



Institut National de Statistique et d'Économie Appliquée

Systeme de Login en Shell Script

Étudiant : LEGSSAIR Issam

Filière : Biostatistique, Démographie et Big Data

Encadrant : M. Anouar Bouchal

Année universitaire : 2024 – 2025

Table des matières

1	Introduction	2
2	Analyse et conception	2
2.1	Définition d'un système de login	2
2.2	Objectifs fonctionnels	2
2.3	Architecture du programme	2
3	Fonction register()	2
3.1	Exécution	4
4	Fonction login()	4
4.1	Connexion réussie	5
4.2	Connexion échouée	6
5	Menu principal	6
5.1	Affichage du menu	7
6	Code source complet	7
7	Comprendre le hachage et SHA256	9
7.1	Pourquoi utiliser un hash ?	9
7.2	SHA256	9
8	Compilation et exécution	9
8.1	Création du fichier	9
8.2	Rendre exécutable	10
8.3	Exécution du script	10
9	Conclusion	10

1 Introduction

Ce rapport présente un projet réalisé dans le cadre du l'élément de Programmation Système, développé spécifiquement sous environnement Debian. L'objectif est de réaliser un mini système de login sécurisé en ligne de commande. Il permet à un utilisateur de s'inscrire, de se connecter, et de protéger ses identifiants à l'aide d'un hachage SHA256.

2 Analyse et conception

2.1 Définition d'un système de login

Un système de login permet à un utilisateur de s'authentifier via un identifiant et un mot de passe. Ce système repose sur une vérification des informations saisies, généralement protégées pour des raisons de sécurité.

2.2 Objectifs fonctionnels

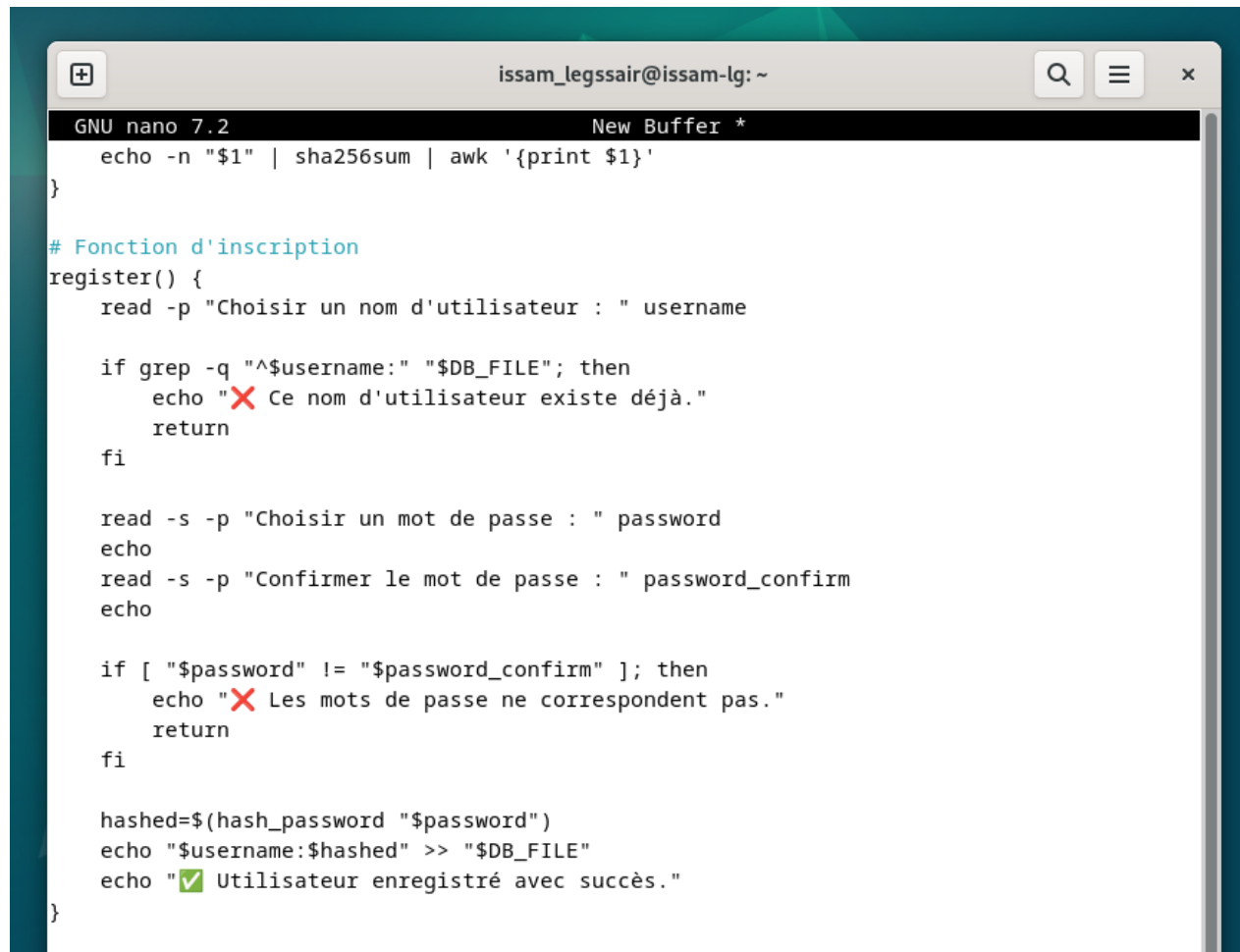
- Lire un nom d'utilisateur et un mot de passe.
- Vérifier l'authenticité des identifiants.
- Protéger les mots de passe via hachage.
- Gérer les utilisateurs à travers un fichier.
- Proposer un menu d'options simple.

