ECOLE SUPERIEURE D'INGENIEUR DE RENNES

Université de Rennes

Shell Course

Cliquet Florian
note taking of online ressources

10 juin 2024

Plagiarism Mention

We attest that the content of this document is original and stems from our personal reflections.

Sommaire

Introduction

This document provides an overview of shell scripting concepts written by Cliquet Florian.

1 Introduction

Le shell, en tant qu'interface primordiale de tout système informatique, tisse un lien étroit avec le système d'exploitation. Chaque dispositif électronique abrite un shell, que ce soit sous une forme native ou adaptée, et certains proposent même une pléthore d'options parmi lesquelles choisir. Bien que leurs nuances puissent varier, ils convergent globalement vers des fonctionnalités similaires, offrant la possibilité de lancer des programmes, d'interpréter les réponses et d'analyser les données.
Dans ce cours, notre attention se portera spécifiquement sur le "Bourne Again SHell", communément désigné sous l'acronyme "bash". En tant que l'un des shells les plus répandus à travers le monde, bash offre une syntaxe familière, proche de celle des autres shells, facilitant ainsi l'apprentissage et la maîtrise de cet outil essentiel.

2 Les bases du Shell

Pour ouvrir un prompt shell, il nous faut accéder au `terminal (cmd)`.

```
$ ls -l
total 0
-rw-r--r-- 1 user group 0 Jun 10 00:00 file.txt
```

Le shell est une interface utilisateur permettant d'interagir avec un système d'exploitation en utilisant des lignes de commande. Dans cette section, nous aborderons les bases essentielles pour travailler avec le Shell.

2.1 Navigation dans le système de fichiers

La navigation dans le système de fichiers est une tâche courante lorsque vous utilisez le Shell. Voici quelques commandes de base :

- `cd` : Change le répertoire courant. Par exemple, pour accéder au répertoire `Documents`, vous pouvez utiliser : `cd Documents`.
- `ls` : Liste les fichiers et répertoires dans le répertoire courant.
- `pwd` : Affiche le chemin du répertoire courant.

2.2 Manipulation des fichiers et répertoires

Vous pouvez manipuler les fichiers et répertoires à l'aide de commandes Shell. Voici quelques exemples :

- `mkdir` : Crée un nouveau répertoire. Par exemple, pour créer un répertoire nommé `NouveauDossier`, vous pouvez utiliser : `mkdir NouveauDossier`.
- `touch` : Crée un nouveau fichier vide. Par exemple, pour créer un fichier nommé `NouveauFichier.txt`, vous pouvez utiliser : `touch NouveauFichier.txt`.
- `rm` : Supprime des fichiers ou des répertoires. Soyez prudent avec cette commande car elle supprime définitivement les fichiers. Par exemple, pour supprimer le fichier `AncienFichier.txt`, vous pouvez utiliser : `rm AncienFichier.txt`.

2.3 Exécution de commandes

Vous pouvez exécuter des commandes et des programmes à partir du Shell. Voici un exemple simple :

- Pour afficher le contenu d'un fichier, vous pouvez utiliser la commande `cat`. Par exemple, pour afficher le contenu du fichier `monfichier.txt`, vous pouvez utiliser : `cat monfichier.txt`.

3 Gestion des processus

3.1 Exécution de commandes en arrière-plan

Lorsque vous exécutez une commande dans le Shell, elle peut s'exécuter en arrière-plan ou en premier plan. Les processus en arrière-plan s'exécutent sans bloquer le Shell, vous permettant de continuer à travailler sur d'autres tâches. Voici comment exécuter une commande en arrière-plan :

```
$ commande &
```

Par exemple, pour démarrer un processus de sauvegarde en arrière-plan, vous pouvez utiliser :

```
$ backup_script.sh &
```

3.2 Surveillance des processus en cours d'exécution

Pour voir quels processus sont en cours d'exécution sur votre système, vous pouvez utiliser la commande `ps`. Par défaut, `ps` affiche les processus associés à votre terminal. Voici quelques options courantes pour la commande `ps` :

- `ps` : Affiche les processus associés à votre terminal.
- `ps aux` : Affiche tous les processus du système avec des informations détaillées.
- `ps -ef` : Affiche tous les processus du système avec des informations détaillées.

