Rapport TP3: Automatisation des tâches avec les scripts Shell sous Linux

Septembre 2024

Ce rapport explore les techniques de création de scripts Shell pour automatiser les tâches répétitives. Il inclut des exemples de scripts pour la gestion des fichiers, des utilisateurs et des services.

1. Partie 1

1. Créer et exécuter un script simple :

• Créez un script nommé bonjour.sh qui affiche :

• Utilisez une substitution de commande pour afficher la date.

Le resultat :

```
  GNU nano 8.1                          bonjour.sh *
#!/bin/bash

echo "Bonjour ! Bienvenue dans ce script."
echo "Nous sommes le : $(date)"
```

```
                                        kh@kh: ~
kh@kh:~$ nano bonjour.sh
kh@kh:~$ chmod +u+x ./bonjour.sh
kh@kh:~$ ./bonjourd.sh
bash: ./bonjourd.sh: No such file or directory
kh@kh:~$ ./bonjour.sh
Bonjour ! Bienvenue dans ce script.
Nous sommes le : Mon Nov 25 02:08:14 PM CET 2024
kh@kh:~$
```

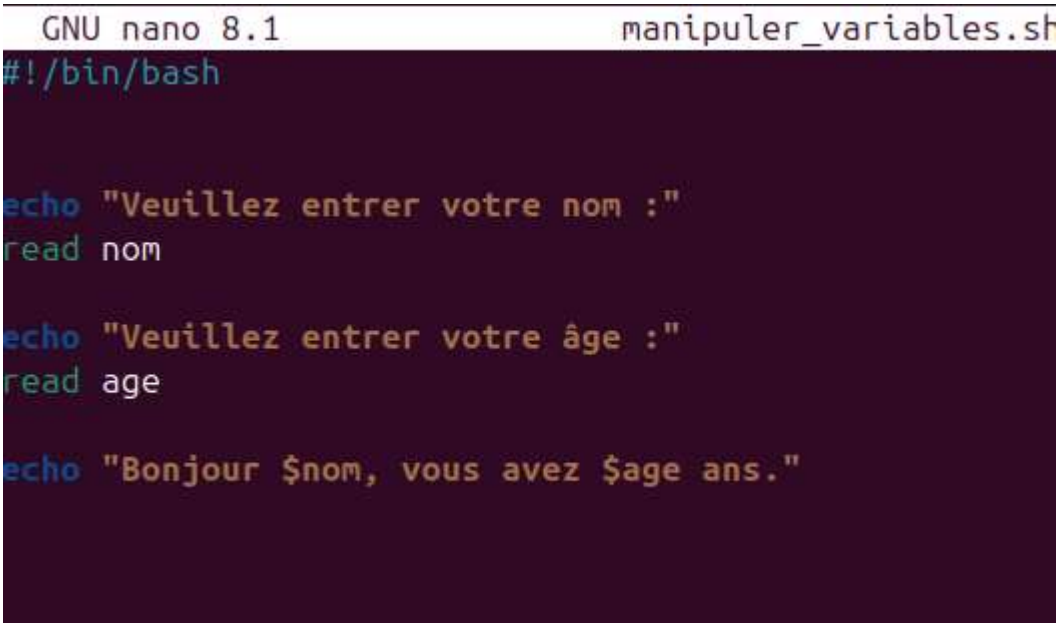
2. Manipulation de variables :