2.3 Architecture du programme

- Script principal : `login_system.sh`
- Fichier de base de données : `users.db`
- Fonctions : `register()`, `login()`, menu

3 Fonction `register()`

La fonction `register()` permet à un utilisateur de créer un compte. Elle vérifie que le nom n'existe pas, demande un mot de passe deux fois, puis enregistre un hachage SHA256 dans un fichier.



The image shows a terminal window with a title bar that reads "issam_legssair@issam-lg: ~". Inside the terminal, the GNU nano 7.2 editor is open, displaying the source code for a function named `register()`. The code is written in Bash and includes comments in French. It prompts the user for a username and password, checks if the username already exists in a database file, and verifies if the passwords match. If successful, it hashes the password and appends the username and hash to the database file.

```
GNU nano 7.2 New Buffer *
echo -n "$1" | sha256sum | awk '{print $1}'
}

# Fonction d'inscription
register() {
    read -p "Choisir un nom d'utilisateur : " username

    if grep -q "^$username:" "$DB_FILE"; then
        echo "✗ Ce nom d'utilisateur existe déjà."
        return
    fi

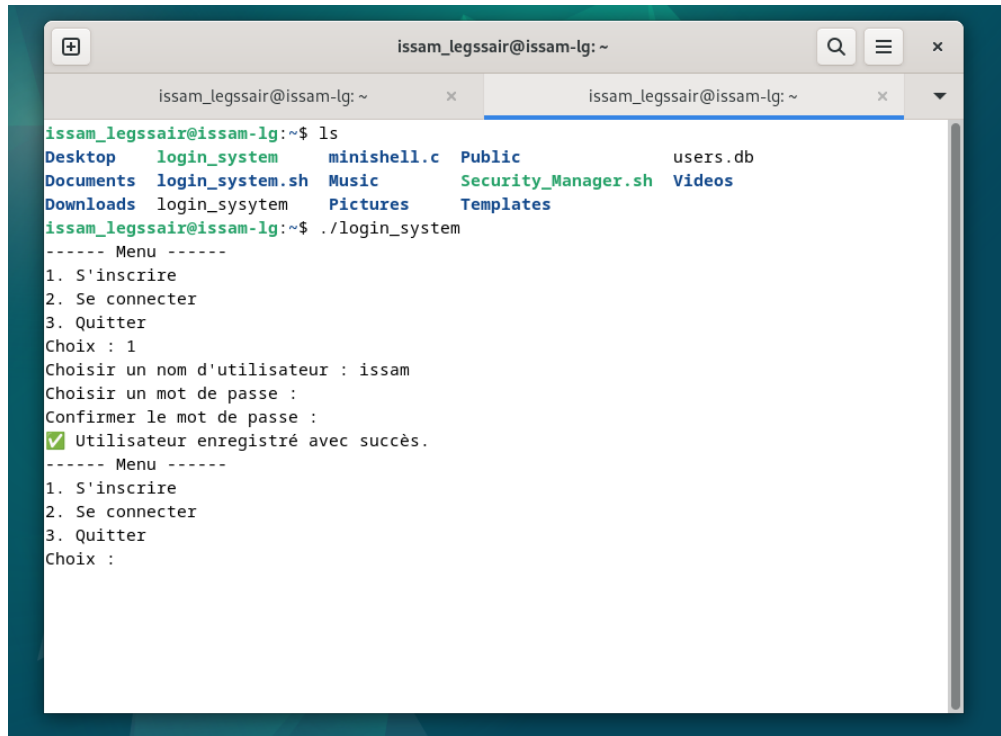
    read -s -p "Choisir un mot de passe : " password
    echo
    read -s -p "Confirmer le mot de passe : " password_confirm
    echo

    if [ "$password" != "$password_confirm" ]; then
        echo "✗ Les mots de passe ne correspondent pas."
        return
    fi

    hashed=$(hash_password "$password")
    echo "$username:$hashed" >> "$DB_FILE"
    echo "✓ Utilisateur enregistré avec succès."
}
```

