PA3: Interprocess Communication Mechanisms

Maria Dmitrievskaia

UIN: 927009911

CSCE 313 – 512

Due Date: October 4th , 2020

Design:

The structure of the code is the following: There is an abstract class “RequestChannel”, of which the FIFO request channel, Message Queue request channel and Shared memory request channel are subclasses of. The request channel class holds the common components of the subclasses, such as name, two IPC objects’ names and the file descriptors. Each subclass, however, has its own cread and cwrite function, as well as its own open_ipc function. The FIFO request channel subclass operates similar to that of the class in PA1.

The Message Queue request channel creates 2 MQ objects, specifies a destructor, and the send and receive functions. The two objects created in the Shared memory subclass are kernel-semaphore protected segments that provide synchronization for the send and receive functions. The Shared memory class also featuerd a destructor and the send and receive functions.

In client.cpp, I added -i and -c to the getopt arguments in order to be able to get the number of channels and the chose IPC method. When creating the child process, I am able to send the buffer capacity and ipc method to the server. Depending on the IPC method, the corresponding class constructor is called and is stored in a variable called “control_chan”. If there are multiple channels, then the “isnewchan” bool value is set to true and multiple channels are created in a similar fassion. I then place the channels in a vector called “chan_vector.” The user can then either request 1 data point, 1000 data points, or a file transfer. If the user is requesting 1 data point from miltiple channels, the data point will be retrieved from all of the requested channels. If 1000 data points are requested, I output the data to the same number of files as there are channels. If a file transfer is requested, each channel transfers the same amount of data to a file sequentially.

On the server side, the server gets the buffer capacity and the chosen IPC method and handles the requests from the client. So, it will check for the buffer capacity and the IPC method. The, the populate_file data function is called. This function creates the people and their data in the BIMDC folder. We then initialize the request channel based on the chosen IPC method(FIFO if none). The handle process loop function is then called, in which we continuosly read from argument “channel” using a buffer.If a Quit message is read from the channel, then we break from the infinite loop and go back to the servers main, in which we delete the server control channel. When we receive a message, we prosses it by calling the “process request” function. This function determines the message type. If data message, we get the person, time, and ecg number from the data message and write it to the channel for the client to read. If file message, we determine whether the message is asking for size of the file, or for a piece of the file. If message is asking for a piece of the file then we read we open the file and use fseek to set the file pointer to the offset value and grab f.length bytes and write that piece to the channel. If There is a new channel request, then we create a new data_channel based on the chosen IPC method.

Time Comparison for IPC for 1000 data points:

FIFO:

Channel 1: 2.79 seconds

Channel 2: 2.88 seconds

Channel 3: 2.88 seconds

Channel 4: 2.85 seconds

Channel 5: 2.89 seconds

Message Queue:

Channel 1: 2.89 seconds

Channel 2: 2.85 seconds

Channel 3: 2.93 seconds

Channel 4: 2.85 seconds

Channel 5: 2.88 seconds

Shared Memory:

Channel 1: 2.82 seconds

Channel 2: 2.89 seconds

Channel 3: 2.94 seconds

Channel 4: 2.87 seconds

Channel 5: 2.88 seconds

Time Comparison for IPC for file transfer with 5 channels:

FIFO:

Channel 1: 0.0071 seconds

Channel 2: 0.0048 seconds

Channel 3: 0.0044 seconds

Channel 4: 0.0114 seconds

Channel 5: 0.0103 seconds

Message Queue:

Channel 1: 0.0029 seconds

Channel 2: 0.0027 seconds

Channel 3: 0.0029 seconds

Channel 4: 0.0028 seconds

Channel 5: 0.0027 seconds

Shared Memory:

Channel 1: 0.0055 seconds

Channel 2: 0.0056 seconds

Channel 3: 0.0029 seconds

Channel 4: 0.0033 seconds

Channel 5: 0.0035 seconds

As seen above, for the 1000 data points the results are comparable, however, for the file request, FIFO IPC is the slowest and the Message Queue is the fastest.

YouTube Video Link:

Part 1: https://youtu.be/vc72gw-Et6I

Part 2: https://youtu.be/4TjAw2iZNyY