1. Bash

Le langage Bash (Bourne Again SHell) est un langage de script largement utilisé dans les systèmes Unix et Linux. Il offre une syntaxe simple et flexible pour automatiser des tâches et exécuter des commandes système. En utilisant Bash, nous pouvons développer des scripts pour simplifier et automatiser la gestion des utilisateurs et des sauvegardes. Le langage Bash offre des fonctionnalités telles que les variables, les structures conditionnelles, les boucles et la manipulation de fichiers, ce qui permet de créer des scripts puissants et efficaces. Il s'agit d'un outil essentiel pour automatiser les tâches répétitives, réduire les erreurs et améliorer l'efficacité dans notre projet de gestion des utilisateurs et des sauvegardes.

2. La réalisation :

Afin de réaliser le projet on avait besoin d'un éditeur de texte pour cela on a utiliser dans notre réalisation l'éditeur de texte nommée Vim -Vim est un éditeur de texte très populaire utilisé principalement dans les environnements Unix et Linux- Grace a Vim on a peu créer notre premier fichier .sh l'extension de fichier Bash qui vas porter notre script qui vas tout d'abord commencer par la création des utilisateurs avec leurs mot de passe, les groupes et même l'ajout et la suppression à partir d'un fichier .CSV grâce à l'ensemble des commande suivante ;

3. L'ajout des utilisateurs :

Tout d'abord pour ajouter un utilisateur on établit le script suivant :

```
if [ "$order2" == "1" ]; then

        read -p "Enterer login du nouveau utilisateur: " login
        read -p "Entrer le mot de passe: " password
        read -p "Entrer le groupe: " groupe

        if grep -q "^$groupe:" /etc/group; then
              echo "L'utilisateur $login a été ajouté au groupe $groupe"
              echo "Login: $login, Password: $password, Groupe: $groupe" >> "$user_file"
              echo "$login" >> $groupe.txt
              echo "AU, $(date '+%Y-%m-%d %H:%M:%S'), $login, $groupe" >> "$gestion_user_log"
              sudo useradd -m -p "$(openssl passwd -1 '$password')" -G "$groupe" "$login"
        else
              echo "Le groupe $groupe n'existe pas. Veuillez d'abord créer le groupe."
              echo "AU, $(date '+%Y-%m-%d %H:%M:%S'), $login, $groupe (raison de l'erreur)" >> "$gestion_user_log"
        fi
```

Ici notre script va lire grâce a la commande read -p l'ensemble des variables login, password et groupe et il vas chercher si le nom du groupe existe avec le condition if et la commande grep, si le groupe existe il vas ajouter l'utilisateur avec la commande sudo useradd et il vas sauvegarder les information dans le fichier.log et dans des fichier texte.

4. L’ajout des groupes :

L’ajout des groupes consiste à :

```
elif [ "$order2" == "2" ]; then

        read -p "Entrer le nom du groupe: " grp
        sudo groupadd "$grp"
        echo "le groupe $grp a bien etait ajouter"
        echo "Groupe : $grp" >> "$grp_ajouter"
        echo "AG, $(date '+%Y-%m-%d %H:%M:%S'), $groupe" >> "$gestion_user_log"
        touch $grp.txt
```

Lire la variable grp qui détermine le nom du groupe a ajouter et l’ajout du groupe avec la commande groupadd la commande touch pour crée un fichier texte avec le nom du groupe qui vas contenir le nom des utilisateurs.

5. L’affichage des utilisateurs d’un groupe :

Afin d’afficher les utilisateurs d’un groupe on met a disposition le script suivant :

```
elif [ "$order2" == "3" ]; then

        echo "Liste des groupes"
        cat "$grp_ajouter"
        read -p "Entrer le nom du groupe" groupe
        echo "les utilisateurs du $groupe"
        getent group $groupe
        cat $groupe.txt
```

Ici on liste l’ensembles des groupes qui sont déjà dans un fichier texte nommée grp_ajouter et on lit le nom de groupe ou on ouvre le fichier texte grâce à la commande cat avec le nom du groupe qui contient tous les utilisateurs de ce dernier.

6. La suppression d’un utilisateur :

Pour supprimer un utilisateur :

```
elif [ "$order2" == "4" ]; then

        cat $user_file
        echo "Entrer le nom d'utilisateur a supprimer"
        read nom
        sudo userdel $nom
        grep -n "$nom" users.txt | cut -d ":" -f 1 | xargs -I {} sed -i '{}d' users.txt
        echo "DU, $(date '+%Y-%m-%d %H:%M:%S'), $login, $groupe" >> "$gestion_user_log"
        echo "l'utilisateur $nom a etait supprimer"
```

Ici l’admin vas taper le nom a supprimer et il vas l’affecter a la variable nom et il vas supprimer l’utilisateur nommée grâce à la commande userdel et il vas supprimer

le ligne du fichier texte grâce à la commande grep… ou le nom de l'utilisateur a supprimer existe.