FIGURE 1 – Code source de la fonction `register()`

3.1 Exécution

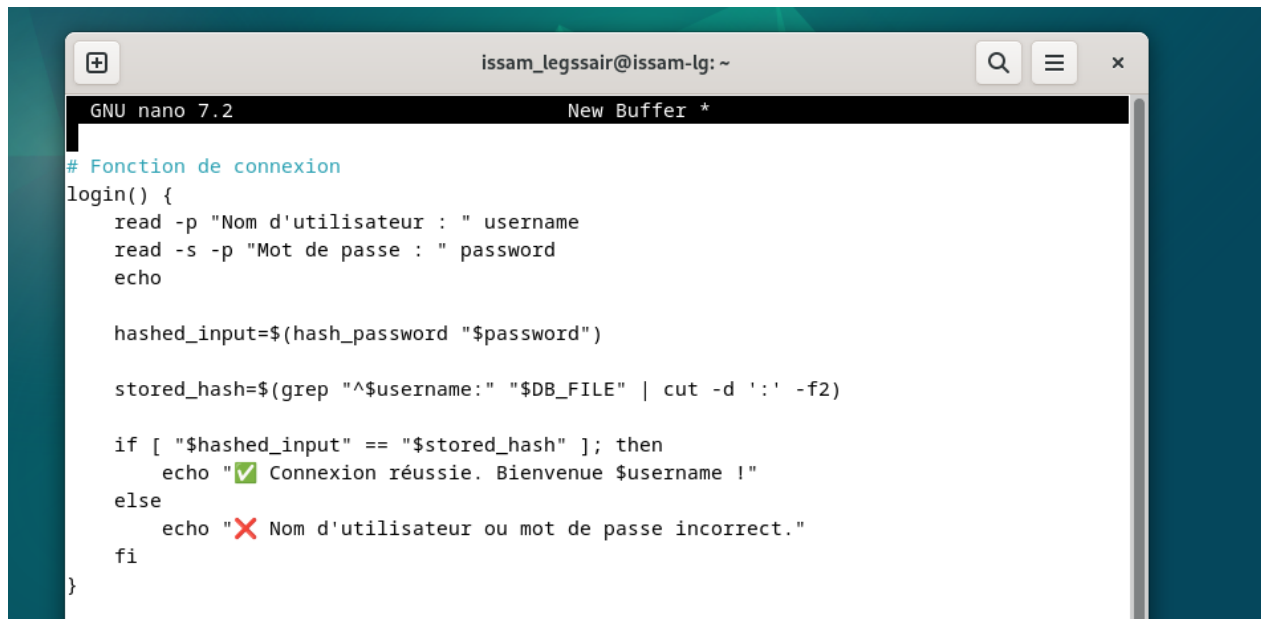


```
issam_legssair@issam-lg: ~  
issam_legssair@issam-lg: ~$ ls  
Desktop  login_system  minishell.c  Public      users.db  
Documents login_system.sh Music        Security_Manager.sh Videos  
Downloads login_sysytem Pictures     Templates  
issam_legssair@issam-lg: ~$ ./login_system  
----- Menu -----  
1. S'inscrire  
2. Se connecter  
3. Quitter  
Choix : 1  
Choisir un nom d'utilisateur : issam  
Choisir un mot de passe :  
Confirmer le mot de passe :  
✓ Utilisateur enregistré avec succès.  
----- Menu -----  
1. S'inscrire  
2. Se connecter  
3. Quitter  
Choix :
```

