v1.4 Projet C-Wire

FILIERE préING2 • 2024-2025
AUTEURS E.ANSERMIN – R.GRIGNON
E-MAILS eva.ansermin@cyu.fr – romuald.grignon@cyu.fr

DESCRIPTION GENERALE

- Ce projet vous demande de réaliser un programme permettant de faire la synthèse de données d'un système de distribution d'électricité.
- Pour cela, vous avez à votre disposition un fichier .csv contenant un vaste ensemble de données détaillant la distribution d'électricité en France, depuis les centrales électriques, à travers les sous-stations de distribution, jusqu'aux entreprises et particuliers, qui sont les clients finaux

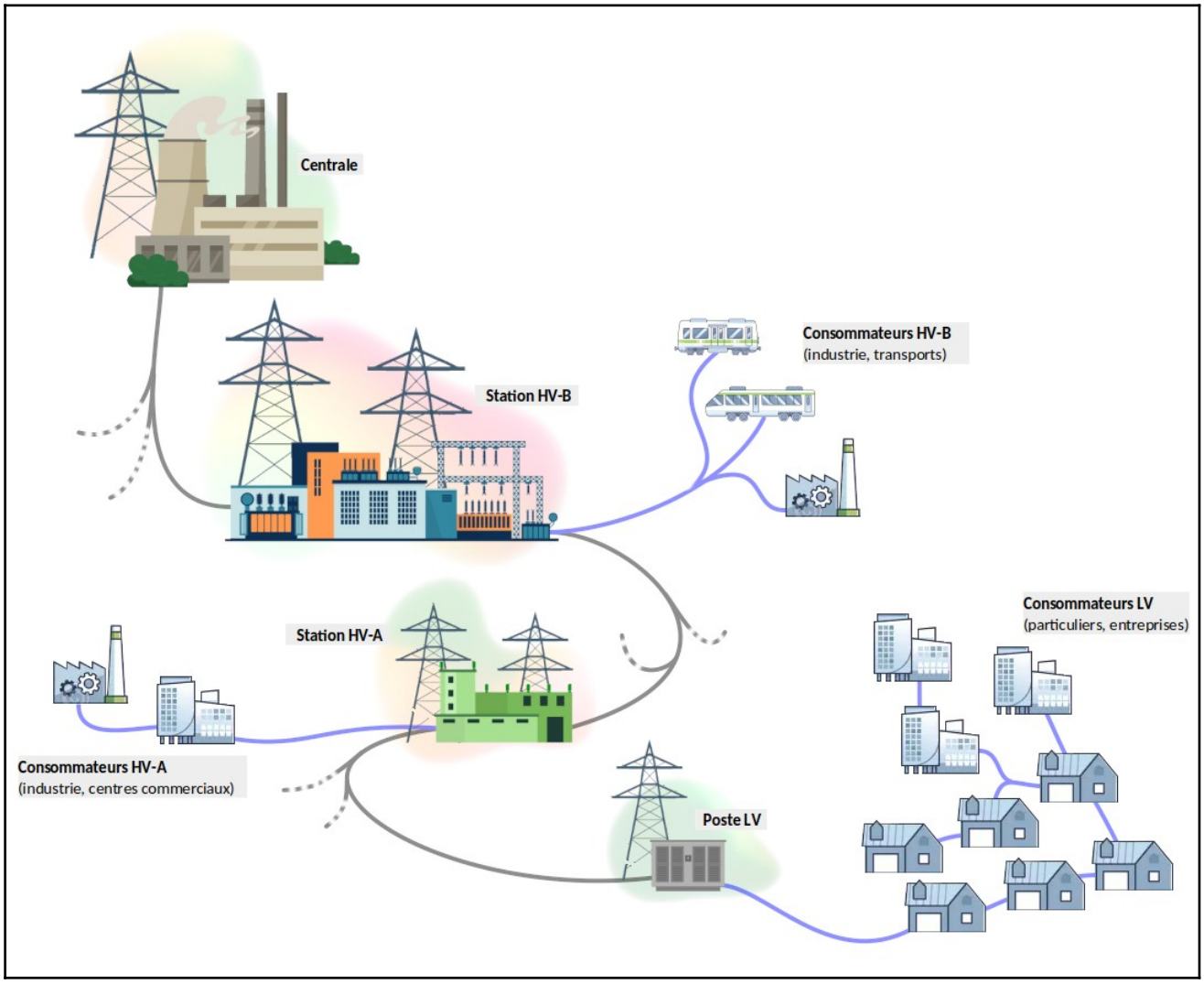
- Les données fournies sont factices mais les ordres de grandeur sont équivalents à celles d'un pays qui ferait 1/4ème de la France métropolitaine, en termes de production, consommation et nombre de consommateurs.

