SAE 3.03 : Réseau et application serveur

Développement d'un shell miniature et gestion de paquets

Jassem TAMOURGH & Yanis CHEBBAH

Janvier 2026

Table des matières

1 Introduction **3**

2 Partie 1 : Le shell miniature mbash **3**
2.1 Approche technique 3
2.2 Gestion des commandes internes 3
2.3 Boucle d'exécution et sortie 3

3 Partie 2 : Mise en œuvre du serveur de dépôts Debian **4**
3.1 Procédure sur le PC Serveur (Machine A) 4
3.1.1 Création du paquet binaire .deb 4
3.1.2 Configuration du service HTTP et de l'indexation 4
3.1.3 Cas particulier de WSL : Redirection de port 5
3.2 Procédure sur le PC Client (Machine B) 5
3.2.1 Ajout du dépôt aux sources APT 5
3.2.2 Installation et vérification 5
3.3 Gestion du cycle de vie (Mise à jour v0.1 vers v0.2) 6

4 Utilisation de l'Intelligence Artificielle **6**

5 Conclusion **6**

1 Introduction

Ce projet s'inscrit dans le cadre de la SAE 3.03. Il consiste à réaliser une version simplifiée de `bash`, nommée `mbash`, et à mettre en place une infrastructure de distribution de logiciels via un serveur de dépôts Debian. L'objectif est de comprendre le cycle de vie d'un logiciel, de sa conception à son déploiement automatisé.

2 Partie 1 : Le shell miniature mbash

2.1 Approche technique

Nous avons choisi d'utiliser la fonction `system()` pour l'exécution des commandes. Ce choix permet de déléguer au shell hôte la gestion complexe du `PATH` et des caractères spéciaux comme l'esperluette (&).

2.2 Gestion des commandes internes

Certaines commandes, comme `cd`, ne peuvent pas être exécutées via `system()` car elles doivent modifier l'environnement du processus parent. Nous avons donc utilisé l'appel système `chdir()` :

Listing 1 – Implémentation de la commande cd

```
if (strncmp(cmd, "cd␣", 3) == 0) {
    char *path = cmd + 3;
    if (chdir(path) != 0) {
        perror("mbash:␣cd");
    }
    continue;
}
```

2.3 Boucle d'exécution et sortie

Le shell fonctionne via une boucle infinie qui lit l'entrée utilisateur. La sortie est gérée soit par la commande `exit`, soit par la détection du caractère de fin de fichier (Ctrl-D) :

Listing 2 – Boucle principale et gestion de la sortie

```
if (fgets(cmd, MAXLI, stdin) == NULL) {
    printf("\ndeconnexion\n");
    break;
}

if (strcmp(cmd, "exit") == 0) {
    break;
}
```

3 Partie 2 : Mise en œuvre du serveur de dépôts Debian

Cette partie détaille l'architecture client-serveur mise en place pour permettre l'installation automatisée de `mbash` via le gestionnaire de paquets `apt`.

3.1 Procédure sur le PC Serveur (Machine A)

Le serveur a pour rôle de construire le paquet binaire et de l'exposer via le protocole HTTP.

3.1.1 Création du paquet binaire .deb

Nous avons d'abord préparé l'arborescence respectant les standards Debian pour placer l'exécutable dans `/usr/bin` :

```
# 1. Compilation du binaire
gcc -o mbash mbash.c

# 2. Creation de la structure du paquet (v0.1)
mkdir -p mbash_0.1/DEBIAN
mkdir -p mbash_0.1/usr/bin
cp mbash mbash_0.1/usr/bin/

# 3. Creation du fichier de controle
cat <<EOF > mbash_0.1/DEBIAN/control
Package: mbash
Version: 0.1
Architecture: amd64
Maintainer: TAMOURGH-CHEBBAH
Description: Shell miniature mbash
EOF

# 4. Generation du paquet
dpkg-deb --build mbash_0.1
```

3.1.2 Configuration du service HTTP et de l'indexation

Le paquet est ensuite publié via Apache2. Pour que `apt` puisse lire le dépôt, un index compressé `Packages.gz` est généré :

```
# Installation d'Apache
sudo apt install apache2

# Mise en place du depot
sudo mkdir -p /var/www/html/debian
sudo cp mbash_0.1.deb /var/www/html/debian/

# Generation de l'index du depot
cd /var/www/html/debian
sudo bash -c "dpkg-scanpackages . /dev/null | gzip -9c > Packages.
   gz"
```

