README

WTF CLIENT

WTFclient.c

Function	What it does
int main(int argc, char** argv)	- Parses the clients commands - Commands that don't require connecting to the server are separate from the ones that do require connecting -
int checkout(char* projname, int sd)	- Sends command "checkout" and project name to server to retrieve project from server - Fails if project doesn't exist in server
int configure(char* ip, char* port)	- Takes the ip and port# passed in through the command line and creates the .configure file containing that info
int charToInt(char* numArr)	- Converts a char array to an int - Created because we don't know how many digits long the int might be - The buffer contains a ':' at the end of the buffer to indicate the end of the number
int create (char **, int)	- Sends command "create" and project name to server to create project in server root - If project already exists in server, error sent back - If project does not exist in server, project created in root - Server sends project back if client does not have

	- Client parses protocol sent and reconstructs project in local repository
int currentversion(char**, int, int)	- Sends command “currentversion” and project name to server to retrieve currentversion of project - Server sends back currentversion in protocol - Contents outputted to stdout
int history (char**, int)	- Sends command “history” and project name to server to retrieve history of project - Server sends back protocol containing history file - History file contents outputted to stdout
int writeToSocket(char**, int)	- Writes bytes to socket to send to server
int searchProject(char * proj)	- Searches to see if project exists in client’s local repository

clientcommands.c

Struct	What it does

`typedef struct manifest{`
`        char* projversion;`
`        char* filepath;`
`        char* vnum;`
`        char* hash;`
`        char* onServer;`
`        char* removed;`
`        struct manifest* next;`
`}Manifest;`
	- Linked list contains info on entries in the manifest
	- Project version is contained in all of the nodes
	- Each value after projversion correspond to a column in the manifest

Function	What it does
char* searchProj(char* projname)	- Searches for the given project on the client side - If it exists, it returns the projpath - Otherwise it returns NULL
void build(char* manpath, Manifest** head)	- Builds a linked list based on the .Manifest file - puts the linked list in head - The very first node created simply contains the heading for the manifest, so that even in the case the manifest is empty, the headings can still be written to the file easily, and the project version is contained somewhere
void writeM(char* manpath, Manifest* head)	- Overwrites the given .Manifest file with all of the information stored in the given linked list - The project version is printed at the very top first - Even if the manifest does not contain any files, it is guaranteed that the headers will still be printed out, because the Manifest* head contains the headings for its values ie head->filepath = “Filepath”, head->vnum = “Ver#”, etc.
void printM(Manifest* head)	- Prints out the manifest linked list exactly as how it would appear in the .Manifest file - Mostly used for debugging
char* hashcode(char* filepath)	- Given a path to a file, a hashcode using SHA256 is generated - Although sha256() returns an unsigned char*, the result is converted to a char* in the function, thus returning a char* to the hashcode
void addM(char* projname, char* filename)	- This function is called inside of WTFclient.c to correspond to the “add” command - Given the project name, it calls searchProj to check that it exists, and gets the path to the project if it does exist - If it does not exists, an error is printed

	- If it does exist, it prints that the project was locally found - Next, “/.Manifest” is appended to the end of the project path to get the path of the manifest - build is called to build the Manifest linked list - The linked list is traversed to search for the file path given as an argument - If the filepath is found in the linked list, the hashcode is updated - If the node is marked as removed (as in removed = “1”) the node is remarked to indicate it is not removed (removed = “0”) and a message is printed to indicate that the file was added to the manifest - If the node is not marked as removed, then a warning that the file exists and the hashcode is updated is printed - At the end of the loop if the pointer is NULL, then the file does not exist in the manifest, so a new node is appended to the end of the linked list to indicate a new file with version 1.0 - The linked list is sent to writeM to write all the data in the Manifest linked list to .Manifest - The linked list is freed using freeManifest - A message that the file was added is printed
void removeM(char* projname, char* filename)	- This function is called inside of WTFclient.c to correspond to the “remove” command - First, searchProj is called to check if the project exists and print an error if it does not - Set up is similar to addM, in that the path to the Manifest is built and the Manifest linked list is created using build - The linked list is traversed - If the filename is found, the removed variable in the node is updated to “1” to indicate that it is removed on the client’s side but the server has not seen it - If the filename is not found, an error

	is printed that it does not exist in the manifest
void freeManifest(Manifest* head)	- Frees the Manifest linked list by traversing it
int destroy(char* projname, int ssd)	- Sends destroy command to server - If any failure is reported by the server, error is printed

commitupdate.c

Function	What it does
int push(char*, int);	- Takes in project name and socket descriptor for server - Edits its own .Manifest and sends over to server - Sends necessary files to server
int commit(char*, int);	- Creates .Commit by comparing server and client .Manifests and sends result to server
int update(char*, int);	- Creates .Update by comparing server and client .Manifests
int upgrade(char*, int);	- Requests files listed on .Update from server
Manifest* createLive(Manifest*, int);	- Generates the live hashes for client's manifest - If called by commit, then all files with a different live hash than Manifest have a new file version - Else if called by update, no new file version are generated

parseprotoc.c

Function	What it does
void *parseProtoc*(char **, int);	- Parses protocol sent over from server to client - Recreates files and directories with the same structure and data as that of server - Recreated in local repository
void createDir(int subdir_size, char**, subdir_name);	- Based on parsing done by parseProtoc, creates empty directory with given name
void createFile(int filename_size, char** filename, int filedata_size, char** filedata);	- Based on parsing done by parseProtoc, recreates a file with proper name and contents
void unzip();	- Unzips a .gz file
int openEmptyZip(char ** bufferbytes, int bufflen)	- Opens an empty .gz file to append compressed contents to

