Sistemas de Operação

(Ano letivo de 2018/2019)

Guiões das aulas práticas

Quiz #IPC/03 Message queues

Summary

Understanding and dealing with concurrency using message queues.
Implementing communication between processes using message queues.

Question 1 *Implementing a producer-consumer application, using a shared FIFO and semaphores.*

(a) Directory `prodcon` *provides an example of a simple producer-consumer application. The application relies on a message queue, used to implement the communication channel between producers and consumers. Each item of information is composed of a pair of integer values, one representing the id of the producer and the other a value produced.*

(b) The given solution is based on System V message queues. The system calls of interest are `msgget`, `msgsnd`, `msgrcv`, *and* `msgctl`. *A message is composed of a field of type* `long`, *representing the message type, followed by 0 or more data bytes.*

(c) Generate the System V version (`make prodcon_msg`*), execute it and analyse the results.*

(d) Look at the code, `queue_msg`, *and analyse it.*

- *Try to understand how message queues are used. Pay close attention to the send and receive functions.*
- *Comparing this code with the implementations in the previous classes, try to understand why functions queueConnect and queueDisconnect are empty.*

(e) Reimplement the `prodcon` *application using POSIX message queues. (See* `man mq_overview`*).*

Question 2 *Designing and implementing a simple client-server application*

(a) Consider a simple client-server system, with a single server and two or more clients. The server consumes requests and produces responses to the requests. The clients produce requests and consume the corresponding responses.

This is a double producer-consumer system, requiring three types of synchronization points:

- *the server must block while there are no requests pending;*
- *A client must block while there is no space to put the request;*
- *A client must block while the response to its request is not in the available.*

A solution to this system can be implemented using two System V message queues, where the type field of the message is used for synchronization purposes. One queue is used for communication from client to server (requests) and another for communication from server to client (responses). Interaction is defined as follows:

(a) The client creates a message, putting its `pid` *in the type field, and sends it to the request queue.*

(b) The server retrieves messages of type 0 (any type) from the request queue, processes it, creates a response with the same type as the request, and puts it in the response queue.

(c) The client retrieves a message with a type equal to its `pid`*.*

Finally, consider that the purpose of the server is to convert a sentence (string) to upper case.

(b) Using the implementation, used in the previous exercice, as a guide line, design and implement a solution to the data structure and its manipulation functions. Consider the following possible functions, where the handle could be the process `pid`*:*

```
HANDLE sendRequest(ITEM);
ITEM getResponse(HANDLE);

(HANDLE, ITEM) getRequest();
sendResponse(HANDLE, ITEM);
```

(c) Think on an implementation using POSIX message queues.