FIGURE 2 – Inscription réussie dans le terminal

4 Fonction login()

Cette fonction récupère le nom d'utilisateur et le mot de passe. Elle hache le mot de passe saisi et le compare avec celui stocké. En cas de succès, l'accès est accordé.



```
issam_legssair@issam-lg: ~
GNU nano 7.2 New Buffer *
# Fonction de connexion
login() {
    read -p "Nom d'utilisateur : " username
    read -s -p "Mot de passe : " password
    echo

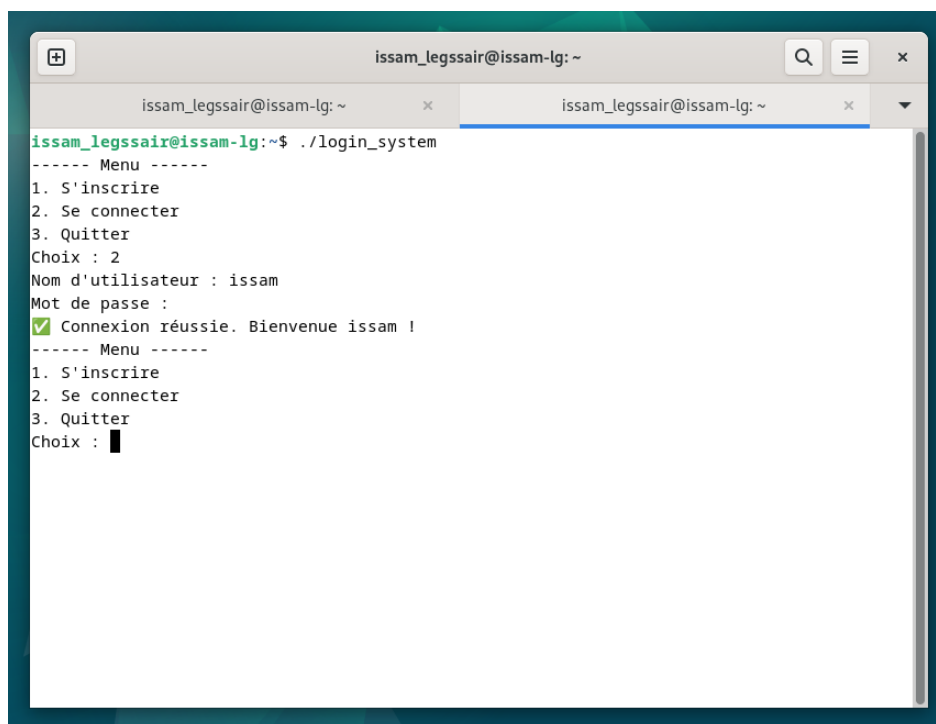
    hashed_input=$(hash_password "$password")

    stored_hash=$(grep "^$username:" "$DB_FILE" | cut -d ':' -f2)

    if [ "$hashed_input" == "$stored_hash" ]; then
        echo "✅ Connexion réussie. Bienvenue $username !"
    else
        echo "❌ Nom d'utilisateur ou mot de passe incorrect."
    fi
}
```

FIGURE 3 – Code source de la fonction login()

