Projet CY-Wildwater

Préing2 MI-4 2025-2026
Groupe I

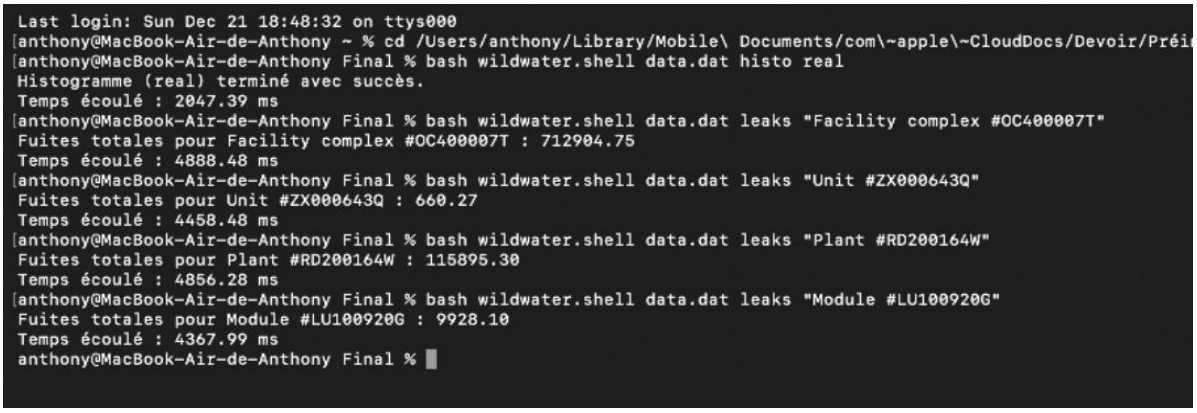
ANCELIN-HARMENIL Anthony
BARTOLO Leo
CAN Axel

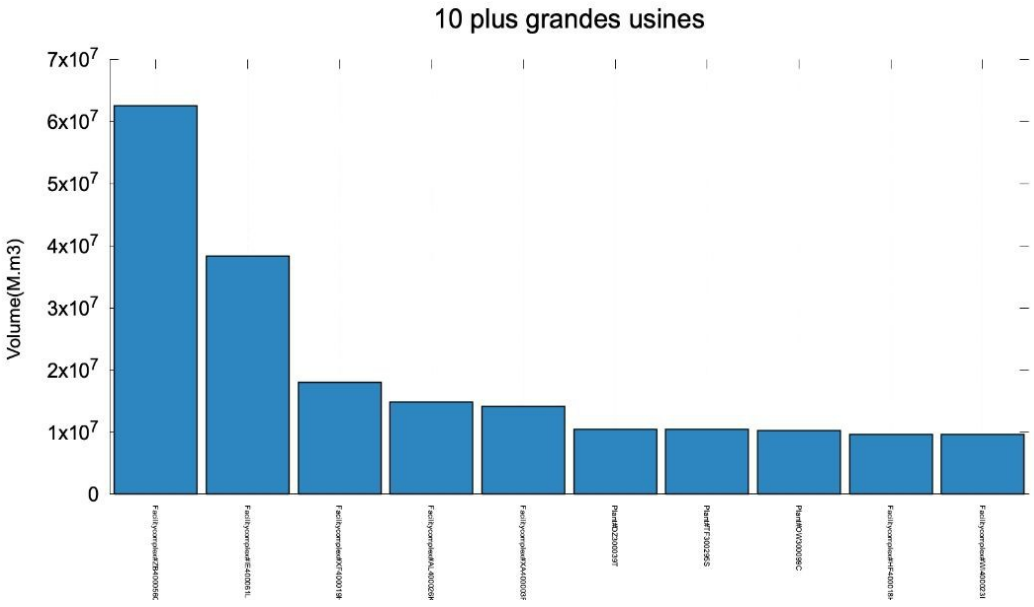
Pour ce projet, les différentes tâches et l'organisation ont étés décidés par Axel, la partie histogramme du C et le makefile ont donc été réalisés par Anthony, la partie fuites du C par Leo, le script shell ainsi que les scripts gnuplots et le document pdf par Axel (photos a la fin du document).