Par exemple, pour afficher tous les processus du système avec des informations détaillées, vous pouvez utiliser :

```
$ ps aux
```

3.3 Arrêt de processus

Parfois, vous devrez arrêter un processus en cours d'exécution. Pour cela, vous pouvez utiliser la commande `kill`, en spécifiant l'identifiant du processus (PID). Voici comment arrêter un processus :

```
$ kill PID
```

Si vous voulez arrêter un processus spécifique par son nom, vous pouvez utiliser `pkill` avec le nom du processus. Par exemple, pour arrêter tous les processus nommés `firefox`, vous pouvez utiliser :

```
$ pkill firefox
```

Ces commandes vous permettent de surveiller et de gérer efficacement les processus en cours d'exécution sur votre système, ce qui est essentiel pour maintenir la stabilité et les performances de votre machine.

4 Scripts Shell

4.1 Introduction aux scripts Shell

Les scripts Shell sont des programmes informatiques qui utilisent le langage de commandes du Shell pour automatiser des tâches. Ils peuvent être utilisés pour exécuter des séquences de commandes, manipuler des fichiers, traiter des données et bien plus encore. Voici quelques caractéristiques des scripts Shell :

- Les scripts Shell sont des fichiers texte contenant une séquence de commandes Shell.
- Ils peuvent être exécutés directement dans un terminal ou inclus dans d'autres scripts.
- Les scripts Shell sont polyvalents et peuvent être utilisés pour une variété de tâches, de simples opérations de fichiers à des tâches plus complexes impliquant des contrôles de flux et des opérations conditionnelles.

4.2 Syntaxe des scripts Shell

La syntaxe des scripts Shell dépend du langage de commandes du Shell utilisé (par exemple, bash). Voici un exemple simple de script Shell :

```
#!/bin/bash

# Ceci est un commentaire
echo "Bonjour, monde !"
```

Ce script Shell commence par un shebang (#!/bin/bash), qui spécifie le Shell à utiliser pour exécuter le script. Ensuite, il y a un commentaire (‘ Ceci est un commentaire‘) et une commande ‘echo‘ qui affiche "Bonjour, monde !".

4.3 Exécution de scripts Shell

Pour exécuter un script Shell, vous devez d'abord lui donner les permissions d'exécution avec la commande ‘chmod‘, puis vous pouvez l'exécuter comme n'importe quel autre programme. Par exemple, si votre script s'appelle ‘mon$_s$*cript.sh*‘, *vouspouvezl'excute*

```
$ chmod +x mon_script.sh
$ ./mon_script.sh
```

Cela donnera les permissions d'exécution au script et l'exécutera.

4.4 Bonnes pratiques pour l'écriture de scripts Shell

Lors de l'écriture de scripts Shell, il est important de suivre certaines bonnes pratiques pour assurer la lisibilité, la maintenabilité et la fiabilité du code. Voici quelques recommandations :

- Utilisez des commentaires pour expliquer le but et le fonctionnement du script.
- Utilisez des noms de variables descriptifs pour rendre le code plus compréhensible.
- Indentez correctement le code pour le rendre plus lisible.
- Gérez les erreurs et les exceptions de manière appropriée pour assurer la robustesse du script.
- Divisez les scripts longs en fonctions pour faciliter la réutilisation du code.
- Testez régulièrement vos scripts pour détecter les erreurs et les bogues.
- Documentez votre code pour aider les autres développeurs à comprendre son fonctionnement.

En suivant ces bonnes pratiques, vous pouvez écrire des scripts Shell efficaces et fiables pour automatiser vos tâches informatiques de manière plus professionnelle et organisée.

5 Gestion des permissions et des utilisateurs

5.1 Gestion des permissions de fichiers et de répertoires

La gestion des permissions de fichiers et de répertoires est une partie importante de l'administration système. Les permissions déterminent qui peut lire, écrire ou exécuter un fichier ou un répertoire. Voici quelques commandes couramment utilisées pour gérer les permissions :

- `chmod` : Modifie les permissions d'un fichier ou d'un répertoire. Par exemple, pour donner la permission de lecture, écriture et exécution à l'utilisateur propriétaire du fichier, vous pouvez utiliser : `chmod u+rwx fichier`.
- `chown` : Modifie le propriétaire et le groupe d'un fichier ou d'un répertoire. Par exemple, pour changer le propriétaire d'un fichier à un utilisateur nommé "utilisateur", vous pouvez utiliser : `chown utilisateur fichier`.
- `chgrp` : Modifie le groupe d'un fichier ou d'un répertoire. Par exemple, pour changer le groupe d'un fichier à un groupe nommé "groupe", vous pouvez utiliser : `chgrp groupe fichier`.

