CY TECH

DÉCEMBRE 2024

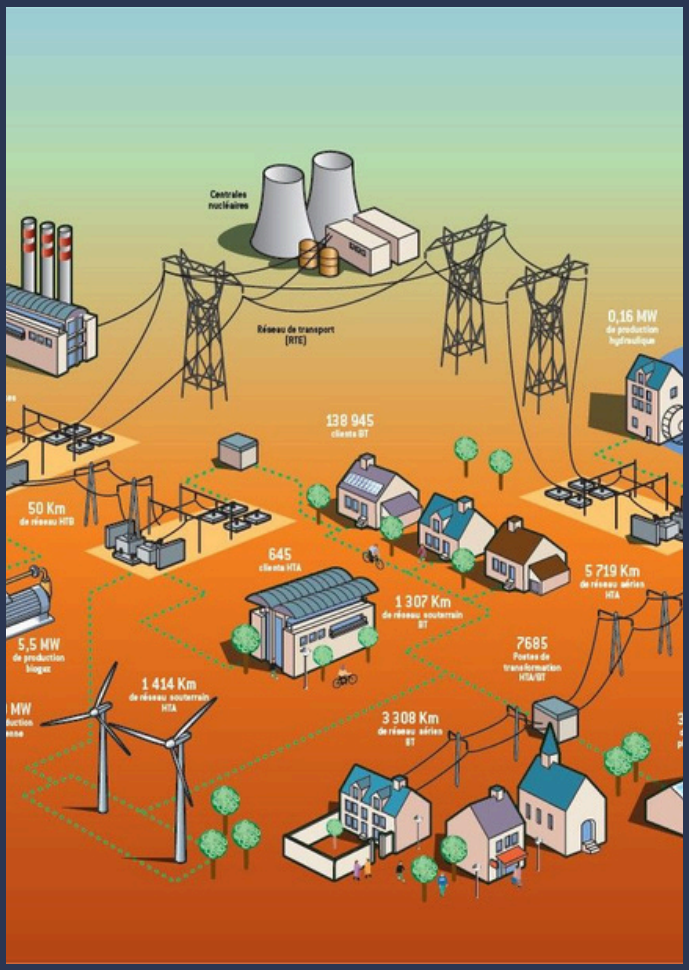
# RAPPORT C_WIRE


**PRESENTED BY**

Rispot Dorian, Houidi Adem, Métaireau Adam

# SOMMAIRE

# INTRODUCTION




C_wire est un programme développé principalement en langage C, conçu pour permettre aux utilisateurs d'effectuer une synthèse des données d'un système de distribution d'énergie électrique.

Grâce à ce programme, un utilisateur souhaitant se concentrer sur une centrale spécifique peut obtenir des données détaillées concernant une station HVB.

En complément, C_wire génère également un graphique, offrant une visualisation claire et accessible des résultats, facilitant ainsi leur compréhension et leur analyse.

# RÉPARTITION DES TÂCHES

Dorian :

- Développement du script Shell, qui assure le filtrage des données et la gestion des entrées utilisateur.
- Rédaction du fichier README, détaillant le fonctionnement et l'utilisation du programme.

Adam :

- Programmation en langage C, avec un accent particulier sur la logique de tri et de gestion des données.
- Création et maintenance du Makefile, garantissant la compilation efficace et structurée du projet.

Adam :

- Programmation en langage C, avec un accent particulier sur la logique de tri et de gestion des données.
- Création et maintenance du Makefile, garantissant la compilation efficace et structurée du projet.

# PLANNING DE RÉALISATION

Pour pouvoir atteindre les objectifs de notre projet dans les délais, nous avons construit un planning de travaille, tout en restant présent face aux imprévus.