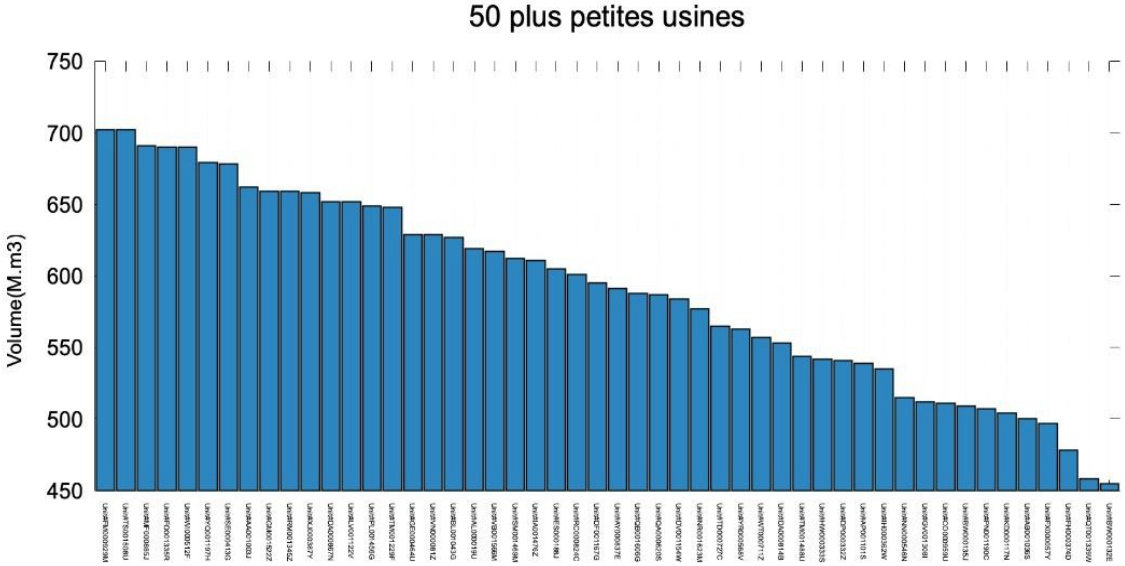
Le projet n'a pas du tout été réalisé de manière homogène dans le temps, et la plupart du projet a été réalisé durant la dernière semaine. La partie shell et la partie C ont étés réalisés séparément et ont étés liés à la vas vite dû à un manque de temps.