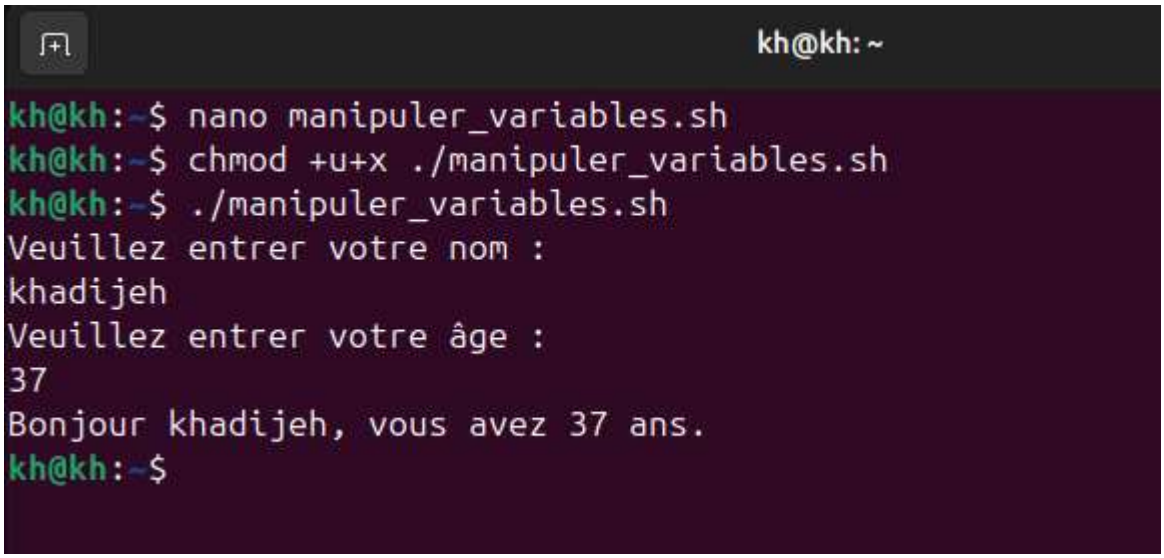
• Créez un script nommé manipuler_variables.sh : j'ai Créé un groupe en utilisant

• Demandez à l'utilisateur d'entrer son nom et son âge.

• Affichez un message avec le nom et l'âge de l'utilisateur

Le resultat :


GNU nano 8.1 manipuler_variables.sh
#!/bin/bash
echo "Veuillez entrer votre nom :"
read nom
echo "Veuillez entrer votre âge :"
read age
echo "Bonjour $nom, vous avez $age ans."


kh@kh: ~
kh@kh:~$ nano manipuler_variables.sh
kh@kh:~$ chmod +u+x ./manipuler_variables.sh
kh@kh:~$ ./manipuler_variables.sh
Veuillez entrer votre nom :
khadijeh
Veuillez entrer votre âge :
37
Bonjour khadijeh, vous avez 37 ans.
kh@kh:~$

Partie 2 :

1. Tests conditionnels :

• Créez un script nommé test_fichiers.sh :

• Demandez à l'utilisateur d'entrer un chemin de fichier.

• Testez si ce fichier :

• Existe.

• Est un répertoire ou un fichier normal.

• Affichez un message adapté à chaque cas.

J'ai utilisé la commande IF avec les arguments suviant :

