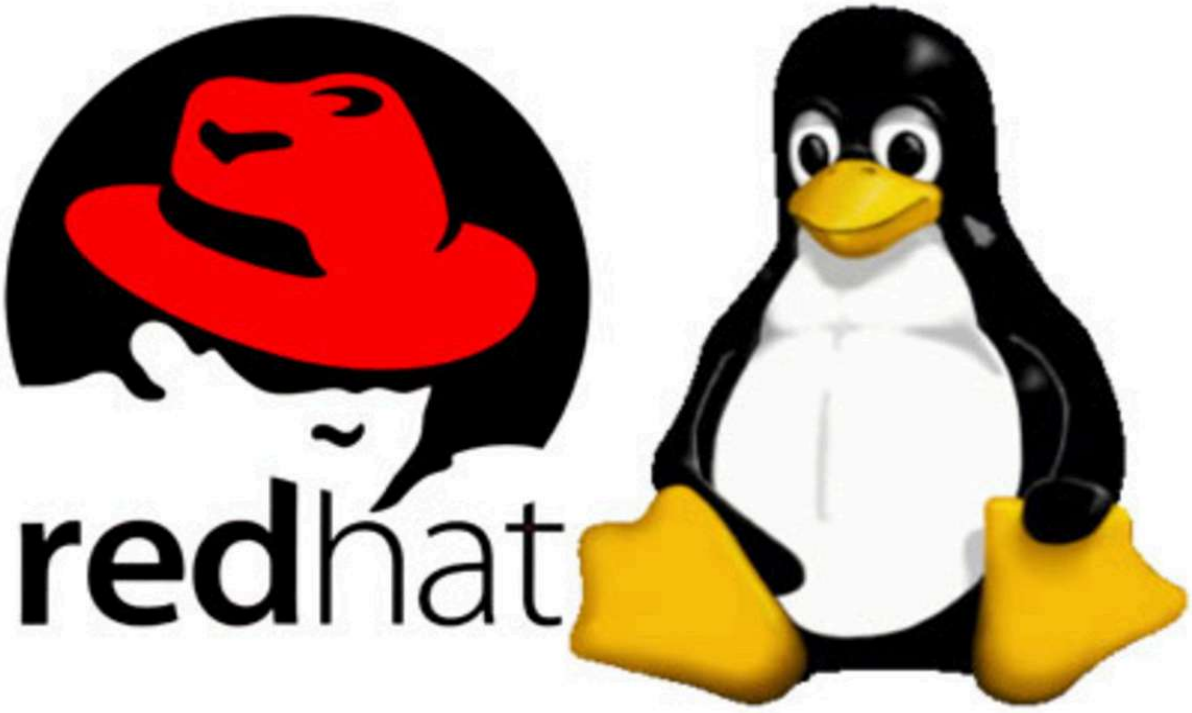
AUTOMATISATION DE TRANSFERTS FTP SOUS RHEL 5 AVEC BASH (CLIENT-SERVEUR)



SOMMAIRE

I.INTRODUCTION

Dans le cadre du cours d'**Administration Linux** dispensé par le Dr. Gouho Bi Jean Baptiste à l'Université Nangui Abrogoua, nous avons travaillé sur la mise en place et la configuration d'un serveur FTP d'entreprise. La première étape consistait à effectuer des transferts de fichiers de manière manuelle, afin de comprendre le fonctionnement de base du protocole FTP et de son intégration dans un environnement Linux.

Dans la continuité de ce travail, nous avons abordé le **scripting Bash** sous Red Hat Enterprise Linux 5, avec pour objectif d'automatiser les tâches répétitives liées aux transferts de fichiers. L'utilisation de scripts permet en effet de remplacer des opérations manuelles par des exécutions automatiques, garantissant ainsi un gain de temps, une meilleure fiabilité et une réduction des erreurs humaines.