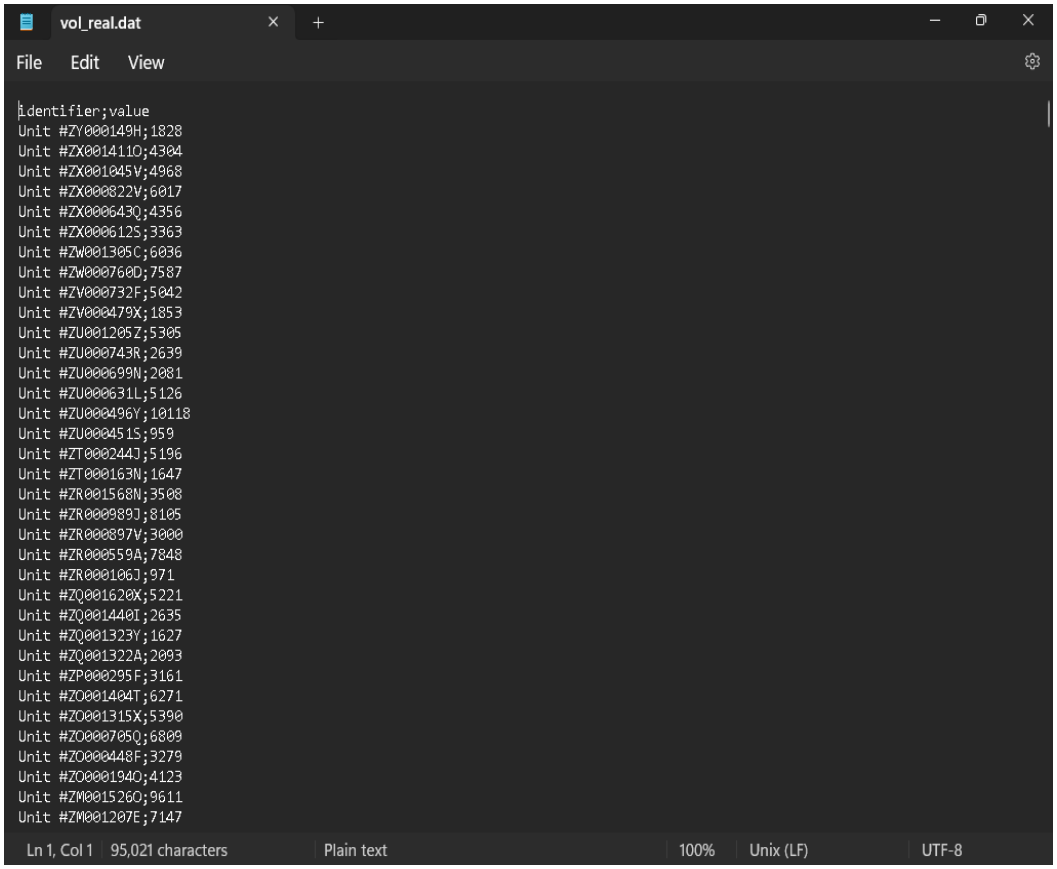
La partie bonus du projet n'a pas été réalisée en raison d'un manque de temps. Pour quelque raison que ce soit le script shell peut générer une erreur lors de la première utilisation mais non lors des suivantes, un problème similaire est présent pour le temps d'exécution du programme C (très long ou très court).
Il manque aussi le fait d'afficher et de rajouter -1 dans le fichier .dat des fuites si l'id de l'usine est incorrect. Les fichiers .dat créés ne contiennent que “identifier;value” sur la première ligne pour chaque type d'histogramme. Les volumes sont dans la mauvaise unité, il faudrait les diviser par 1000 mais cela casse les volumes en dessous de 1 M.m3. Le fichier leaks.dat ne contient pas de première ligne pour indiquer les différentes colonnes.

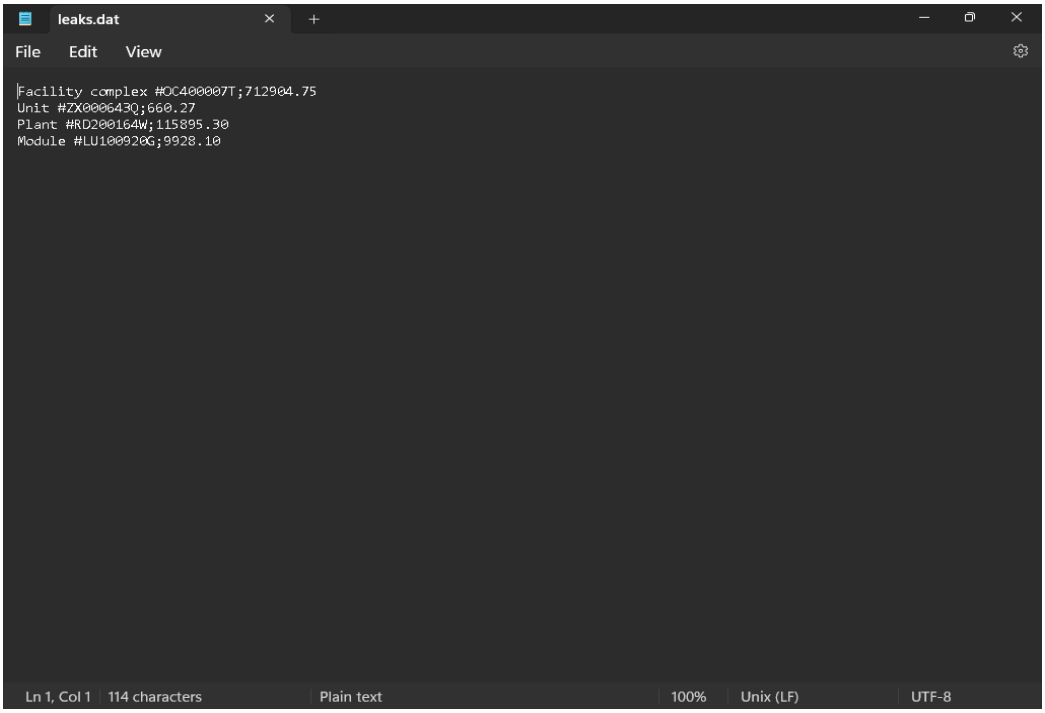
Exemples d'exécution :