- Séance à l'école:
  - Chaque mercredi de 16h30 à 18h et chaque jeudi de 8h30 à 11h45, nous avons consacré nos séances de cours à l'avancement du projet, en sollicitant l'aide de notre enseignant lorsque nécessaire. Ces créneaux nous permettaient de valider certains choix techniques et de commencer à résoudre les nouveaux problèmes rencontrés.
- Séances de travail en équipe :
  - En dehors des cours, nous nous réunissions sur Discord pour travailler au minimum deux heures chaque soir, y compris le samedi. Ces appels nous ont permis de progresser régulièrement tout en restant informés des avancées de chacun et des nouveautés implémentées dans le code.
- Phases de développement :
  a. Nous avons choisi de commencer par le développement du Shell, qui était une base nécessaire pour tester notre programme. Cette étape était primordiale pour garantir une structure solide.
  b. Une fois le Shell suffisamment fonctionnel, nous avons concentré nos efforts sur le programme en C, en priorisant les fonctionnalités de filtrage et le travail sur les AVL (arbres binaires équilibrés).
  c. Troisième phase (Bonus) : Lorsque les fonctionnalités principales étaient presque finalisées, nous avons commencé à travailler sur les bonus. Chacun proposait ses idées sur notre groupe Discord, et ensemble, nous mettions en forme et intégrions ces nouvelles fonctionnalités, telles que l'ajout de graphiques ou d'améliorations esthétiques comme les couleurs.

# FONCTIONNALITÉS IMPLÉMENTÉES

1. Lecture et Traitement des Données
- Lecture d'un fichier de données :
  - Le programme ouvre un fichier texte contenant des données structurées avec des délimiteurs (;).
  - La première ligne est ignorée pour ne pas inclure l'en-tête dans le traitement.
- Analyse des lignes :
  - Chaque ligne du fichier est découpée en segments grâce à la fonction strtok (C).
  - Les segments contiennent des informations telles que :
    - ID de la station.
    - Type de consommateur (individuel ou entreprise).
    - Charges, capacités, et autres métriques liées aux stations.
- Construction d'un AVL :
  - Les données traitées sont stockées dans une structure d'arbre AVL pour permettre une organisation et une recherche efficace.

2. Export des Données
- Fichiers CSV pour les résultats :
  - Les données triées et filtrées sont exportées dans un fichier CSV (tests/<station>_<type>_<id>.csv) pour une analyse approfondie.
  - Les en-têtes de fichier incluent les colonnes suivantes : Station ID, Capacity, et Load.
- Fichier pour graphiques :
  - Un fichier spécifique (Graphs/graphique.csv) est généré avec une structure adaptée pour créer des graphiques à partir des données analysées.

3. Gestion et Compilation du Code
- Compilation dynamique via un Makefile :
  - Le script Shell vérifie et compile le programme C en utilisant un Makefile, garantissant une gestion simple des dépendances et des mises à jour.
- Lancement du programme C :
  - Une fois compilé, l'exécutable est exécuté avec les arguments fournis pour produire les fichiers de sortie.

4. Fonctionnalités Avancées dans le Shell
- Vérification des arguments :
  - Le script Shell valide les paramètres fournis par l'utilisateur :
    - Type de station : hvb, hva, ou lv.
    - Type de consommateur : comp, indiv, ou all.
- Gestion des répertoires :
  - Les dossiers Graphs et temp sont vérifiés et nettoyés ou recréés pour chaque exécution.
  - Cela garantit une structure de travail propre pour la génération des graphiques.
- Aide contextuelle :
  - Une option -h affiche des instructions détaillées sur l'utilisation du programme.

5. Création de Graphiques
- Les données exportées dans le fichier Graphs/graphique.csv sont prêtes pour être utilisées dans des outils de visualisation.
- Ces graphiques permettent une interprétation visuelle des charges et capacités des stations selon les types de consommateurs.

6. Optimisation et Sécurité
- Vérification des erreurs :
  - Gestion robuste des erreurs lors de l'ouverture des fichiers et de la validation des arguments.
  - Le programme s'arrête avec un message d'erreur clair en cas de problème (fichier manquant, paramètres invalides, etc.).
- Nettoyage des ressources :
  - Les arbres AVL sont libérés après usage pour éviter les fuites de mémoire.