5.2 Gestion des utilisateurs et des groupes

La gestion des utilisateurs et des groupes est essentielle pour assurer la sécurité et l'organisation d'un système. Voici quelques commandes pour gérer les utilisateurs et les groupes :

- `useradd` : Ajoute un nouvel utilisateur au système. Par exemple, pour ajouter un utilisateur nommé "nouvel_utilisateur", vous pouvez utiliser : `useradd nouvel_utilisateur`.
- `usermod` : Modifie les paramètres d'un utilisateur existant. Par exemple, pour changer le répertoire de connexion d'un utilisateur, vous pouvez utiliser : `usermod -d /nouveau/repertoire utilisateur`.
- `userdel` : Supprime un utilisateur du système. Par exemple, pour supprimer l'utilisateur "ancien_utilisateur", vous pouvez utiliser : `userdel ancien_utilisateur`.
- `groupadd` : Ajoute un nouveau groupe au système. Par exemple, pour ajouter un groupe nommé "nouveau_groupe", vous pouvez utiliser : `groupadd nouveau_groupe`.
- `groupmod` : Modifie les paramètres d'un groupe existant. Par exemple, pour changer le nom d'un groupe, vous pouvez utiliser : `groupmod -n nouveau_nom groupe`.
- `groupdel` : Supprime un groupe du système. Par exemple, pour supprimer le groupe "ancien_groupe", vous pouvez utiliser : `groupdel ancien_groupe`.

La gestion des permissions et des utilisateurs est une partie cruciale de l'administration système. En comprenant ces concepts et en utilisant ces commandes avec précaution, vous pouvez sécuriser et organiser efficacement votre système.

6 Redirection et tubes (pipes)

6.1 Redirection de la sortie standard

La redirection de la sortie standard vous permet de rediriger le flux de sortie d'une commande vers un fichier ou un autre processus. Voici quelques opérateurs de redirection couramment utilisés :

- `>` : Redirige la sortie standard vers un fichier. Par exemple, pour rediriger la sortie d'une commande vers un fichier nommé "sortie.txt", vous pouvez utiliser : `commande > sortie.txt`.
- `»` : Redirige la sortie standard vers la fin d'un fichier (ajout). Par exemple, pour ajouter la sortie d'une commande à la fin d'un fichier existant nommé "sortie.txt", vous pouvez utiliser : `commande » sortie.txt`.
- `2>` : Redirige la sortie d'erreur standard vers un fichier. Par exemple, pour rediriger les messages d'erreur vers un fichier nommé "erreurs.txt", vous pouvez utiliser : `commande 2> erreurs.txt`.
- `&>` : Redirige à la fois la sortie standard et la sortie d'erreur vers un fichier. Par exemple, pour rediriger à la fois la sortie et les erreurs vers un fichier nommé "sortie$_e$*rreurs.txt*", *vouspouvezutiliser* :

6.2 Redirection de l'entrée standard

La redirection de l'entrée standard vous permet de rediriger le flux d'entrée d'une commande à partir d'un fichier ou d'un autre processus. Voici un opérateur de redirection couramment utilisé :

- `<` : Redirige l'entrée standard à partir d'un fichier. Par exemple, pour lire l'entrée d'une commande à partir d'un fichier nommé "entree.txt", vous pouvez utiliser : `commande < entree.txt`.

6.3 Les tubes pour chaîner des commandes

Les tubes (pipes) vous permettent de rediriger la sortie d'une commande vers l'entrée d'une autre commande, vous permettant ainsi de chaîner des commandes ensemble. Voici comment utiliser les tubes :

- `|` : Utilisez l'opérateur pipe pour rediriger la sortie d'une commande vers l'entrée d'une autre. Par exemple, pour filtrer la sortie d'une commande avec une autre commande, vous pouvez utiliser : `commande1 | commande2`.

Les redirections et les tubes sont des fonctionnalités puissantes du Shell qui vous permettent de manipuler les flux de données et de créer des pipelines complexes pour automatiser des tâches et traiter des données de manière efficace.

7 Variables d'environnement et alias

7.1 Variables d'environnement

Les variables d'environnement sont des variables globales qui définissent le comportement et les paramètres du système. Elles sont utilisées par les programmes et les processus pour obtenir des informations sur l'environnement dans lequel ils s'exécutent. Voici quelques exemples de variables d'environnement couramment utilisées :

- `PATH` : Spécifie les répertoires où le système recherche les exécutables des commandes.
- `HOME` : Spécifie le répertoire personnel de l'utilisateur.
- `USER` : Spécifie le nom de l'utilisateur actuel.
- `PWD` : Spécifie le répertoire de travail actuel.

