CECS 326-01 Operating Systems

William Luu

Assignment 2

Due Date: 10/15/2024

Submission Date: 10/15/2024

Program Description

The three programs—master.cpp, sender.cpp, and receiver.cpp—work together to implement a message-passing system using System V message queues in Linux. The master program first creates a message queue and spawns one sender process and multiple receiver processes based on a user-specified number. The sender collects user input, where the user enters a message and specifies which receiver should receive it, then sends this message through the queue. Each receiver retrieves its corresponding message from the queue, displays it, and sends an acknowledgment back to the sender. Once all child processes have completed, the master removes the message queue and terminates. This system allows for structured communication between processes via message queues, with the sender sending messages to specific receivers and receiving acknowledgment in return.

master.cpp

The master takes a command line argument of the number of receivers it wants to create, assigning this value to a variable x. It also gets its own process id and outputs it. The master program then creates the message queue using ftok() and msgget(), outputting and assigning this to an integer variable named msgid. Using fork(), it creates a child process for the sender process, and execl to execute the sender using the command line. Using a for loop, it loops x times creating receiver child processes. After creating all necessary child processes, the master process waits until all child processes terminate, deleting the message queue at the end.

sender.cpp

The sender process creates a structure named msg_buffer which takes a message type to filter which message to send/receive and the contents of the message in a character array acting as a buffer. Looping x times, for x is the number of receivers created, it takes a user input as a string for the message to send, and the receiver it wants to be sent to. Then the process creates a msg_buffer structure named message, and assigns the message type to be the user's receiver choice every loop. The message contents assigned by the user then gets copied into the message content of the message struct. This is sent to the message queue waiting for the receiver to send an acknowledgement message back.

receiver.cpp

The receiver process has the same structure msg_buffer that holds the message type and message content the same way sender.cpp has it. By using msgrcv(), the process takes the message from the message queue with the corresponding message type. This message is then stored into a msg_buffer variable and then is printed. The process then creates another msg_buffer called acknowledgement with a message type of 999 and the contents stating the receiver acknowledges receipt of the message sent. This is sent to the message queue for the sender to pick up via the message type 999. The process then terminates afterwards.