Rapport de Projet : C-WIRE

Membres du Groupe :

- COLMONT Arthur
- ARCHI Adame
- ERRABHI Amine

Introduction :
Ce rapport présente les résultats du projet « C-WIRE », réalisé par le groupe MI-2_E. Ce projet a été mené sur une période d'environ trois semaines, à la fois durant les séances de TD qui y étaient consacrées et à distance.

Planning :

Première Semaine :
Durant cette première semaine, après avoir créé et bien organisé le dépôt GitHub, nous avons décidé de répartir les tâches afin d'être les plus efficaces et rapides possible. Arthur, qui se sentait plus à l'aise avec le Shell, a choisi de travailler sur cette partie, qui nous semblait la plus importante. De leur côté, Adame et Amine se sont occupés de la partie C.
Les progrès réalisés au cours de cette première semaine étaient relativement flous : nous n'étions pas certains que nos efforts allaient aboutir. Cependant, nous avons tout de même avancé de manière significative en posant les bases du projet.

Deuxième Semaine :
Au début de la deuxième semaine, nous avons décidé de relier la partie C et la partie Shell afin de tester si notre programme fonctionnait correctement. C'est à ce moment-là que nous avons rencontré de nombreuses erreurs :

- La partie C recevait des informations qu'elle ne pouvait pas traiter correctement.
- La partie Shell n'effectuait pas les tris souhaités.

Durant cette semaine, chacun de nous a travaillé à corriger ces erreurs et à ajuster certaines lignes de code pour obtenir un programme fonctionnel. Chaque fois qu'un problème était résolu, nous en informions les autres membres du groupe afin que tout le monde reste au courant des avancées et des points restant à corriger.

Dernière Semaine :
Dès le début de la dernière semaine, nous avions un programme fonctionnel, mais son temps d'exécution était beaucoup trop long. Le problème se situait dans la partie Shell. Nous avons cherché des moyens de réduire au maximum le temps d'exécution du programme. Nous avons d'abord essayé de remplacer certaines boucles par la fonction AWK, mais sans succès, car nous ne savions pas encore bien l'utiliser.
Bloqués, nous avons demandé l'aide de notre professeur (M. Grignon), qui nous a expliqué le fonctionnement d'une autre commande : GREP. Grâce à cette fonction, nous avons pu réduire le temps d'exécution de cinq minutes à environ vingt secondes.
Une fois ce problème réglé, nous avons pu nous concentrer sur les détails et certains

bonus, comme la génération de graphiques. C'est également durant cette semaine que nous avons commencé à rédiger le rapport et le fichier README.

Limitations et Difficultés Rencontrées :

Compréhension du Sujet :
La première difficulté a été de comprendre le sujet dans son ensemble, notamment les différentes fonctions nécessaires pour relier deux parties écrites dans des langages différents. De plus, il nous a fallu trier les bonnes informations et déterminer par où commencer avec un sujet d'une douzaine de pages, dense en informations.

Les Tris :
La gestion des tris a été le principal problème rencontré durant ce projet. Initialement, notre idée de tri fonctionnait mais nécessitait un temps d'exécution considérable (plusieurs minutes). Or, le programme devait idéalement s'exécuter en moins de 30 secondes.
Trouver une solution pour optimiser et réduire ce temps d'exécution a été une tâche longue et difficile. Finalement, grâce à la commande GREP, nous avons réussi à réduire ce temps en ayant cependant un programme plutôt lent (entre 30 et 90 secondes).

Ce qui marche :

Programme C
- Compilation (Amine et Adame)
- Entrée des données dans le programme C via la lecture d'un fichier (Amine et Adame)
- Création de l'arbre AVL (Amine et Adame)
- Insertion des stations dans l'arbre AVL (Amine et Adame)
- Tri des données par AVL (Amine et Adame)
- Calcul de la consommation totale des composants de chaque station (Amine et Adame)
- Ecriture dans un fichier de la consommation totale des composants de chaque station du type choisi (Amine et Adame)

Shell
- Vérification des arguments en entrée et de si ces arguments sont cohérents entre eux (ex : hva all impossible) (Arthur)
- Sélection de la centrale dans les arguments (Arthur)
- Vérification de la présence de l'exécutable dans CodeC (Arthur)
- Affichage du menu help -h (Arthur)
- Vérification et création des dossiers nécessaires au programme notamment tmp et graphs (tous)
- Tri par capacité de chacune des stations pour le fichier final dans le shell (Amine et Adame)
- Affichage du temps en secondes pour le traitement total (Arthur)
- minmax lv all et tri par différence entre la consommation et la capacité (Amine et Adame)
- Bonus : Graphe pour lv all (Arthur) Aide : documentation Gnuplot

Explication dossier tests :

Le dossier test contient les résultats des exécutions précédentes. Elle contient notamment à titre d'exemple également les fichiers temporaires (data_to_process.csv, data_process.csv, sortdiff.csv et

sortload.csv) et le graphique de l'option lv all 1 soit le fichier avec la consommation totale de tous les composants des stations lv de la centrale 1 en colonne 3 reproductible avec la commande : bash c-wire.sh (chemin relatif par rapport à c-wire.sh)/c-wire_v25.dat lv all 1

Il contient enfin les fichiers finaux des différentes exécutions comme lv comp, hva comp…
Lorsque le programme est lancé, c'est ici que les fichiers sont stockés. En cas de fichier avec le nom (soit les mêmes options choisies, par exemple on prend l'option lv comp alors qu'il existe dans le fichier tests lv_comp.csv) , le fichier le plus ancien est remplacé par le nouveau fichier.
De plus, le séparateur de nos fichiers csv est ':' et pour plus de précision sur les fichiers temporaires leur utilité est indiquée dans les commentaires du programme shell.

Temps d'exécution du programme moyen :
A noter que les temps peuvent être plus ou moins long car fait sur un autre pc que ceux pour l'évaluation d'où les fourchettes larges :
hvb comp : 1 seconde
hva comp : 11-25 secondes
lv comp : 15-30 secondes
lv indiv : 30-80 secondes
lv all : 30-80 secondes

Conclusion :
Ce projet nous a permis d'approfondir nos connaissances sur l'interaction entre le langage C et les scripts Shell. Malgré les défis rencontrés, notamment en matière de tri et d'optimisation, nous avons réussi à produire un programme fonctionnel et rapide. Cette expérience a également renforcé notre esprit d'équipe, la communication ayant été un élément clé pour résoudre les problèmes rencontrés.