

Report – Password Management system.

Report- Password Management System

Introduction: In the world of computer and web-applications, security & authentication has become a very crucial part of technology. This requires passwords for authentication. As the authentication has increased it has become very difficult to remember all the passwords and keeping the same password is vulnerable. So, storing the password somewhere user can access is becoming crucial. And this project is a Password Management System using Bash. This system will store various passwords of user.

What is it all about ...

- This is going to maintain the login IDs and password of the users , it create a separate directory for a user and sub-directory according to the requirement.
- User can check the list of data stored , in a tree structure.
- Update and remove can also be a part of the password management system.
- To get the data related to application ,user can get it on one click.
- Multiple users can use this system at the same time, system should be host on a client side . As multiple users will access this system it should encrypt contents while sending to server and should decrypt the contents while accessing the content of the files.

Project Description in details:

The basic architecture and the shells include in the project.

Architecture Design

----- Consists of seven scripts

- `init.sh` - Create user : To create the directory. It validates whether the user has inserted a parameter i.e. the name of the directory and then creates the directory, it also checks whether same is trying to create directory again.
Syntax : `./init.sh $user`
- `Insert.sh`- Create and update a service : To create the service and store login details.
Syntax : `./insert.sh $user $service $choice $payload`
- `rm.sh` - Remove a service : To remove a service.
Syntax : `./rm.sh $user $service`

Architecture Design -contd.

- `show.sh` - To show the login credentials.
syntax : `./show.sh $user $service`
- `ls.sh` - List the contents of the directory.
syntax : `./ls.sh $user $service`

The structure tree of the project directory:

```
cs19200438@csserver: ~  
cs19200438@csserver:~$ tree project  
project  
├── client.sh  
├── init.sh  
├── insert.sh  
├── ls.sh  
├── remove.sh  
├── server.sh  
└── show.sh  
  
0 directories, 7 files  
cs19200438@csserver:~$
```

1. ./init.sh

Basic commands used:

touch : to create the file init.sh

chmod u+x : to make it executable

mkdir : to create the directory for user.

```
cs19200438@csserver:~/project$ ./init.sh arshad  
User arshad created  
cs19200438@csserver:~/project$
```

2 ./insert

Basic commands used:

pwd : to get the current path.

Cd : for target movement

```
link=`pwd`
link_t="$link/$2"
#echo "$link_t is the link"
file_name="${link_t##*/}"
#echo "$file_name is the file"
directory_name="${link_t//$file_name/}"
#echo "$directory_name is the directory"
if [ "$3" = "i" ];then #to check its insert or update
    if [ -d "$directory_name" ];then
        cd "$directory_name"
        if [ -f "$file_name" ]; then
            echo "Error: Service already exist"
        else
            touch "$file_name"
            echo "$4" >> "$file_name"
            echo "Ok: Service Created"
        fi
    else
        mkdir -p "$directory_name"
        cd "$directory_name"
        touch "$file_name"
        user=`echo $4 | cut -d ":" -f 1`
        password=`echo $4 | cut -d ":" -f 2`
        echo -e "login: $user\npassword: $password" >> "$file_name"
        echo "Ok: Service Created"
    fi
fi
```

Different arguments and the response:

```
cs19200438@csserver:~/project$ ./insert.sh
Error:the number of parameters is wrong
cs19200438@csserver:~/project$ ./insert.sh arshad bank/ireland/aib.ie i "mylogin password"
Ok: Service Created
cs19200438@csserver:~/project$ ./insert.sh arshad bank/ireland/aib.ie i "mylogin password"
Error: Service already exist
cs19200438@csserver:~/project$ ./insert.sh xyz bank/ireland/aib.ie i "mylogin password"
Error: User Doesn't exist
cs19200438@csserver:~/project$
```

3../show.sh

Basic command:

cat :to display the content of the file.

```
if [ -d "$directory_name" ];then
    cd "$directory_name"
    if [ -f "$file_name" ]; then
        cat $file_name
    else
        echo "Error: Service does not exist"
    fi
else
    echo "Error: Service does not exist"
fi
```

Different arguments and the response captured:

```
cs19200438@cssserver:~/project$ cs19200438@cssserver:~/project$ ./show.sh xyz bank/ireland/aib.ie
Error: User Doesn't exist
cs19200438@cssserver:~/project$ ./show.sh arshad bank/ireland/aib.ie
login: mylogin password
password: mylogin password
cs19200438@cssserver:~/project$ ./show.sh arshad bank/ireland/aib.i
Error: Service does not exist
cs19200438@cssserver:~/project$
```

4../ls.sh

Basic command:

tree :to display the tree view of the directory

```
if [ $# -eq 2 ]; then
    if [ -d $1 ]; then
        cd "$1"
        if [ -d $2 ]; then
            tree $2
        else
            echo "Error : No such services available" >&2
            exit 3
        fi
    else
        echo "Error : No such services available" >&2
        exit 3
    fi
fi
```

Different arguments and the response captured:

```
cs19200438@csserver:~/project$ cs19200438@csserver:~/project$ ./ls.sh arshad bank
bank
├── ireland
│   └── aib.ie
└── 1 directory, 1 file
cs19200438@csserver:~/project$ ./ls.sh arshad
arshad
├── bank
│   ├── ireland
│   │   └── aib.ie
└── 2 directories, 1 file
cs19200438@csserver:~/project$
```

