Setting up

On Windows

Running Lua Demon requies C++ Redistributables 2017 (vcredist) and the LuaJIT DLL (lua51.dll). This DLL can either be downloaded or compiled yourself and needs to placed in the same directory as the LuaDemon.exe. LuaDemon is currently only available as 64-bit for Windows.

On Linux / Unix / OSX

These systems look for a libluajit-5.1.so.2 which has to be placed in your systems library directory. Copying it to /usr/lib/ works to but is not a reccomended option.

Make sure you are using the right lib file for your system (x86/x64 or ARM).

Configuration

To display all available commands simply launch the LuaDemon executable with -help. Available commands are:

- -dir <directory> *required for operation*
    - Sets the working directory for the Lua environment. The LuaDemon process needs permission to read from this directory. The path MUST end with a slash (/). The program will automatically convert backslashes to forward slashes.
    - Using System links or remote directories is untested. File watching and automatic reloading may not work under these circumstances. See the -nofilespy argument to disable them.
- -nofilespy
    - Disables the file watching function. Hotloading of Lua files is no longer possible. Use this if it causes instabilites or if file realoading is unwated.
    - Its still possible to manually reload the environment using the Lua function `forceReload()`
- -nodebug
    - Hides the Cyan/Blue debug messages in the console.

Writing Lua for Lua Demon

Lua Demon is built on LuaJIT 2.0.5 and will run any plain Lua code. All classic Lua functions are available without restrictions. LuaDemon is designed not to impair the performance of Lua.

A Lua program must not block at any point and has to exit or finish!

For example a `while true` without an exit would mean it will never finish to initialize the lua environment and automatic reloading or Hooks would no longer work.Writing Lua for Lua Demon

Reloading and Hotloading

LuaDemon fully supports Hotloading of Lua. You can reload as many times as you want without increasing memory or CPU cycles. This means while LuaDemon is executing your Lua code you can press "safe" in your text editor and LuaDemon will reload the code with the new changes. The global data and functions are preserved. Reloading on file changes can be disabled using `-nofilespy`. A reload can also be initiated by Lua itself using `forceReload()`.

Keep this in mind when creating Hooks or when using global data – You may have to clean up to avoid interference with old data.

Available Lua Functions and Variables

LuaDemon exposed some functions to the Lua environment. Not all functions or variables are available on all systems and configurations. Check first in Lua before using them.

`nil` means the function returns nothing or expects no arguments

`nil include(string RelativePath)`

Allows inclusion of other Lua files. Exception safe. If the specified file is unavailable it will do nothing but output a warning to the console.

`nil forceReload(nil)`

Will force the Lua environment to reload all Lua files the same way a file change would. Global Lua data is preserved after a Reload.