Objectif du projet : concevoir un script Bash capable de simplifier et fiabiliser les transferts de fichiers pour plusieurs utilisateurs, en automatisant la récupération des données depuis un serveur FTP et en les déposant dans un répertoire local défini. Ce projet illustre l'importance de l'automatisation dans l'administration système et met en évidence les avantages d'une gestion centralisée et reproductible des processus.

II. ENVIRONNEMENT

Le projet a été réalisé dans un environnement académique et technique spécifique, afin de mettre en pratique les notions d'administration système et de scripting Bash.

- **Système d'exploitation** : Red Hat Enterprise Linux 5 (RHEL 5), utilisé comme plateforme principale pour l'administration et l'exécution des scripts.
- **Shell** : Bash (version 3.x), choisi pour sa compatibilité avec RHEL 5 et ses fonctionnalités de scripting.

- **Protocole de transfert** : FTP (File Transfer Protocol), utilisé pour la gestion des transferts de fichiers entre le serveur et les clients.
- **Serveur FTP** : configuré sur une machine Linux, permettant l'accès aux répertoires utilisateurs et la récupération des données.
- **Fichiers externes** :

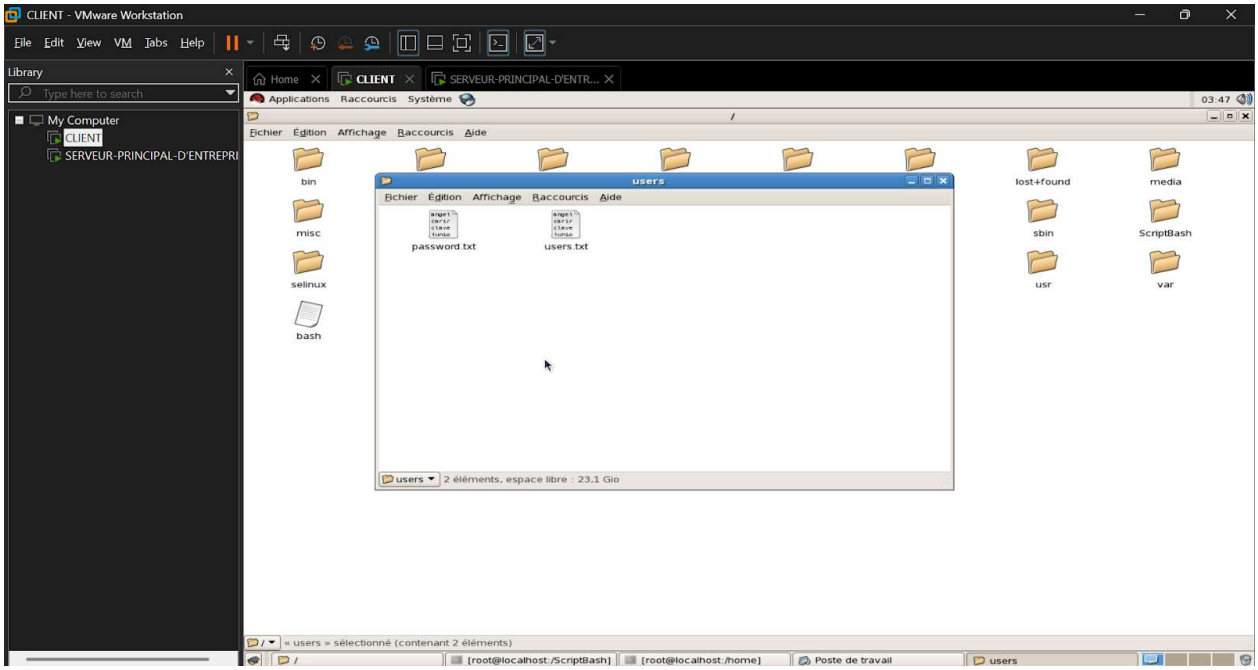
  `users.txt` : liste des utilisateurs autorisés.

  `password.txt` : liste des mots de passe correspondants.