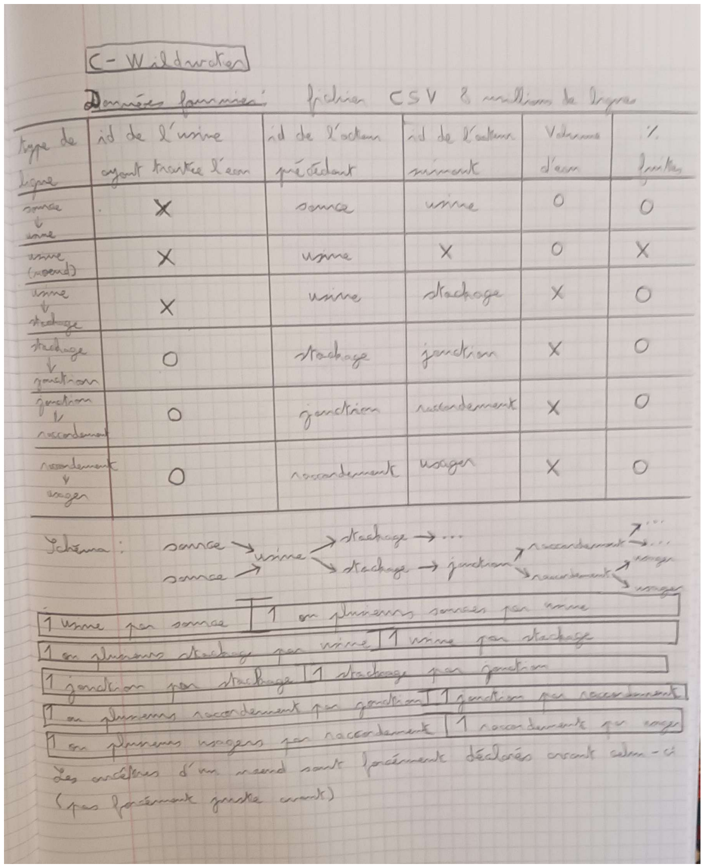

10 plus grandes usines
Volume(M.m3)
7x10^7
6x10^7
5x10^7
4x10^7
3x10^7
2x10^7
1x10^7
0


50 plus petites usines
Volume(M.m3)
750
700
650
600
550
500
450


vol_real.dat
File Edit View
identifier;value
Unit #ZY000149H;1828
Unit #ZX001411O;4304
Unit #ZX001045V;4968
Unit #ZX000822V;6017
Unit #ZX000643Q;4356
Unit #ZX000612S;3363
Unit #ZW001305C;6036
Unit #ZW000760D;7587
Unit #ZV000732F;5042
Unit #ZV000479X;1853
Unit #ZU001205Z;5305
Unit #ZU000743R;2639
Unit #ZU000699N;2081
Unit #ZU000631L;5126
Unit #ZU000496Y;10118
Unit #ZU000451S;959
Unit #ZT000244J;5196
Unit #ZT000163N;1647
Unit #ZR001568N;3508
Unit #ZR000989J;8105
Unit #ZR000897V;3000
Unit #ZR000559A;7848
Unit #ZR000106J;971
Unit #ZQ001620X;5221
Unit #ZQ001440I;2635
Unit #ZQ001323Y;1627
Unit #ZQ001322A;2093
Unit #ZP000295F;3161
Unit #ZO001404T;6271
Unit #ZO001315X;5390
Unit #ZO000705Q;6809
Unit #ZO000448F;3279
Unit #ZO000194O;4123
Unit #ZM001526O;9611
Unit #ZM001207E;7147
Ln 1, Col 1 95,021 characters Plain text 100% Unix (LF) UTF-8


leaks.dat
File
Edit
View
Facility complex #OC400007T;712904.75
Unit #ZX000643Q;660.27
Plant #RD200164W;115895.30
Module #LU100920G;9928.10
Ln 1, Col 1
114 characters
Plain text
100%
Unix (LF)
UTF-8