-e : Vérifie si un fichier ou un répertoire avec le chemin donné existe

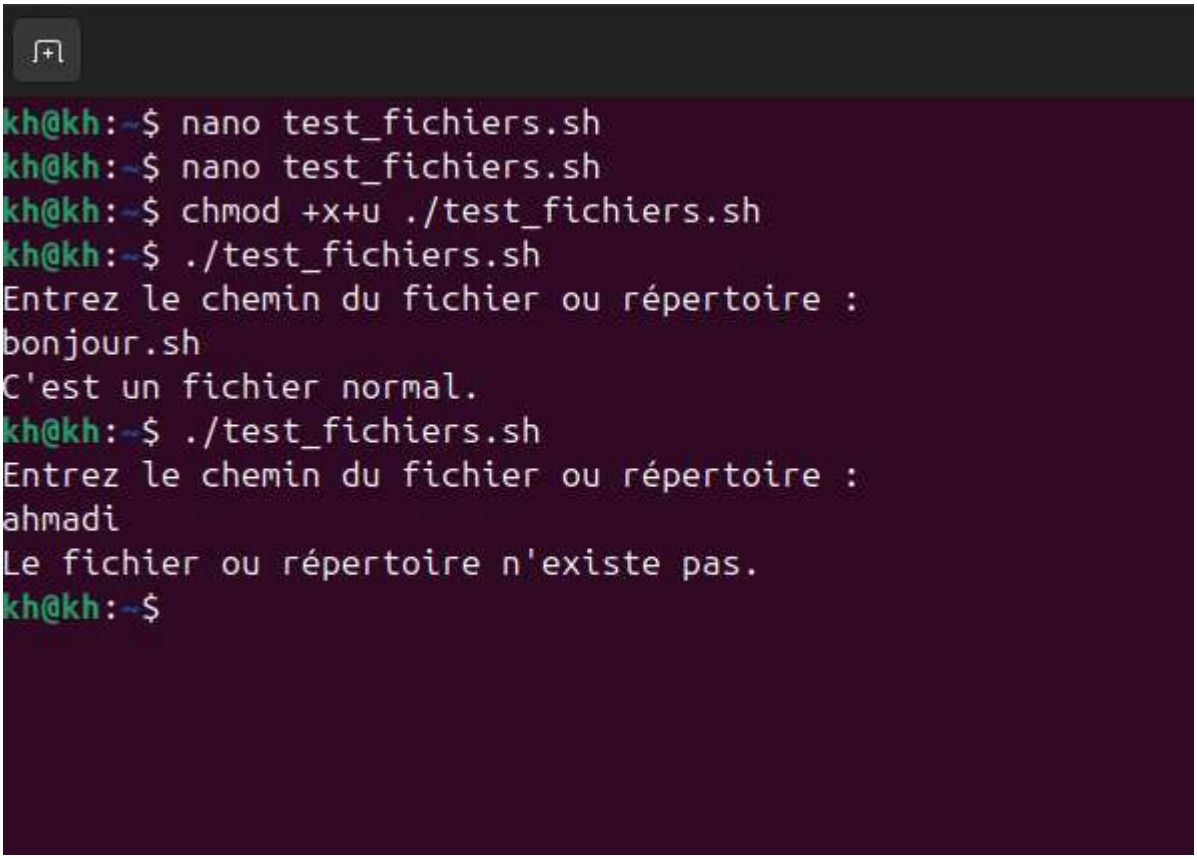
-d : vérifie si le chemin est un répertoire

-f : vérifie si le chemin pointe vers un fichier normal

```
  GNU nano 8.1                    test_fichiers.sh
#!/bin/bash


echo "Entrez le chemin du fichier ou répertoire :"
read chemin

if [ -e "$chemin" ];
then
    if [ -d "$chemin" ];
 then
        echo "C'est un répertoire."
    elif [ -f "$chemin" ];
 then
        echo "C'est un fichier normal."
    else
        echo "C'est un type de fichier spécial."
    fi
else
    echo "Le fichier ou répertoire n'existe pas."
fi
```



2. Boucles avec calculs :
• Créez un script nommé calcul_factoriel.sh :
• Demandez un entier à l'utilisateur.
• Utilisez une boucle while pour calculer et afficher le factoriel de ce nombre.

- ✓ Tout d'abord, j'ai vérifieé le nombre saisi en utilisant la commande If et les argumante lt pour qu'il ne soit pas inférieur à zéro car il n'y a pas de factorielle pour les nombres plus petits.
- ✓ -lt est un opérateur de comparaison utilisé pour vérifier si deux valeurs numériques sont plus petites.
- ✓ Finament j'ai utilisé la commande while

Le resultat :

GNU nano 8.1 calcul_factoriel.sh

```
#!/bin/bash

echo "Entrez un nombre entier :"
read nombre

if [ "$nombre" -lt 0 ]; then
    echo "Le factoriel n'est pas défini pour les nombres négatifs."
    exit 1
fi

factoriel=1
i=$nombre

while [ $i -gt 1 ]; do
    factoriel=$((factoriel * i))
    i=$((i - 1))
done

echo "Le factoriel de $nombre est $factoriel."
```

```
kh@kh:~$ nano calcul_factoriel.sh
kh@kh:~$ chmod +x+u /.c^C
kh@kh:~$ chmod +x+u calcul_factoriel.sh
kh@kh:~$ ./calcul_factoriel.sh
Entrez un nombre entier :
5
Le factoriel de 5 est 120.
kh@kh:~$ ./calcul_factoriel.sh
Entrez un nombre entier :
3
Le factoriel de 3 est 6.
kh@kh:~$
```

