<u>Systems Programming: Spring 2017</u>
Assignment 3: Wherefore Art Thou, File?

Contributors: Enkai Ji & Joseph Moussa

Implementation of a file server in connection with a client program using sockets and threads. Basic functionality includes the standard netopen(), netread(), and netwrite() commands to work over socket connection in the server.

DESIGN:

1. **<u>client.c</u>**
   This is the main program that the user will interact with. It starts with a netserverinit() call to verify there is a host to connect to. If there is it proceeds to make the net function calls to the server, if there isn't it stops and prints the errno;

2. **<u>libnetfiles.c</u>**
   This is the library of net commands that will be used by the client to be sent to the server. The basic commands that can be called are netopen(), netread() and netwrite(). There is also netserverinit() that validates the existence of a host.
3. **<u>netfileserver.c</u>**
   This holds the server program and thread function that manages each of the calls. Essentially the main function will create a socket, then bind it to an address and then listen with a limit of 5 connections. As it listens the client will then connect and the server will reply with an accept call. From there a thread is created, the arguments for the call are then passed onto the heap and into the function pointer for the thread. Before the accept function and during the threading process a lock is placed on it, as you will see in the code with MUTEX flanking the while(1) loop. The while loop ensures that the server keeps listening and doesn't stop and wait for a return message with new connections. A helper method str_split() is implemented to break up the message received from the client and turn them into arguments. The final method is the function used for every new thread (service_single_client). It takes the socket and message from the client and detects the operation (open, read, write) as well as the parameter and executes the call and sends the results back to the client.