Programme : *chat-bot*

Objectif du programme Bash : *chat-bot*

L'objectif du programme Bash est de simuler un *chat-bot* interactif. Pour ce faire, nous allons utiliser le programme *chat* pour communiquer. Bien évidemment, le mode *--bot* doit être activée. Il devra être capable de répondre aux commandes envoyées par l'utilisateur et doit être capable de rediriger les flux standard de chat afin d'envoyer et recevoir les messages.

Appel de fonction

Nous avons créé une liste associative (un dictionnaire en Python), où les clés sont les noms des fonctions, et les valeurs leurs appels. De cette manière, le programme exécute des fonctions spécifiques selon ce que l'utilisateur demande.

Par ailleurs, cette implémentation est très intéressante car nous n'avons pas à modifier la liste mais seulement l'implémentation de la fonction lorsqu'il y aura un bug,. Cela nous permet donc d'avoir un code modulaire.

Gestion des pipes

Ce sont grâce aux pipes nommés que les processus puissent communiquer entre eux. Dans ce projet, nous avons utilisé la fonction *coproc* qui permet d'exécuter des programmes parallèles. Nous avons décidé d'utiliser cette fonction au lieu de *mkfifo*. En effet, cette dernière nécessite un chemin d'accès, alors que *coproc* crée automatiquement des pipes temporaire. Cela facilitera la tâche concernant la création et la suppression de ces fichiers.

Ainsi, le script fonctionne ainsi :

- Lorsque l'utilisateur entre une commande dans le terminale, le script analyse et valide les arguments ;
- Selon la commande (*liste*, *li*, *find_word*, etc.), le programme utilise les pipes pour les transmettre à un processus fils ;
- Enfin, les résultats sont retournés à l'utilisateur via la sortie standard *stdout*.

Gestion de la mémoire

Il est important de libérer la mémoire pour éviter les fuites de mémoire ou de problèmes concernant l'utilisation prolongée de variables. De ce fait, lorsque l'utilisateur arrête d'interagir avec le programme, avant de terminer, nous utilisons la fonction *unset* afin de libérer la mémoire allouée pour les fichiers *stdin*, *stdout* et les commandes de l'utilisateur.

Par ailleurs, certaines variables sont déclarées en tant que *local* car cela permet de nettoyer la mémoire automatiquement à la fin de l'exécution d'une fonction. De plus, cela favorise également l'encapsulation car les variables déclarés comme *local* sont accessibles seulement à l'intérieur de la fonction.

Problème de compatibilité

Lorsque nous avions effectué des tests individuels du script bash, nous avons remarqué un soucis au niveau de la compatibilité lors de l'exécution. En effet, lorsque deux personnes

travaillant sur Ubuntu/Macos exécute le code écrit par un collègue travaillant sur Windows/WSL, ce message d'erreur s'affichait dans le terminale : « *bash: ./chat-bot : ne peut exécuter : le fichier requis n'a pas été trouvé»*. Cela est dû au fait que l'encodage de retour à la ligne sur Windows est différente que sur Unix. De ce fait, l'interpréteur bash ne peut pas retrouver le shebang et ne peut donc pas reconnaitre les noms de fonctions utilisé ainsi que sa syntaxe.

Pour remédier à ce problème, nous avons utilisé la commande « *:set ff=unix* » de l'éditeur de texte Vim afin de pouvoir exécuter sur la machine tournant sur Windows/WSL et sur Ubuntu/Macos.