3. Menu interactif :

• Créez un script nommé menu.sh :

• Affichez un menu avec les options suivantes :

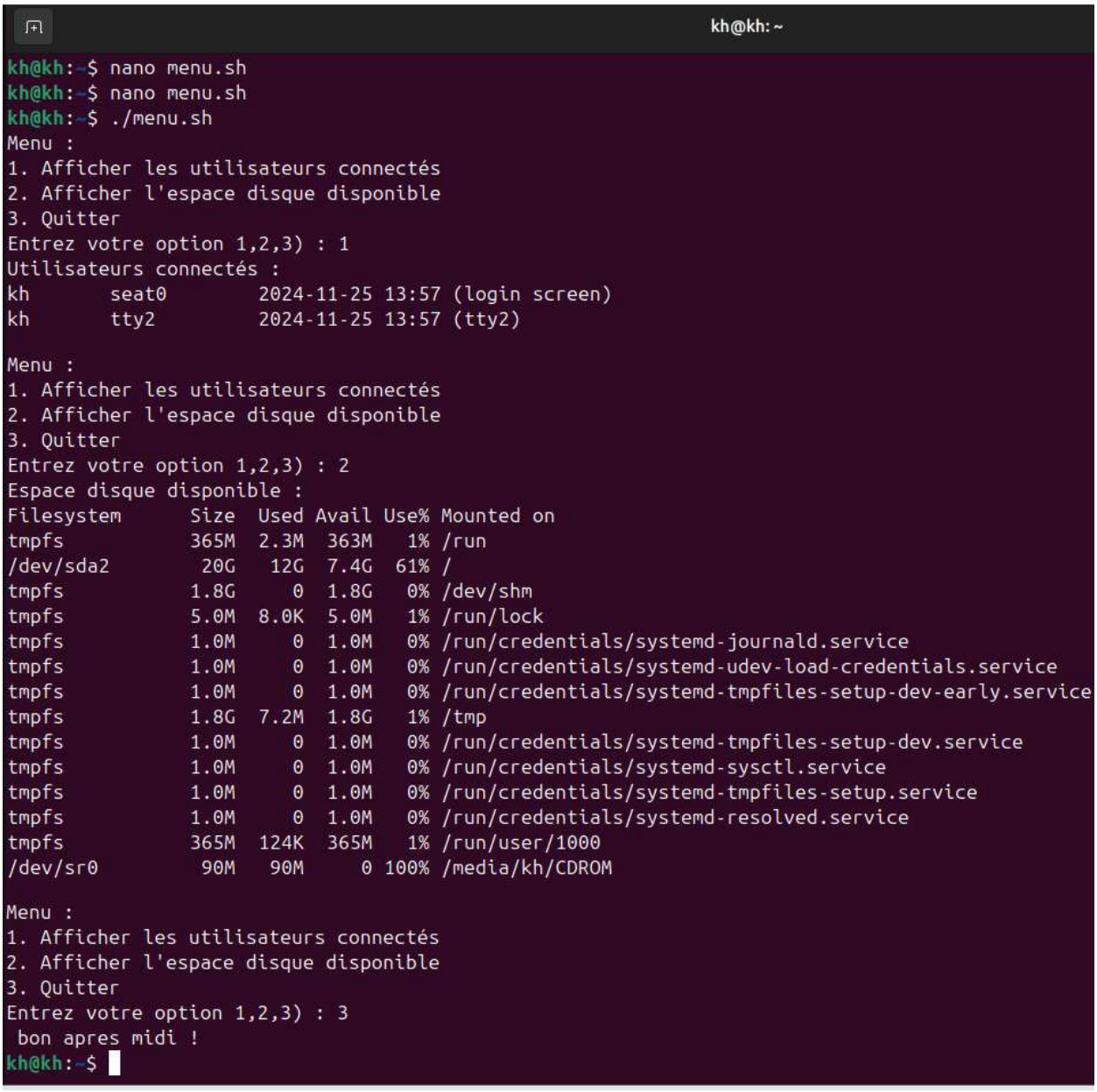
1. Afficher les utilisateurs connectés.
2. Afficher l’espace disque disponible.
3. Quitter.