Vous pouvez afficher la valeur d'une variable d'environnement en utilisant la commande `echo`, par exemple : `echo $PATH`.

7.2 Alias

Les alias sont des raccourcis que vous pouvez créer pour des commandes ou des ensembles de commandes fréquemment utilisées. Ils permettent de simplifier les tâches répétitives en remplaçant une commande longue par un mot-clé plus court. Voici comment définir un alias :

- `alias` : Utilisez la commande `alias` suivie du nom de l'alias, du signe égal (=) et de la commande à exécuter. Par exemple, pour créer un alias nommé `ll` qui affiche les fichiers avec des détails, vous pouvez utiliser : `alias ll='ls -l'`.

Vous pouvez lister tous les alias définis en utilisant la commande `alias` sans aucun argument. Pour supprimer un alias, utilisez la commande `unalias` suivi du nom de l'alias à supprimer.

Les variables d'environnement et les alias sont des outils puissants pour personnaliser votre environnement Shell et simplifier vos tâches quotidiennes. En les utilisant de manière appropriée, vous pouvez améliorer considérablement votre efficacité et votre productivité en ligne de commande.

8 Communication avec d'autres systèmes

8.1 Connexion à des serveurs distants via SSH

La connexion à des serveurs distants via SSH (Secure Shell) est une pratique courante pour administrer des serveurs à distance de manière sécurisée. Voici comment vous pouvez vous connecter à un serveur distant via SSH :

8.2 Connexion à des serveurs distants via SSH

La connexion à des serveurs distants via SSH (Secure Shell) est une pratique courante pour administrer des serveurs à distance de manière sécurisée. Voici comment vous pouvez vous connecter à un serveur distant via SSH :

— `ssh` : Utilisez la commande `ssh` suivie de l'adresse IP ou du nom d'hôte du serveur distant. Par exemple, pour vous connecter à un serveur distant avec l'utilisateur "utilisateur" et l'adresse IP "192.168.1.100", vous pouvez utiliser : `ssh utilisateur@192.168.1.100`.

Une fois connecté, vous pouvez exécuter des commandes sur le serveur distant comme si vous étiez physiquement connecté à celui-ci.

8.3 Transfert de fichiers via SCP ou SFTP

Le transfert de fichiers via SCP (Secure Copy Protocol) ou SFTP (SSH File Transfer Protocol) est une autre tâche courante lors de la communication avec d'autres systèmes. Voici comment vous pouvez transférer des fichiers via SCP ou SFTP :

8.4 Transfert de fichiers via SCP ou SFTP

Le transfert de fichiers via SCP (Secure Copy Protocol) ou SFTP (SSH File Transfer Protocol) est une autre tâche courante lors de la communication avec d'autres systèmes. Voici comment vous pouvez transférer des fichiers via SCP ou SFTP :

— `scp` : Utilisez la commande `scp` pour copier des fichiers entre votre système local et un système distant via SSH. Par exemple, pour copier un fichier local vers un serveur distant, vous pouvez utiliser : `scp fichier.txt utilisateur@192.168.1.100:/chemin/destination`.

— `sftp` : Utilisez la commande `sftp` pour établir une connexion interactive avec un serveur distant et transférer des fichiers de manière sécurisée. Vous pouvez utiliser des commandes similaires à celles de FTP pour naviguer dans le système de fichiers distant et transférer des fichiers.

En utilisant ces outils, vous pouvez facilement communiquer avec d'autres systèmes de manière sécurisée et efficace.

9 Optimisation des performances et gestion des ressources

9.1 Surveillance de l'utilisation des ressources

La surveillance de l'utilisation des ressources est essentielle pour optimiser les performances et assurer le bon fonctionnement d'un système. Voici quelques outils que vous pouvez utiliser pour surveiller l'utilisation des ressources :

9.2 Surveillance de l'utilisation des ressources

La surveillance de l'utilisation des ressources est essentielle pour optimiser les performances et assurer le bon fonctionnement d'un système. Voici quelques outils que vous pouvez utiliser pour surveiller l'utilisation des ressources :

- `top` : Utilisez la commande `top` pour afficher en temps réel les processus en cours d'exécution et leur utilisation des ressources telles que le processeur et la mémoire.
- `htop` : Une alternative améliorée à la commande `top` avec une interface utilisateur plus conviviale et des fonctionnalités supplémentaires telles que la navigation interactive.
- `free` : Utilisez la commande `free` pour afficher les informations sur l'utilisation de la mémoire, y compris la mémoire libre, utilisée et disponible.
- `df` : Utilisez la commande `df` pour afficher l'utilisation de l'espace disque sur les différents systèmes de fichiers montés.