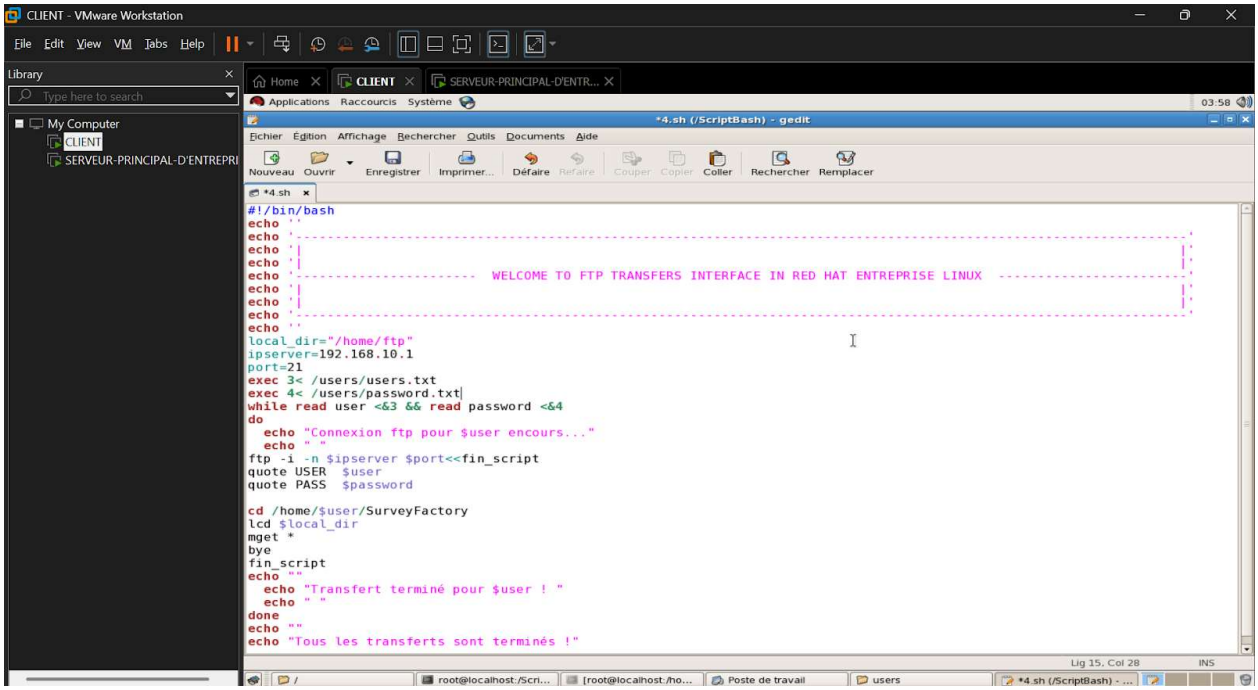
  Ces fichiers permettent d'externaliser les informations sensibles et de

  faciliter la maintenance du script.

- **Répertoire local de dépôt** : défini via la commande `lcd`, afin de centraliser les fichiers téléchargés dans un espace dédié.
- **Éditeur de texte** : gedit, utilisé pour la rédaction et la modification du script.
- **Machine virtuelle** : VMware Workstation, servant de support pour l'installation et la simulation de l'environnement RHEL 5.



III. CONCEPTION DU SCRIPT



La conception du script Bash repose sur une approche modulaire et automatisée, permettant de remplacer les opérations manuelles par des instructions exécutées de manière séquentielle et fiable. Les principales étapes de conception sont les suivantes :

Externalisation des identifiants

- Les informations sensibles (utilisateurs et mots de passe) sont stockées dans deux fichiers distincts (`users.txt` et `password.txt`). Cette méthode facilite la maintenance et permet d'ajouter ou de supprimer des comptes sans modifier le script.

Lecture parallèle des fichiers

- Afin d'associer chaque utilisateur à son mot de passe correspondant, le script utilise la commande `exec` pour ouvrir les deux fichiers en parallèle, puis une boucle `while read` qui lit une ligne de chaque fichier simultanément. Cette approche garantit une correspondance 1 "à" 1 entre utilisateurs et mots de passe.

