Lab 4 Design Document

Group 2
Aaron Meurer
Oran Wallace
Sheng Lundquist

Introduction

Interprocess communication is essential for a workstation that supports multiple processes. The operating system must provide a tool for processes or managers to communicate with each other. This communication can be used for synchronization, multi-threaded programing, and information sharing.

The IPC

Each manager will be essentially a queue, using our queue manager. Each message queue will be length five. This small size will make things easier to test. Also, it should be a reasonable number in a system where we can only have up to 20 processes. It will be easy to change in the future on a per-manager basis if we decide it is too large or too small.

We will have to modify our queue manager to handle messages, rather than just process_control_block. This will be done by copying the queuemanager and modifying it to use message structs instead of process control block structs.

The message will be a struct, making it easier to change the structure of the message in the future. For now, it will contain a variable noting which manager was the source manager, and a char array of size 256 for the message, as well as the necessary variables to be in a queue (prev and next).

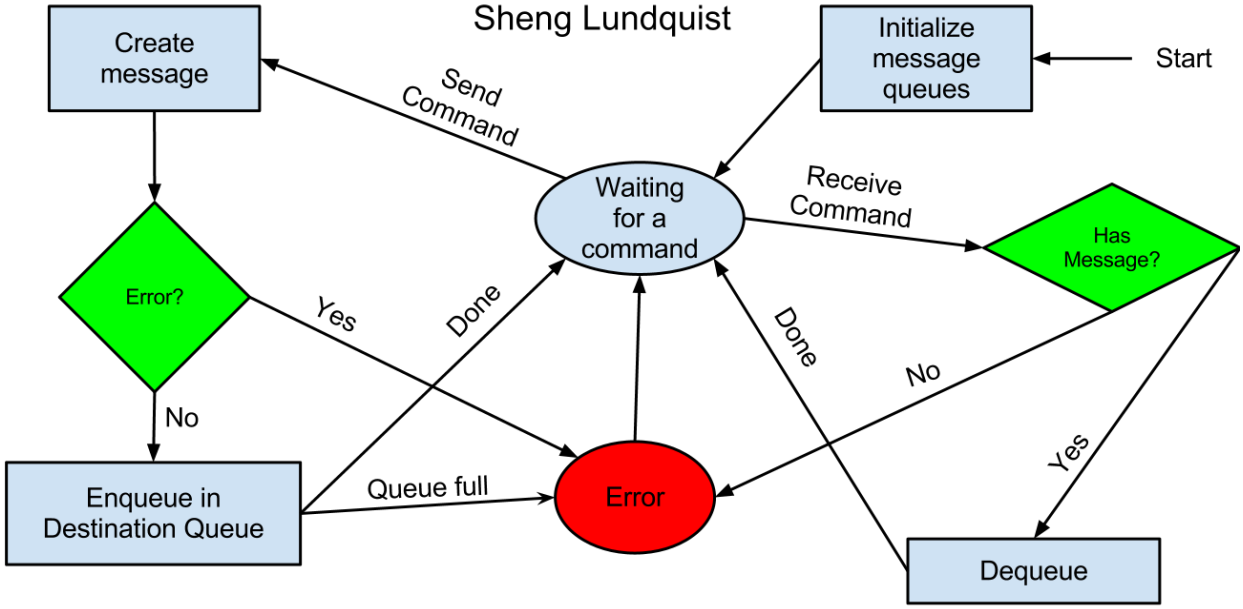
Each message queue will be referenced by an integer from 0 to 9. This may later be adapted to use an enum once we know what exactly they are, which would be exactly the same anyway. This method of reference will be used both internally and in the commands to the test manager.

