Installing and Updating ChatScript

Revision 2/09/2017 cs7.2

Installing (Windows, Mac, Linux)

Installing Windows

Windows is simple. Unzip the zip into a folder of your choosing (typically called ChatScript), preserving directory structure. You are ready to execute ChatScript.exe immediately.

Installing Mac

Life is not so easy for a Mac user since I don't have a mac. After unzipping into a folder (typically called ChatScript), you need to compile the src in SRC. The alternate technology is to beg in the ChatScript forum on ChatBots.org for a mac user to send you their executable copy of the latest build. To compile means using XCODE and making an appropriate make file for it based on the make file in SRC. And you have to install curl and optionally postgres. If you don't have or want them you can in your build file do defines of: DISCARDDATABASE and DISCARDJSON.

Alternatively you can use the binary in `BINARIES/MacChatScript`, but it is at least one version behind unless you are grabbing chatscript directly from github.

There is also an Xcode project in the `NON-WINDOWS NON-C/Xcode` directory that you can launch and build your own binary with. This is probably the prefered method for advanced users.

See the `Chatscript on a Mac` in `Overviews and Tutorials` directory of the docs for a more indepth discusion of compiling and using ChatScript on a mac.

Installing Linux

If you have a 64-bit machine, generally you can the LinuxChatScriptxx64 binary file directly, after first doing "chmod +x LinuxChatScript64" to make it executable by Linux. You may need to install "curl" as well if you use JSON webqueries. If you have a 32-bit machine or don't install curl or have other issues running, you may need to compile the system yourself. This means installing make and g++, then go stand in the SRC directory and type make server or make pgserver (for postgress). This will copy into the main CS directory ChatScript and ChatScriptpg. Note this is different from the names that come in the zip, which are LinuxChatScript64 and LinuxChatScriptpg64. The only thing you have to pay attention to is that if you install the cron job in the LINUX folder,

be aware that it names the Linux names and you either need to edit that cron or rename your files that you build.

Updating ChatScript

ChatScript evolves on a regular basis. New functionality is added, new bugs are created, and old bugs are fixed. Sometimes old functions or concepts get deprecated or removed. The top-level file changes.txt advises on new functionality and incompatible changes. You should read it. When functions get changed or deprecated, generally if you try to compile your old script that uses them, the compiler will complain and you just fix your source. When concepts are renamed or deleted, the compiler will issue warnings and you edit your script. Of course if you get zillions of warnings and pay no attention, you will miss that change.

The simple-minded way to install a new version is to unzip it and then move your bot's RAWDATA folder and `buildxxx.txt` file into the new CS folder. Then do :build of whatever. That's not the ideal way however.

The ideal way to do ChatScript is to have a folder that somehow contains your bot data AND contains CS as a subfolder. Like this:

```
MYFOLDER/
    startserver.sh
    BOTDATA/
        Smooth/
            smoothcontrol.top
        Fuzzy/
            fuzzycontrol.top
            flea.top
        AllBots/
            shopping.top
    filesmybot.txt
    ChatScript/
```

In your `filesmybot.txt` you name the path to your `BOTDATA` files and folders appropriately. Then, to update ChatScript, you remove the ChatScript folder and drop in the new one. ChatScript automatically looks above itself to find your `filesxxx.txt` file if it can't find it within.

Example filesmybot.txt for the directory structure above:

```
# multibot example
../BOTDATA/Smooth/
../BOTDATA/Fuzzy/
../BOTDATA/AllBots/
# load in standard chatscript quibbles
RAWDATA/QUIBBLE/
```

You then start the server in the 'MYFOLDER' directory with a script like this on Mac or Linux, windows is similar:

```
#!/bin/sh
cd ChatScript && BINARIES/MacChatScript livedata=../LIVEDATA english=LIVEDATA/ENGLISH syste
```

Similarly, if you have your own `LIVEDATA` files, you can give a reference to your copy of files in `MYDATA/LIVEDATA` so you don't have to worry that CS copies may have changed. You want to allow CS to use its own copy of `LIVEDATA/SYSTEM` and `LIVEDATA/ENGLISH`. And you may prefer to use the substitution files of ChatScript. If you have your own canon file data you want to use and your own substitutions, consider adding them to your script file as `canon:` and `replace:` (see `:build` documentation).

If you actually go an edit CS code to add your own engine functions, don't modify CS engine code. Instead compile with the PRIVATE_CODE define and name in your include path where to find your files: privatesrc.cpp and privatetable.cpp so you can add code and table entries for them w/o touching the original CS source. You can augment the existing ChatScript dictionary in script, merely by declaring concepts with pos-tagging data (both generic and specific tags). E.g.

```
concept: ~morenouns ~NOUN ~NOUN_SINGULAR (webview webvan)
```