• Faites un traitement en fonction du choix de l’utilisateur et utilisez la commande break pour quitter.

- ✓ J’utilise la commande while true car L'utilisateur doit pouvoir sélectionner à plusieurs reprises différentes options de menu.
- ✓ J’utilise la commande case pour faire l’option
- ✓ J’utilise la command who pour afficher les utilisateurs connectés.
- ✓ J’utilise la command df pour afficher l’espace disque disponible.
- ✓ J’utilise la command break pour quitter.

Le resultat :

```
  GNU nano 8.1
#!/bin/bash
while true; do
    echo "Menu :"
    echo "1. Afficher les utilisateurs connectés"
    echo "2. Afficher l'espace disque disponible"
    echo "3. Quitter"
    echo -n "Entrez votre option 1,2,3) : "
    read option
    case $option in
        1)
            echo "Utilisateurs connectés :"
            who
            ;;
        2)
            echo "Espace disque disponible :"
            df -h
            ;;
        3)
            echo " bon apres midi !"
            break
            ;;
        *)
            echo "option invalide, essayez encore."
            ;;
    esac
    echo ""
done
```



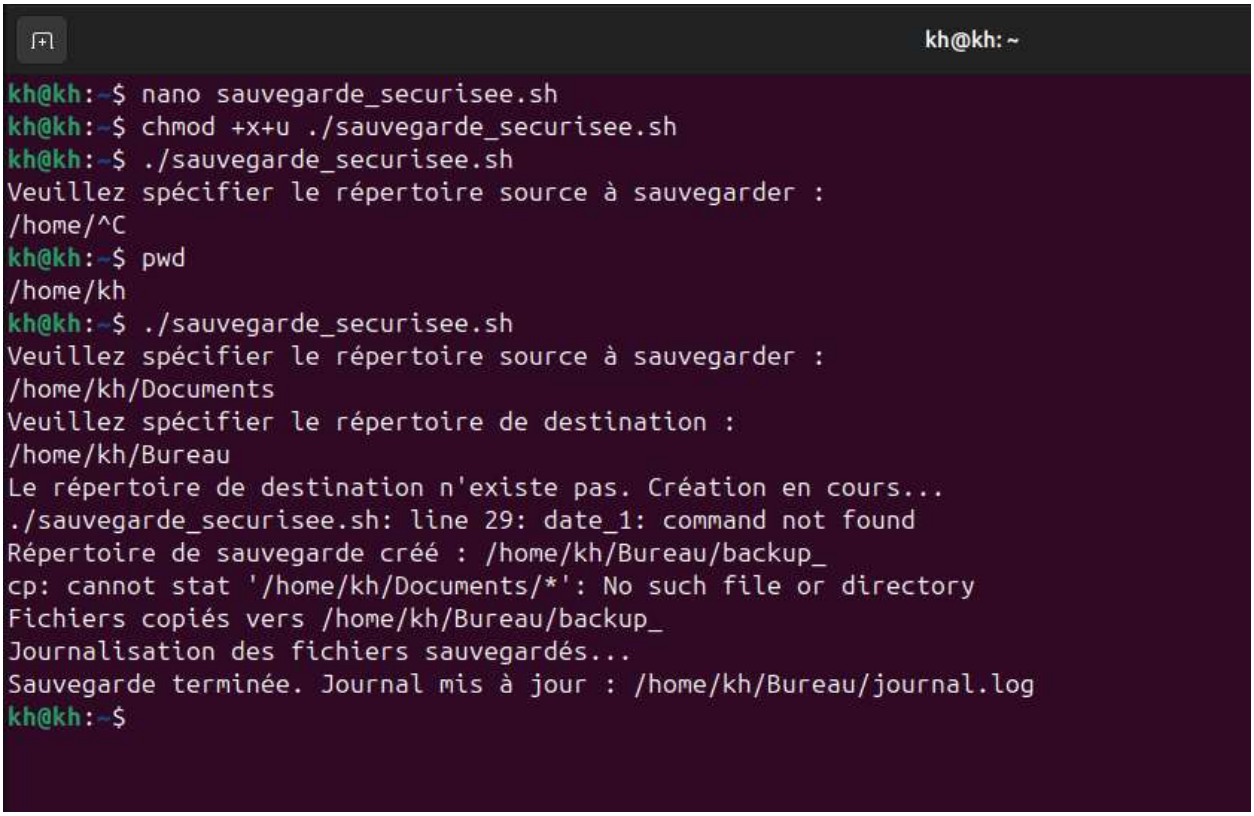
Partie 3 :

