WTF(1)

NAME

WTF - Where's That File, a version control system

SYNOPSIS

WTF [COMMAND] |OPTIONS|

DESCRIPTION

WTF has 3 main functions, their descriptions are listed below:

configure

Saves the IP address and port of the server to later be used to connect to the server. This is written then to a .configure file

checkout

Will attempt to fetch the current version of the given project from the server. Fails if the project already exists locally, or if the project doesn't exist.

update

Fetches the server's .Manifest and compares with the client's .Manifest and sees what files need to be updated. Will check for any conflicts and prepare a .Update file if the command runs successfully.

upgrade

Sends the .Update file to the server and fetches any files that need to be added/modified and removes any files that need to be removed. Fails if there is a .Conflict file.

commit

Fetches the server's .Manifest and compares with the client's .Manifest and sees what files need to be pushed to the server. Fails if there is a .Conflict file or if the server .Manifest is ahead of the client's .Manifest

push

Sends the .Commit file to the server along with any files that need to be added/modified. Will also remove any files that need to be on the server's version. Increments the project version and file versions.

create

Creates the project on the servers end and sends the .Manifest to the client. Fails if the project already exists.

destroy

Destroys the project on the servers end. Does not destroy the project locally. Fails if the project doesn't exist.

add

Adds a new file to the clients .Manifest along with a new hashcode and version number. Fails if the

remove

Removes a file from the clients manifest.

currentversion

Requests the current state of the project from the server. Prints out all files in the .Manifest along with their versions. Fails if the project doesn’t exist. Does not require project to exist locally

history

Requests the projects full push history. Prints out the full history of the project. Fails if the project doesn’t exist. Does not require the project to be stored locally.

rollback

Will request the server to rollback the project to the version number. Fails if the version number is invalid or if the project does not exist

Author

Written by Andrew Park and Ryan Davis.

Reporting Bugs

Email rgd51@scarletmail.rutgers.edu or ap1614@scarletmail.rutgers.edu

SYNOPSIS

The project consists of mainly 7 files: **server.c**, **server.h**, **client.c**, **client.h, simpleIO.c, simpleIO.h**, **Makefile**

- **server.c** contains the code for the server.
- **Server.h** contains the function headers for the server
- **client.c** contains the code for the client
- **client.h** contains the function headers for client
- **simpleIO.c** contains the code for the simpleIO library
- **simpleIO.h** contains the function headers for the simpleIO library
- **Makefile** will automatically clean and compile WTF along with WTFserver, and any test files that need to be made

DESIGN

In order to ensure that multiple clients are able to communicate with the server safely. The server uses threads for each client connected to the server. In addition to this, the server uses mutex locks in order to ensure that different clients are not able to open the same project at the same time in order to ensure thread safety. This occurs whenever a thread accesses any of the files within a project.

In addition to thread safety, data compression was used to keep disk space and network traffic at a minimum. All previous versions of projects are stored and compressed into a single file. Along with this, all files sent over the network are compressed in order to reduce network traffic.

With both of these techniques implemented, we keep thread safety as a top priority while ensuring that network traffic and file sizes are kept to a minimum. The data compression was done using system().