- Votre travail est de filtrer et traiter ces données au moyen d'un script shell (partie filtrage) et d'un programme C (partie traitement/calcul).

DISTRIBUTION D'ENERGIE

- Chaque **centrale** envoie cette énergie électrique avec une très haute tension (~400kV) vers plusieurs **sous-stations HV-B** (High Voltage B) qui abaissent la tension (> 50kV) et fournissent en énergie une grande zone géographique.
  Chaque station HV-B transfère l'énergie vers plusieurs **sous-stations HV-A** (High Voltage A) qui abaissent encore cette tension (> 1000V) pour fournir l'énergie à un niveau plus régional.
  Puis chaque sous-station HV-A transfère l'énergie vers plusieurs **postes LV** (Low-Voltage) qui s'occupent de distribuer l'énergie aux particuliers ou aux petites entreprises (tension 230V).

- On dispose donc d'une topologie qui est un arbre enraciné, donc un graphe orienté sans cycle à partir de **chaque** centrale. Le schéma d'un arbre pour **une** centrale est le suivant :



Un arbre représentant **une** centrale est donc composé de :

- **La Centrale (racine de l'arbre)**

  La centrale électrique est la **racine** de l'arbre, qui alimente plusieurs stations HV-B. Ces dernières, réparties dans différentes zones, redistribuent l'énergie vers d'autres stations intermédiaires ou directement vers des consommateurs très énergivores.

- **Les stations intermédiaires hiérarchisées**

  Stations HV-B (nœuds principaux de premier niveau)
  - Ces stations sont les premières à recevoir l'énergie de la centrale. Il peut y avoir plusieurs stations HV-B, chacune couvrant une zone géographique ou un secteur spécifique.
  - Elles alimentent :
    - Des **consommateurs très énergivores** (comme de grosses entreprises exploitant beaucoup d'électricité, ex. : SNCF, aciéries, usines).
    - Des **stations HV-A**, qui prennent en charge des consommateurs intermédiaires.

Stations HV-A (nœuds secondaires)

- Chaque station HV-A est alimentée par une station HV-B et dessert des **entreprises moyennes ou grosses, ayant une consommation modérée** (comme des centres commerciaux ou des zones industrielles moins énergivores).
- Il peut y avoir plusieurs stations HV-A.

Postes LV (stations basse tension)

- Chaque poste LV est alimenté par une station HV-A. Il transforme l'énergie pour la rendre compatible avec les faibles besoins des utilisateurs finaux (particuliers, petites entreprises).
- Il existe de nombreux postes LV, souvent proches des zones résidentielles et des petites zones commerciales.

➢ **Consommateurs finaux (feuilles de l'arbre)**

- Les **grosses entreprises énergivores** sont directement connectées à des stations HV-B.
- Les **entreprises moyennes/grosses** et certains centres commerciaux sont desservis directement par des stations HV-A.
- Les **particuliers et petites entreprises** sont alimentés via les postes LV. Ce sont aussi des **feuilles** de l'arbre, où l'énergie est consommée sans redistribution ultérieure.

Chaque poste LV est relié à une seule station HV-A, chaque station HV-A est reliée à une seule station HV-B, et chaque station HV-B est reliée à une seule centrale. Il y a plusieurs centrales.

➢ Dans l'intégralité des données à disposition, nous avons :

- 5 centrales
- ~118 sous-stations HV-B
- ~512 sous-stations HV-A
- +180k postes LV
- +1.25M consommateurs (entreprises)
- +7.6M consommateurs (particuliers)

➢ Chacun de ces acteurs (distributeur ou consommateur d'énergie) sera associé à un numéro d'identifiant qui est unique dans sa catégorie.

