LuCI on lighttpd

This article explains how to run the LuCI web interface on the lighttpd web server. LuCI is the main web administration utility for OpenWrt. By default it is installed with uHTTPd. You can of course use any other web server for LuCI. There are a couple available in the OpenWrt archives: http.overview This is the howto for lighttpd.

See also:

- http.lighttpd
- luci.essentials

Installation

The following procedure brings you a minimal LuCI-Installation (instead of the full installation that is obtained when using the meta-package "**luci**" or "**luci-ssl**", which also automatically installs the webserver "**uHTTPd**")

First, install lighttpd and LuCI packages:

```
opkg update
opkg install lighttpd lighttpd-mod-cgi luci-mod-admin-full libiwinfo
```

NOTE: The **libiwinfo** package is only needed if your router has wireless.

Then choose a LuCI theme

```
opkg install luci-theme-openwrt
```

or

```
opkg install luci-theme-bootstrap
```

Afterwards check that `/etc/lighttpd/conf.d/10-cgi.conf` loads the **mod_cgi**-module:

```
server.modules += ( "mod_cgi" )
```

Now we need to tell lighttpd to process requests for the web interface using Lua. The LuCI administation package installs a file `/cgi-bin/luci`, which is the default CGI gateway for LuCI. This is a script (with shebang line) that can run LuCI independently and calls Lua by itself. To tell lighttpd that it needs to load everything starting with `/cgi-bin/luci` by that script simply add

```
"cgi-bin/luci" => ""
```

to the cgi configuration file (`/etc/lighttpd/conf.d/10-cgi.conf`):

```
cgi.assign = ( ".pl" => "/usr/bin/perl", ".cgi" => "/usr/bin/perl", ".rb" => "/usr/bin/ruby", ".erb" => "/usr/bin/eruby", ".py" =>
"/usr/bin/python", "cgi-bin/luci" => "" )
```

This makes LuCI work as a CGI process. See http://redmine.lighttpd.net/projects/1/wiki/docs_modcgi [http://redmine.lighttpd.net/projects/1/wiki/docs_modcgi] for an explanation of the `cgi.assign` syntax. Note that the index.html file in the document root `/www` that comes with the LuCI package redirects to `/cgi-bin/luci`, thereby allowing access to the web interface by just loading the address of your OpenWrt device (e.g. 192.168.1.1 instead of having to load 192.168.1.1/cgi-bin/luci).

If you want lighttpd's logs in the output of "**logread**", add this:

```
## enable syslog server.errorlog-use-syslog = "enable"
```

Enable lighttpd (so that it will be started at every boot) and start it:

```
/etc/init.d/lighttpd enable
/etc/init.d/lighttpd start
```

Add the following for the server to listen on ipv4 and ipv6

```
#Listen on ipv4
$SERVER["socket"] == ":80" {
}
#Listen on ipv6
$SERVER["socket"] == "[::]:80" {
        server.use-ipv6 = "enable"
}
```

LuCI and Another Website Simultaneously on lighttpd (Listening on Multiple Ports)

You can easily add other content to the document root and use it as your website. For example, you might aggregate the web interfaces of other OpenWrt programs under the same socket listening instance in lighttpd. However, you may want to have the web interface LuCI listening on another socket altogether. This is useful for example if you want to run a website next to the LuCI interface on another port. Or if you want to run an SSL website next to a regular one. Another reason is to prevent people that visit your website to see the LuCI log in screen, if they happen to access the document root, which normally accesses LuCI. (Another way to just prevent this, is to hide it a bit more by not serving LuCI on the docroot by removing the redirect HTML file explained above. And instead of putting the CGI script under `/cgi-bin/luci` you might come up with a more secret path for that as well.)

Listening on a second port with its own document root is very straigtforward in lighttpd. Here we consider the case where we run a simple static website next the LuCI web interface.

Suppose we make a directory for the website that we want to run next to LuCI and put some content in it:

```
mkdir /website
echo "<P>It works!</P>" > /website/index.html
```

Now we simply need to use lighttpd's "matching on socket" mechanism (see the documentation on conditional configuration [http://redmine.lighttpd.net/projects/1/wiki/Docs_Configuration#Conditional-Configuration]), for which you have to put the following somewhere in your lighttpd.conf:

```
$SERVER["socket"] == ":80" { server.document-root = "/website/" }
```

After restarting lighttpd your website is on port 81 and LuCI is still on port 80 using lighttpd. Naturally, you can interchange these two ports such that the website is on the default HTTP port 80 and set the web interface on port 81.

