UNIVERSITÉ DE ROUEN NORMANDIE

Projet d'Algorithemique

Edouard.H Théo.R.V

2022-2023

Résumé

Ce document constitue notre compte rendu du projet d'Algorithemique Lines Identical (lnid).

Dans une première partie, nous aborderons une brève description du projet et de ses objectifs. Puis, nous présenterons les divers problématiques qu'entraine ce sujet. Enfin, nous verrons les solutions trouvées à ces problèmes par le biais d'une présentation complète de notre implémentation. Suivie d'une mise en pratique avec des tests intéressant et des tests de performance. Pour conclure, nous aborderons les diverses limitations et possible amélioration de cette implémentation, mais aussi nous dieffrons de la production de ce projet.

Table des matières

I Introdution

Ce projet consiste en la réalisation d'un programme C, élaborant le traitement suivant :

> Si un seul fichier lui est fourni, alors affiche la liste des numéros des lignes redondante puis affiche cette ligne.
>
> Sinon si, plusieurs fichiers lui sont fournit, affiche pour tous les fichiers le nombre d'occurrence de ligne commune et affiche ces lignes.
>
> Sinon si aucun fichier n'est fournie alors effectue le traitement de fichier unique sur les lignes présente sur l'entrée standart.

Voir Figure ? ? pour un exemple d'execution. Le programme devra aussi avoir les même spécification que les commandes linux comme '-' qui lis sur l'entrée standart ou encore '--' qui prévient que l'argument suivant est un fichier. De plus diverses options doivent être mises en œuvre, ajoutant des fonctionnalités, voir Figure **??** pour leur spécification.

`-ht`	`-help`	
Affiche sur la sortie standart la documentation de l'executable.		
`-nc`	`-no-color`	
Enlève les couleurs de l'affichage du programme.		
`-u`	`-uppercasing`	
Tous les caractères traiter seront en sortie en majuscule.		
`-a`	`-avl`	
Utilise les avls pour la gestion du projet.		
`-f CLASS`	`-filter CLASS`	Avec `CLASS` l'un des suffixes ... des douze test d'appartenance à une catégorie de caractères is... de l'en-tête standard <ctype.ht>
Ne prend en compte au traitement que les charactères répondan au test de présence dans l'enssemble CLASS, fonction de test étant is.CLASS.		
`-s WORD`	`-sort WORD`	Avec `WORD` valant soit `standard` soit `locale`
Trie les valeurs sur la sortie en prenant en compte l'ordre `WORD` avec `standart` correspondant à l'ordre du "C" et `locale` celui du système de l'utilisateur.		

FIGURE 1 – Tableau des options.

II Problèmatique

— Lecutre d'une ligne de longueur quelconque.
— Gestion des différents compteur.
— Gestion des options.
— Cas de la ligne vide dans la lecture.
— Gestion message d'erreur. (string litéral)
— Suppression des "valeurs magiques" présent dans le code.
— Mettre en place le même comportement que pour une commande linux (<– fichier>)

III Nos Solutions

1 Da

De nombreux problème dans ce projet réside dans l'ajout d'un nombre quelqu'onque d'élement à une structure de donné. Notament dans la lecture des lignes, le comptage des occurences de celle-ci ou leur numérotations mais aussi dans la gestion de la liste de fichier à traiter. C'est pour cela que nous avons opter pour l'extention d'un type de donner abstrait (TDA), qui représenterai notre objet, que l'on nomera donc DA ("Dynamique Array" := "tableau dynamique"). Vous trouverez dans la figure **??** la spécification de ce TDA.

Afin d'être en accord avec l'implémentation des tableaux en C, la spécification du module DA promet que les éléments stockés sont contingue dans la mémoire et sans offset.

Le point négatif d'une telle condition, est une perte de séparation entre l'implémentation et la spécification du module. Or, elle permet une simplification de l'uitilisation du module mais aussi des gains de performance. On peut notament cité le traitement des lignes lus.

Une ligne lu est stoké dans un buffeur de type da. Sans la contrainte il nous est impossible de la comparer au chaine de charactère déjà lu (les chaines déjà lu n'étant pas des da, car stocké des da ne permetterait pas l'utilisation de fonction de comparaison comme strmcp et strcoll). Il nous serait obligatoire d'allouer une nouvelle chaine de la taille de la ligne présente dans le buffer puis d'y recopier la ligne pour enfin pourvoir la comparer. Or de ce traitement on comprend qu'il faudra allouer une chaine dans touts les cas, même si cette ligne est déjà lu ou même dans le cas ou le fichier qui est lu n'est pas le premier (Les lignes lu sur des fichiers qui ne sont pas le premier ne seront jamais sauvegrader voir section **??**).

Nous voillont bien que dans ces deux cas, cette restriction permet d'économiser une allocation d'un doublon pour le permier et pour le deuxième une d'allocation. Par exemple, pour deux fichier. Dont le premier est composer que d'une ligne et le deuxième de n ligne. On obtient les allocations suivantes :

	Premier fichier	Deuxième fichier	Total d'allocation
Avec restriction	1	0	1
Sans restriction	1	n	n + 1

FIGURE 2 – Comparaison nombre d'allocation sans prendre en compte celle du buffer

2 Opt

Nous avons décider de gérer les options a l'aide d'un module, opt ("option"). Notre objetctif a été de disgner un module génréque donc réutilisable à souhait. Mais aussi de les gérers "à la linux" (notament avec les cas spéciales de "–", "-" et que la valeur d'une option soit déterminé par sa dernière définition).