En surveillant régulièrement l'utilisation des ressources, vous pouvez identifier les goulets d'étranglement et les problèmes de performances potentiels, ce qui vous permet de prendre des mesures correctives pour optimiser les performances du système.

Cette section fournit une explication détaillée sur l'utilisation d'outils tels que `top`, `htop`, `free` et `df` pour surveiller l'utilisation des ressources système, avec des exemples pratiques pour chaque outil.

10 Sécurité

10.1 Gestion des utilisateurs et des permissions

La gestion des utilisateurs et des permissions est un aspect crucial de la sécurité d'un système. Voici quelques bonnes pratiques à suivre pour renforcer la sécurité :

10.2 Gestion des utilisateurs et des permissions

La gestion des utilisateurs et des permissions est un aspect crucial de la sécurité d'un système. Voici quelques bonnes pratiques à suivre pour renforcer la sécurité :

- Limitez l'accès aux utilisateurs et aux groupes autorisés en utilisant des permissions appropriées sur les fichiers et répertoires.
- Utilisez des mots de passe forts et encouragez la rotation régulière des mots de passe.
- Désactivez les comptes d'utilisateur inutilisés et supprimez-les si nécessaire.
- Surveillez les journaux d'authentification pour détecter les tentatives d'accès non autorisées.
- Utilisez des outils de chiffrement pour protéger les données sensibles lorsqu'elles sont stockées ou transitent sur le réseau.

En suivant ces bonnes pratiques, vous pouvez renforcer la sécurité de votre système et réduire les risques de violation de la sécurité.

11 Débogage et gestion des erreurs

11.1 Journalisation des erreurs

La journalisation des erreurs est un aspect essentiel de la gestion des erreurs dans un système. Voici quelques bonnes pratiques pour la journalisation des erreurs :

11.2 Journalisation des erreurs

La journalisation des erreurs est un aspect essentiel de la gestion des erreurs dans un système. Voici quelques bonnes pratiques pour la journalisation des erreurs :

— Utilisez des niveaux de journalisation pour catégoriser les messages en fonction de leur gravité, tels que INFO, WARNING, ERROR, etc.
— Enregistrez les messages d'erreur dans un fichier de journal dédié pour faciliter l'analyse et le dépannage ultérieurs.
— Incluez des informations contextuelles telles que la date, l'heure, l'emplacement et la description de l'erreur dans les messages de journal.
— Utilisez des outils de surveillance des journaux pour surveiller les messages d'erreur en temps réel et recevoir des alertes en cas de problèmes critiques.

En suivant ces bonnes pratiques, vous pouvez améliorer la visibilité et la traçabilité des erreurs dans votre système, ce qui facilite leur résolution et réduit les temps d'arrêt.

12 Étendre les fonctionnalités avec des outils externes

12.1 Utilisation de paquets et de bibliothèques externes

L'utilisation de paquets et de bibliothèques externes est un moyen efficace d'étendre les fonctionnalités d'un système. Voici quelques bonnes pratiques à suivre :

12.2 Utilisation de paquets et de bibliothèques externes

L'utilisation de paquets et de bibliothèques externes est un moyen efficace d'étendre les fonctionnalités d'un système. Voici quelques bonnes pratiques à suivre :

— Recherchez des paquets et des bibliothèques externes bien établis et largement utilisés dans la communauté.
— Assurez-vous de comprendre la licence et les restrictions d'utilisation des paquets externes avant de les intégrer dans votre système.
— Suivez les bonnes pratiques de gestion des dépendances pour éviter les conflits et les problèmes de compatibilité entre les différentes versions des paquets.
— Utilisez des outils de gestion de paquets tels que apt, yum, pip, npm, etc., pour installer, mettre à jour et supprimer des paquets de manière efficace.

En utilisant des paquets et des bibliothèques externes de manière appropriée, vous pouvez étendre les fonctionnalités de votre système tout en minimisant les efforts de développement et en assurant la stabilité et la fiabilité du système.

13 Conclusion

Thanks for reading the entire pdf, i hope it's usefull to you. If you need any help contact me on Github