**Exercise 3 – MtaCoin with mqueues**

**Objective:**

Practice multi process communication using POSIX message queues.

Transform the already written application from assignment #2 into message based multi process application.

Same server and miner logic should be used.

Get rid of the mutexes and condition variables and use POSIX message queues instead to exchange data between the miners and the server.

The code should be divided into at least miner.c, server.c, include.h(shared definitions and structures) and launcher.c(practice fork() a little bit, fork server process and X miner processes, see example below).

**Note:** server.out and miner.out should also be possible to be executed independently without the launcher.

**Algorithm:**

On execution server should wait for connections on a hard-coded queue name known both to server and miner.

On execution the miner shall send a “connection request” to the server with his ID and his queue name.

Server should send the newly added miner the current block and miner should start mining the next block.

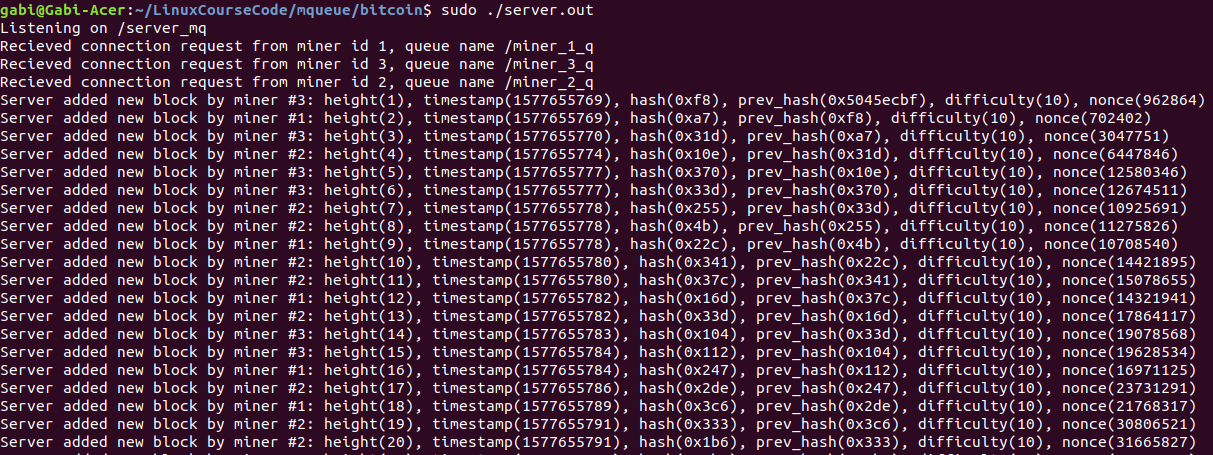
Upon successful work miner shall send the block to the server.

The server verifies the block and if block is approved, he sends the new block to all the miners.