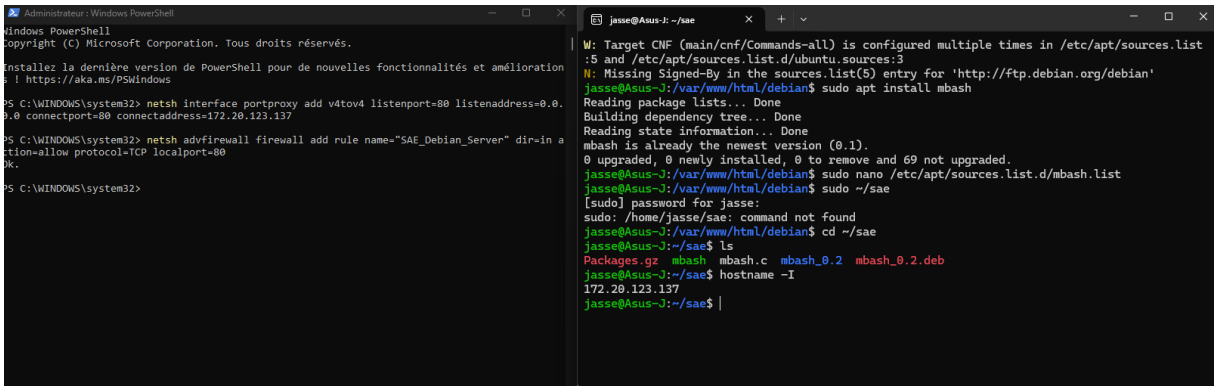

FIGURE 1 – Redirectin de port sur Powershell avec IP du WSL

3.1.3 Cas particulier de WSL : Redirection de port

Le serveur tournant sous WSL, nous avons dû configurer une redirection de port sur l'hôte Windows pour rendre le service accessible aux autres machines du réseau local :

```
# Commande executee en PowerShell Administrateur sur l'hote Windows
netsh interface portproxy add v4tov4 listenport=80 listenaddress
   =0.0.0.0 connectport=80 connectaddress=172.20.123.137
```

3.2 Procédure sur le PC Client (Machine B)

Le client doit être configuré pour reconnaître le serveur comme une source de logiciels fiable.

3.2.1 Ajout du dépôt aux sources APT

Nous avons créé un fichier de configuration dédié dans `sources.list.d`. L'option `[trusted=yes]` est utilisée car le dépôt n'est pas signé numériquement :

```
# Remplacement de IP_SERVEUR par l'IP Windows du serveur
echo "deb [trusted=yes] http://IP_SERVEUR/debian ./" | sudo tee /
   etc/apt/sources.list.d/mbash.list
```

3.2.2 Installation et vérification

L'installation s'effectue via les commandes standards. Le système télécharge l'index puis le paquet binaire :

```
sudo apt update
sudo apt install mbash

# Verification de l'emplacement d'installation
which mbash # Doit afficher /usr/bin/mbash
```

3.3 Gestion du cycle de vie (Mise à jour v0.1 vers v0.2)

Pour démontrer l'évolution du logiciel, nous avons produit une version 0.2 de `mbash`. La procédure sur le serveur consiste à reconstruire le paquet avec un fichier `control` mis à jour et à régénérer le fichier `Packages.gz`. Sur le client, la mise à jour est immédiate :

```
sudo apt update
sudo apt upgrade # mbash sera mis a jour vers la version 0.2
```

4 Utilisation de l'Intelligence Artificielle

L'intelligence artificielle a été utilisée comme un *thought partner* tout au long du projet pour optimiser notre temps de travail,.

- **API :** L'IA nous a aidé à arbitrer entre l'utilisation de `execvp()` et `system()`, en nous permettant de valider que `system()` répondait aux contraintes de syntaxe tout en respectant le temps imparti. Nous avons aussi utilisé l'IA pour des infos sur l'API (methodes...).
- **Résolution de problèmes système :** Lors de la configuration du serveur, l'IA a aidé à identifier l'origine des erreurs de droits d'accès lors de la redirection de sortie vers le fichier `Packages.gz`.
- **Rédaction et mise en forme LaTeX :** Nous avons fourni à l'IA le détail de chaque étape du projet, le code source ainsi qu'une liste des éléments à inclure dans le rendu (déjà partiellement rédigés). L'IA a alors servi de support pour structurer ces informations et les convertir proprement au format LaTeX.
- **Redirection de portavec Powershell :** Nous avons nous-même identifié que le problème de serveur provenait du WSL, nous avons utilisé l'IA pour obtenir la commande powershell permettant la redirection de port.

L'ensemble du code produit a été revu et testé manuellement pour garantir sa conformité avec l'environnement Linux local.

5 Conclusion

Ce projet a permis de valider les compétences en programmation système C et en administration réseau Linux. Nous avons réussi à simuler un cycle complet de mise à jour logicielle de la version 0.1 à la version 0.2 via les outils standards Debian.