7. La suppression d'un groupe :

```
elif [ "$order2" == "5" ]; then

        echo "Liste des groupes"
        cat "$grp_ajouter"
        echo "Entrer le nom du groupe a supprimer"
        read nom
        while read -r username; do
            sudo userdel "$username"
            grep -n "$username" $nom.txt | cut -d ":" -f 1 | xargs -I {} sed -i '{}d' $nom.txt
        done < $nom.txt
        sudo groupdel -f $nom
        grep -n "$nom" grps.txt | cut -d ":" -f 1 | xargs -I {} sed -i '{}d' grps.txt
        grep -n "$nom" users.txt | cut -d ":" -f 1 | xargs -I {} sed -i '{}d' users.txt
        echo "DG, $(date '+%Y-%m-%d %H:%M:%S'), $groupe" >> "$gestion_user_log"
        rm $nom.txt
```

Afin de supprimer un groupe avec ces utilisateurs on commence par lister les groupes disponibles grâce a la commande cat après on entre le nom du groupe a supprimer et on lance une boucle qui vas lire tout les utilisateurs dans le groupes et les supprimer grâce a la commande userdel et a la fin on supprime le groupe avec la commande groupdel.

8. L'ajout à partir d'un fichier CSV :

```
elif [ "$order2" == "6" ]; then

        echo "Debut ajout"
        while IFS=',' read -r username password group
        do
          if [[ $username == "Username" ]]; then
              continue
          fi
          sudo useradd -m "$username"
          echo "$username:$password" | sudo chpasswd
          sudo usermod -a -G "$group" "$username"
          echo "User $username created with password $password and added to group $group"
        done < "$csv_file"
        echo "ajout complet"
```

Ici le scripte va lire le fichier csv et il va séparer le contenu par des virgules pour capter le username password et le group et on établit l'ajout avec useradd.

9. La suppression à partir d'un fichier CSV :

Pour établir la suppression à partir d'un fichier CSV on a utilisé le script suivant :

```
elif [ "$order2" == "7" ]; then

      echo "Debut supression"
      echo "Entrer le chemain du fichier CSV si c'est dans le meme dossier juste le nom:"
      read csv_file

      while IFS=',' read -r username group
      do
       sudo userdel -r "$username"
       sudo groupdel "$group"
       echo "Deleted user: $username, group: $group"
      done < "$csv_file"
      echo "Suppression complete"
```

Le script va lire les noms et le mot de passe des utilisateurs a partie du fichier .csv et il lance une boucle pour établir la suppression des utilisateurs et des groupes grâce à userdel et groupdel.

10.L'archivage des répertoires :

Pour établir l'archivage on présente un menu de trois éléments :

```
echo "1. Archiver un repertoire unique"
echo "2. Archiver pleusieurs repertoire"
echo "3. Archiver a partir d'un fichier text"
read order3
```

Si ont choisi le choix numéro 1 on vas déclencher le script suivant :

```
if [ "$order3" == "1" ]; then

   read -p "Enter the directory name to be archived: " source_directory
   archive_name="$source_director.tar.gz"
   full_source_directory="$(pwd)/$source_directory"
   if [ -d "$full_source_directory" ]; then
      tar -czf "$archive_directory/$archive_name" -C "$(dirname "$full_source_directory")" "$(basename "$full_source_directory")"
      echo "Directory archived successfully."
   else
      echo "Error: Directory '$full_source_directory' does not exist."
      exit 1
   fi
```

Grace au script suivant on demande d'entrer la source ou le chemin de répertoire a fin de l'archiver grâce à la commande qui vas l'archiver dans le dossier Archive et lui donner l'extension .tar.gz.

Si ont choisi 2 :

```
elif [ "$order3" == "2" ]; then

  read -p "Enter the directories to be archived (separer par espaces): " -a directories
  for dir in "${directories[@]}"
  do
  if [ -d "$dir" ]; then
      archive_name="$(basename "$dir").tar.gz"
      archive_path="$archive_directory/$archive_name"
      tar -czf "$archive_path" -C "$(dirname "$dir")" "$(basename "$dir")"
      if [ $? -eq 0 ]; then
         echo "Successfully archived $dir"
      else
         echo "Failed to archive $dir"
      fi
  else
      echo "Directory $dir does not exist"
  fi
  done
fi
```

Ici on va lire les chemins entrer grâce a une boucle for qui vas archiver tous les répertoires mentionner.

