Portfolio C (2018-2019)

Guillaume Cau

Introduction :

Les travaux demandés sont séparé par TP et par exercice dans des répertoires. Dans chaque répertoire se trouve un makefile. Ce makefile permet de faire les commandes suivantes :

- make (compilation)

- make clear (suppression des binaires et de l'exécutable.)

- make run (lance l'exécutable.)

Malheureusement, si l'exécutable demande des paramètres, il faut le lancer manuellement.
L'exécutable se nomme main sauf si un autre nom est demandé dans le sujet.

TP n°2 :

Le but de ce TP était de ce familiariser avec la syntaxe du langage C et ses particularités. Même en ayant déjà fais du C, j'ai appris des chose en faisant ce TP.
Dans mon IUT, nos enseignants nous ont appris à utiliser les fonctions read/write/open et close qui sont les fonctions du système UNIX. Nous n'avions eu que peu d'exercices sur les fonctions utilisant le fonction FILE de la librairie stdio.h.
Ce TP m'a donc permis d'apprendre à utiliser ces fonctions qui doivent être utilisées en priorité.
Mon expérience d'utilisation des fonctions read/write etc m'ont permis de comprendre plus facilement le fonctionnement des fonctions de stdio.

TP n°3 :

Le but de ce TP était de s'entraîner à utiliser le C en utilisant des algorithmes. J'avais déjà implémenté ces algorithmes lors de mon DUT. Je n'ai donc pas eu trop de problèmes lors de ce TP.
Il y a cependant la recherche par dichotomie qui m'a posé problème. En effet, j'ai du mal à me visualiser comment couper le tableau en 2 sans dessins. J'ai donc perdu de ce temps de ce côté là.
Après, retourner le bon indice a été plus ardu que prévu. Les indices que je renvoyai lorsque l'élément était trouvé était celui du tableau local. Donc celui du sous-tableau et non du tableau principal. J'ai

donc été obligé de faire la somme de l’indice trouvé avec la taille du tableau divisé par deux lorsque l’élément se trouvait dans la partie de « droite » lors d’un appel récursif.
Lorsque l’élément ne se trouvait pas dans le tableau, la fonction renvoyait -1. Mais, la valeur -1 s’additionnait désormais avec la taille du tableau divisée par 2 lorsque la fonction était rappelé à droite du tableau. J’ai donc du faire une condition pour éviter d’effectuer une somme avec une valeur -1.

TP n°4 :

Le but de ce TP était d'apprendre à utiliser une pile (LIFO). Je n'ai pas vraiment eu de difficulté à créer ses fonctions associées.
Le seul problème était de savoir dans quel sens effectuer les opérations binaire (à deux arguments).

TP n°5 :

TP n°6 :

Je n'avais jamais fait de backtracking jusqu'à aujourd'hui. Ce TP m'a donc permis de comprendre comment procéder. De plus, j'ai appris une manipulation sympa. Lorsque l'on manipule un gros tableau, il est impossible de le placer dans une fonction car son existence même va faire exploser la pile. Il faut donc le placer en tant que variable globale. C'est peut être plus moche, mais ça fonctionne. Du moins, tant qu'on utilise pas le tas.

Sinon, dans l'exercice du sudoku, j'ai utilisé les divisions et les modulos pour déterminer les coordonnées plutôt qu'une boucle. De cette manière, je visualisait plus facilement l'algorithme parcourir le tableau case par case.
Dans l'exercice précédent, j'avais fait l'inverse. J'avais utilisé des boucles.

Avec le recul que j'ai pris, je ne suis pas certains d'avoir utiliser les méthodes les plus efficaces. J'ai utilisé les méthodes que je comprenais.