FORMAT DES DONNEES

➢ Le fichier **DATA_CWIRE.csv** est un fichier contenant les informations de distribution et de consommation de tous les acteurs cités au-dessus.
Au vu de leur nombre, ce fichier est très lourd et il n'est pas envisageable de le traiter manuellement ou à l'aide d'un tableur
Cette partie explicite son contenu afin de pouvoir automatiser son traitement.

- Le fichier de données est un fichier CSV contenant les noms de colonnes suivants :
  - ***Power* plant** : identifiant d'une centrale électrique (producteur)
  - **HV-B Station** : identifiant d'une station HV-B
  - **HV-A Station** : identifiant d'une station HV-A
  - **LV Station** : identifiant d'un poste LV
  - **Company** : identifiant d'une entreprise (consommateur)
  - **Individual** : identifiant d'un particulier (consommateur)
  - **Capacity** : quantité d'énergie produite par une centrale ou transférée par une station HV-B, HV-A ou LV (en kWh)
  - **Load** : quantité d'énergie consommée par le consommateur final entreprise ou particulier (en kWh)

- Chaque ligne du fichier représente un acteur ou un lien entre 2 acteurs de ce réseau de distribution (centrale, HV-B, HV-A, LV, entreprise ou particulier)

- De nombreuses données dans ce fichier seront vides. Par exemple, les acteurs fournisseurs d'électricité incluent les centrales, les stations HV-B, HV-A et les postes LV. Dans le fichier, les lignes correspondant à ces instances contiendront une valeur dans la colonne « Capacity » (capacité de production/transfert), tandis que la colonne « Load » (indiquant la consommation) sera vide.
  À l'inverse, pour les lignes relatives aux consommateurs d'électricité (entreprises et particuliers), la colonne « Load » sera renseignée, et la colonne « Capacity » restera vide.
- Concernant les colonnes d'identifiants, pour chaque ligne, la colonne correspondant au type d'acteur sera renseignée avec le numéro d'identifiant de l'acteur concerné. Par exemple, une ligne relative à une station HV-A aura la colonne « HV-A Station » remplie avec l'identifiant de cette station. De plus, l'identifiant de la station qui lui fournit directement de l'énergie est également indiquée (voir les exemples ci-dessous). Cela permet de pouvoir remonter sur la provenance de l'énergie distribuée

- Exemples de lignes du fichier CSV
  - **Centrale**
    `1;-;-;-;-;-;17972235418;-`
    - permet de définir la racine d'un des arbres
    - contient l'identifiant de la centrale (1)
    - contient la production maximale (17972235418 kWh)

  - **Station HV-B**
    `3;73;-;-;-;-;554172263;-`
    - permet de définir un nœud HV-B
    - contient l'identifiant de la centrale parente (3)
    - contient l'identifiant de la station HV-B (73)
    - contient la capacité de transfert (554172263 kWh)

➔ **Consommateur HV-B** (entreprise)
`1;17;-;-;23;-;-;123823310`
- permet de définir un consommateur HV-B (feuille)
- contient l'identifiant de la centrale racine (1)
- contient l'identifiant de la station HV-B parente (17)
- contient l'identifiant du consommateur (23)
- contient la consommation du client (123823310 kWh)

➔ **Station HV-A**
`2;31;119;-;-;-;284930294;-`
- permet de définir un nœud HV-A
- contient l'identifiant de la centrale racine (2)
- contient l'identifiant de la station HV-B parente (31)
- contient l'identifiant de la station HV-A (119)
- contient la capacité de transfert (284930294 kWh)

➔ **Consommateur HV-A** (entreprise)
`2;-;117;-;2211;-;-;3389540`
- permet de définir un consommateur HV-A (feuille)
- contient l'identifiant de la centrale racine (2)
- contient l'identifiant de la station HV-A parente (117)
- contient l'identifiant du consommateur (2211)
- contient la consommation du client (3389540 kWh)

➔ **Poste LV**
`3;-;274;103733;-;-;112040;-`
- permet de définir un nœud LV
- contient l'identifiant de la centrale racine (3)
- contient l'identifiant de la station HV-A parente (274)
- contient la capacité de transfert (112040 kWh)

