INF2132 : Systèmes d'Exploitation

TP8 - Construction de Scripts Shell Basiques

Nom de l'enseignant :
Pr. Ilias Tougui

Nom de l'assistant :
Pr. Yasser Aderghal

Table des matières

Lisez attentivement cette page

I Objectifs du TP

À la fin de ce TP, vous serez capable de :

* Écrire un script bash pour automatiser la création d'une structure de répertoires complexe
* Utiliser des variables locales et globales dans un script
* Comprendre la différence entre variables locales et globales
* Maîtriser les tests conditionnels bash : `-d`, `-f`, `-e`
* Vérifier l'existence de fichiers et répertoires avant manipulation
* Utiliser les structures conditionnelles `if-then-else`
* Comprendre le concept de code de sortie (exit code)
* Implémenter des mécanismes de protection contre les erreurs

II Prérequis

Avant de commencer ce TP, vous devez :

* Assister au CM : Programmation Shell sous Linux.
* Maîtriser les commandes de base Linux.
* Connaître les bases du shell bash.

1 Préparation de l'Environnement de travail

Durée : 40 min, Note : 7 points

En utilisant les commandes de base de Linux et les concepts de scripting shell, suivez les consignes ci-dessous pour créer automatiquement l'environnement de travail structuré de ce TP.

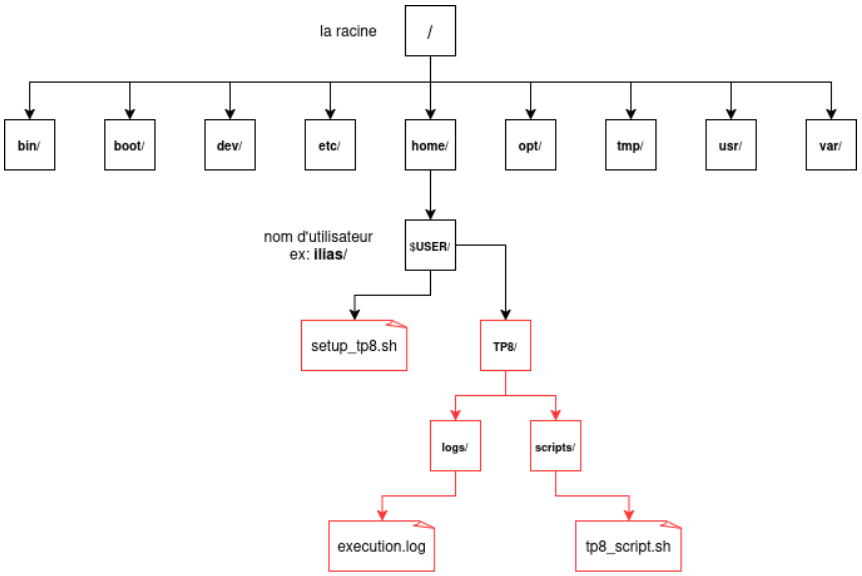

FIGURE 1 – Structure de l'environnement de travail.

1. Déplacez-vous vers le répertoire racine.
2. Déplacez-vous ensuite dans votre répertoire personnel.
3. Créez un fichier de script nommé `setup_tp8.sh`.
4. Rendez le script `setup_tp8.sh` exécutable en utilisant le mode symbolique.
5. Téléchargez le contenu du script `setup_tp8.sh` fourni sur `Connect UIR`.
6. En utilisant un éditeur de texte de votre choix, ouvrez et complétez le script en remplaçant les pointillés (. . . . .) par les commandes appropriées basées sur les commentaires définis.
7. Sauvegardez et exécutez le script `setup_tp8.sh`.
8. En utilisant les commandes `set`, le `pipe` et `grep`, listez et trouvez les variables d'environnement que vous avez créées dans le script.
9. Quelle est la différence entre la variable `utilisateur` (créée dans le script) et la variable système `$USER` ?
10. Comment afficher seulement les variables d'environnement globales au niveau de votre système ?
11. Comment rendre les variables locales globales ? Proposez un exemple avec les variables `nom` et `prenom`.
12. Qu'est-ce que la substitution de commande ? Donnez deux exemples à partir du fichier `tp8_script.sh`.
13. Appelez le professeur pour montrer le travail accompli.

2 Analyse et Amelioration du Script

Durée : 40 min, Note : 8 points (**Voir le Cours**)

Dans cet exercice, vous allez corriger le script `setup_tp8.sh` étape par étape pour qu'il gère correctement les réexécutions multiples.

1. Accédez au répertoire contenant le script `setup_tp8.sh`.
2. Exécutez le script une deuxième fois.
3. Identifiez et notez toutes les lignes qui affichent des erreurs.
4. Combien d'erreurs de type `"File exists"` apparaissent ?
5. Ces erreurs empêchent-elles le script de continuer son exécution ?

6. En bash, par convention :
   * Code de sortie `0` = ...
   * Code de sortie `non-zéro` (1, 2, etc.) = ...
7. La variable spéciale `$?` contient quoi exactement ?

8. Créez une copie du script : `cp setup_tp8.sh setup_tp8_v2.sh`
9. Ouvrez `setup_tp8_v2.sh` avec `nano`.
10. Localisez la ligne contenant `mkdir TP8`.
11. Ajoutez **avant** cette ligne un test pour vérifier si `TP8` existe :

```
if [ -d "TP8" ]; then
    echo "xxxxx Le répertoire TP8 existe déjà xxxxx"
else
    mkdir TP8
    if [ $? -eq 0 ]; then
        echo "***** Répertoire TP8 créé avec succès *****"
    else
        echo "Erreur lors de la création de TP8"
        exit 1
    fi
fi
```

12. Que fait la ligne `if [ $? -eq 0 ]` ?
13. Pourquoi utilise-t-on `-eq` au lieu de `==` pour comparer des nombres ?
14. Que fait la commande `exit 1` ?
15. Sauvegardez et testez : `./setup_tp8_v2.sh`
16. L'erreur `"cannot create directory 'TP8'"` a-t-elle disparu ?
17. Que signifie le test `-d` en bash ?

18. Dans le même fichier `setup_tp8_v2.sh`, localisez les lignes :

```
mkdir scripts logs
```

19. Remplacez ces deux lignes par une seule commande utilisant l'option `-p` :

```
mkdir -p scripts logs
echo "***** Sous-répertoires vérifiés/créés *****"
```

20. Que fait l'option `-p` de la commande `mkdir` ?
21. Pourquoi le code de sortie de `mkdir -p scripts logs` est `0` même si le répertoire existe ?
22. Sauvegardez et testez à nouveau le script.
23. Les erreurs pour `scripts` et `logs` ont-elles disparu ?

24. Localisez la ligne contenant : `date > logs/execution.log`
25. Quelle est la différence entre `>` et `»` ?
26. Dans votre script, modifiez la gestion du log comme suit :

```
# Si le fichier existe, ajouter une séparation
if [ -f "logs/execution.log" ]; then
    echo "" >> logs/execution.log
    echo "--- Réexécution ---" >> logs/execution.log
fi

# Ajouter les nouvelles informations
date >> logs/execution.log
echo "Ce script a été créé par $prenom $nom sous Linux avec le compte
$utilisateur"
```

27. Que signifie le test `-f` en bash ?
28. Sauvegardez et exécutez le script plusieurs fois.
29. Vérifiez le contenu de `logs/execution.log`. Que constatez-vous ?
30. Pourquoi est-il important de tester l'existence des fichiers/répertoires ?
31. Comment vérifier si la dernière commande a réussi ou échoué ?
32. Appelez le professeur pour validation.