CS 214: Systems Programming
Assignment 2: WTF

Authors:

Jonathan Konopka
NETID: jk1549
RUID: 178005220
Nicholas Rytelewski
NETID: nr548
RUID: 197000952

Project Description :

This project acts as a very simplified version of github, where a server contains a collective version of a repository, and clients have their own version of the repository locally. In order to keep versions of the code the same amongst the group, a client can upload (push), and download (pull) each other's versions of the code to and from the server. Our WTF system included the commands 'configure', 'checkout', 'update', 'upgrade', 'commit', 'push', 'create', 'destroy', 'add', 'remove', 'currentversion', 'history', and 'rollback', all of which were stated in the original assignment PDF.

Thread Synchronization :

Our program made use of threads by calling a fork() command whenever a new client connection was accepted on the server side. Our fork() was assigned to a pid int and if the int equaled zero on the server (indicating that the process currently happening is a child of the server's main process), then the process would carry on executing, and completing the requests sent by the server until the operation was over. The main process (PID != 0) would never be the one executing client requests.

Functions :

These are a list of functions implemented into both WTF.c and WTFserver.c. This list does not include a description of the functions for each command, as they are outlined in the project description (it would otherwise be redundant to explain what they do).

void removefl(char* projectName, char* fileName)

Takes the project name and a specific file name related to the project. This function is designed to open the servers .Manifest of the project name supplied and remove the entry that contains the file name.

void download_files(int socketfd)

Takes a socket file descriptor as an argument. This was a helper function used to send files over from either the client to server or server to client (both WTFserver.c and WTF.c had this function). Our protocol to send files was to send the server/client an int of how many files were about to be downloaded, and then for each file, send the number of bytes in the file name, the actual file name, and the bytes in the file one byte at a time.

char* makehash(char* filepath)

Uses MD5 from the openssl library to compute a hash for the given file path\ to be put into a manifest of used as the live hash of a file for comparisons.

int getfilelength(char* filepath)

Returns the number of bytes in a file. This was a helper function that was used to loop through the entirety of a file.

int checkIfProjectDirExists(char* projectName)

Used by the server to see if the given project name already exists on the server, otherwise commands such as create will not be able to be run. If the directory existed it would return 0, otherwise it would return a -1.

void sendManifest(char* projectName, int clientfd)

Sends the server's manifest of the given project name to the client side, necessary for the update/upgrade and commit/push commands so that comparisons would occur between the client and server. When sent to the client, it would appear with the name '.ManifestServer' and be deleted as soon as it was done being used for comparisons.

void quitfunc()

Closes all file descriptors and shut down the server. The server is generally supposed to close when it receives a SIGINT (Ctrl+C); this function makes sure that all file descriptors and clients are shut down properly in the case that the SIGINT is supplied.