



# Projet Shell

## Sujet

Mini Tableur en BASH

## Auteur

*KATCHALA MELE Abdoulaye*

*L3 informatique Groupe TD2-C*

# Ce qu’il faut savoir

Le projet est réalisé seul (monôme) et est divisé en trois fichiers distincts : bases.sh, utilitaires.sh et main.sh

Dans le fichier bases.sh, les fonctions de base sont implémentées pour effectuer les opérations demandées par l'utilisateur dans la feuille de calcul. Ces fonctions n'apportent aucune modification directe à la feuille de calcul, et la plupart d'entre elles n'ont même pas la feuille en tant que paramètre. Ces fonctions représentent la liste des opérations demandées dans l'énoncé.

Le fichier utilitaires.sh regroupe les fonctions permettant d'effectuer des opérations sur la feuille de calcul et de la manipuler. La plupart des fonctions de ce fichier entraînent des modifications directes sur la feuille de calcul et renvoient la version modifiée en sortie. Il faut noter que ces fonctions utilisent les fonctions de bases dans le fichier précédent.

Le fichier main.sh quant à lui correspond au programme principale du projet contenant le script qui permet de traiter la feuille passée en paramètre.

# L’Algorithme principal (main.sh)

L’algorithme de notre programme principale est constitué de 8 étapes :

### *Étape 1 : Gestion des options*

On parcourt les paramètres du programme pour avoir les options. À chaque itération, un Switch est utilisé pour déterminer laquelle des options est activée. En l'absence de toute option, il s’agit de la feuille. On gère aussi les éventuelles erreurs dans les options. Si l’option -in est mis on affiche le fichier et on stocke la feuille dans une variable

### *Étape 2 : Remplacement des séparateurs d’entrée*

Après avoir géré les options, le programme parcourt la feuille de calcul pour remplacer les séparateurs de lignes et de colonnes spécifiés par l'utilisateur par la tabulation et le retour à la ligne. Afin d’éviter tout problème, on remplace les tabulation écrite par l’utilisateur dans les cellules par le séparateur en entrée. Il convient de noter que ce sont les séparateurs par défaut utilisés dans toutes les fonctions du projet.

### *Étape 3 : un Premier parcours*

Après remplacement des séparateurs d’entrée par les séparateurs par défaut, on parcours la feuille de calcul pour rechercher des éventuels cases à afficher avec la fonction ***display.*** Si on trouve une cellule contenant l’appel à cette fonction, on ajoute l’intervalle à afficher dans une variable qui contiendra toutes les cases à afficher.

### *Étape 4 : Traitement de la feuille*

Ensuite, le programme procède au traitement de la feuille de calcul en appelant la fonction ***Traite_intervalle_cellule***, qui prend en charge chaque cellule faisant partie de l'intervalle passé en paramètre. Plus de détails sur cette fonction seront abordés ultérieurement. Si la sortie de la fonction n'est pas égale à zéro, cela indique la présence d'une erreur, et par conséquent, le programme est interrompu.

### *Étape 5 : Inversion des cellules (Option -inverse)*

Si l'option -inverse a été spécifié, on procédera à l'inversion des cellules. La ligne deviendra la colonne et la colonne deviendra la ligne.

### *Étape 6 : Récupération des cellules à afficher seulement (fonction display)*

On parcourt la variable précédente '**display**' afin d'afficher seulement les cellules que l'utilisateur a spécifier dans les éventuelles fonctions display.

### *Étape 7 : Remplacement des séparateurs de sortie*

Si le traitement des cellules de la feuille s'est déroulé correctement, on procède ensuite au remplacement des séparateurs par défaut par les séparateurs de sortie spécifiés par l'utilisateur. On commence par remplacer le séparateur d'entrée par la tabulation (\t), qui a été précédemment substitué. Ceci afin de traiter l'éventuel problème de la tabulation s'il est pas séparateur.

### *Étape 8 : Affichage du résultat*

Le résultat du traitement sera affiché sur la console en l'absence de spécification d'un fichier de sortie via l'option -out. Dans le cas contraire, la feuille sera recopiée dans le fichier de sortie spécifié.

## Les fonctions de traitement (utilitaire.sh)

Dans le fichier utilitaire.sh sont utilisée plusieurs fonctions de traitement de la feuille :

- ***ligne_colonne()*** : cette fonction permet d'avoir la ligne et colonne d'une notation licj. Elle utilise sed afin de remplacer l et c par des espaces
- ***valide_ligcol()*** : permet de voir si le numéro de ligne et de colonne est valide. C'est à dire ne dépasse pas la feuille
- ***appartient()*** : permet de voir si le numéro de ligne et de colonne est appartient à un intervalle de cellule. C'est à dire si elle vient après la première cellule de l'intervalle et vient avant la deuxième cellule de l'intervalle
- ***remplace_cellule()***: permet de remplacer le contenu de la cellule par un autre. Elle prend la ligne du contenu et refait la ligne en remplaçant le contenu par le nouveau en utilisant sed.
- ***nom_paramètres()*** : permet d'avoir le nom et les paramètres d'une fonction écrite sous forme textuelle. Elle fait sortir le nom en utilisant cut et '(' comme séparateur et

parcours la seconde partie pour afficher tous les paramètres de la fonction. Elle prend en compte le nombre de parenthèse ouvrante et fermante.