5../remove.sh

Basic command:

rm : to delete the file

rm -r : to delete the directory

rmdir: to delete the empty directory

```
file_name="${link_t##*/}"
directory_name="${link_t//$file_name/}"
if [ -d "$directory_name" ];then
    cd "$directory_name"
    if [ -f "$file_name" ]; then
        rm -f $file_name
        echo "Ok: Service Removed"
    else
        echo "Error: Service does not exist"
    fi
else
    echo "Error: Service does not exist"
fi
```

Different arguments and the response captured:

```
cs19200438@csserver:~/project$ cs19200438@csserver:~/project$ ./remove.sh arshad bank/ireland/aib.ie
Ok: Service Removed
cs19200438@csserver:~/project$ tree arshad
arshad
├── bank
│   └── ireland
2 directories, 0 files
cs19200438@csserver:~/project$ ./remove.sh arshad
Error:the number of parameters is wrong
cs19200438@csserver:~/project$
```

6../server.sh

How to access the above-mentioned services.

- All the service are registered to a server i.e. server.sh ,which is handling the request from the client. And using the above services desired response were echoed,the synchronization was happening using semaphores. While showing the login details of user the files should be decrypted.
- Client.sh is performing the platform between the user and server, the client side and server side should be synchronous with each other. Pipes are used to multiple user to access. Working on the FIFO alogorithm.

Basic command:

```
PIPE_SERVER="server.pipe"
#creating server pipe if not exists
if [ ! -e "$PIPE_SERVER" ]; then
    mkfifo "$PIPE_SERVER"
fi

trap "exit_interrupted" INT

function exit_interrupted()
{
    rm -f "$PIPE_SERVER"
    exit 0
}
```

```
read input < "$PIPE_SERVER"
echo "Input Received: $input"

clientId=`echo $input | cut -d " " -f 1`
request=`echo $input | cut -d " " -f 2`
```


7../client.sh

Basic command:

```
pipe_server="server.pipe"

if [ ! -e "$pipe_server" ]; then
    echo "Error: Server not started"
    exit 2
fi

trap "exit_interrupted" INT

function exit_interrupted()
{
    rm -f "$pipe_server"
    exit 0
}

client_id=$1
pipe_client="$client_id".pipe
option=$2

rm -f "$pipe_client"
mkfifo "$pipe_client"

case "$option" in
    init)
        if [ $# -ne 3 ]; then
            echo "Error: parameters problem"
            rm -f "$pipe_client"
            exit 3
        fi
        user_id="$3"
        echo "$client_id $option $user_id" > $pipe_server
        cat "$pipe_client"
        ;;
    insert)
        if [ $# -ne 4 ]; then
            echo "Error: parameters problem"
            rm -f "$pipe_client"
            exit 4
        fi
        echo "Please write login:"
        read login
        if [ -z "$login" ]; then
            echo "Error: login is not valid."
            rm -f "$pipe_client"
            exit 1
        fi
        echo "Please write password:"
        read password
        if [ -z "$password" ]; then
            echo "Error: login is not valid."
            rm -f "$pipe_client"
        fi
    esac
```

Sample screen shot from client.sh and server.sh running shell.

```
arshad@gmail.com
Please write password:
xyz@3
Ok: Service Created
cs19200438@csserver:~/project$ ./client.sh client1 rm john bank/ireland/aib.ie
Ok: Service Removed
cs19200438@csserver:~/project$ ./client.sh client2 update arshad bank/ireland/aib.ie "mylogin pas
Error: parameters problem
cs19200438@csserver:~/project$ ./client.sh client2 update arshad bank/ireland/aib.ie
Please write login:
john@#
Please write password:
@#45
Ok: Service Created
cs19200438@csserver:~/project$ ./client.sh client2 update arshad bank/ireland/aib.ie
Please write login:
johhu
Please write password:
565fygv
Ok: Service Updated
cs19200438@csserver:~/project$ ./client.sh client2 show arshad bank/ireland/aib.ie
johhu:565fygv
cs19200438@csserver:~/project$ ./client.sh client2 ls arshad
arshad
├── bank
│   └── ireland
│       └── aib.ie
2 directories, 1 file
cs19200438@csserver:~/project$ ./client.sh client2 insert arshad social/facebook
Please write login:
ars@
Please write password:
89u
Ok: Service Created
cs19200438@csserver:~/project$ ./client.sh client2 show arshad social/facebook
login: ars@
password: 89u
cs19200438@csserver:~/project$ ./client.sh client2 update arshad social/facebook
Please write login:
rruu
Please write password:
jhbhb
Ok: Service Updated
cs19200438@csserver:~/project$ ./client.sh client2 show arshad social/facebook
rruu:jhbhb
cs19200438@csserver:~/project$ ./client.sh client2 update arshad social/facebook
Please write login:
asdsad
Please write password:
asdsadsaefe
Ok: Service Updated
cs19200438@csserver:~/project$ ./client.sh client2 show arshad social/facebook
login: asdsad
password: asdsadsaefe
```

Conclusion:

Conclusion

- Password Management System stores login details of the user. This was a complicated project, but I have learnt how the Linux server works and the labs worked out the most helpful completing this assignment.
- It as nice experience doing the project in Linux from the scratch and this built confidence over bash.