Luadch v0.08 Manual

by blastbeat

October 18, 2008

Contents

1 Installation **3**

1.1 Requirements 3

1.2 Luadch vs. Luadchjit 3

1.3 Windows Installation 3

1.3.1 Binary 3

1.3.2 Compiling 3

1.4 Linux/Unix Installation 4

1.4.1 Compiling 4

1.4.2 Installation 4

2 Configuration **4**

2.1 Getting started 4

2.2 Directory Overview 4

2.3 UTF8 Notes 5

2.4 The cfg.tbl 5

2.5 The user.tbl 5

2.6 Advanced configuration 6

2.6.1 Using ADCS 6

2.6.2 Using Scripts 6

2.6.3 Language Files 7

3 Running Luadch **7**

3.1 General Notes 7

3.2 Hub Commands 7

4 Scripting **8**

4.1 Conventions 8

4.2 Data Types 9

4.3 Listeners 9

4.4 Script Modules 9

5 Links **10**

1 Installation

1.1 Requirements

Luadch was tested on Windows XP, Windows Server 2003, Debian Etch x86/AMD64, FreeBSD 7.0 and Ubuntu Hardy. Luadchjit will only work on x86. The hardware requirements will depend on the usage of the hub; high user counts will require more CPU, RAM and bandwidth. For 100 users, the following should suffice:

- 128 MB RAM
- 500 MHz CPU
- broadband connection with 128 kbit/s upstream

1.2 Luadch vs. Luadchjit

Luadch uses Lua as basis and Luadchjit uses Luajit. Luajit is a JIT compiler for Lua, which can improve the performance a lot. But as mentioned, it runs only on x86. So without performance problems, I suggest to use Luadch because Luadchjit is only a branch. Anyway, core and scripts of both are the same, you don't have to change anything. More information about Lua and Luajit can be found here:

`http://www.lua.org`
`http://luajit.org`

1.3 Windows Installation

1.3.1 Binary

Download Luadch/jit binaries from Sourceforge:

`http://sourceforge.net/project/showfiles.php?group_id=214423`

Unzip the archive somewhere, no further installation is needed. Note: on Windows 2003 I got some errors like 'dll xy wasn't found'. In this case I suggest to copy the files from a Windows XP installation.

1.3.2 Compiling

Compiling Luadch on Windows will cost a bit more. A small how-to can be found here:

`http://luadch.sourceforge.net/forum/viewtopic.php?f=10&t=5`

1.4 Linux/Unix Installation

1.4.1 Compiling

Install Subversion (SVN), OpenSSL, libssl-dev and GCC. Checkout the latest Luadch source:

```
svn co https://luadch.svn.sourceforge.net/svnroot/luadch luadch
```

Change to luadch/trunk/luadch and use the following script for compiling:

```
compile_with_gcc.sh
```

The compiled hub can be found in the folder 'build_gcc'.

1.4.2 Installation

Because I am a dumb Windows user, Luadch does not provide a makefile or similar for installation on *nix. Feel free to send me a good one.

2 Configuration

2.1 Getting started

First of all, to run an ADC hub, you will need some basic knowledge about ADC. You should know what SID, PID, CID and TIGR means. Learn more about ADC here:

```
http://adc.sourceforge.net/ADC.html
http://www.adcportal.com/wiki/index.php/Main_Page
```

You don't have to read or understand the whole thing, but you should know the differences of the addressed issues. The next step would be some knowledge about the programming language Lua. It isn't necessary but very helpful, because Luadch is written in Lua.
Some useful links for Lua:

```
http://www.lua.org
http://lua-users.org
```

2.2 Directory Overview

- cfg/ - contains the configuration file (cfg.tbl), user database (user.tbl) and backups of both
- core/ - contains the basic hub logic
- scripts/ - contains optional user scripts
- scripts/cfg/ - contains optional cfg files for the scripts

- scripts/lang/ - contains optional lang files for the scripts
- log/ - contains log files
- lib/ - contains library files
- certs/ - contains your certificates
- lang/ - contains language files for the hub core

2.3 UTF8 Notes

All user files, scripts, settings in Luadch must be UTF8 encoded. So make shure to save your files in UTF8 mode, because at the moment Luadch will only complain about non UTF8 strings, but not convert them.

2.4 The cfg.tbl

In cfg/cfg.tbl you can find general settings for Luadch. The file is in fact a Lua script, which returns a Lua table (therefore cfg.tbl) The most important setting for the beginning are the ports. Change them in the default cfg.tbl, which should look like this:

```
return {

hub_hostaddress = "your.host.addy.org", -- your hostadress (string)
hub_website = "http://yourwebsite.org", -- your website (string)
hub_network = "your hubnetwork name", -- your hubnetwork (string)
hub_owner = "you", -- hubowner (string)

...

}
```

The settings can be changed according to their description. Luadch will check the file for syntax/type errors, but nothing more.

2.5 The user.tbl

In cfg/user.tbl you can find the user database of your hub. Similar to cfg.tbl, its a Lua script, which returns a Lua array. The default user.tbl should look like this:

```
return {

    { nick = "dummy", password = "test", rank = "2", level = "100" },
    { nick = "dummy2", password = "test", rank = "2", level = "100" },

}
```

Every line contains the information of a reg user. You should not edit that file manually. Use the hub commands like reg, delreg, upgrade instead. You should also delreg the default dummy users. Hub bots will be regged in user.tbl with a random password too.