Connexion FTP automatisée

- Le script utilise un *here-document* (`<< fin_script`) pour envoyer automatiquement les commandes FTP au serveur. Les instructions incluent :
  - `quote USER` et `quote PASS` pour l'authentification.
  - `cd` pour accéder au répertoire distant de l'utilisateur.
  - `lcd` pour définir le répertoire local de dépôt des fichiers.
  - `mget *` pour télécharger l'ensemble des fichiers disponibles.

Boucles et automatisation

- La boucle principale parcourt chaque paire utilisateur/mot de passe et exécute les commandes FTP correspondantes. À la fin de chaque itération, un message de confirmation est affiché pour indiquer la réussite du transfert.

Gestion des répertoires locaux

- Le script transfert automatiquement le répertoire de dépôt local où les fichiers téléchargés sont centralisés dans un espace dédié sans risque d'erreur .

En résumé

La conception du script illustre une démarche progressive : externalisation des données sensibles, association correcte des identifiants, automatisation des transferts FTP et gestion centralisée des fichiers. Cette approche assure la fiabilité, la maintenabilité et la reproductibilité des opérations dans un environnement Linux.

IV. FONCTIONNEMENT

Le script Bash développé pour l'automatisation des transferts FTP suit une séquence d'exécution bien définie. Chaque étape contribue à garantir la fiabilité et la simplicité du processus :

1. Initialisation

    - Le script commence par afficher un message d'accueil afin de contextualiser son utilisation.
    - Les variables principales sont définies : adresse IP du serveur FTP, port de connexion, et répertoire local de dépôt des fichiers.

2. Ouverture des fichiers d'identifiants

    - Les fichiers `users.txt` et `password.txt` sont ouverts en parallèle grâce aux commandes `exec 3<` et `exec 4<`.
    - Cette méthode permet de lire simultanément une ligne de chaque fichier, assurant la correspondance entre utilisateur et mot de passe.

3. Boucle principale

    - Une boucle `while read` parcourt les deux fichiers ligne par ligne.
    - À chaque itération, un utilisateur et son mot de passe sont extraits et utilisés pour établir une connexion FTP.

4. Connexion au serveur FTP

- Le script lance le client FTP en mode non interactif (`ftp -i -n`).
- Les commandes sont envoyées via un *here-document* :
  - `quote USER` et `quote PASS` pour l'authentification.
  - `cd` pour accéder au répertoire distant de l'utilisateur.
  - `lcd` pour définir le répertoire local où les fichiers seront déposés.
  - `mget *` pour télécharger l'ensemble des fichiers disponibles.
  - `quit` pour fermer la session.

5. Confirmation et suivi
   - Après chaque transfert, un message est affiché pour confirmer la réussite de l'opération.
   - La boucle continue jusqu'à ce que tous les utilisateurs aient été traités.

