# Assignment 2 - Computing Systems

## Short introduction

Hello again, my name is Dawid Szczesny  (21300293), and I worked on this assignment with Brian Moyles again.

BashBook version 2 is finally out. It like v1 is a terminal based, Facebook-like, client-run application where friends and family can add each other and post on each other's walls, all from just their shared computer. But now with they all have their own profiles and everything works with a locally run server.

## Organizational Structure

Our organizational structure in this assignment was the exact same as the previous assignment (partnership) as it worked the best for us. We both got together and worked on the pseudo code, in order to plan out our new approach. We then divided up the work and contacted each other whenever we got stuck.

In terms of the organizational structure of the program, we divided it into many little scripts. server.sh is still the main script, however, in this version we implemented a client side script as well and two extra scripts with that also.

## Implementations

### Use of less over cat

We use less in display.sh over cat, as when using cat it returned a single

## Making pipes

Firstly, we needed to connect the server side to the client side. We did this using named pipes in bash

```
mkfifo <name>.pipe
```

We had a main pipe in the server called server.pipe, which was used to let users send commands to the server.

## Using pipes with client

To implement the client side we needed to reliably be able to receive and send information from and to the server. We ended up using two separate scripts called receiver.sh and sender.sh respectively. We then ran the two scripts simultaneously from client.sh.

→ implemented by Dawid

## Managing several clients

As the server knew there was the possibility of many users being connected at once, but not knowing how many or where, we needed a way to send confidential information directly to the user rather than to ever client.

This was achieved by creating separate client pipes that the sever used to send back the information. The server acquired the correct client pipe from whichever client made the request to the server. We made it a default argument that the clients id was passed so that the server could identify who is contacting it.

→ implemented by Brian

## Clean up

With our implementation and using named pipes, after the users left the server and even after the server was closed, there was a lot of garbage left over after the program was finished. The most common way uses left the server and the server was closed was by using the control c command to interrupt the terminal process. Knowing this we trapped the control c command and used it to run a cleanup function

```
ctrl_c(){
  # clean up code
}
trap ctrl_c INT
```

→ implemented by Dawid

## Difficulties

- Connecting the server to the clients so that it would simultaneously read and send data from the user to the sever and vice versa. This was a major problem that we couldn’t figure out for a while. We fixed it by using the receiver and sender sh files to simultaneously send and receive the information

## Synchronization

The server allows users to send and receive commands simultaneously to and from the server. With testing we experienced no synchronization issues, hence, we did not need to implement any lock mechanisms in the design.

## Conclusion

From this assignment I certainly developed my skills in script writing in bash. Not only that but also the importance of planning a project. The importance of making scalable code, to make it easy to build upon and implement more and more features.

In Bashbook version 1, the code was scalable and made implementing (once we figured out the syntax) relatively easy.