

Setting up

On Windows

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

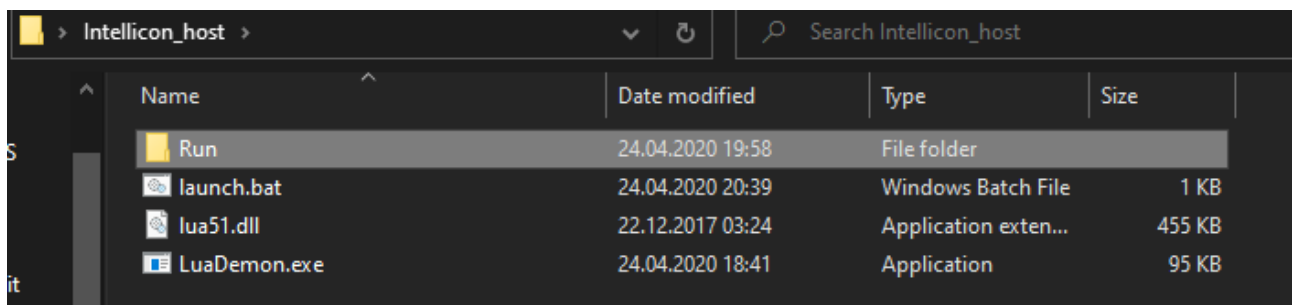
On Linux / Unix / OSX

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 but is not a recommended option.

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

The given directory MUST contain an init.lua as an entry point!

A typical setup on Windows looks like this:



The screenshot shows a Windows File Explorer window with the address bar set to 'Intellicon_host'. The window displays a list of files and folders. The 'Run' folder is selected. The list includes 'Run' (File folder), 'launch.bat' (Windows Batch File, 1 KB), 'lua51.dll' (Application extension, 455 KB), and 'LuaDemon.exe' (Application, 95 KB).

Name	Date modified	Type	Size
Run	24.04.2020 19:58	File folder	
launch.bat	24.04.2020 20:39	Windows Batch File	1 KB
lua51.dll	22.12.2017 03:24	Application extension...	455 KB
LuaDemon.exe	24.04.2020 18:41	Application	95 KB

```
title Intellicon HOST
.\LuaDemon.exe -dir C:\Users\Florian\Desktop\Intellicon_host\Run\
pause
```

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 but may work. File watching and automatic reloading may not work under these circumstances. See the `-nofilespy` argument to disable them.
- `-nofilespy`
 - Disables the file watching function. Automatic reloading of Lua files will no longer happen. Use this if it causes instabilities or if file reloading is unwanted.
 - It is still possible to manually reload the environment using the Lua function `forceReload()`
- `-filespyinterval` (*Linux only*)
 - Defaults to 500ms
 - Sets the interval to periodically check files for changes. Lower values will increase resources usage!
- `-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/break would mean it would never finish to initialize the Lua environment. Since LuaDemon would never start and initialize, automatic reloading or Hooks would not work.

Reloading and Hotloading

LuaDemon fully supports Hotloading of Lua. You can reload as many times as you want without increasing memory usage or CPU time. It cleans up the old Lua instance while keeping all global variables (stack/heap is unchanged). Reloading is triggered by file changes in the given directory. A reload may also be triggered manually from inside Lua by calling `forceReload()`. This behaviour may be changed by adding `-nofilesy` to the launch parameters (manual reloads still work).

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

Note for Linux users: The FileSpy on Linux uses `inotify()`. This means it will not see any file changes or updates made to files created in the directory after the program startup. On program startup it recursively seeks through the directory tree. It only sees changes made by the local system – if the files are changed remotely (by FTP, SMB, or on a NAS...), nothing will happen. Some editors use swap files when saving, this can lead to multiple unintentional reloads. LuaDemon reacts when a file is modified and then closed but it cannot differentiate what or if something changed. Some editors seem to be saving completely randomly. Multiple subsequent reloads do not affect the Lua state negatively, however your Lua program should be written with this in mind.

Available Global Lua Functions and Variables

LuaDemon exposes some global 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. The path is relative to the file its being called in. 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. Keep in mind you may cause a soft-lock if used recursively so avoid using it outside of debug applications.

nil setConsoleTitle(string NewTitle)

(Windows only)

Changes the window title in the Windows desktop GUI. This function is available on Linux for cross-platform compatability, but calling it will not do anything. Refrain from using special characters, see Windows reference for WinAPI SetConsoleTitle().

LuaDemon Libraries

LuaDemon adds several libraries to Lua:

- - File
 - For basic file handling
- - Network
 - Used for TCP/UDP network communication
- - Serial
 - Used for Serial (COM) Port communication
- - Std (global)
 - General utility functions
- - Think
 - Basic timing hooks and functions

Some library functions may behave differently on different architectures and OS. Some functions may not be available on all operating systems.

Note that the Network and Serial libraries work by polling.

Library: Serial (serial.)

Basic communication for serial ports. Ports are referred to in Lua by their name.

(ex: "COM4")

Name	Arguments	Return	Description
Discover <i>(Windows only)</i>	nil	nil	Returns a list of all available serial ports.
Open	string Port, number Speed=9600, number Bytesize=8, number Stopbits=1	bool true	Opens and initiates a serial port.
Send	string Port, var Data	string Data	Sends complete string. (null characters in string will be sent)
Receive	string Port, function Callback(nil)	nil	Calls the function whenever a new byte is received on that port. Only one callback per port.
Available	string Port	string Data	Returns
Read	string Port	string Data	Reads a file and returns its contents
ReadAll	string Port	string Data	Reads a file and returns its contents
ListOpen	nil	string Data	Returns a lua table listing all active ports.

Library: File (file.)

A very basic but useful file IO class.

Name	Arguments	Return	Description
readAll	string FileName	string Data	Reads a file and returns its contents
writeAll	string Filename, string Data, bool NoOverwrite= true	nil (bool true)	Writes string to a file and closes it. Returns true if unsuccessful. Save to use with large files.

Lua Net library (udp.)

The UDP library adds a new pseudo type: "**socket_udp**". This object points to an internal socket. You may use tostring() on this type.

Overview

Name	Arguments	Return	Description
dumpUDP	string IP, short Port, string Data	bool Success	sends a UDP packet
open	short Port, bool reuseOnReload, function Callback(string Data, string IP)	socket_udp	Opens new UDP socket and returns it
close	short Port	nil	Closes a socket
close	socket_udp Port	nil	Closes a socket
list	nil	socket_udp[]	Returns a table with all active/open sockets