➔ **Consommateur LV** (entreprise)
`1;-;-;10520;69817;-;-;20659`
- permet de définir un consommateur LV (feuille)
- contient l'identifiant de la centrale racine (1)
- contient l'identifiant du poste LV parent (10520)
- contient l'identifiant du consommateur (69817)
- contient la consommation du client (20659 kWh)

➔ **Consommateur LV** (particulier)
`1;-;-;10520;-;411970;-;5270`
- permet de définir un consommateur LV (feuille)
- contient l'identifiant de la centrale racine (1)
- contient l'identifiant du poste LV parent (10520)
- contient l'identifiant du consommateur (411970)
- contient la consommation du client (5270 kWh)

OBJECTIF

- Votre projet devra permettre d'analyser les stations (centrales, stations HV-A, stations HV-B, postes LV) afin de déterminer si elles sont en situation de surproduction ou de sous-production d'énergie, ainsi que d'évaluer quelle proportion de leur énergie est consommée par les entreprises et les particuliers.

- Cet objectif nécessite que :
  - que l'utilisateur puisse définir simplement ses **paramètres** d'observation (le type de station qu'il souhaite analyser) et les catégories de clients à examiner.
  - que votre projet **filtre** les informations pertinentes dans le fichier .csv.
  - que votre solution **calcule** la somme de la consommation de tous les clients associés à ces stations.
  - que les résultats soient **sauvegardés** de manière structurée (voir la description des fichiers de sortie ci-après).

- **Fichiers de sorties**

  Les fichiers de sortie attendus doivent contenir sur chaque ligne, 3 colonnes :

  - identifiant de la station
  - capacité en kWh
  - consommation en kWh

  Le fichier de sortie attendu doit contenir une première ligne qui indique le contenu des colonnes en fonction des paramètres qui sont donnés :

  - `Station HVB` ou `Station HVA` ou `Station LV`
  - `Capacité`
  - `Consommation (entreprises)` ou `Consommation (particuliers)` ou `Consommation (tous)`

  Ces fichiers seront des fichiers au format CSV avec un caractère ' :' (deux-points) comme séparateur, et porteront le nom de l'option de la station (hvb, hva, lv) suivi d'un underscore '_' suivi de l'option des consommateurs (comp, indiv ou all). Dans le cas où il y a un filtrage supplémentaire par numéro de centrale, alors le numéro de centrale sera rajouté à la suite.

  Ex : ***hva_comp.csv*** ou ***lv_all.csv*** ou ***lv_all_2.csv***

  Ces données seront triées par Capacité croissante (la plus petite capacité en premier).

- Etant donné le nombre très important de postes LV, dans le cas des options ***lv all*** le script devra effectuer un traitement supplémentaire qui sera de stocker dans un autre fichier seulement les 10 postes LV avec le plus de consommation et les 10 postes LV avec le moins de consommation.
  Ces informations concernant les 20 postes LV seront triées par quantité absolue d'énergie consommée en trop (du poste le plus chargé à celui le moins chargé). En d'autres termes, on calculera la différence entre la capacité totale et la consommation totale, et le tri sera fait en fonction de la valeur croissante de cette différence.
  Le nom de ce fichier sera ***lv_all_minmax.csv***.

- Pour répondre à cet objectif global, vous allez devoir crer un script Shell ainsi qu'un programme C dont les objectifs plus précis vous sont détaillés dans les sections suivantes.

SCRIPT SHELL

- Votre script Shell sera nommé **c-wire.sh**
  Votre script Shell va prendre en paramètre le chemin du fichier CSV d'entrée contenant les données. Il prendra également d'autres paramètres qui seront les choix des traitements à faire
  Les différents traitements en fonction des paramètres sont décrits plus loin dans cette section.
- Les différents paramètres du script sont les suivants (**dans l'ordre**) :
  - **chemin du fichier** de données
    - obligatoire
    - indique l'endroit où se trouve le fichier d'entrée
  - **type de station** à traiter
    - obligatoire
    - valeurs possibles :
      hvb (high-voltage B)
      hva (high-voltage A)
      lv (low-voltage)
  - **type de consommateur** à traiter
    - obligatoire
    - valeurs possibles :
      comp (entreprises)
      indiv (particuliers)
      all (tous)
    - ATTENTION : les options suivantes sont interdites car seules des entreprises sont connectées aux stations HV-B et HV-A :
      hvb all
      hvb indiv
      hva all
      hva indiv