Flow Charts/State Diagrams

IPC Message Flow Chart

IPC Flow Chart

Group 2
Aaron Meurer
Oran Wallace
Sheng Lundquist



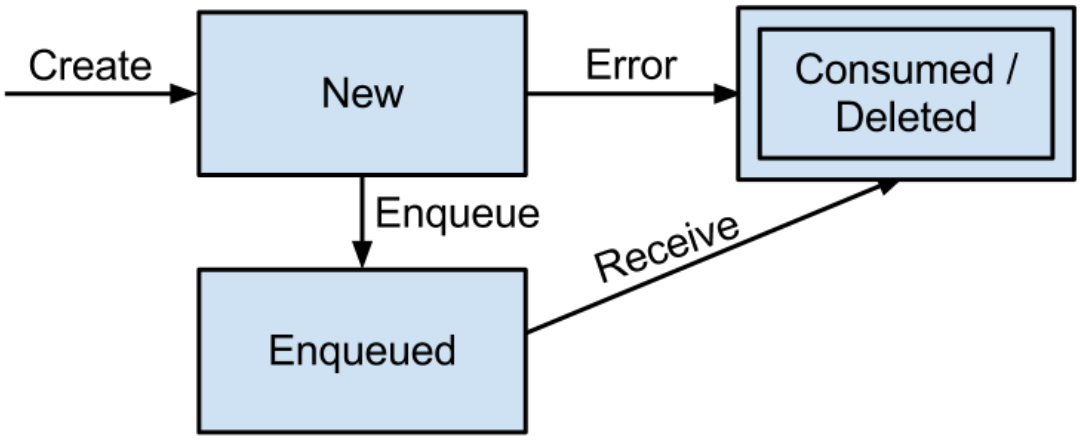
This flowchart describes the flow of events a message will experience in our IPC Manager. A send command will signify when a message is to be created. During the message creation if any errors are encountered (such as a malformed message), the manager will discard the message and return the error accordingly. If no errors arise during the message creation process the message is enqueued into the corresponding destination queue. Upon a receive command being issued for the destination queue, the message will be dequeued from that queue. If there are no messages in the queue, that is an error. In the test manager, the message will be printed to the terminal.

IPC State Diagram

IPC Message State Diagram

Group 2

Aaron Meurer
Oran Wallace
Sheng Lundquist



This state diagram describes the states and transition that are related to a message within our IPC Manager. There are three possible states that a message can be in, “New,” “Enqueued,” and “Consumed/Deleted.”

When a message is first created, it is in the “New” state. This means that the IPC manager is in the process of allocating space for the message and checking it for errors. If there are any errors, it is deleted, so it goes to the “Consumed/Deleted” state. Possible errors could be malformed message, bad source or destination, or if the IPC queue that the message is to go in is full.

If there are no errors, it is enqueued in the proper queue. It remains here until it reaches the top of the queue and the destination receives it. When this happens, it is consumed, and the space used by the message is reallocated. As far as the IPC manager is concerned, consumed and terminated are equivalent, because in both cases, the memory that was used for the message is reallocated, and the IPC manager is no longer aware of the message itself. Hence, they are the same state.

In other words, when a message is received, we consider this to be the end of the message’s life. If the same message is sent again somewhere else, we consider that to be the creation of a new (identical) message.

In the “New” state, little can be asserted about the message, other than that memory has

been allocated for it (we do not enter the diagram at all if there is no memory available). In the “Enqueued” state, it can be asserted that a message has no errors, including any errors about source and destination. Furthermore, the message is in a queue. In the “Consumed/ Deleted” state, we can assert that the memory for a message is free again, and it cannot be used further.

Test Cases

The following things are tested in the test interface for each command.

- INIT_IPC
  Check is init_ipc command works correctly. Check if multiple message queues can be initialized. Check that message queue cannot be created for an invalid manager.
- SEND
  Check if send command works correctly. Check that a manager cannot send to itself. Check that a message can’t be sent to a manager either uninitialized or invalid. Check that a message cannot be sent to a full queue.
- RETRIEVE
  Check that retrieve command works correctly. Check that a message cannot be dequeued from a empty/uninitialized queue.
- HAS_MESSAGE
  Check that has_message command works correctly. Check that command cannot be performed on a uninitialized/invalid queue.
- LIST
  Check if list command works correctly. Check if list command displays uninitialized queues along with initialized queues.

How to run the program: Use the HELP command to view each command the the arguments it takes. Use the Makefile to compile everything, following this use the ipc_test.o to actually run the test interface. If a file name is used for an argument with ipc_test it will run that test file(ipc_tests.txt) , otherwise it will take command line arguments.