Annexe :



C-Wildwater

Données fournies : fichier CSV 8 millions de lignes

type de ligne	id de l'usine ayant traitée l'eau	id de l'acteur précédant	id de l'acteur suivant	Volume d'eau	% fuites
source → usine	X	source	usine	O	O
usine (noeud)	X	usine	X	O	X
usine → stockage	X	usine	stockage	X	O
stockage → jonction	O	stockage	jonction	X	O
jonction → raccordement	O	jonction	raccordement	X	O
raccordement → usager	O	raccordement	usager	X	O

Schéma : source → usine ← source ; usine → stockage → … ; usine → stockage → jonction → raccordement → … / usager ; jonction → raccordement → usager / usager

1 usine par source | 1 ou plusieurs sources par usine

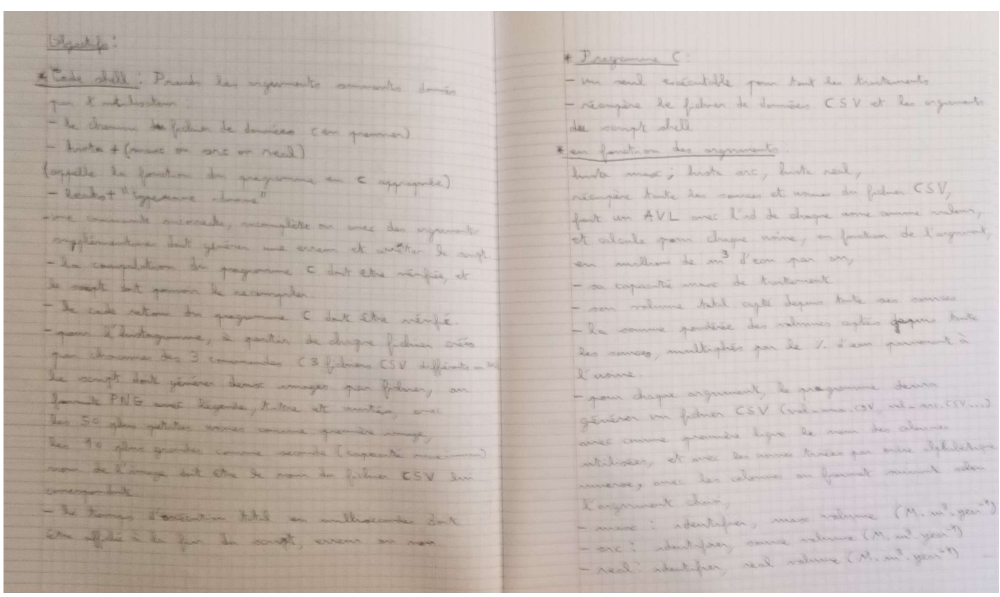
1 ou plusieurs stockage par usine | 1 usine par stockage

1 jonction par stockage | 1 stockage par jonction

1 ou plusieurs raccordement par jonction | 1 jonction par raccordement

1 ou plusieurs usagers par raccordement | 1 raccordements par usager

Les ancêtres d'un noeud sont forcément déclarés avant celui-ci (pas forcément juste avant)



Objectifs :