Si ont choisi 3 :

```
elif [ "$order3" == "3" ]; then

   read -p "Entrez le chemin vers le fichier texte : " file_path
   if [ -f "$file_path" ]; then
       while IFS= read -r dir
       do
       if [ -d "$dir" ]; then
           archive_name="$(basename "$dir").tar.gz"
           archive_path="$archive_directory/$archive_name"
           tar -czf "$archive_path" -C "$(dirname "$dir")" "$(basename "$dir")"
           if [ $? -eq 0 ]; then
              echo "Répertoire $dir archivé avec succès"
           else
              echo "Échec de l'archivage du répertoire $dir"
           fi
       else
           echo "Le répertoire $dir n'existe pas"
       fi
       done < "$file_path"
  else
       echo "Le fichier $file_path n'existe pas"
  fi
```

Ici on va lire les chemin grâce a une boucle et on les archive par ordre.

11.Le menu de navigation :

Pour l'établir on a mis une boucle avec les choix disponible a réaliser et quitter comme dernier choix :

```
elif [ "$order2" == "9" ]; then

        break

fi
done
```

12.Le code :

```
#!/bin/bash

user_file="users.txt"
grp_ajouter="grps.txt"
csv_file="users.csv"
archive_directory="/home/Archive"
gestion_user_log="GestionUser.log"
archivage_log="Archivage.log"

echo "1. Entrer comme admin"
read order

if [ "$order" == "1" ]; then

    echo "entrer votre mot de passe admin"
    read password

    if [ "$password" == "admin" ]; then
            while true
            do
            echo
        echo "1. Ajouter un utilisateur"
        echo "2. Ajouter un groupe"
        echo "3. Voir les utilisateurs d'un groupe"
        echo "4. Pour supprimer un utilisateurs"
        echo "5. Pour supprimer un groupe et ces utilisateurs"
        echo "6. Pour ajout avec un fichier csv"
        echo "7. Pour suppression avec un fichier csv"
        echo "8. Gerer les archives"
        echo "9. Quit"
        read order2
        echo
```

```
45.
46.         if [ "$order2" == "1" ]; then
47.
48.             read -p "Enterer login du nouveau utilisateur: " login
49.             read -p "Entrer le mot de passe: " password
50.             read -p "Entrer le groupe: " groupe
51.
52.             if grep -q "^$groupe:" /etc/group; then
53.                   echo "L'utilisateur $login a été ajouté au groupe
    $groupe"
54.                   echo "Login: $login, Password: $password, Groupe:
    $groupe" >> "$user_file"
55.                   echo "$login" >> $groupe.txt
56.                   echo "AU, $(date '+%Y-%m-%d %H:%M:%S'), $login, $groupe"
    >> "$gestion_user_log"
57.                   sudo useradd -m -p "$(openssl passwd -1 '$password')" -G
    "$groupe" "$login"
58.             else
59.                   echo "Le groupe $groupe n'existe pas. Veuillez d'abord
    créer le groupe."
60.                   echo "AU, $(date '+%Y-%m-%d %H:%M:%S'), $login, $groupe
    (raison de l'erreur)" >> "$gestion_user_log"
61.             fi
62.
63.         elif [ "$order2" == "2" ]; then
64.
65.                 read -p "Entrer le nom du groupe: " grp
66.                     sudo groupadd "$grp"
67.                 echo "le groupe $grp a bien etait ajouter"
68.                 echo "Groupe : $grp" >> "$grp_ajouter"
69.                 echo "AG, $(date '+%Y-%m-%d %H:%M:%S'), $groupe" >>
    "$gestion_user_log"
70.                 touch $grp.txt
71.
72.         elif [ "$order2" == "3" ]; then
73.
74.                 echo "Liste des groupes"
75.                 cat "$grp_ajouter"
76.                 read -p "Entrer le nom du groupe" groupe
77.                 echo "les utilisateurs du $groupe"
78.                 getent group $groupe
79.
80.         elif [ "$order2" == "4" ]; then
81.
82.                 cat $user_file
```

```
83.                 echo "Entrer le nom d'utilisateur a supprimer"
84.                 read nom
85.                 sudo userdel $nom
86.                 grep -n "$nom" users.txt | cut -d ":" -f 1 | xargs -I {}
   sed -i '{}d' users.txt
87.                 echo "DU, $(date '+%Y-%m-%d %H:%M:%S'), $login, $groupe"
   >> "$gestion_user_log"
88.                 echo "l'utilisateur $nom a etait supprimer"
89.
90.         elif [ "$order2" == "5" ]; then
91.
92.                 echo "Liste des groupes"
93.                 cat "$grp_ajouter"
94.                 echo "Entrer le nom du groupe a supprimer"
95.                 read nom
96.                 while read -r username; do
97.                     sudo userdel "$username"
98.                     grep -n "$username" $nom.txt | cut -d ":" -f 1 | xargs
   -I {} sed -i '{}d' $nom.txt
99.                 done < $nom.txt
100.                      sudo groupdel -f $nom
101.                      grep -n "$nom" grps.txt | cut -d ":" -f 1 | xargs -I
   {} sed -i '{}d' grps.txt
102.                      grep -n "$nom" users.txt | cut -d ":" -f 1 | xargs -
   I {} sed -i '{}d' users.txt
103.                      echo "DG, $(date '+%Y-%m-%d %H:%M:%S'), $groupe" >>
   "$gestion_user_log"
104.                      rm $nom.txt
105.
106.             elif [ "$order2" == "6" ]; then
107.
108.                     echo "Debut ajout"
