Name: Deepkumar Patel, NetID: dgp52

Name: Viraj Patel, NetID: vjp60

Base Program and Extension A

****Makefile commands: (1) make (2) make clean. Before running (1), be sure that 'client.c' is present in the same folder in order for libnetfiles to compile****

Base Program:

int netserverinit(char * hostname, int filemode)

- Netserverinit checks if the host name is a valid host, it returns zero if it successfully found the host name, otherwise it returns a negative one. Since we did "Extension A", it also takes in the file mode. The integer for each file modes, that is unrestricted mode, exclusive mode, and transaction mode is 0, 1 and 2 respectively. If the input modes are not valid then this function also returns a negative one. All the files that will be open by the client will have the same mode, unless later on if the client chooses to call this function again, then it will change the mode to the most recent function call. Depending on the errors, this function will set the h_error for host and file mode errors.

netopen, netread, netwrite and netclose

- All of the these functions have five things in common, they all create their own socket, connects to the server, every input parameters to these functions are turned into strings as a buffer, between each inputs of buffer there will be a space for parsing purposes on the server side, they all read from socket. Once the buffer string arrives from the server, it checks to see if the last character is -1, if so, then there's an error, and it sets the respective errnos. However, if it's anything other than -1 then it returns the respective values of each functions. At the end it closes the socket and returns the value.

Server: How does it connect with client?

- As we compile and run the sever file, it creates a socket and listens for the connections, we have a while loop which will listen for the connections forever. Once we receive a request we send all the necessary arguments to the worker thread as a structure.

Server: How does it know the functions call?

- The worker thread will have the buffer that was sent by the user, and every buffer will either have a number 1, 2, 3 or 4. Those numbers represent open, read, write, and close respectively. Therefore, while we split the buffer between spaces, we just look for the

last character and it will go to the corresponding functions. Once the computation is done, the worker thread will write back to the listening client, if there was an error, then it will set the errno in both server and client.

Linked List: How extension A works?

- All the clients that connect to the server and send buffer will be added to the head of the linked list, however, if the permissions aren't allowed then it will return the file descriptor as -1 and set the permission errors. Net open sends the mode along with the buffer. The worker thread will get the buffer and it will decode the message, after that it checks to see if the file exists, it will set the errno if it doesn't exist otherwise it will proceed further and it will create the head of the linked list. The very first client will be added automatically in the linked list along with its file name, ip address, permission/mode, flag, and the file descriptor. If the client is second and onward then depending on the current user mode it will call the following functions unrestrictedmode(), exclusivemode(), and transactionmode(), those function will return one if it's safe to open without any permission issue, otherwise it will return non negative value. Clients will be added to the linked list as long as permissions are valid.

Linked list: How to close and remove node/client from linked list?

- While closing the file we match the client's ip and file descriptor of the client who called this function with the linked list data structure we created. If it matches at some nth position in the linked list then we remove that client/node from the list. In another words, it tells the server that this client is done using the file, and it could safely remove the node.

  Client must close the file that was opened by them, trying to open the file with similar mode and flag will result in error. The buffer size is 1024 bytes, sending more data over to the server might result in unpredictable behavior. The port number we used is 9672, and it is hard coded in both libnetfiles and netfileserver.