- ***traite_cellule()*** : La fonction permet de traiter le contenu d'une cellule en la remplaçant. Tout d'abord, elle vérifie la validité de la ligne et de la colonne. Ensuite, elle s'assure qu'il n'y a pas de récursion infinie sur une cellule en utilisant une variable globale $traitement. La fonction récupère le contenu de la cellule et vérifie s'il commence par '='. Si ce n'est pas le cas, la fonction se termine. En revanche, si le contenu commence par '=', la cellule est ajoutée à la variable globale $traitement pour éviter un nouvel appel. Ensuite, la fonction est exécutée par la fonction ***execute_fonction()***, et la cellule est remplacée par le résultat. Finalement, la nouvelle feuille est retournée.
- ***traite_intervalle_cellule()*** : permet de traiter un intervalle de cellule. Elle utilise la fonction précédente sur les cellules concernées.
- ***execute_fonction()*** : La fonction execute_fonction() prend en charge l'exécution d'une fonction spécifiée sous forme textuelle en tant que paramètre. Tout d'abord, elle sépare le nom de la fonction de ses trois éventuels paramètres en utilisant la fonction nom_paramètre. Ensuite, elle traite le premier, le deuxième et le troisième paramètre à l'aide de la fonction ***traite_parametre()*** et grade les valeurs dans des variables val. Ensuite, elle utilise un Switch sur le nom de la fonction pour choisir la fonction appropriée lors de l'exécution. Selon la fonction à exécuter, elle traite un certain nombre de cellules en cours de route, en conservant à chaque étape la nouvelle feuille.
  En fin de compte, la fonction retourne la nouvelle feuille et le résultat de l'exécution de la fonction.
- ***traite_parametre() :*** Cette fonction nous permet d'éviter des répétitions en traitant une paramètre d'une fonction passée sous forme textuelle. Elle regarde si le paramètre est une fonction. Si oui, elle appelle récursivement la fonction execute_fonction() sur le paramètre et conserve le résultat. Après cela, elle vérifie s'il est au format 'licj'. Si c'est le cas, elle traite la cellule en appelant la fonction traite_cellule() et stocke le résultat ainsi que la nouvelle feuille.
  Si le premier paramètre est une valeur, elle le conserve simplement. Ensuite elle affiche la nouvelle feuille éventuellement modifiée, la valeur du paramètre et la valeur issue du calcul associé au paramètre.

# Les fonctions de bases (bases.sh)

Les fonctions de base correspondent à la liste des fonctions demandées dans l'énoncé. Les algorithmes de ces fonctions sont naturels et relativement simples à mettre en œuvre. Chacune de ces fonctions intègre une gestion des erreurs, signalant un message d'erreur et renvoyant une sortie différente de zéro en cas de problème. Les principales fonctions comprennent :

***cellule()*** pour avoir la valeur d'une cellule***, additionne()*** pour additionner 2 nombres***, soustrait()*** pour calculer la soustraction entre 2 nombres***, multiplie()*** pour calculer la

multiplication de 2 nombres***, divise()*** pour calculer la division entre 2 nombres***, puissance()*** pour calculer l'exposant d'un nombre***, logarithme()*** pour calculer le logarithme d'un nombres***, exponentielle()*** pour calculer l'exponentielle d'un nombre***, racine()*** pour calculer la racine d'un nombre***, somme()*** pour calculer la somme d'un intervalle de cellule***, moyenne()*** pour calculer la moyenne d'un intervalle de cellule***, variance()*** pour calculer la variance d'un intervalle de cellule***, ecartype()*** pour calculer l'écart-type d'un intervalle de cellule***, mediane()*** pour calculer la médiane d'un intervalle de cellule***, minimum()*** pour avoir le minimum d'un intervalle de cellule***, maximum()*** pour avoir le maximum d'un intervalle de cellule, ***concat()*** pour concatener deux chaines***, length()*** *pour connaître la longueur d'une chaine****, substitute()*** pour remplacer une sous chaine dans une chaine par une chaine***, size()*** pour avoir la taille d'un fichier***, lines()*** pour avoir le nombre de ligne d'un fichier***, shell()*** pour pouvoir executer une commande bash***, display()*** pour pouvoir afficher des cellules.

# Notice

La syntaxe de la commande est :

***tableur [-in file] [-out file] [-scin sep] [-scout sep] [-slin sep] [-slout sep] [-inverse] [-help]***

Notons que on ne peut pas utiliser un caractère comme séparateur et mettre le meme caractère comme contenu dans une cellule, car il sera interprété.
On ne peut pas non plus utiliser les caractères '/' ou '\' comme séparateur car ils sont interprété par la fonction sed au niveau des remplacement.

Si l'utilisateur utilise l'option help, la syntaxe de la commande lui sera affiché.

Si des fonctions de traitement de chaînes de caractère comme ***concat, length, substitute, size, lines, shell*** contienne une virgule ',' dans leurs paramètres. Il y aura erreur car on utilise les virgules comme séparateurs des paramètres.

Mise à part ces bugs, tous les autres bugs sont gérés