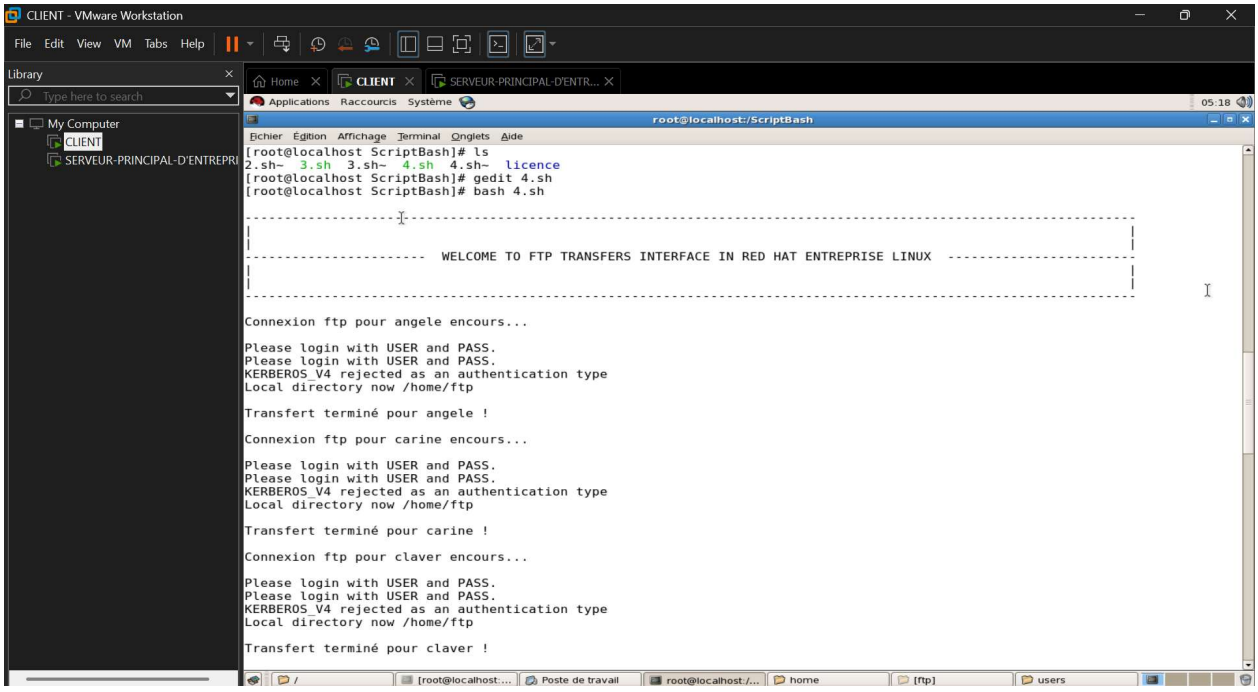
6. Clôture du script
   - Une fois la boucle terminée, le script affiche un message final indiquant que l'ensemble des transferts est achevé.
   - Les fichiers téléchargés sont disponibles dans le répertoire local défini.