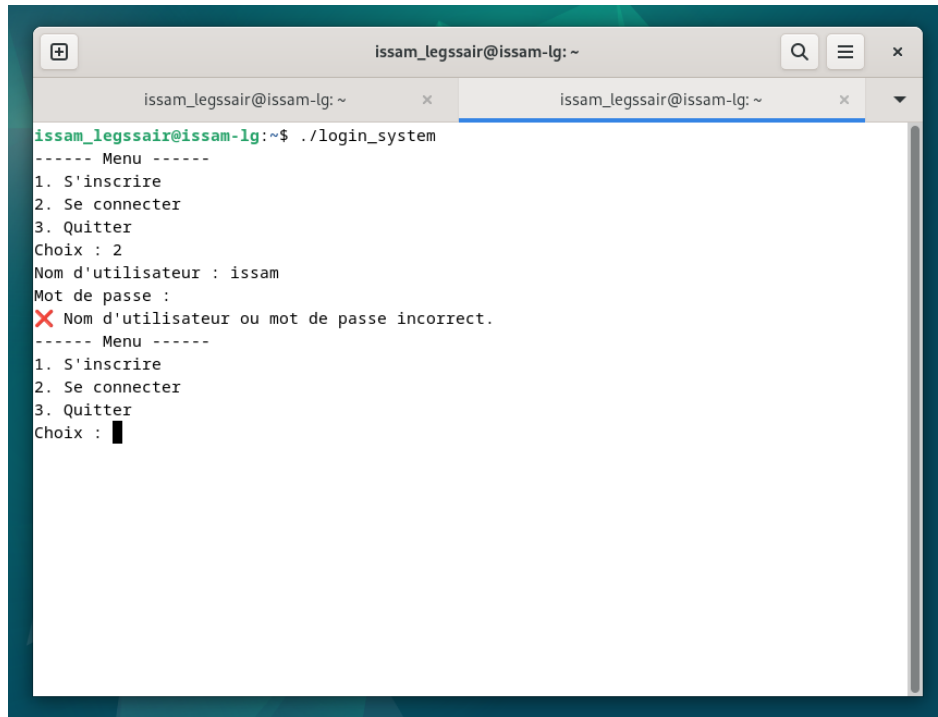
4.1 Connexion réussie



```
issam_legssair@issam-lg: ~
issam_legssair@issam-lg:~$ ./login_system
----- Menu -----
1. S'inscrire
2. Se connecter
3. Quitter
Choix : 2
Nom d'utilisateur : issam
Mot de passe :
✅ Connexion réussie. Bienvenue issam !
----- Menu -----
1. S'inscrire
2. Se connecter
3. Quitter
Choix : █
```

FIGURE 4 – Connexion validée – message de bienvenue

4.2 Connexion échouée

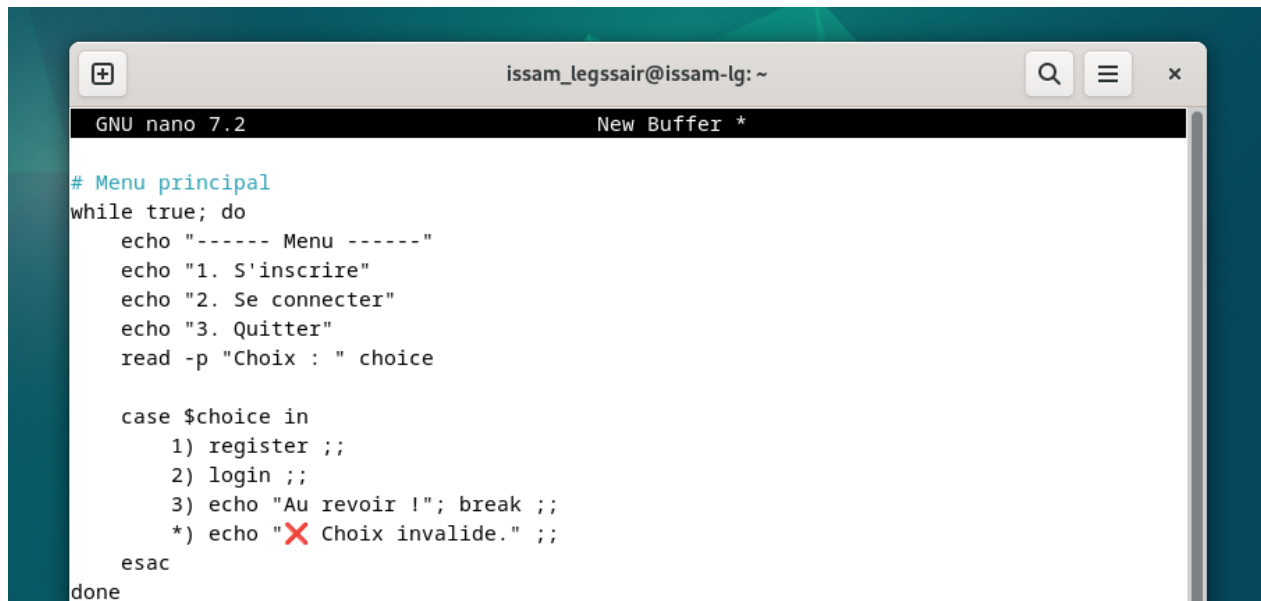


```
issam_legssair@issam-lg: ~  
----- Menu -----  
1. S'inscrire  
2. Se connecter  
3. Quitter  
Choix : 2  
Nom d'utilisateur : issam  
Mot de passe :  
✗ Nom d'utilisateur ou mot de passe incorrect.  
----- Menu -----  
1. S'inscrire  
2. Se connecter  
3. Quitter  
Choix : █
```