En terme de code, j'ai eu un problème lors du développement du sudoku. J'avais mis la vérification pour savoir si la case était vide dans la boucle des 9 chiffres. Le temps d'exécution était très lent. Je n'ai pas tout de suite aperçu ce problème. Je ne l'ai même jamais remarqué. Mon algorithme ne marchait pas, il ne terminait jamais . Je me suis dis que j'allais l'optimisé et c'est là que j'ai corrigé le problème sans réfléchir. La fonction s'est mise à marcher. Je ne comprends toujours pas pourquoi ça a fonctionner vu que c'est au plus une dizaine de comparaisons supplémentaires par récursion. Et pourtant, le temps d'exécution était bien plus que 10 fois plus lent. Enfin, je viens de comprendre en écrivant ces lignes… C'est plutôt exponentiel le nombre de comparaison en plus dans la récursion.

TP n°7 :

Malheureusement, je n’ai pas eu le temps de beaucoup travailler sur le sudoku. Je n’ai donc pas eu le temps de faire des fonctions propres pour toutes les fonctionnalités. Il en résulte que le code n’est pas très compréhensible.

Sur tous ce TP, je me suis basé sur des points de coordonnés choisi arbitrairement. Après, il suffisait de faire quelques calculs pour convertir les coordonnées de l’écran en coordonnées de tableau.

TP n°8 :

real 1m24,296s

user 1m24,293s

sys 0m0,000s

TP n°9 :

Le but de ce TP était d’apprendre à utiliser l’allocation mémoire. J’avais déjà beaucoup utilisé ces fonctionnalités dans ma formation précédente. Je n’ai donc pas eu trop de difficultés à réaliser ce TP.

J’ai tout de même essayer de percer les secrets de malloc. Il semblerai qu’en plus de la zone mémoire qui nous est attribuée, malloc créer aussi ses propres bloc pour se rappeler quels zones mémoires ont été attribué. J’ai aussi remarqué que l’on peut réécrire sur les blocs généré par malloc. Et j’ai aussi pu me rendre compte que malloc attribue une zone minimale. C’est à dire que pour 1 octet, il en affecte plus (Je ne me rappelle plus combien exactement).

TP n°10 :

La manipulation des void* ne m’a pas trop posé de problème lors de ce TP. En effet, les cours d’assembleur du début de l’année m’ont permis de comprendre comment la machine opérait pour manipuler les données que ce soit dans le tas ou sur le pile d’exécution.

Le deuxième exercice m'a posé plus de problème. Le tri par nom et par age m'a pris un peu de temps. Je n'ai compris que plus tard que pour effectuer ce double tri, il suffisait d'effectuer le premier tri, et si deux noms sont identiques, faire le tri sur l'age.

Sinon, la manipulation de structure ne m'a pas posé beaucoup de problèmes.

Deuxième difficulté, lors de ma formation précédente, pour lire et écrire dans les fichiers, nous utilisions les fonction read et write. J'ai donc su apprendre à utiliser les fonctions fread et fwrite.

Avec ce TP, j'ai appris, et je l'espère pour toujours, que pour faire un double tri, il ne faut pas faire les deux tris successivement mais en même temps.

TP n°11 :

Le but de ce TP était de créer un jeu de taquin en utilisant la librairie MLV. Je trouve cette librairie simple d'utilisation. En effet, les fonctions sont bien documentées et comprendre leurs fonctionnement est aisé.

J'ai eu quelques difficultés à gérer les coordonnées des dalles. En effet, le parcours des dalles et des pixels des images ne doit pas être effectuer de la même façon. Les images doivent forcément être lu en horizontal puis en vertical. Du coup, je n'ai pas pu faire mes boucles comme d'habitudes (for (i…) for (j…)). J'ai donc été obligé de m'imposer une sorte de norme pour ne plus avoir de problème.

Le mouvements des dalles s'effectue via les flèches du clavier. Il faut s'imaginer que l'on essaye de bouger la case noire. Si on appuie sur la touche haut, la dalle noire monte vers le haut etc.

Ce TP m'a appris l'importance de prévoir à l'avance comment implémenter l'application. En effet, le fait de l'avoir développé sans avoir réfléchi au préalable m'a rendu très difficile la gestion des coordonnées. (Quelles dimension du tableau représente les x etc.).

J'utilise des variables globales dans ce TP. J'aurais pu les mettre dans le main. Je m'en excuse.

