Assignment 3: A Better Open

Malay Shah mhs145

Hammad Iqbal hi60

Base Program:

For the base program we created a way to easily access files across networks. We created 'netopen', 'netread', 'netwrite', 'netclose', and 'netserverinit'. We used the hostname and the file mode for the netserverinit function. We made sure the file mode was valid, the IP address was valid, and if the connection worked. If these things did not work we printed an error statement in the command prompt. For netopen was used the pathname and flags to make sure the socket could be write and read to correctly. We printed error statements otherwise. Netread used the file descriptor, buffer, and byte size. It would return the number of bytes written to the file that was directly associated with the file descriptor. Netwrite had the same parameters as netread and was implemented in a similar way with char *tok and strcmp. Netclose used the file descriptor and printed the errors whether or not we could write or read to the socket.

Extension A + C:

This part of the project would provide three types of file connections. Unrestricted mode allows any number of clients to have the same file open with any permissions. Exclusive mode allows any number of clients to have the same file open in read mode but only one open in write mode. Transaction mode allows one distinct client to a given file at any time. These different clients have the same file open in different modes simultaneously. Unrestricted mode = 0, Exclusive mode = 1, Transaction mode = 2.

The next extension checked if the client wants to open a certain mode and if the file server should check to see if the file is closed. Therefore, once the file is closed the next mode is queued up and will succeed. If a filesystem error should occur the request will not be requeued. All error codes for this part of the extension are implemented when needed.

We built a struct called threadArg which had to due with the client struct length, client socket file descriptors, and the buffer size. We built a struct called fileInfo which had the flag, mode, file descriptor, and file name. We used a function SendE to send in an error. We used a function errorChange to change the error. We built a initialization of the socket to make sure the socket connection worked and was binded. The function isTransacation checks if a file is open in transaction mode. The function checkFileOpen checks if the file is open. The function isExclusive checks if the file is in exclusive mode. We used the file information and a pointer to the file descriptor while checking the flag. The function checkUnrestrictedWrite was implemented in a similar way and checked if it was in unrestricted mode. The functions checkCanOpen will check the cases for transaction, exclusive, and unrestricted mode and see if they are able to open. Simulatenetopen was implemented using the file information and file descriptor with malloc. We checked the mode and the flag while returning the file descriptor. With the data given we decoded the request using the function decodeRequest and if the

error was an invalid file mode we exited the thread. If the error were any of the other codes we would send the error and exit the thread as well. We made sure the file index was found and the client buffer was also found.