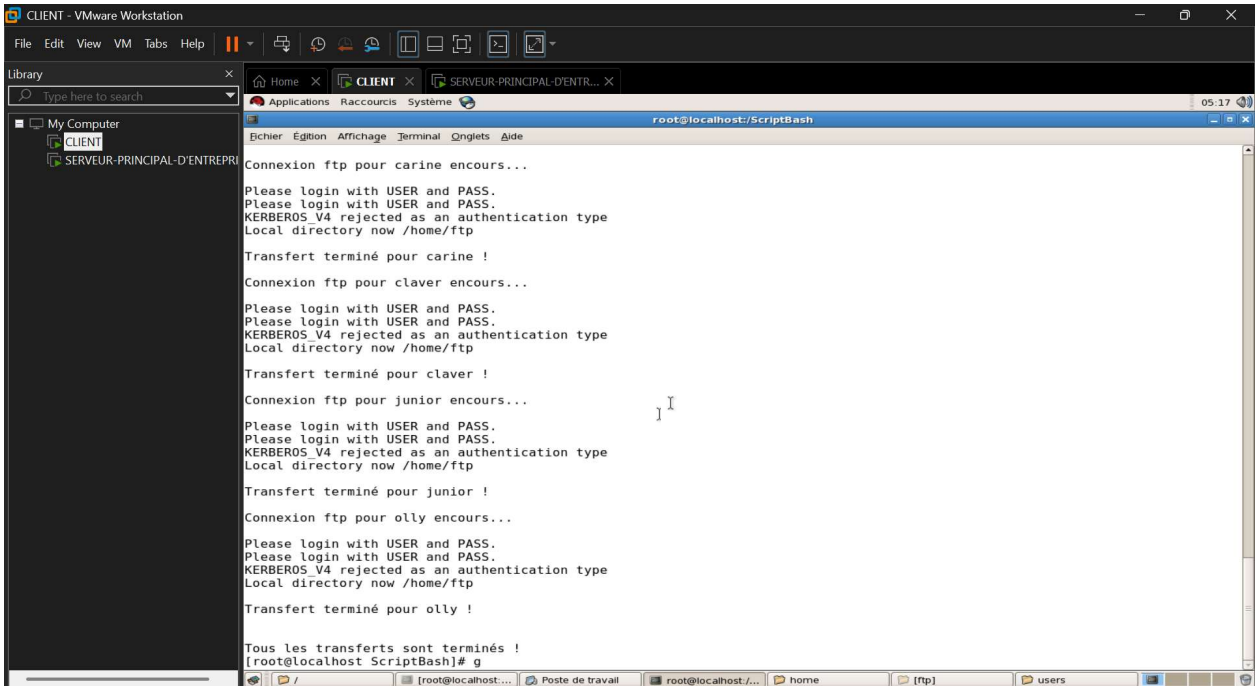
Résumé

Le fonctionnement du script repose sur une logique simple mais efficace : lecture des identifiants, connexion automatisée au serveur FTP, transfert des fichiers et confirmation des opérations. Cette automatisation permet de réduire considérablement les tâches répétitives et d'assurer une gestion centralisée des données.