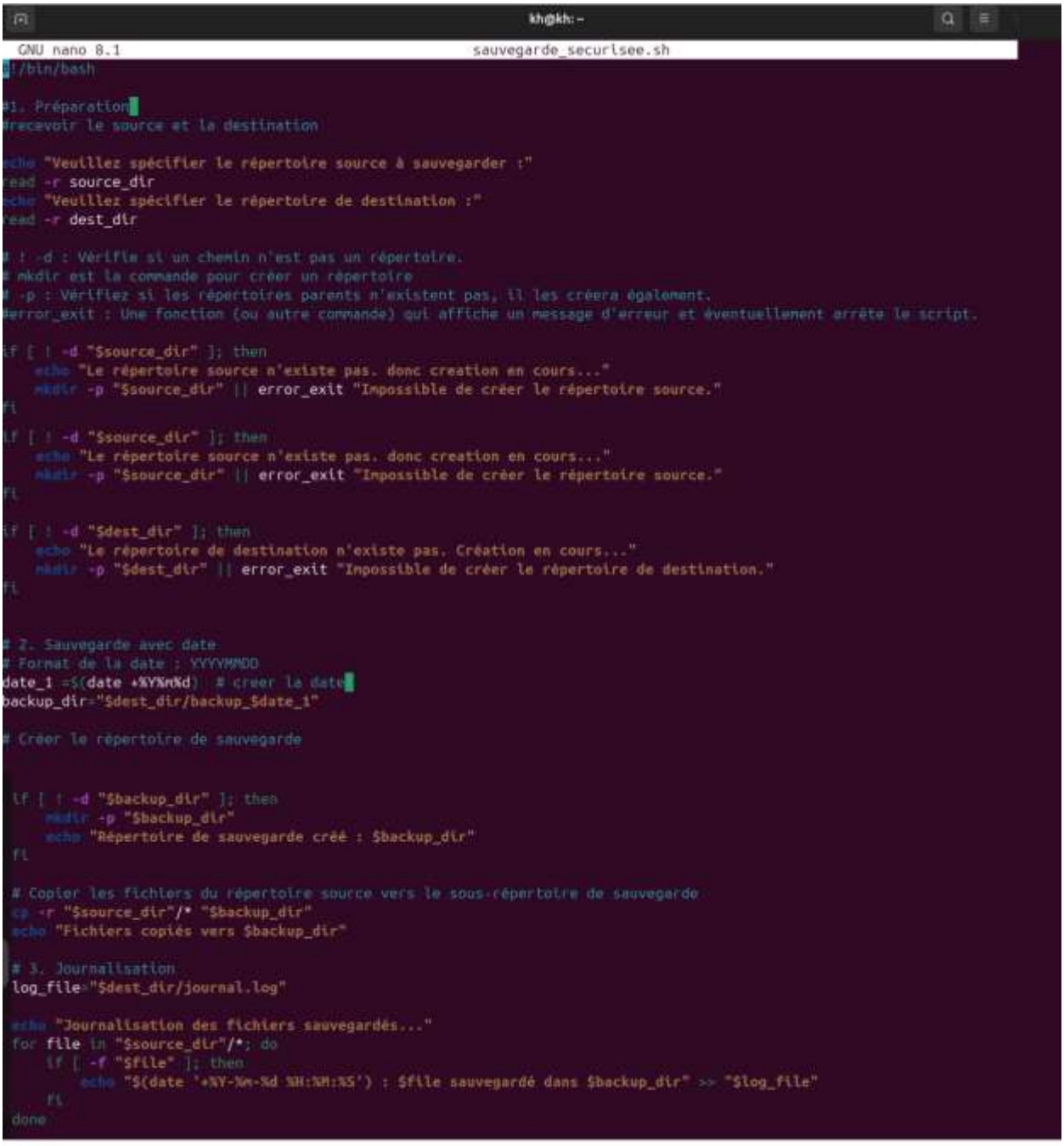
But : Automatiser sauvegarde sécurisée.