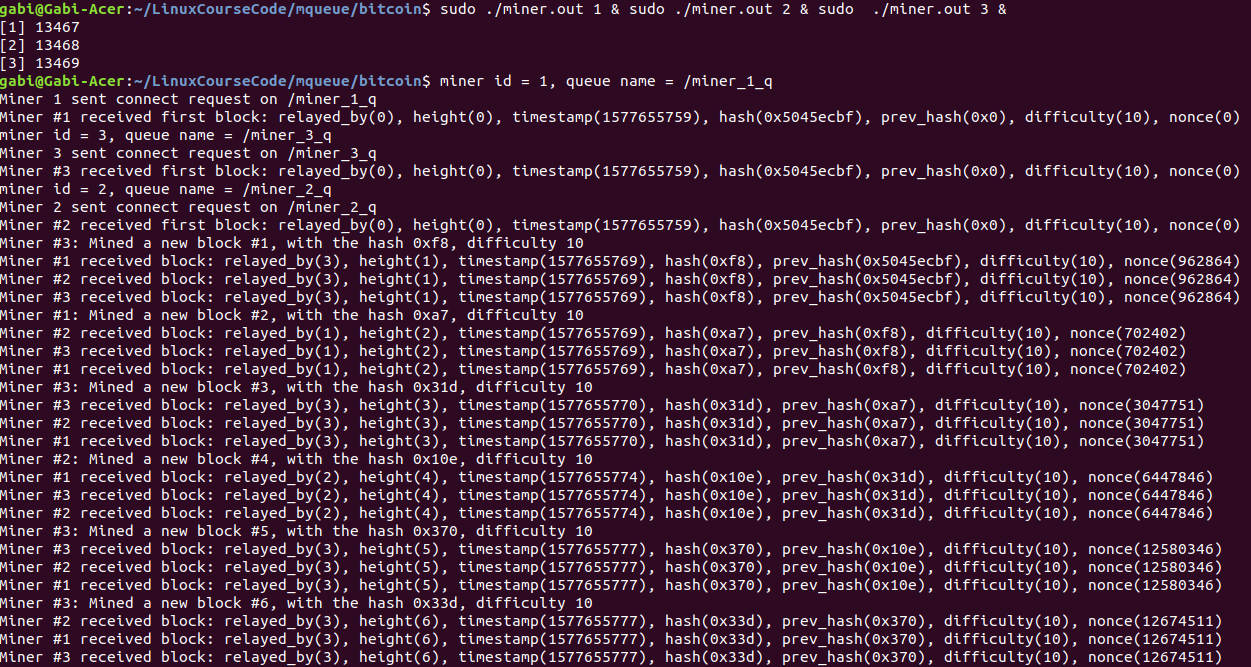
Miner shall stop mining based on the old block and move to mine based on the new block.

\* It should be possible to launch and kill some miners dynamically while others are working

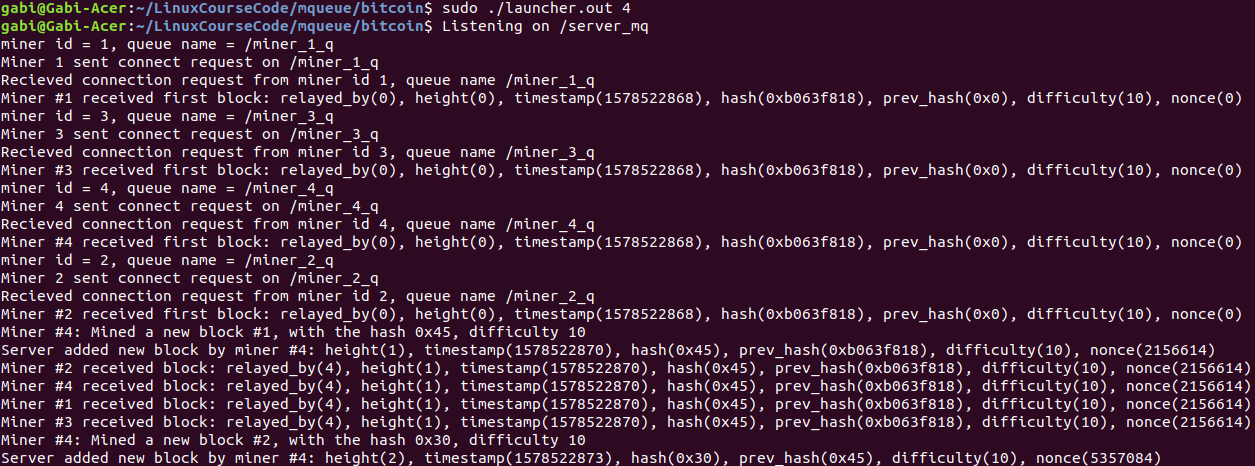
**Server:**



**Miners:**



**Launcher:**



**Bonus (10 pts):** load and run the above applications to virtual embedded device

Create a new Yocto layer called meta-mta.

Create a new image called core-image-base-mtacoin

Add a recipe that compiles and loads(installs) the executables to the image file system at /usr/bin/:



Build to beaglebone-yocto machine.

Only meta-mta shall be submitted.