FIGURE 5 – Connexion échouée – mot de passe incorrect

5 Menu principal

Le script propose un menu interactif avec 3 choix : s'inscrire, se connecter, ou quitter. Le menu se répète jusqu'à un choix de sortie.

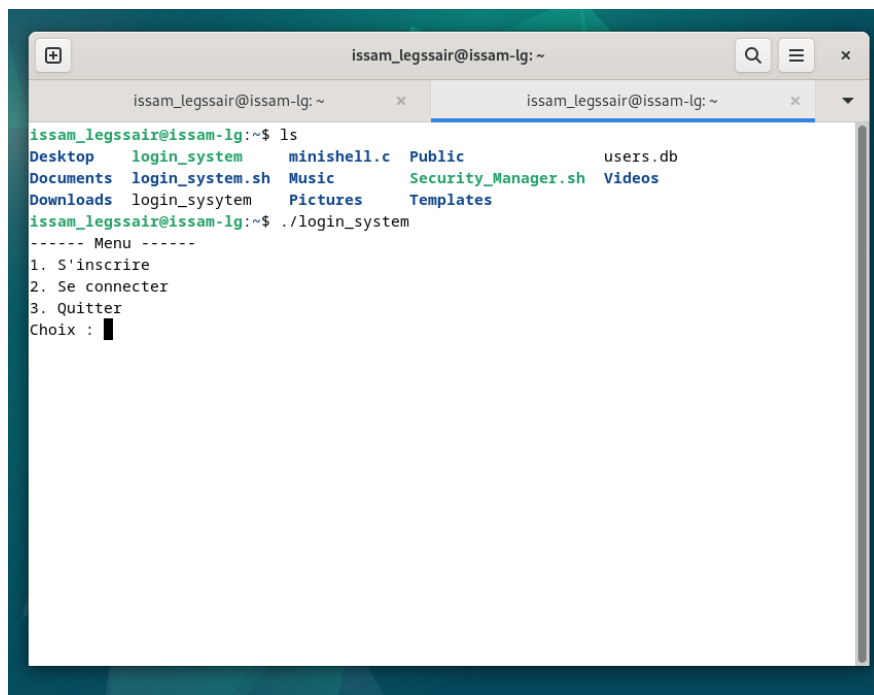
A screenshot of a terminal window with a dark background. The window title is 'issam_legssair@issam-lg: ~'. The editor is GNU nano 7.2. The code is as follows:

```
# Menu principal
while true; do
    echo "----- Menu -----"
    echo "1. S'inscrire"
    echo "2. Se connecter"
    echo "3. Quitter"
    read -p "Choix : " choice

    case $choice in
        1) register ;;
        2) login ;;
        3) echo "Au revoir !"; break ;;
        *) echo "❌ Choix invalide." ;;
    esac
done
```