2.6 Advanced configuration

2.6.1 Using ADCS

First of all you need some general information about SSL:

```
http://en.wikipedia.org/wiki/Transport_Layer_Security
http://www.openssl.org/docs/
```

To use SSL in hub - client connections, you have to do 2 things:

- create certs
- activate SSL in cfg.tbl

A example configuration for certificates can be found in certs/. Use make_cert.sh/bat to create some 'pem' files. In cfg.tbl, the key 'use_ssl' has to be true, and 'ssl_ports' needs an entry. The key 'ssl_params' must be a table according to the luasec specs, which can be found here:

```
http://www.inf.puc-rio.br/~brunoos/luasec/0.2/reference.html
```

2.6.2 Using Scripts

Luadch provides some prewritten scripts. In fact virtually all useful facilities of Luadch are scripts in the scripts/ directory. Without them, simple things won't work. That means also, when you miss a feature, you have to script it. Every script you want to use must be added to 'scripts' in cfg.tbl A script usually starts with a short description and a settings part. When you want to fully customize your hub, open every script you want to use and change the settings. Make sure to save everything in UTF8.
The first thing you have to think about are user levels. The default levels are:

```
levels = { -- your levels with level names (array of strings)

    [ 0 ] = "UNREG",
    [ 10 ] = "GUEST",
    [ 20 ] = "REG",
    [ 30 ] = "VIP",
    [ 40 ] = "SVIP",
    [ 60 ] = "OPERATOR",
    [ 80 ] = "ADMIN",
    [ 100 ] = "HUBOWNER",

}
```

You may also use 1000 profiles, but you should choose them carefully. To change them later can be very painful. You also have to edit the following keys in cfg.tbl:

```
reg_level = 20, -- min level to be a reg user (integer)
key_level = 40, -- min level to get a key (integer)
bot_level = 60, -- level of bots (integer); match this with your scripts
```

An important script you should check is the 'usr_ranks.lua': here you can define which level gets a key or is reg user in the client tag. With your profiles, you can step through the scripts and edit the permission tables.

2.6.3 Language Files

The lang/ and scripts/lang/ directories contain the language files of hub and scripts. At the moment only English and German is available. To change the language of the whole hub, use key 'language' in cfg.tbl. Default is 'en' for English. You can also change manually the language of every single script.

3 Running Luadch

3.1 General Notes

You can run Luadch in release or debug mode. Release mode means, no event messages are shown or logged, only script or errors messages according to the cfg.tbl. This makes the hub a lot of faster. To run luadch in release mode type

```
luadch -r
```

Debug mode is the default mode. According to your cfg.tbl also events are logged. This lets the event.log growing very large in short time.
To connect to your ADC/ADCS hub with DC++, connect via

```
adc://your.addy:port
adcs://your.addy:port
```

Often users forget 'adc' in the addy, and although connected, they cannot login.

3.2 Hub Commands

Type '+help' in the main chat to get all available according to your level. The most commands are also accessible as user commands. Important hub commands are:

```
setpas
Usage: [+!#]setpas sid|nick|cid <SID>|<nick>|<CID> <password>

ban
Usage: [+!#]ban sid|nick|cid <SID>|<nick>|<CID> [<time> <reason>]
```

```
unban
Usage: [+!#]unban ip|nick|cid <IP>|<nick>|<CID>

reg
Usage: [+!#]reg nick <nick> <password> <level>

delreg
Usage: [+!#]delreg nick|cid <nick>|<CID>

errors
Usage: [+!#]errors

reload
Usage: [+!#]reload

restart
Usage: [+!#]restart

shutdown
Usage: [+!#]shutdown
```

4 Scripting

Most of all important things about the scripting API can be found in docs/api.txt. To avoid repeating myself I only explain a bit in the following sections.

4.1 Conventions

I tried to introduce a certain scripting style in Luadch and the prewritten scripts for better readability and less errors. In general I handle core scripts more strict.
Some conventions of core/ :

- no global vars
- every function/module has to be imported with 'use' (to avoid globals)
- declaration of all locals with script wide scope at the top

Some conventions of scripts/ :

- no global vars
- name of the script should be as following: type_name.lua; for example 'cmd_reg.lua'

- a script should have at most one hub command. instead of writing a script with commands '+reladd' or '+reldel' use parameters like '+release add', '+release del' and name the script 'cmd release.lua'
- usage of language files which should stored in lang/ folder
- usage of script modules, for example 'ucmd', 'hubcmd', 'report', 'help'

You don't have to follow these suggestions, but for me they made life easier.

4.2 Data Types

Luadch has some custom data types, which are of course more symbolic and not really comparable with the build in data types of Lua. Some functions will do type checks during runtime, but don't count on it. You have to take care to pass the right data to the functions. You can find a list of all data types in docs/api.txt

4.3 Listeners

Scripts are usually listening for certain events. For these events they have to set listeners, which will be executed when the events happen. The general syntax to set a listener looks like this:

```
hub.setlistener( event, key, func )
```

This means set the function 'func' with identifier 'key' (which can be a table or an unique string) as listener to event 'event'. Of course every event passes other arguments in the listeners functions. An example for listing on main chat messages would be

```
hub.setlistener( "onBroadcast", { },
    function( user, adccmd, msg )
        hub.debug( msg )    -- prints the message in hub window
    end
)
```

You can find a list of all events in docs/api.txt

4.4 Script Modules

A nice feature in Luadch is the ability to export scripts as modules. So different scripts can use each other. One script needs to return something, and another can then import it. There are some prewritten script modules:

- 'cmd_help.lua'
- 'etc_report.lua'
- 'etc_hubcommands.lua'

- 'etc_usercommands.lua'

You can only import a module 'onStart', that means you have to set a listener for that event:

```
local help    -- no globals...

hub.setlistener( "onStart", { },
    function( )
        help = hub.import "cmd_help.lua"
        assert( help )
        do_something_with( help )
    end
)
```

5 Links

- Luadch project:

  `http://sf.net/projects/luadch`

- Luadch forum:

  `http://luadch.sf.net/forum`