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.

Design Decisions

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

This flow chart 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

This state diagram describes the states and transition that are related to a message within our IPC Manager. Following the state state, an epsilon transition is made into the “Creation” state. In this state a message is simply created and if any errors arise in the creation process (for example: invalid destination or malformed message) an error will be produced and the message will be discarded.

From the “Creation” state, a message will transition to the “Sending” state. In this state a simple check is made to see if the destination queue is full. If it is an error is thrown and the message is discarded, if not the message transitions to the next state.

Following the “Sending” state a message is enqueued to its corresponding destination queue which will transition to the “Waiting in message queue” state. If a message has transitioned to this state it is implied that destination queue was not full. “Waiting” means that it is waiting to get to the top of the queue and be dequeued.

Finally, when a message reaches the top of the destination queue and a receive command is issued, it moves to the “Received” state. 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.