Computer Systems and organisation Assignment 1 Report

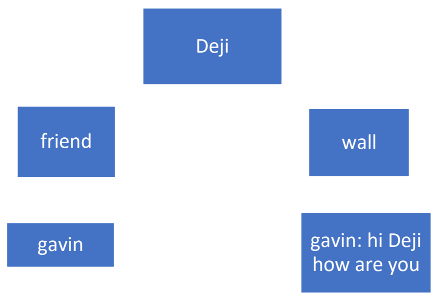
Introduction – BashBook

In this assignment we have implemented a Facebook-like social media server called BashBook in Bash. All users have their respective walls, users can become friends and write on each other's walls. This assignment helps us implement what we have learned from our lab sessions. In order to successfully complete this assignment we must have a strong understanding of how to make directories/files/scripts. We must know multiple commands and concepts such as spacing, first file line, loop structures and conditional expressions, bash variables, parameters and more. We worked in pairs to complete this assignment working through each step. First we have to create a user, we must check if there is only one parameter. We must use if loop to check if the user already exists else create user. Next we add a friend to the user. In order to do this we check if the user exists, friend exists and if the friend is already in the list of friends for that user. If any of them are false we exit. To post the message we create 3 parameters $sender, $receiver and message. We must check if all relevant parties exist (id, friend) and that there are 3 parameters. Then we send a message from sender to the wall of the receiver and we can display the walls using the script of dispay_wall.sh. There will also be a server to detail the commands and a readme file to describe how the project runs.

Organisation of the system

First we make a directory Assignment1 using mkdir, we then create a user (Deji) inside this directory (cd Assignment1) by using our create.sh script. Once we call the ./create.sh Deji, the file for the user Deji is created along with friends and wall files that go inside the Deji file. Inside the friends file is where we see the list of names added by add_friends.sh that are now friends with Deji, in this case you'll see 'gavin' inside Deji-friend. Inside the walls file you'll see the messages posted by the friends of Deji using the post_message.sh script. Any message sent by gavin will post onto the wall of deji-wall.txt. Each user contains friends and walls file inside their repositories. These files then let us see who is friends with who and what messages are being posted. Gavin must be a user in order to add him to Deji's friend

File structure:



List of Implemented features

Important features:

Structure of a Bash Script: #!/bin/bash gives the kernel the path to the command processor
Conditional expressions structure:

```
if cond1; then
    instructions
elif cond2; then
    instructions
else instructions
fi
```

create.sh:

Firstly Gavin made a create.sh inside our main directory file using the terminal**. Cd** assignment1 – **touch** create.sh. our create.sh file uses conditional statements to check all relevant fields such as parameters and if the user already exists. We use echo to print our relevant messages to the terminal to indicate the output. If the user already exists ("user already exists") or the parameter is not equal to 1("no identifier provided") we **exit 1** and print the relevant strings otherwise we create the user $id and create a friend file and a wall file inside the user directory. This will print "ok: user created". This is equivalent to someone making a new account on their social media platform.

Add_friend.sh:

Deji made the add_friend.sh file which is used to add friends to a user's friend file. We first have to check 1. If user exists. 2. If friend exists 3. If the friend is in the friend list of the user already (if grep "^$2" "$1"/friend > /dev/null;). We do this by using if statements. Exit 0 = true and exit 1 = false. If all are true we add friend and print "ok: friend added". End the if statements with "fi".

Post_message.sh:

Both team members worked on post_message.sh together. First we must use our knowledge of parameters. We let sender=$1, receiver=$2 and message=$3 (don't use spaces). We check if the parameters = 3, that the sender and receiver exist in the depositories and if the sender is inside the friends list of the receiver by using if grep. Grep searches the given file for lines containing a match to a given command. If we pass through these checks we print "ok: message posted!" otherwise print the failed messages.

Display_wall.sh

Gavin created the display_walls.sh where we have to check if the user exist so we can print the messages inside the file using the cat keyword (cat "$1"/wall.txt) otherwise the user must not exist, print "nok: user $1 does not exist."

Server.sh

In our server script we create an infinite loop with parameters which details the various commands in order to create, add, post and display in BashBook. In here Deji made if and elif statements indicating how a user would use the server. This is to make it clear and easy to use.

Challenges faced and solutions

We faced many problems during this assignment, which we managed to overcome. The first problem we faced is that we lacked the general knowledge of bash and of the commands to start off the assignment properly. We had to go through the notes and complete all the labs to fully have a grasp knowledge of what was needed for the assignment. This took us a quite a while to get all done and took off time that we could've used to complete the assignment.

The second problem we faced was procrastination we left the assignment for too long before starting it because we had many assignments to do at first and we thought we'd still have a lot of time when we finally started the assignment. This was foolish of us as it made us speed through the assignment which caused us to make a lot of mistakes along the way.

The biggest problem we had was with syntax. Since this was our first time properly doing bash script we would make silly mistakes like not putting "then" after "if" statements, not closing the if statements with "fi" or putting quotation marks on the "nok" statements even though they weren't needed. To notice these errors that we were making we had to learn how to read and understand the error list when they would show up in the terminals. This allowed us to be able to go to the exact line of the error and change what was needed to make the correct the code.

We also faced some problems with properly assigning roles with each other for the assignment. We split the coding up evenly, but I can't say it was totally even as if one of us needed help to get their part of the code working the other would take over and if worse came to worst, we'd ask the tutor to help us understand our problem with the code.

Conclusion

This assignment we worked as a team testing our skills in bash scripting. We used the terminal to create directories/files and scripts, to do this we must have strong knowledge of the appropriate commands i.e **touch create.sh** made the script, we made it executable with **chmod u+x create.sh,** and we can run it with **./create.sh**. In our text editors we made short programs that created the functionalities of our BashBook. Our BashBook can create users, add friends, post messages to friends walls and display the walls. We had to made sure we had appropriate checks within our scripts to prevent duplication and allowing for a clean program i.e we don't want to duplicate friends so that "Gavin" is displayed multiple times in Deji's friend file. Each user has a simple structure, inside each user they have a friend file where the user's friends are displayed and a wall file where friends can post messages. We also created a server.sh script which details the various command needed to run your BashBook and carry out the important functionalities. In conclusion, this assignment tested our critical thinking and problem solving, constantly testing and running our scripts, fixing any issues or errors that occurred as a team.