TP n°12 :

L'objectif de ce TP était de réussir à compter le nombre de mots différents dans une fichiers texte. Pour ce faire, il fallait utilisait : une liste chaînée, puis une table de hachage.

Je n'ai pas trouvé ce TP particulièrement difficile. En effet, j'ai compris depuis quelque temps le fonctionnement des listes chaînées. Je n'ai donc pas eu de problème a créer les structures et les fonctions qui les manipules pour les mots.
La table de hachage ne m'a pas posée de problème non plus. Il suffisait juste d'associer à chaque case d'un tableau une liste chaînée (pour gérer les collisions) puis de créer une fonction qui calcule dans quels case du tableau il faut de positionner.

Sinon, après avoir essayé les deux méthodes, on se rend compte que la table de hachage est bien plus efficace qu'une liste chaînée dans cette situation.

TP n°13 :

L'objectif de ce TP était d'implémenter le jeu des huit dames.

Ce TP se basait sur les opérations bits à bits. J'avais déjà énormément travaillé dur ces opérations lors de ma formation précédente. Je n'ai donc eu aucun problème à implémenter les fonctionnalités qui utilisait le bit à bit. Je garde cependant quelques doutes sur l'efficacité des fonctions que j'ai réalisé.

La principale difficulté que j'ai rencontré était la réalisation de l'attaque en diagonale des dames. L'algorithme qui permet la création des diagonales a été le challenge le plus dur que j'ai rencontré cette année. J'ai peut-être passé deux heures à le terminer.

Malgré cette fonctionnalité, je n'ai pas eu d'autres problèmes pour terminer ce TP.

J'ai malheureusement utilisé un flag pour sortir de la boucle principale. Je ne savais pas comment stopper le programme autrement si l'utilisateur clique sur le bouton.

Autre problème, la fonction qui est appelé lors de la fermeture de la fenêtre fait segfault le programme. Je ne comprends toujours pas pourquoi.

TP n°14 :

Le but de ce TP était de créer un programme qui trouve les 10 fichiers les plus volumineux à partir d'un point donné.
J'ai eu du mal à comprendre comment faire interagir la fonction passée en argument à ftw avec mes structures de données. J'ai donc eu recours à une variable globale. Je n'ai pas trouvé d'autres méthodes.

Mise à par ça, ce TP ne m'a pas trop posé de problèmes.

Je n'ai pas créé le paquet Debian. Je suis sur ArchLinux, j'ai essayé de faire le paquet pour Arch, mais j'ai abandonné par manque de temps.

Les fichiers normaux de ce TP se trouvent dans le répertoire Src.

Je pense que j'aurais aussi pu mieux organiser mon code en le sous-divisant en plus de fonctions.

TP n°15 :

L'objectif de ce TP était de de créer une calculette en polonais inversé en utilisant readline.

Ce TP ne m'a pas posé trop de problèmes. Juste quelques difficultés à lire convenablement la ligne renvoyée par readline.

Mise à par ça j'ai essayé de tout bien free, mais Valgrind m'indique que de nombreuses allocations n'ont pas étés libérés. Je suppose qu'elles sont liés à la fonction readline et aux fonction de la librairies string.h que j'utilise.

J’ai le même regret sur ce TP que celui sur le TP précédent. J’aurais pu mieux organiser mon code et créer plus de fonctions.

Ensuite, je ne suis pas sûr des solutions que j’ai employés pour déterminer l’opération voulue par l’utilisateur. Sont-elles efficaces ? Je ne sais pas.

Conclusion :

Cette année m’a permis de beaucoup m’améliorer dans le langage C. Je fais beaucoup moins d’erreur qu’auparavant, les SegFault apparaissent moins souvent et j’ai beaucoup plus de facilités à bien présenter mon code. (Sauf pour les deux derniers TP…). Ce module m’a permis de me rendre compte que j’aime comprendre comment un système fonctionne et adapter mes programmes en conséquences. Je me suis rendu compte que l’optimisation et la compilation m’intéresse particulièrement. C’est donc très probablement dans ces domaines que je continuerai mes études à la fin de ce diplôme.