- **Identifiant de centrale** :
  - optionnel
  - filtre les résultats pour une centrale spécifique
  - si cette option est absente, les traitements seront effectués sur toutes les centrales du fichier

- **Option d'aide (*-h*) :**
  - optionnel et prioritaire
  - si présente, toutes les autres options sont ignorées, quelle que soit la position de l'option d'aide
  - affiche une aide détaillée sur l'utilisation du script (description, fonctionnalités, options possibles, ordre des paramètres, etc…)

- Le script SHELL doit s'assurer que toutes les options obligatoires sont présentes, et que leurs valeurs sont cohérentes, avant de lancer un traitement. En cas de mauvaises options, un message d'erreur doit s'afficher avec le détail de l'erreur. En plus du message d'erreur, l'aide doit aussi s'afficher en dessous, comme si l'utilisateur avait tapé l'option ***-h***

- Le script Shell devra vérifier la présence de l'exécutable C sur le disque dur et, si ce dernier n'est pas présent, devra lancer la compilation et vérifier que cette dernière s'est bien déroulée. Si ce n'est pas le cas, un message d'erreur doit être affiché. Une fois la compilation effectuée il pourra effectuer le traitement demandé en argument.
- Le script Shell vérifiera la présence du dossier **tmp** et **graphs** (cf. *Informations Complémentaires*) : si ces derniers n'existent pas il devra les créer. Si le dossier tmp existe déjà, il devra le vider **avant** l'exécution des traitements.

- La durée de chaque traitement à effectuer devra être affichée en **secondes** à la fin. Peu importe si le script s'est déroulé correctement ou a rencontré une erreur, les durées devront être affichées systématiquement. Les durées ne comprendront pas la phase de compilation du programme C ou la création des dossiers effectuée au départ. Les temps affichés doivent être des temps utiles de traitement des données. Si le programme échoue sur les valeurs des paramètres, aucun traitement n'aura été lancé le temps affiché sera donc de 0.0sec.

- Une fois les traitements de données terminés, le script Shell devra créer des fichiers et/ou des graphiques qui contiendront les données de sortie du traitement. Pour les graphiques vous pourrez utiliser le programme ***GnuPlot*** *(cf. Bonus)*.

- Lorsque le script demande un type de station (hvb, hva, lv), le but final sera de **créer un fichier** contenant une liste de ces stations avec la valeur de capacité (la quantité d'énergie qu'elle peut laisser passer) et la somme des consommateurs branchés directement dessus.

- La somme des consommateurs ne tiendra compte que de ceux qui sont demandés (comp : entreprises uniquement, indiv : particuliers uniquement, all : tous les consommateurs).
- Pour avoir la somme des consommateurs d'une station, il va falloir construire un **arbre de recherche binaire équilibré**. Ceci est le rôle de la partie de votre projet en C. Ce traitement (la somme) doit être fait avec le programme C **obligatoirement**. Si votre script effectue tout ou partie de cette somme dans le script Shell, votre note sera pénalisée.