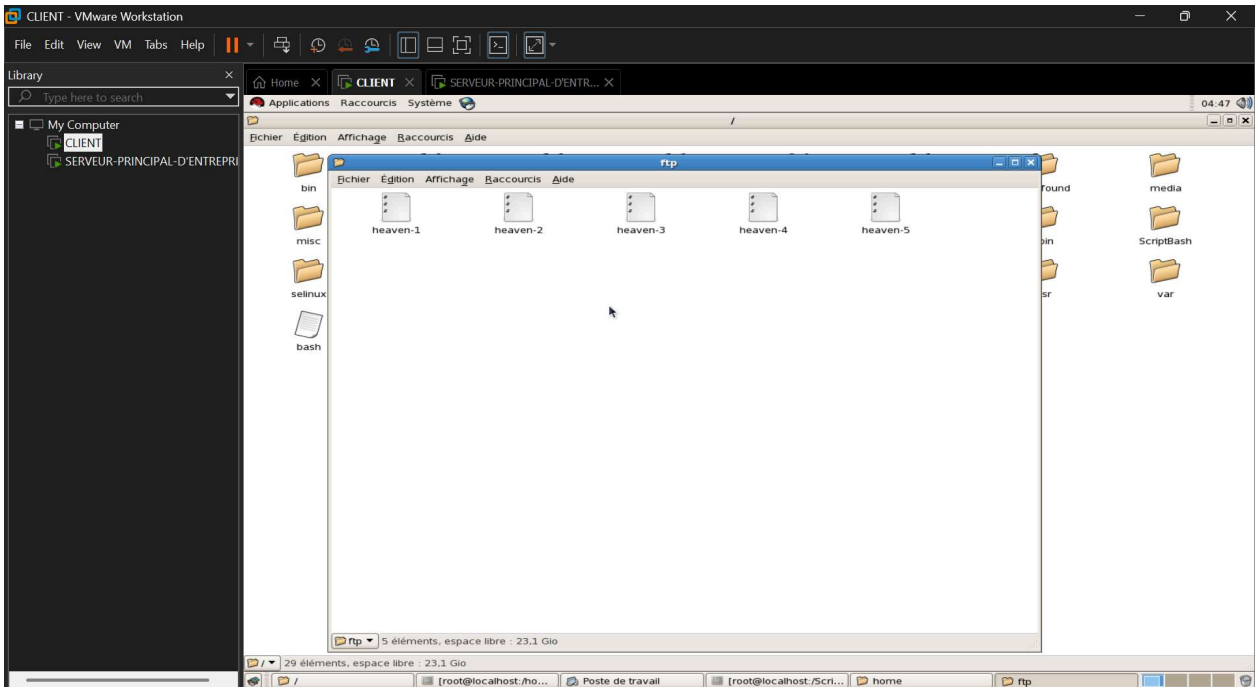
V. RÉSULTATS

EXÉCUTION DU SCRIPT SUR LE TERMINAL

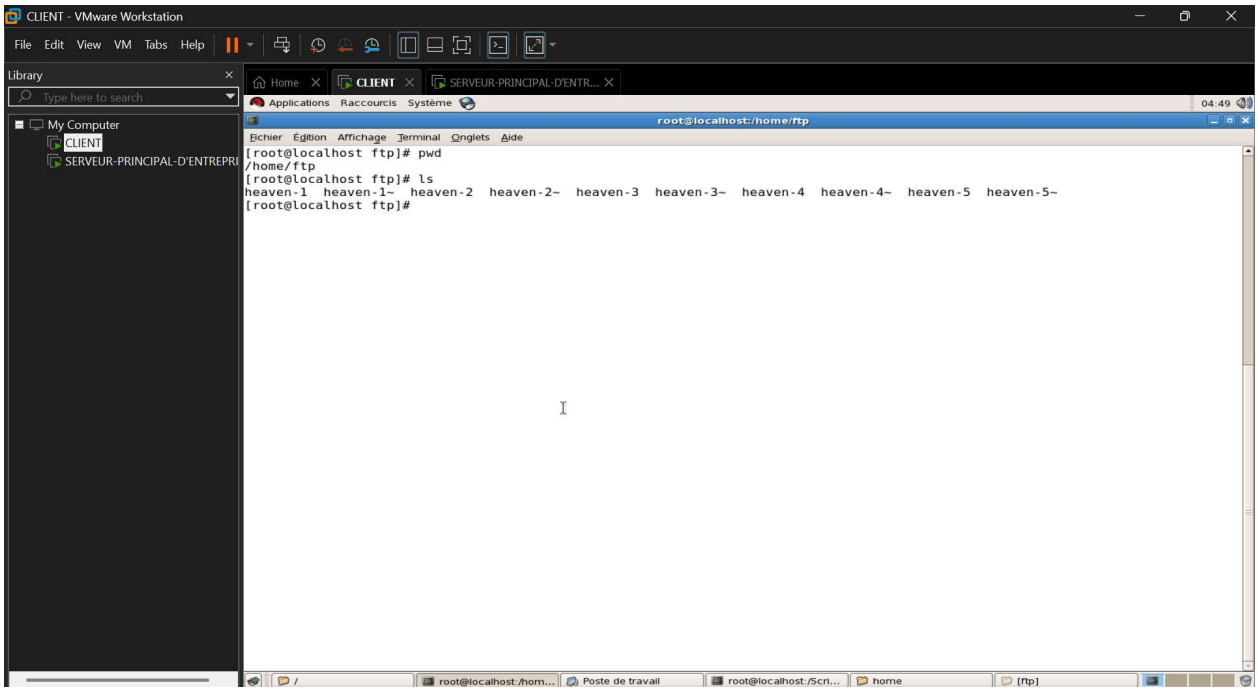




VUE GRAPHIQUE DES FICHIERS RÉCUPÉRÉS



VUE SUR TERMINAL DES FICHIERS RÉCUPÉRÉS



VI. CONCLUSION

Ce projet d'automatisation des transferts FTP sous **Red Hat Enterprise Linux 5** a permis de mettre en pratique les compétences acquises en administration système et en scripting Bash. À partir d'une procédure initialement manuelle, nous avons conçu un script capable de gérer automatiquement les connexions FTP multi-utilisateurs, d'assurer la correspondance entre identifiants et mots de passe, et de centraliser les fichiers téléchargés dans un répertoire local défini.

Les résultats obtenus démontrent l'efficacité de l'automatisation dans la réduction des tâches répétitives, l'amélioration de la fiabilité et la simplification de la maintenance. Ce travail illustre également l'importance de l'externalisation des données sensibles et de la modularité dans la conception de scripts robustes.

Au-delà de l’aspect technique, ce projet met en évidence la capacité à travailler dans des environnements contraints (Bash 3, RHEL 5) et à proposer des solutions adaptées aux besoins d’une infrastructure. Il constitue une étape importante dans le développement de compétences en **administration Linux**, en **automatisation des processus systèmes** et en **gestion des infrastructures réseau**.