Interprocess communication

Ahmed Ahres (0978238)
Maciej Kufel (0944597)

The inter-process communication was achieved using *farmer.c, worker.c* and *common.h*. The farmer contains the initialization of the request and response queue as well as the request messages and response messages which are data structures defined in *common.h*.

The request message is a struct containing the hash of type *uint128_t* as well as a starting character of type *char*, while the response message contains the hash of type *uint128_t* and a character array of length MAX_MESSAGE_LENGTH containing the result string corresponding to the hash. After opening and renaming the message queues, the next step for the farmer is to create *NROF_WORKERS* child processes using the method *fork()*. A separate function *create_children()* was implemented for this, which calls *fork()* *NROF_WORKERS* times. Since *fork()* returns 0 when the process is a child process, *worker* is executed each time 0 is returned using the function *execlp()*.

The worker (child) first starts by receiving the request message sent by the farmer (parent) using *mq_receive()*. Then after waiting for some time (*rsleep(10000)*), it generates all possible strings up to and including a length of 6 characters and hashes each of the strings using *md5s*. If one of the hashed values corresponds to the request message which is passed as a parameter in the function generating all the strings, then it is sent as a response message to the parent using *mq_send()*. If none of the hashed values corresponds to the request message, then the message is skipped and the next one starts being received. The parent then receives the correct string. This communication is done in a loop such that the jobs are sent until all of them are sent. Finally the message queues are cleaned and closed using *mq_close* and *mq_unlink*.

The response message data structure contains the received hash in order to return not only the string corresponding to the hash but also the md5 hash itself in order to connect the result to the corresponding hash.
When a request message is pushed to the request queue, the first free child grabs the request and does the worker job. The worker will generate all possible strings of a length up until 6 that starts with the starting character in the request message corresponding to the hashed value. If there are more requests in the request queue than there are children, then the pending jobs are grabbed by a children that have finished its current job.

The algorithm for generating strings was inspired from the internet, with some changes in order to hash the returned string and compare it with the received hash value:
https://stackoverflow.com/questions/4764608/generate-all-strings-under-length-n-in-c