WTF SERVER

Global Variables

th_container threads [BACKLOG];	An array used to keep track of threads, where BACKLOG is a macro (BACKLOG = 50)
int numthreads;	Indicates how many threads currently working
pthread_t mainboi;	Main thread that manages threads
int sockfd;	Socket file descriptor used to communicate with server

PROJECT * PROJECTS_LL = NULL;	Linked list to keep track of all files present in server's root directory - Use of mutexes here to ensure different threads don't make changes to the same project at a given time

WTFserver.c

Function	What it does
int checkoutProj(int cfd)	
int charToInt(char* numArr)	- Converts a char array to an int - Created because we don't know how many digits long the int might be - The buffer contains a ':' at the end of the buffer to indicate the end of the number
int main(int argc, char** argv)	- Sets up server and networking - socket, setsockopt, bind, listen, & accept used here - Creating and maintaining threads here - Sets up sigset_t so that SIGINT gets blocked - Joins and canceled on done threads
int searchProj(char * proj)	- Searches for a project in server's root directory - If exists, returns 0. If not, returns -5
int createProj(int cfd)	- Creates project in server root if it doesn't already

	exists. - Alerts Client once created - Waits for “OK” from client to send over project so that client can have it in it’s local repository
int writeToSocket(char ** buf, int cfd)	- Allows server to communicate with client and send over bytes of data to client
int fetchHistory(int cfd)	- Sends History of a project over to the client in a protocol
int currver(int cfd)	- Sends currentversion of a project over to the client in a protocol
void * handleClient(void * thr_cont)	- Each thread executes this function - Handles client and executes requested commands - Once thread done carrying out tasks, closes communication file descriptor, marks thread as done, and calls pthread_exit
void * handle_sigs(void * args)	- Thread created in main and assigned to execute handle_sigs - Thread will remain blocked until SIGINT becomes pending - Once SIGINT becomes pending, thread shutdowns socket file descriptor with shutdown call, and calls pthread_exit
void addToLL(char ** projname)	- Adds a project to the project global linked list
PROJECT ** searchinLL(char ** projname, PROJECT ** projstruct)	- Searches the project global linked list to see if a project exists

createprotocol.c

Function	What it does
void createGzip();	- Creates a gzip file using zlib
void traverseDir(char ** path)	- Traverses a project directory and generates metadata information. - appends metadata info to a protocol file
void destroyProtocolFile()	- Removes protocol file
int openrequested(char ** path, char ** cmd)	- Opens requested file path - Determines if it’s a file or a directory - If file: appends send(file, hist, or mann): generates associated protocol - If directory - Appends senddir: and associated metadata protocol - Calls traverseDir to generate and append rest of protocol based on inner contents of directory
void createProtocol(char **path, char **cmd, int sockd)	- Calls openrequested to generate protocol - Opens protocol file after generation, reads in contents, sends across client file descriptor to client to parse.

threadsetup.c

STRUCT

STRUCT	What it Does

`typedef struct th_Container{ pthread_t thread_id; int is_done; int cfd; }th_container;`	Contains information about a thread, its current status, and the client file descriptor the server can use to communicate with the client.
`typedef struct sig_waiter_args{ sigset_t * set; int sockfd; }sig_waiter_args;`	Contains arguments passed to handle_sigs method - Contains blocked signals (SIGINT) - Contains socket file descriptor of server to be used to shutdown server once SIGINT becomes pending
`typedef struct PROJECT{ pthread_mutex_t proj_lock; char * projname; pthread_mutex_t numthread_lock; int num_threads; int destroy; struct PROJECT * next; }PROJECT;`	Contains name, locks, the number of threads currently working/waiting on a project, and destroy status of a project.

Commands.c

numCommits = 0

- Global variable keeping track of the number of commits created while server is running
- Incremented each time .Commit is active

int deleteCmts(char*);

- Deletes all .Commits in project folder besides the one sent by the client

int validCmt(char*);

- Checks if given .Commit matches one of the .Commit# in the project folder

int upgrade(int);

- Sends over files requested by the client

int destory(int);

- Checks that the requested project exists, and calls deleteDir

int deleteDir(char*);

- Recursively deletes a directory and all its contents

int commit(int);

- Sends .Manifest to client, waits for .Commit and saves it as .Commit# where # is the number of .Commits created up to that point

int push(int);

- Completes all commands on .Commit file and moves old project into ./prev