CS214 Assignment 3

John Russell and Andrew Dos Reis

Functionality

We were able to successfully implement the net versions of the required system calls, but no extensions were implemented.

Message Protocol

Messages were sent using a fixed-length length-prefix protocol. When data is sent between the server and client, a 4-byte length-prefix is first sent indicating how much data should be read. If the 4 bytes successfully arrive, it is converted into an integer *length*, and then *length* bytes are read from the socket. If *length* bytes are not read and an error code is received from the call to read(), then an error was encountered.

Message Formats

There are different formats for messages that are sent between client and server depending on the system call.

Open

Client sends: *o!<flag>!<pathname>*

Server responds: *<-1>!<error>* if open fails, or *<filed>* if it succeeds

Close

Client sends: *c!<filed>*

Server responds: *-1!<error>* if close fails, or *0* if it succeeds

Read

Client sends: *r!<nbyte>!<filed>*

Server responds: *-1!<error>* if read fails, or *<nbyte>!<buffer>* if it succeeds

Write

Client sends: *w!<filed>!<nbyte>!<buffer>*

Server responds: *-1!<error>* if write fails, or *<nbyte>* if it succeeds

Libnetfiles

This library provides net versions of the open, close, read, and write function calls. They are called with the same parameters as the system calls, but before they can be used, a call to netserverinit() with the hostname must be made.

The port for both the server and libnetfiles are hardcoded at 40690

Server

For every client that connects to the server, a new thread is spawned. In this thread, the server reads and handles messages from the client and sends the appropriate responses. When the client closes the connection or an error is encountered, the thread exits and closes the connection from the server end.