

CS 471 – Operating Systems – Fall 2004
Department of Computer Science
George Mason University

Individual Project – Prof. Daniel A. Menascé

This project will require you to be able to create processes in UNIX as well as to use UNIX message passing (see sample codes in the course home page). In this project, you will implement a simple message server that can be used by client processes to exchange messages. Client processes are identified by names, strings of at most 10 characters. The message server (MS) and each client are separate UNIX processes.

The message server provides the following services to clients:

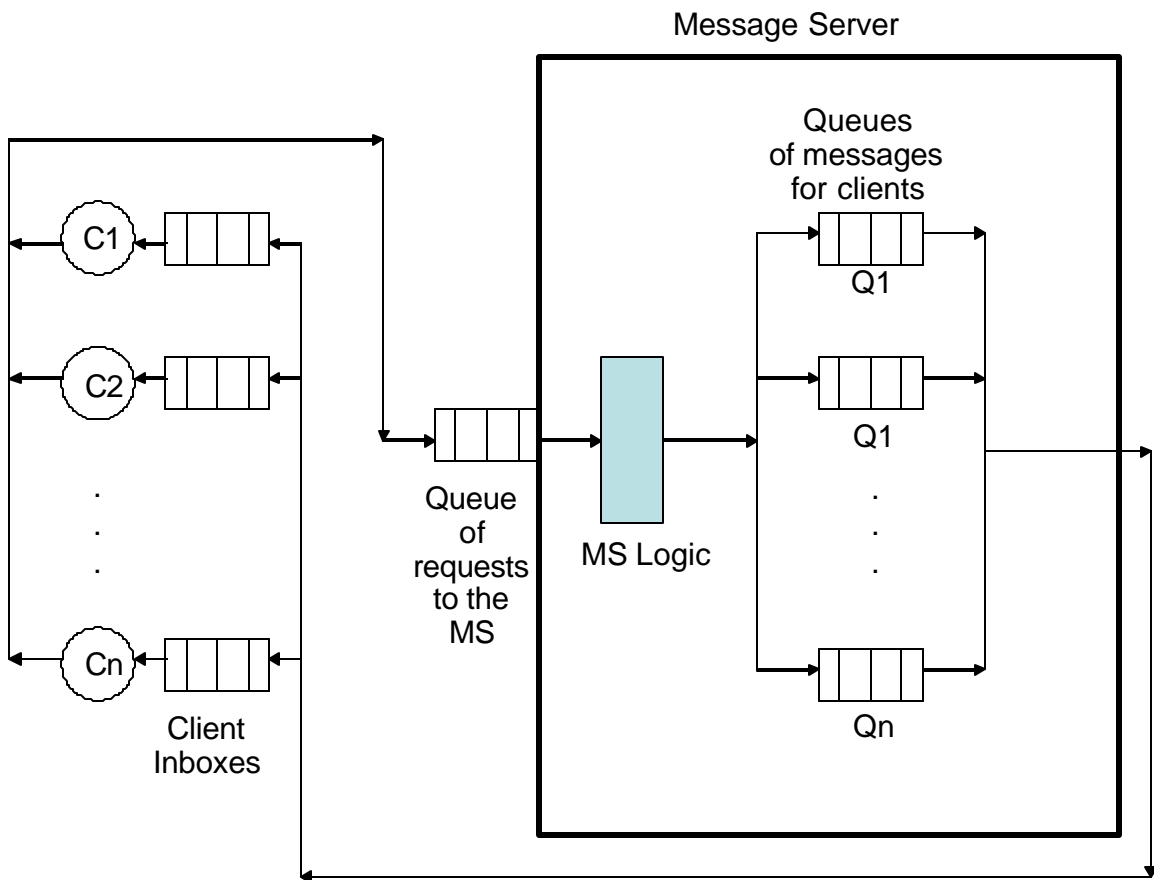
- RegisterClient (ClientName): used by a client to request the MS to register the client with a name ClientName. This MS sends a return code to the client (in a message), which can be:
 - 0: success
 - 1: failure; client with that name already exists
 - 2: failure; server could not create a mailbox
- SendMessage (DestinationClient, Message): sends message Message to the client named DestinationClient. If DestinationClient does not exist, the MS sends the sending client a message indicating that the destination does not exist.
- ReceiveMessages (ClientName): receives all messages in the inbox for client ClientName.

The structure of messages and processes is illustrated in the figure below (see next page). There is a queue for each client process where the messages for the process (inbox) are stored. There is a common queue where all clients send requests to the MS. The MS logic takes one request at a time from its queue of requests. If the request is to send a message, the MS determines if the recipient exists. In the affirmative case, the MS places the message in the message queue "inside the MS" corresponding to the destination client. If the request is for receiving messages, the MS sends all messages in that client's MS queue "inside the MS" to the client's inbox queue.

The MS process stays in a loop in which it takes a request from its queue and processes it. Client processes have to register first. Then, they will send and receive a large number (100) of messages to other processes.

For the purpose of this project, use 3 clients only. Messages can be fixed-size strings. The content does not matter. But, for testing purposes it would be useful

to sends messages such as "Message no. xx from client y to client z.", where xx is a sequence for messages generated by client y.



The pseudo-code for a client k is:

```

RegisterClient (k);
MessageNo = 0;
Loop 100 times
    X= Randomly selected destination client;
    MessageNo = MessageNo + 1;
    SendMessage (X, "Message " + String(MessageNo) +
        "from client k to client " + String (X))
    At each 10 messages sent ReceiveMessages (k);
End Loop
    
```

Deliverables: Report with the following sections:

1. Introduction
2. Problem Description

3. Pseudo code for each process (MS and client (more detailed than the one I described)
 4. Description of major data structures
 5. Description of test results (make sure to indicate how you tested your implementation)
 6. Concluding remarks (include lessons learned)
- Appendix A: Source code
Appendix B: Sample runs

Your code should be available in case it is needed for grading purposes. Make sure to document your code very well. Your report must be well-organized and well-written. Use spell checkers and grammar checkers.