109.                     while IFS=',' read -r username password group
110.                     do
111.                       if [[ $username == "Username" ]]; then
112.                           continue
113.                       fi
114.                       sudo useradd -m "$username"
115.                       echo "$username:$password" | sudo chpasswd
116.                       sudo usermod -a -G "$group" "$username"
117.                       echo "User $username created with password
   $password and added to group $group"
118.                     done < "$csv_file"
119.                     echo "ajout complet"
120.
```

```
121.             elif [ "$order2" == "7" ]; then
122.
123.                   echo "Debut supression"
124.                   echo "Entrer le chemain du fichier CSV si c'est dans
   le meme dossier juste le nom:"
125.                           read csv_file
126.
127.                   while IFS=',' read -r username group
128.                   do
129.                    sudo userdel -r "$username"
130.                    sudo groupdel "$group"
131.                    echo "Deleted user: $username, group: $group"
132.                   done < "$csv_file"
133.                   echo "Suppression complete"
134.
135.             elif [ "$order2" == "8" ]; then
136.
137.                  echo "1. Archiver un repertoire unique"
138.                  echo "2. Archiver pleusieurs repertoire"
139.                  echo "3. Archiver a partir d'un fichier text"
140.                  read order3
141.
142.                      if [ "$order3" == "1" ]; then
143.
144.                         read -p "Enter the directory name to be
   archived: " source_directory
145.                         archive_name="$source_director.tar.gz"
146.                         full_source_directory="$(pwd)/$source_directory
   "
147.                         if [ -d "$full_source_directory" ]; then
148.                            tar -czf "$archive_directory/$archive_name"
   -C "$(dirname "$full_source_directory")" "$(basename
   "$full_source_directory")"
149.                            echo "Directory archived successfully."
150.                         else
151.                            echo "Error: Directory
   '$full_source_directory' does not exist."
152.                            exit 1
153.                         fi
154.
155.                      elif [ "$order3" == "2" ]; then
156.
157.                         read -p "Enter the directories to be archived
   (separer par espaces): " -a directories
158.                         for dir in "${directories[@]}"
```

```
159.                             do
160.                             if [ -d "$dir" ]; then
161.                                 archive_name="$(basename "$dir").tar.gz"
162.                                 archive_path="$archive_directory/$archive_n
    ame"
163.                                 tar -czf "$archive_path" -C "$(dirname
    "$dir")" "$(basename "$dir")"
164.                                 if [ $? -eq 0 ]; then
165.                                    echo "Successfully archived $dir"
166.                                 else
167.                                    echo "Failed to archive $dir"
168.                                 fi
169.                             else
170.                                 echo "Directory $dir does not exist"
171.                             fi
172.                             done
173.                         fi
174.
175.                         elif [ "$order3" == "3" ]; then
176.
177.                             read -p "Entrez le chemin vers le fichier texte
    : " file_path
178.                             if [ -f "$file_path" ]; then
179.                                 while IFS= read -r dir
180.                                 do
181.                                 if [ -d "$dir" ]; then
182.                                     archive_name="$(basename
    "$dir").tar.gz"
183.                                     archive_path="$archive_directory/$archi
    ve_name"
184.                                     tar -czf "$archive_path" -C "$(dirname
    "$dir")" "$(basename "$dir")"
185.                                     if [ $? -eq 0 ]; then
186.                                        echo "Répertoire $dir archivé avec
    succès"
187.                                     else
188.                                        echo "Échec de l'archivage du
    répertoire $dir"
189.                                     fi
190.                                 else
191.                                     echo "Le répertoire $dir n'existe pas"
192.                                 fi
193.                                 done < "$file_path"
194.                             else
195.                                 echo "Le fichier $file_path n'existe pas"
```

```
                        fi


            elif [ "$order2" == "9" ]; then

                    break

            fi
            done

        else
            echo "mot de passe incorrect"
            fi
        fi
```