était un bazar mais on a quand même réussi à s'en sortir petit à petit.

Ensuite on avait une première version du tri, n'utilisant pas le tri rapide, qui triait plus ou moins de la même manière sauf au niveau du nombre d'occurrences et au sens lexicographique. Lors de l'ajout d'un mot, on comparait 1 à 1 chaque mot depuis le début du fourre-tout pour le placer au bon endroit. Donc on a vite été confronté à la lenteur du programme sur de gros fichiers à cause de la vitesse linéaire du tri (un maximum d'une dizaine de secondes sur une bonne machine à plusieurs minutes sur d'autres).

Comme dit dans l'introduction, au début on avait créé notre propre module qui nous permettait d'enregistrer les fword ainsi que les mots. On avait trié tout ça d'une certaine manière, tout fonctionnait mais ce n'était pas optimisé et on se prenait trop la tête pour quelque chose de simple.

V - Limites

Le programme a certaines limites prédéfinies et d'autres qui sont modifiables avec l'utilisation des options.

- Un maximum de 32 fichiers sont acceptés
- Le nombre de mots affichés sont gérés par la variable ***nbword***, modifiable avec l'option ***-t X***
- La taille des mots est limité par la variable ***prefix***, modifiable avec l'option ***-i X***

On a pas pu vérifier la limite du compteur de mots, même avec de très gros fichiers remplis du même mot, on a un résultat de plusieurs billions d'occurrences.