Réalisation et déploiement d'un Shell

SAE 3.03 - Réseau et application serveur

Sommaire

I. Généralités **1**

II. Programmation du mBash **2**
A. Réalisation 2
B. Bilan 2

III. Déploiement sur un serveur **3**
A. Réalisation 3
B. Bilan 3

I. Généralités

Comme convenu avec M. Colnet, j'ai travaillé seul sur les deux parties du projet. J'ai également utilisé l'outil de gestion de projet Trello pour gérer l'organisation de mes tâches.

II. Programmation du mBash

A. Réalisation

- J'ai d'abord initialisé un dépôt GitHub et j'y ai importé le programme initial fourni.
- J'ai commencé par programmer un affichage au démarrage de mBash, ainsi que la commande exit qui permet de terminer l'exécution du mBash.
- Ensuite, il s'agissait d'analyser les commandes et de les exécuter si ce sont des commandes externes, en utilisant execve.
- Puis j'ai modifié l'analyse de la commande, de sorte à prendre en compte le caractère * (étoile) comme caractère joker.
- J'ai ensuite programmé les fonctions cd et help.
- Enfin, j'ai modifié le code pour pouvoir lancer une commande externe en arrière-plan lorsque l'utilisateur la termine par & (esperluette).

B. Bilan

Ainsi, en lançant mBash, l'utilisateur peut :

- Exécuter n'importe quelle commande intégrées au système (ls, pwd, emacs…)

- Exécuter des commandes intégrées à mBash :
  - ❖ cd <répertoire> pour changer de répertoire
  - ❖ help pour afficher un message d'aide
  - ❖ exit pour quitter mBash

- Utiliser le caractère * pour désigner n'importe quel caractère dans une commande.

- Utiliser le suffixe & pour exécuter une commande en arrière-plan

<!> Bug non-résolu : L'utilisation de & sur une commande entraîne l'exécution en arrière-plan pour toutes les commandes suivantes.

III. Déploiement sur un serveur

A. Réalisation

Sur le serveur :

- J'ai d'abord installé apache2 avec snap
- Puis j'ai généré une clé GPG
- J'ai ensuite transformé mon application en fichier .deb. Pour cela j'ai :
  - Installé les librairies nécessaires
  - Copié mon fichier mbash.c dans un répertoire respectant la syntaxe *<nom>-<version>* et généré l'archive tar compressée de même nom
  - Généré un répertoire *debian* en utilisant la commande *dh_make* et l'archive
  - Construit le package (.deb) à partir de ce répertoire, en utilisant *dpkg-buildpackage*
- Il s'agissait ensuite de créer un dépôt Debian. Pour cela j'ai :
  - Installé *reprepro* avec apt
  - Configuré une branche
  - Construit le référentiel en y ajoutant le package de mon application
  - Je me suis arrêté à cette partie car j'étais bloqué et par manque de temps…

B. Bilan

Je n'ai pas réussi à terminer cette partie mais j'étais déjà bien avancé sur la configuration de mon dépôt Debian.

Note: Vous trouverez sur le dépôt GitHub ci-dessous des captures d'écran de certaines commandes : Hugo-COLLIN/BUT_S3_SAE_ReseauAppServeur (github.com)