# LIMITATIONS FONCTIONNELLES

-Compatibilité restreinte des stations et consommateurs

Les stations HV-B et HV-A ne peuvent traiter que des consommateurs de type entreprise (comp). Les options "all" ou "indiv" sont interdites pour ces stations.

-Validation des entrées :

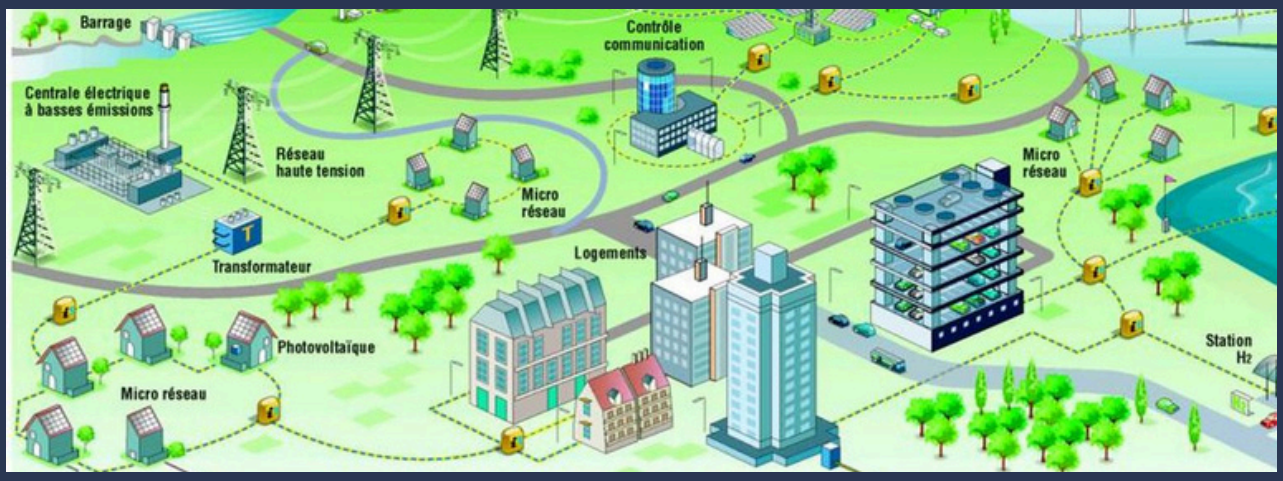
Les stations acceptées sont hvb, hva, et lv, et les types de consommateurs doivent être comp, indiv, ou all. Toute valeur incorrecte est rejetée.

-Génération de fichiers :

Les fichiers CSV générés suivent une structure définie et sont écrasés à chaque exécution. Aucune gestion des versions ou des sauvegardes n'est prévue.

- Compilation et exécution :

La génération de l'exécutable via le Makefile est obligatoire. En cas d'échec, le programme ne peut pas fonctionner.



# CONCLUSION

Le programme permet de traiter et analyser des données provenant de stations électriques, offrant une interface via un script shell.

Il prend en charge la lecture de fichiers, le stockage dans une structure AVL, l'exportation en CSV et la génération de graphiques. L'utilisateur peut facilement spécifier les paramètres de traitement, avec une gestion dynamique des erreurs.

Le programme est robuste avec des vérifications d'entrées et des limitations pour éviter les erreurs, tout en offrant une certaine flexibilité. L'ajout de la génération graphique améliore la visualisation des données.

Malgré certaines limitations (comme le nombre de stations et de consommateurs), le programme est efficace pour organiser et exporter des données. Des améliorations peuvent être apportées pour étendre ses fonctionnalités et sa flexibilité.

CY TECH

https://github.com/Dorianrst/C_WIRE