PROGRAMME C

- Le but de ce programme est de faire la somme des consommations d'un type de station. Etant donnée les nombreuses données et stations, nous allons passer par un AVL afin de limiter le temps de traitement.
- Chaque nœud de l'AVL représente une station et va donc contenir l'identifiant de la station ainsi que ses différentes données comme sa capacité, ou bien la somme de ses consommateurs qui sera mise à jour au fur et à mesure de la lecture des données par votre programme.
- Le format des données d'entrée du programme C est laissé libre. A vous de voir si vous passez tout le fichier de données, ou si vous voulez le filtrer (lignes / colonnes) à l'aide du script Shell avant de passer les informations au programme C.
- Le format des données de sortie du programme C est également laissé libre du moment qu'il vous permet de récupérer les informations nécessaires à votre traitement.
- Le programme C a pour objectif de calculer la somme des consommations pour une station HV-B, HV-A ou un poste LV, en utilisant obligatoirement un arbre AVL (arbre binaire de recherche équilibré). Une solution générique doit être envisagée pour traiter ces trois types de stations avec un programme (optimiser la transmission des données).
- Toute implémentation utilisant uniquement un ABR non équilibré entraînera une pénalité.
- Le programme C devra retourner un code d'erreur avec une valeur strictement positive si un problème est rencontré, et 0 sinon. Ce programme ne devra jamais s'arrêter de manière inattendue : c'est à vous de bien vérifier que toutes les données que vous traitez sont correctes. Si vous détectez un problème, vous devez stopper le programme et renvoyer un code d'erreur.
- Le code du programme C devra donc être robuste.
- Il vous est également demandé de limiter au mieux la taille mémoire utilisée. Pour ce faire, vous devrez définir des variables dans vos structures ayant le moins d'empreinte mémoire possible, tout en garantissant le fonctionnel demandé bien sûr.
  De plus toutes les allocations dynamiques doivent être libérées avant de terminer le programme C correctement. Par contre si le programme C rencontre une erreur, il n'est pas demandé de libérer toute la mémoire explicitement.
- Enfin, la compilation du programme C devra se faire avec l'utilitaire '*make*' en utilisant un '*Makefile*'. La seule instruction de compilation attendue par votre script Shell est l'appel à l'exécutable 'make'. Pas d'appel direct à gcc depuis votre script.

INFORMATIONS SUPPL.

- **Le fichier de données CSV**

Le fichier « data.csv » fourni contient l'ensemble des données..
C'est un fichier volumineux (+150Mo) avec plus de 5 millions de lignes.

Il va donc être très difficile voire impossible d'avoir une vision d'ensemble précise de ce fichier avec des outils de bureautique classiques. Il va vous falloir passer par un programme informatique pour extraire les données dont vous aurez besoin.

- **Organisation des fichiers de votre projet**

  Le fichier de données d'entrée devra être copié dans un dossier '***input***'
  Le programme C et tous les fichiers qui s'y rapportent (makefile, exécutable, …) devront être situés dans un dossier '***codeC***'
  Les graphiques, si il y en a seront stockés dans des images sur le disque dur dans un dossier '***graphs***'
  Les fichiers intermédiaires nécessaires à votre application (si il y en a) seront placés dans un dossier '**tmp**'.
  Les résultats d'exécutions précédentes seront dans le dossier '***tests***'.
  Le script Shell sera quand à lui placé à la racine de votre projet.

BONUS

- Dans le cas du traitement ***lv all***, il vous est demander en bonus de créer un graphique en barre des 10 postes LV les plus chargés, et les 10 postes LV les moins chargés.
- Ces 20 barres seront affichés avec un visuel explicite au niveau des couleurs (rouge/vert) pour indiquer quelle quantité d'énergie est consommée en trop ou si il y a de la marge.
- Pour créer un graphique dans une image, vous utiliserez l'utilitaire en ligne de commande sous linux ***GnuPlot***
- *Vous pouvez rajouter les bonus de votre choix (autre option, interface, etc.) du moment que le reste du cahier des charges vous semble rempli.*

CRITERES DE NOTATION

- Le rendu du travail sera un **lien github** menant au projet. Inutile d'envoyer les fichiers par email/Teams ou toute autre moyen : le seul livrable attendu et qui sera évalué sera le lien du dépôt git dans lequel doivent se trouver tous les fichiers de votre projet. Avant la date de rendu vous pouvez configurer ce dépôt en « privé » pour ne pas laisser d'autres personnes vous plagier.
  A la date de rendu, ce dépôt devra être **visible publiquement** pour que vos chargés de projet puissent y accéder librement. Tout retard d'accès à ce dépôt public aura un impact négatif sur votre note finale.

- Le dépôt de code contiendra en plus des fichiers de code, un fichier texte **ReadMe** contenant les instructions pour compiler et pour utiliser votre application. Il contiendra aussi un document au format **PDF** présentant la répartition des tâches au sein du groupe, le planning de réalisation, et les limitations fonctionnelles de votre application (la liste de ce qui n'est pas implémenté, et/ou de ce qui est implémenté mais qui ne fonctionne pas correctement/totalement).