* Code shell : Prends les arguments suivants donnés par l'utilisateur :
- le chemin du fichier de données (en premier)
- histo + (max ou src ou real)
(appelle la fonction du programme en C appropriée)
- leaks + "identifiant usine"
- une commande incorrecte, incomplète ou avec des arguments supplémentaires doit générer une erreur et arrêter le script.
- la compilation du programme C doit être vérifiée, et le script doit pouvoir le recompiler.
- le code retour du programme C doit être vérifié.
- pour l'histogramme, à partir de chaque fichier créés par chacune des 3 commandes (3 fichiers CSV différents au max) le script doit générer deux images par fichier, au format PNG avec légende, titre et unités, avec les 50 plus petites usines comme première image, les 10 plus grandes comme seconde (capacité maximum) nom de l'image doit être le nom du fichier CSV lui correspondant
- le temps d'exécution total en millisecondes doit être affiché à la fin du script, erreur ou non

* Programme C :
- un seul exécutable pour tout les traitements
- récupère le fichier de données CSV et les arguments du script shell

* en fonction des arguments :
histo max ; histo src, histo real,
récupère toute les sources et usines du fichier CSV, fait un AVL avec l'id de chaque usine comme valeur, et calcule pour chaque usine, en fonction de l'argument, en million de m^3 d'eau par an,
- sa capacité max de traitement
- son volume total capté depuis toute ses sources
- la somme pondérée des volumes captés depuis toute les sources, multipliés par le % d'eau parvenant à l'usine.
- pour chaque argument, le programme devra générer un fichier CSV (vol_max.CSV, vol_src.CSV...) avec comme première ligne le nom des colonnes utilisées, et avec les usines triées par ordre alphabétique inverse, avec les colonnes au format suivant selon l'argument choisi,
- max : identifier, max volume ($M.m^3.year^{-1}$)
- src : identifier, source volume ($M.m^3.year^{-1}$)
- real : identifier, real volume ($M.m^3.year^{-1}$)

lack "type usine id usine",
- le programme prend le fichier de données CSV et construit un arbre classique avec une liste chaînée de noeuds enfants (arbre à nombre de noeuds enfants indéterminé)
- faire aussi un AVL afin de connaître l'adresse du noeud parent.
- le programme parcourt l'ensemble des liens depuis l'usine jusqu'aux usagers, et fait la somme des quantités d'eau perdues dans chaque lien, en million de m^3 par an.
- si l'id n'est pas trouvée (mais le format est correct) le volume d'eau sera -1
- pour chaque lien la quantité d'eau est répartie équitablement entre les enfants.
- un fichier .dat devra être créé, ou complété, (pour chaque usine ?) avec comme colonnes, identifier, leak volume ($M.m^3.year^{-1}$) inclure

* Le programme C doit retourner une valeur d'erreur strictement positive, 0 sinon, le code doit être robuste.
* Il faut limiter la mémoire utilisée et libérer les allocations (sauf en cas d'erreur).
* Le code C est séparé en modules (.c, .h, sans-dossier) et est compilé par un makefile.
* Le code est commenté [illegible] par bloc et indenté, avec tout dans la même langue.
* Fichier readme avec instructions pour utiliser le programme.

* Document PDF avec la répartition des tâches, le planning, et les limitations fonctionnelles.
* Donner test avec images et fichiers pré-générés et présentés dans le PDF

AVL node :
char id_aval ← valeur ABR
node ARBRE* pointeur vers noeud réel
node AVL* sous_arbre_droit
node AVL* sous_arbre_gauche
int facteur

ARBRE node :
float / fuites
int V_eau
liste chaînée noeuds_enfants
↓
de ARBRE node

↖ créer un noeud racine vide de départ

lignes utilisées :

source → usine
usine → stockage
stockage → jonction
jonction → raccordement
raccordement → usager

C S V.

si la ligne est une source on regarde si l'id enfant correspond, sinon on regarde si la ligne id_usine correspond
↓
on recherche l'id de l'acteur parent dans l'AVL, on crée un noeud et on le rajoute dans la liste chaînée enfant du noeud parent pointé,
on rajoute dans la liste chaînée du noeud racine si id pas trouvé dans l'AVL
↓
on créer un nouveau noeud dans l'AVL avec id acteur aval comme valeur, + pointeur vers le noeud réel ainsi créé

On parcours chaque noeud de l'arbre avec parcours largeur, et on met dans le volume de chaque noeud enfant le volume parent divisé par le nb d'enfants