FIGURE 6 – Code source du menu principal

5.1 Affichage du menu

A screenshot of a terminal window with a dark background. The window title is 'issam_legssair@issam-lg: ~'. The terminal shows the following commands and output:

```
issam_legssair@issam-lg:~$ ls
Desktop  login_system  minishell.c  Public  users.db
Documents login_system.sh Music  Security_Manager.sh Videos
Downloads login_sysytem Pictures  Templates
issam_legssair@issam-lg:~$ ./login_system
----- Menu -----
1. S'inscrire
2. Se connecter
3. Quitter
Choix : █
```

FIGURE 7 – Affichage du menu dans le terminal

6 Code source complet

```
#!/bin/bash
```



```
DB_FILE="users.db"
touch "$DB_FILE$"

# Fonction pour hacher un mot de passe
hash_password() {
    echo -n "$1" | sha256sum | awk '{print $1}'
}

# Fonction d'inscription
register() {
    read -p "Choisir un nom d'utilisateur : " username

    if grep -q "^$username:" "$DB_FILE"; then
        echo "    Ce nom d'utilisateur existe d j ."
        return
    fi

    read -s -p "Choisir un mot de passe : " password
    echo
    read -s -p "Confirmer le mot de passe : " password_confirm
    echo

    if [ "$password" != "$password_confirm" ]; then
        echo "    Les mots de passe ne correspondent pas."
        return
    fi

    hashed=$(hash_password "$password")
    echo "$username:$hashed" >> "$DB_FILE"
    echo "    Utilisateur enregistr avec succ s."
}

# Fonction de connexion
login() {
    read -p "Nom d'utilisateur : " username
    read -s -p "Mot de passe : " password
    echo

    hashed_input=$(hash_password "$password")

    stored_hash=$(grep "^$username:" "$DB_FILE" | cut -d ':' -f2)

    if [ "$hashed_input" == "$stored_hash" ]; then
        echo "    Connexion r ussie. Bienvenue $username !"
    else
        echo "    Nom d'utilisateur ou mot de passe incorrect."
    fi
}

# Menu principal
while true; do
    echo "----- Menu -----"
    echo "1. S'inscrire"
```

```
echo "2. Se connecter"
echo "3. Quitter"
read -p "Choix : " choice

case $choice in
    1) register ;;
    2) login ;;
    3) echo "Au revoir !"; break ;;
    *) echo "    Choix invalide." ;;
esac
done
```

7 Comprendre le hachage et SHA256

Un **hash** (ou hachage) est une transformation irréversible d'une donnée (comme un mot de passe) en une suite de caractères de longueur fixe. Cette opération produit une « signature numérique » unique.

7.1 Pourquoi utiliser un hash ?

- Pour ne jamais stocker les mots de passe en clair.
- Pour protéger les identifiants même si le fichier est volé.
- Pour vérifier une identité sans stocker le mot de passe réel.

7.2 SHA256

SHA256 est une fonction de hachage cryptographique qui produit un résultat de 256 bits, soit 64 caractères en hexadécimal. Elle fait partie de la famille SHA-2, largement utilisée dans la cybersécurité.

Exemple :

```
echo -n "motdepasse" | sha256sum
```

Cela retourne un hash unique, qui ne peut pas être inversé. Même un petit changement dans le mot de passe modifiera complètement le résultat.

Propriétés de SHA256 :

- **Irréversible** : on ne peut pas retrouver le mot de passe original.
- **Stable** : la même entrée donne toujours le même hash.
- **Longueur fixe** : toujours 64 caractères (en hexadécimal).

Limite actuelle : Le SHA256 seul (sans salt) peut être vulnérable aux attaques par dictionnaire ou rainbow tables. C'est pourquoi il est souvent recommandé d'ajouter un *sel* aléatoire à chaque mot de passe.

8 Compilation et exécution

8.1 Création du fichier

```
nano login_system.sh
```

8.2 Rendre exécutable

```
chmod +x login_system.sh
```

8.3 Exécution du script

```
./login_system.sh
```

9 Conclusion

Ce projet a permis de mettre en pratique les concepts fondamentaux de la programmation shell. Le système développé utilise des fonctions Bash, la lecture d'entrée, la gestion de fichiers, et le hachage SHA256.

Des améliorations possibles :

- Ajout d'un salt pour renforcer la sécurité.
- Blocage après plusieurs échecs.
- Interface plus conviviale.