C: WTF CLIENT MANUAL

WRITTEN BY ARYAN SHAH AND ALLEN ZHANG

NAME

Configure - Saves the IP/hostname and the port

SYNOPSIS

```
./WTF configure [IP/hostname] [port]
```

DESCRIPTION

Writes the IP/hostname and port to a .Configure file stored within the project for use by later commands.

If [IP/hostname] or [port] arguments are missing, return fatal error. If [port] is invalid, return fatal error.

NAME

checkout - Requests the current version of the project

SYNOPSIS

./WTF checkout [project name]

DESCRIPTION

Requests the entire project from the server, which will send over the current version of the project, .Manifest as well as all the files that are listed in it. Afterwhich, the client constructs corresponding directories.

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project name doesn’t exist on the server, the client can't communicate with the server, if the project name already exists on the client side or if configure was not run on the client side.

NAME

update - Compares client and server manifest for updates

SYNOPSIS

./WTF update [project name]

DESCRIPTION

Fetches server’s .Manifest for the specified project and compares every entry in it to the client’s .Manifest and sees if there are any changes on the server side for the client. If there are, it adds a line to a .Update file to reflect the change and outputs information about what needs to change. If there is an update but the user changed the file that needs to be updated, update instead writes to a .Conflict file and deletes any .Update file. If the server has no changes for the client, update will inform the client that the version is up to date.

RETURN VALUE

On a full success case, update will return “Up to Date” to std out. In the partial success case, update will write to the .Update file and print the changes needed to be made to std out. Returns a fatal error if the project name doesn’t exist on the server and if the client can not contact the server.

NAME

upgrade - applies the changes in the .Update file

SYNOPSIS

./WTF upgrade [project name]

DESCRIPTION

Applies the changes listed in the .Update to the client's local copy of the project. Deletes the entry from the client's .Manifest for all files tagged with a "D", fetches from the server and then writes or overwrites all files on the client side that are tagged with a "M" or "A", respectively. Upon completion, it deletes the .Update file. If the .Update is empty, it informs the user that the project is up to date and deletes the empty .Update file. If no .Update file exists, it tells the user to first do an update. If .Conflict exists, it tells the user to first resolve all conflicts and update.

RETURN VALUE

Returns success message upon proper completion. Returns fatal error if the project name doesn't exist on the server, if the server can not be contacted, if there is no .Update on the client side or if .Conflict exists.

NAME

commit - Compares client and server manifest for commits

SYNOPSIS

./**WTF commit** [project name]

DESCRIPTION

Fetches the server's .Manifest and checks if the .Manifest versions match. If they do not match, it asks the user to update its local project first. If the versions match, it runs through its own .Manifest and computes a live hash for each file listed in it. An entry is written out for every file whose live hash is different from the stored hash saved in the client's local .Manifest to a .Commit with its file version number incremented. Send its .Commit to the server and reports success

RETURN VALUE

Returns success message upon proper completion. Returns fatal error, if the project name doesn’t exist on the server, if the server can not be contacted, if the client can not fetch the server's .Manifest file for the project, if the client has a .Update file that isn't empty (no .Update is fine) or has a .Conflict file.

NAME

push - Updates the server files with client’s .Commit

SYNOPSIS

```
./WTF push [project name]
```

DESCRIPTION

Sends its .Commit and all files listed in it to the server. The server locks the repository so no other command can be run on it. While the repository is locked, the server checks to see if it has a stored .Commit for the client and that it is the same as the .Commit the client just sent. If this is the case, the server expires all other .Commits pending for any other clients, duplicates the project directory, writes all the files the client sent to the newly-copied directory (or remove files, as indicated in the .Commit), updates the new project directory's .Manifest by replacing corresponding entries for all files uploaded (and removing entries for all files removed) with the information in the .Commit the client sent, and increases the project's version. The server then unlocks the repository and sends a success message to the client. If there is a failure at any point in this process, the server deletes any new files or directories created, unlocks the repository and sends a failure message to the client. The client erases its .Commit on either response from the server.

RETURN VALUE

On a full success case, update will return a success message to std out. Returns a fatal error if the project name doesn’t exist on the server, if the client can not communicate with the server or if the client has no .Commit file.

NAME

create - Creates a new project folder with the given name

SYNOPSIS

```
./WTF create [project name]
```

DESCRIPTION

Creates a project folder with the given name, initializes a .Manifest for it and sends it to the client. The client sets up a local version of the project folder in its current directory and places the .Manifest the server sent in it.

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project name already exists on the server or the client can not communicate with the server.

NAME

destroy - Deletes all files and directories in the project

SYNOPSIS

```
./WTF destroy [project name]
```

DESCRIPTION

On receiving a destroy command the server locks the repository, expires any pending commits, deletes all files and subdirectories under the project and sends back a success message.

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project name doesn’t exist on the server or the client can not communicate with it.

NAME

add - Creates an entry in client manifest for the file

SYNOPSIS

```
./WTF add [project name] [filename]
```

DESCRIPTION

Adds an entry for the file to its own .Manifest with a new version number and hashcode.

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project does not exist on the client.

NAME

removes - Removes the entry for the file in client manifest

SYNOPSIS

`./`**`WTF remove`**` [project name] [filename]`

DESCRIPTION

Removes the entry for the given file from its own .Manifest.

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project does not exist on the client.

NAME

currentversion - Requests the current state of the project

SYNOPSIS

`./`**`WTF currentversion`**` [project name]`

DESCRIPTION

Requests from the server the current state of a project from the server. Outputs a list of all files under the project name, along with their version number (i.e., number of updates).

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project does not exist on the server.

NAME

history - Sends a file containing all successful operations

SYNOPSIS

./WTF history [project name]

DESCRIPTION

Sends over a file containing the history of all operations performed on all successful pushes since the project's creation.

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project doesn’t exist on the server or the client can not communicate with it.

NAME

rollback - Reverts current version of server project

SYNOPSIS

./WTF rollback [project name] [version]

DESCRIPTION

Reverts its current version of the project back to the version number requested by the client by deleting all more recent versions saved on the server side.

RETURN VALUE

Returns a success message to std out on success. Returns a fatal error if the project name doesn’t exist on the server, the client can't communicate with it, or the version number given is invalid.