Pour cela, nous avons décider de mettre en pratique l'encapsulation des donners à l'aide d'un type (optparam). Représentant une options, sur la base des opitions linux nous avons donc une représentation longue et court mais aussi une descritpion de celle-ci. Pour le traiment, l'utilisateur fournit sa liste d'option et la liste d'argument de son programme et notre module traitement chaqu'un des arguments avec le traitement adéquemtement fournit par l'utilisateur. Ce traitement s'organise de la façon suivant, avec le traitement de l'option et de l'argument représenter par les fonctions hdl de la structure opt et other tout deux passer en parametre dans la fonction opt_init :

Pré-condition tab-args, tableau d'argument d'un programme.
Pour chaque arg **de** tab-args **faire**
Si est-option arg **alors**
Traitement de l'option
Sinon
Traitement de l'argument
Fin Si
Fin Pour chaque

FIGURE 3 – Algorithme principal de la fonction opt_init

Dans notre implémentation la fonction est-option est opt_parse. Cette fonction à donc deux traitement différent en fonction d'une option courte et d'une longue. Une option courte est de la forme :

Dans ce cas, si notre option courte est à l'indice k du tableau argv de opt_init il en vient alors que sa possible valeur est à l'indice k + 1, or le traitement opt_parse va traiter les chaine k et k + 1. De ce faite opt_init ne devra pas traiter l'argument d'indice k + 1, c'est donc pour cela que opt_parse prend un pointeur sur l'indice de l'option. Alors si, l'option courte apprait mais aussi sa valeur à l'indice k + 1, dans opt_init le k prendra la valeur k + 2 après sont tour de boucle de traitement de l'indice k. Ne traitent pas une chaine qu'il ne faut pas traiter.

Une option longue est de la forme (avec LONG_JOIN aillant pour valeur dans notre cas '=') :

Dans ce cas, opt_parse récupère la chaine puis teste au maximum avec toutes les options longues si l'option longeue est préfixe de la chaine. Si l'option nécessite un argument dans ce cas un test d'égalier entre le caractère suivant le préfixe et LONG_JOIN alors le traitement de la value s'effectura sur le reste de la chaine. Sinon, le traitement s'effectue avec la fonction défiene pour cette option.

L'utilisateur choisie ou non en fonction de son besoin de modifier son evyroenemt par le traitemnt de ces option a l'aide du paramettre context.

Ce module étant inspirer des option linux, il contient par défaut l'option "help", décrivant l'utilisation et le comportement du programme de l'utilisateur (nom de cette option qui peut être modifier par l'utilisateur dans "opt.h").

Due a l'encapsulation des donners, ce module doit recourir a des alloctions dynamique pour chauqe option mais ces allocations sont raisonable en vue du nombre d'option en moyenne d'un programme. De plus étant donné que le module vise à être le plus générique possible, il est en devient complexe à sa comprehenseion, notament dans la spécification de opt_init.

3 hastable / avl

La laison entre une ligne et son nombre d'occurence ou son numéro d'appartion peut être résolue par l'utilisation de structure de donné qui visent à joindre ces informations pour pouvoir les récupérers. Nous decider de mettre deux type de ces structures à dispostion, une d'arbres binaire et l'autre d'une table de hachage.

3.1 hastable

Constituant l'implémentation d'une table de hachage, cette structure permet a partir d'une clé une valeur. Dans notre cas la clé est ligne déjà lu et la valeur ces numéro de lignes (dans le cas d'un seul fichier) ou son nombre d'occurence (dans le cas de pluseir fichier). Cette structure est toutes choisite pour repondre a ce problème. En effet, théoriquement une table de hachage permet un accées presque constant à une valeur par sa clé. Malgrès tout en pratique cette accés dépends exclusivement de la focntion de hachage utilisé, c'est pour cela que dans la section de test nous métons à disposition divers exemple d'execution avec des fonction de hachages différentes. De plus, en pratique l'ajout dans une table de hachage pour entraiténer un agrandisement de celle-ci et donc un recalcule de toutes les valeurs en sont sein.

3.2 Avl

L'utilisation des arbres binaires, nous a demander la création d'un type hcell pour permtettre de stocké les deux valeurs (chaine et compteur). Contrairement à la table de hachage notre implémentaion des arbres binaires garant une recherche au maximum en temps logarithemique. Mais un ajout dans un avl peut entrainer au plus deux rotation de temps constant après avoir effectuer sont ajout en bout de chemin ce qui donne une compléxité logarithemique.

Divers test, aillant pour but d'identifier la meilleur option sont effectuer dans la section test.

4 main

IV Mise en Pratique

1 Fichier lourd

```
$ cat —help
```

V Conclusion

VI Remerciment

Merci à Erwan LIEVIN pour avoir trouvé le noms du module da (dynamique array)

Merci à Ilyas [nom de famille] pour la correction de nos nombreuse fautes d'ortographe notament dans les spécifications