- Il vous est demandé d'effectuer une livraison sur le dépôt 1 fois après chaque séance de TD sur le projet au minimum, même si le dépôt de code n'est pas fonctionnel.
  Si vous avez des évolutions au niveau de votre code d'une séance à l'autre, vous devrez avoir effectué un commit la veille de la séance, pour bien fixer l'historique de vos modifications, et pour pouvoir travailler dessus avec votre chargé de TD.
  De fait, au minimum il y aura un commit juste avant et juste après la séance obligatoirement.
  Si vous avez des modifications, même non fonctionnelle elles ne doivent pas rester plusieurs jours sans être stockées sur le dépôt de code, et éviter ainsi une perte catastrophique dans l'hypothèse où votre machine viendrait à tomber en panne. Il est de votre responsabilité d'archiver régulièrement vos modifications sur le dépôt. Aucune perte de code ne pourra être prise en compte si vous n'utilisez pas votre dépôt.
  Cela vous forcera également à développer en équipe avec un dépôt central que vous vous partagez et qui contient les dernière modifications du projet.

- Votre rendu contiendra des exemples d'exécution de votre application. Vous mettrez dans le dossier '*tests*', les images, fichiers intermédiaires et finaux, et vous présenterez ces résultats dans le document PDF cité précédemment. Ces éléments doivent donc être reproductibles en utilisant votre programme.

- Le rendu est un travail de groupe : si des similitudes entre groupes sont trouvées, et/ou si des exemples disponibles sur Internet sont découverts sans être sourcés, une procédure de fraude à un examen pourra être envisagée. Le but pédagogique de ce projet est que vous réalisiez par vous-même ce programme, et que vous maitrisiez l'ensemble du code fourni.

- Le code en langage C sera séparé en **modules** (fichiers .c et .h, sous-dossiers).

  Il ne faut pas que votre programme C soit d'un seul tenant.
- Un fichier **Makefile** sera présent et il permettra de compiler l'exécutable. La première cible permettra de compiler le projet. Ce fichier inclura entre autres une cible '**clean**' qui permet d'effacer les fichiers générés.

- Votre code sera **commenté** (modules, fonctions, structures, constantes, …) et correctement **indenté**.

- Les **symboles** du code (variables, fonctions, types, fichiers, …) seront dans la **même langue** que les **commentaires** (soit tout en anglais, soit tout en français, mais pas de mélange entre les langues utilisées).

- Le programme C et le script Shell doivent respecter toutes les consignes décrites dans ce document.

- Le programme C et le script Shell ne doivent en aucun cas générer d'erreur inattendue. Il ne doit y avoir aucune **erreur de segmentation**, **erreur de syntaxe**, ou de nom de **commande inconnue**, etc.
  Ce critère sera extrêmement punitif sur la note finale : il est donc de votre responsabilité de tester votre programme correctement pour pouvoir corriger ces situations avant la livraison.
- Si une erreur est détectée par votre programme/script, un message d'erreur doit s'afficher pour indiquer la cause à l'utilisateur, et un code retour avec une valeur strictement positive doit être retournée.

- Les structures allouées temporairement dans votre programme doivent être désallouées explicitement avant la fin. La quantité de mémoire non libérée à la fin de l'exécution, sera évaluée. Pensez à appeler la fonction ***free(…)*** quand il faut !
- De plus, la quantité de mémoire vive consommée par votre programme, sera également notée : pensez donc à limiter votre empreinte mémoire. Attention tout de même à faire en sorte que votre programme fonctionne : l'optimisation mémoire reste un plus. La première étape étant de faire quelque chose de fonctionnel.
- Vous disposez d'un fichier de données CSV qui est figé. Il est donc possible pour un groupe d'étudiants de « coder en dur » les résultats attendus. Pour éviter ce cas de triche, il est possible que l'évaluation de votre programme se fasse avec un fichier de données CSV différent du votre (mais similaire en terme de structure, de taille, …). Pensez donc à faire un programme véritablement générique pour éviter une mauvaise surprise lors de l'évaluation.

RESSOURCES UTILES

GitHub
- site Web : https://github.com/

Format CSV
- site Web : https://fr.wikipedia.org/wiki/Comma-separated_values

GnuPlot
- site Web : http://gnuplot.info/