CT213 University of Galway
Computing Systems & Organisation 2022/2023

*Lab Project*

# BashBook Part 2: Enabling Connection of Multiple Users

Assignment 1. Version v1.1

## 1 Project Description

In this Part 2 of the *BashBook* project, you will have to expand your work from Part 1 (either your own code or the one provided by the Lecturer) by enabling multiple users to connect to the server and run commands at the same time (thus, there might be **synchronisation** issues).

To do that, you will have to create a script that each client could run independently on their terminal (call it `client.sh`). Each client script client will interact with the server (`server.sh`) using **named pipes**.

## 2 Client Script

First, you will need to create the last component of your system (i.e., the client application), which will send requests to the server. You will write a script `client.sh id`, which gets only one parameter `$id` representing a user.

First your script has to check if a parameter has been given and if yes it enters an endless loop. Every time it enters the loop, the script reads a request from the user(requests have the form `req args`). If the request is well formed, then the script sends the request `req id args` to the server with `$id` being the `$id` given as parameter of the script `client.sh` (this avoids that the user types their `$id` every time). The client has to then read the server's response and show it in a user friendly way on the terminal, e.g.,

- if the server replies `ok: user created!`
  then the client should print `SUCCESS: user created!`
- if the server replies `nok: user already exists`
  then the client should print `ERROR: user already exists`
- if the server replies with the display of a user's wall
  then the client should only print the content of the wall and not print the lines `start_of_file` and `end_of_the_file`

## 3 Connecting Server and Clients

Since the server does not know how many clients there are (not where they are), all clients should write their requests into a unique named pipe and the server should keep reading from that same pipe. However, as we do not want the clients to know what the server replies to other clients, the server should reply to each client on a dedicated named pipe.

To achieve that, the server will create a named pipe called `server.pipe` and each client will create a named pipe called `$id.pipe` with `$id` the id given as parameter of the `client.sh` script. Modify the scripts `server.sh` and `client.sh` with named pipes.

Figure 1 shows an overview of client-server communication architecture using named pipes.

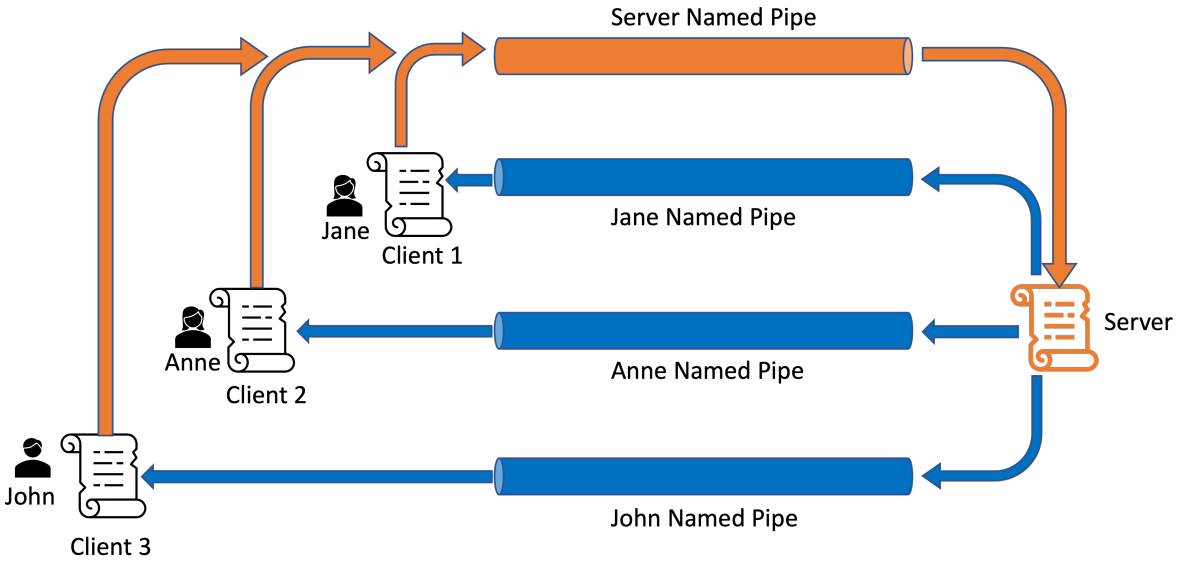


Figure 1: Overview of BashBook communication architecture

### 3.1 Extra Challenge (not mandatory but will get you extra points)

It is likely that so far you have been using control + c to stop the scripts (and many users are likely to do just that). If a user closes their terminal, the user pipe their client created might never be deleted. Search how you could delete the named pipes whenever a user quits a script (i.e., trap the command control + c). Use the following idea in your programs to do so: trap control + c and then delete the named pipe and then exit with the exit code 0

## 4 Handling Synchronisation Issues

The server should enable each user to run any command in the server at any time. Therefore, you will need to manage synchronisation issues that might arise.

You can use locks to achieve this (remember Lab 5?). However, you will need to decide how to name the locks and when/where to acquire/release each of them. There are many ways to implement this, but with varying effects.

## 5 General Instructions

- This assignment is to be completed in pairs (the same as in Part 1).
- You are encouraged to collaborate with your peers on this project, but all written work must be your own. In particular we expect you to be able to explain every aspect of your solution if asked.
- **Both pair members must submit an archive** (zip or tar.gz) of your solution on Blackboard:
  1. Code/scripts
  2. README.txt file describing how to run your programs
  3. PDF report of your work (5 pages max, no need to include code in it).
- The report should include the following sections:
  1. A short introduction
  2. A section that describes the organisation of your system.
  3. A list of implemented features highlighting the contribution of each group member.

4. Description of the locking strategy for each request with comments on performance and correctness: what is the name of used lock?, why did you choose that name?, why did you use it at that level?. If you did not implement the locking mechanism, you could describe how you would implement it in words (it might get you some points).
5. A description the different challenges you faced and your solutions.
6. A short conclusion

- Due Date: November $25^{th}$ 2022, at 5pm. 10 Marks penalty for $<$1 week late submissions. 20 Marks penalty for $<$2 weeks late submissions. No submissions will be accepted after 2 weeks.
- Both members of the group must submit their project.