You can of course also add other statements to this conditional configuration (conditional on listening and serving requests on another port), such as another CGI package or a different PHP configuration. For more information, see the following from the lighttpd website:

- http://redmine.lighttpd.net/projects/1/wiki/docs_modcgi [http://redmine.lighttpd.net/projects/1/wiki/docs_modcgi]
- http://redmine.lighttpd.net/projects/1/wiki/Docs_Configuration [http://redmine.lighttpd.net/projects/1/wiki/Docs_Configuration]
- http://redmine.lighttpd.net/projects/1/wiki/Docs_SSL [http://redmine.lighttpd.net/projects/1/wiki/Docs_SSL]
- http://redmine.lighttpd.net/projects/lighttpd/wiki/HowToRedirectHttpToHttps/9 [http://redmine.lighttpd.net/projects/lighttpd/wiki/HowToRedirectHttpToHttps/9]

Remark: Instead of opening a 2nd port for lighttpd it should be possible to activate the module "**mod_simple_vhost**" and serve your website (in a vhost) in friendly coexistence of LuCI via port 80. Please document this here if you get it to work.

> *Please document this here if you get it to work.*

Some hints for achieving "friendly coexistence" of luci, ssl and php-based website, but without using mod_simple_vhost: First, file layout (notice the luci → /www links):

```
root@OpenWrt / # tree /srv/www
/srv/www
|-- default
|   |-- example.org -> ../example.org
|   `-- luci -> /www
`-- example.org
    |-- index.html
    |-- luci -> /www
    `-- phpinfo.php
```

luci-related (and php-related) lines in /etc/lighttpd/lighttpd.conf

```
server.modules = (
        "mod_rewrite", # needed for luci
        "mod_fastcgi", # needed for php
        "mod_cgi" # needed for luci
)

# luci - no handler means invoking the file
cgi.assign = ( "cgi-bin/luci" => "" )

# php via socket-driven fastcgi.
# to get cgi-php working one needs to comment out (or update
# accordingly) doc_root and user_dir in /etc/php.ini
fastcgi.server = (
        ".php" =>
```

```
        ((
                "bin-path" => "/usr/bin/php-fcgi",
                "socket" => "/tmp/php.socket"
        ))
)

# port 80 is default, so this line can be commented out
server.port = 80

# this enables https
$SERVER["socket"] == "0.0.0.0:443" {
        ssl.engine = "enable"
        ssl.pemfile = "/etc/lighttpd/server.pem"
}

# I prefer to keep sites here
server.document-root = "/srv/www/default/"

# rewrites to keep luci code untouched
url.rewrite = ( "^/luci$" => "/luci/", # helper only
                "^/cgi-bin/luci.*" => "/luci$0",
                "^/luci-static/.*" => "/luci$0" )

# block to catch accessing luci via http and "kind" redirect to https
$HTTP["scheme"] == "http" {
        $HTTP["url"] =~ "^/(cgi-bin/)?luci" {
                url.redirect = (".*" => "https://%0$0" )
        }
}

# rules for lan access
$HTTP["host"] =~ "^10.0.0.1(\:[0-9]*)?$" {
        dir-listing.activate = "enable"
}

# rules for remote acces by example.org and www.example.org
$HTTP["host"] =~ "^(www.)?example.org(\:[0-9]*)?$" {
        server.document-root = "/srv/www/example.org/"
}
```

The "minimalistic" setup for putting own page in **/srv/www/** and getting luci by http://…/luci I expect to work is to execute

```
mkdir -p /srv/www
ln -s /www /srv/www/luci
```

and add to /etc/lighttpd/lighttpd.conf file the following:

```
server.document-root = "/srv/www/"
server.modules += ("mod_rewrite", "mod_cgi")
cgi.assign = ( "cgi-bin/luci" => "" )
url.rewrite = ( "^/luci$" => "/luci/",
                "^/cgi-bin/luci.*" => "/luci$0",
                "^/luci-static/.*" => "/luci$0" )
```