But : Automatiser une sauvegarde sécurisée.
Créez un script complet nommé sauvegarde_securisee.sh :
1. Préparation :
• Demandez à l'utilisateur de spécifier un répertoire source à sauvegarder et un répertoire de destination.
• Vérifiez si les deux répertoires existent. Si l'un des deux est absent, créez-le.
2. Sauvegarde avec date :

• Ajoutez un sous-répertoire dans le répertoire de destination nommé selon la date du jour (backup_YYYYMMDD) (formattage de la date).
• Copiez tous les fichiers du répertoire source vers ce sous-répertoire.
3. Journalisation :
• Créez ou mettez à jour un fichier journal.log dans le répertoire de destination :
• Enregistrez les noms des fichiers sauvegardés avec la date et l'heure.

Le resultat:




kh@kh: ~
GNU nano 8.1
sauvegarde_securisee.sh
#!/bin/bash
#1. Préparation
#recevoir le source et la destination
echo "Veuillez spécifier le répertoire source à sauvegarder :"
read -r source_dir
echo "Veuillez spécifier le répertoire de destination :"
read -r dest_dir
! -d : Vérifie si un chemin n'est pas un répertoire.
mkdir est la commande pour créer un répertoire
-p : Vérifiez si les répertoires parents n'existent pas, il les créera également.
#error_exit : Une fonction (ou autre commande) qui affiche un message d'erreur et éventuellement arrête le script.
if [! -d "$source_dir"]; then
echo "Le répertoire source n'existe pas. donc creation en cours..."
mkdir -p "$source_dir" || error_exit "Impossible de créer le répertoire source."
fi
if [! -d "$source_dir"]; then
echo "Le répertoire source n'existe pas. donc creation en cours..."
mkdir -p "$source_dir" || error_exit "Impossible de créer le répertoire source."
fi
if [! -d "$dest_dir"]; then
echo "Le répertoire de destination n'existe pas. Création en cours..."
mkdir -p "$dest_dir" || error_exit "Impossible de créer le répertoire de destination."
fi
# 2. Sauvegarde avec date
# Format de la date : YYYYMMDD
date_1 =$(date +%Y%m%d) # creer la date
backup_dir="$dest_dir/backup_$date_1"
# Créer le répertoire de sauvegarde
if [! -d "$backup_dir"]; then
mkdir -p "$backup_dir"
echo "Répertoire de sauvegarde créé : $backup_dir"
fi
Copier les fichiers du répertoire source vers le sous-répertoire de sauvegarde
cp -r "$source_dir"/* "$backup_dir"
echo "Fichiers copiés vers $backup_dir"
# 3. Journalisation
log_file="$dest_dir/journal.log"
echo "Journalisation des fichiers sauvegardés..."
for file in "$source_dir"/*; do
if [-f "$file"]; then
echo "$(date '+%Y-%m-%d %H:%M:%S') : $file sauvegardé dans $backup_dir" >> "$log_file"
fi
done