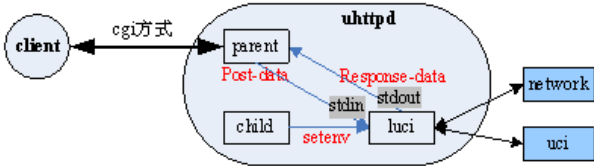


Luci实现框架

2013-05-14 22:01 by zmkeil, 12753 阅读, 7 评论, 收藏, 编辑

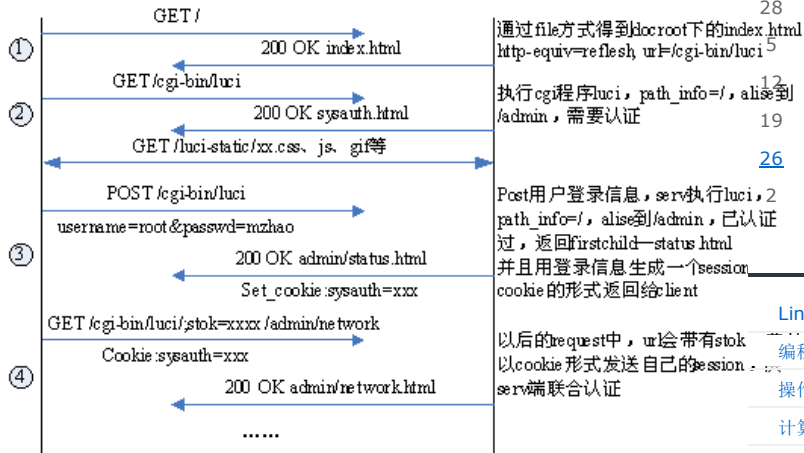
1. 总述

上一篇总结了uhttpd的工作方式，openwrt中利用它作为web服务器，实现客户端web页面配置功能。对于request处理方式，采用的是cgi，而所用的cgi程序就是luci，工作框架如下图所示：



Client端和serv端采用cgi方式交互，uhttpd服务器的cgi方式中，fork出一个子进程，子进程利用execl替换为luci进程空间，并通过setenv环境变量的方式，传递一些固定格式的数据（如PATH\_INFO）给luci。另外一些非固定格式的数据（post-data）则由父进程通过一个w\_pipe写给luci的stdin，而luci的返回数据则写在stdout上，由父进程通过一个r\_pipe读取。

下面的图描述了web配置时的数据交互：



- 1. 首次运行时，是以普通的file方式获得docroot/index.html，该文件中以meta的方式自动跳转到cgi的url，这是web服务器的一般做法。
- 2. 然后第一次执行luci，path\_info="/，会alise到/admin'（/会索引到tree.rootnode，并执行其target方法，即alise(/admin)，即重新去索引adminnode，这在后面会详细描述），该节点需要认证，所以返回一个登录界面。
- 3. 第3次交互，过程同上一轮的，只是这时已post来了登录信息，所以serv端会生成一个session值，然后执行/admin'的target（它的target为firstchild，即索引第一个子节点），最终返回/admin/status.html，同时会把session值以cookie的形式发给client。这就是从原始状态到得到显示页面的过程，之后主要就是点击页面上的连接，产生新的request。
- 4. 每个链接的url中都会带有一个stok值（它是serv生成的，并放在html中的url里），并且每个新request都要带有session值，它和stok值一起供serv

About

昵称: [zmkeil](#)  
园龄: [3年3个月](#)  
粉丝: [37](#)  
关注: [0](#)  
[+加关注](#)

SEARCH

最新评论

Re:Luci实现框架

您好，想请教一个问题，我想将Luci的admin-full下面的syslog显示功能移植到admin-mini，请问怎么实现？ -- zyzferrari

日历

随笔档案

2013年5月						
<	日	一	二	三	四	五
				1	2	3
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8						

<a href="#">2016年5月(2)</a>
<a href="#">2016年2月(1)</a>
<a href="#">2015年11月(1)</a>
<a href="#">2015年2月(1)</a>
<a href="#">2015年1月(1)</a>
<a href="#">2013年8月(3)</a>
<a href="#">2013年5月(9)</a>
<a href="#">2013年4月(13)</a>

随笔分类

- [Linux开发杂记\(4\)](#)
- [编程语言C/C++/JAVA\(5\)](#)
- [操作系统\(4\)](#)
- [计算机架构\(1\)](#)
- [算法\(2\)](#)
- [网络相关\(15\)](#)
- [信号处理DSP\(2\)](#)
- [有感而发\(4\)](#)

推荐排行榜

- [1. Linux下的虚拟Bridge实现\(4\)](#)
- [2. 网络嵌入式设备\(2\)](#)
- [3. 关于uC/OS的简单学习\(2\)](#)
- [4. Luci实现框架\(2\)](#)
- [5. uhttpd的实现框架\(2\)](#)

阅读排行榜

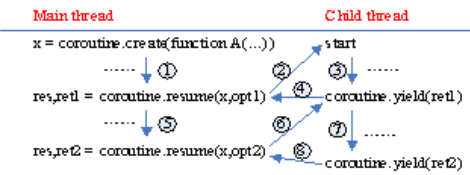
- [1. Luci实现框架\(12752\)](#)
- [2. uhttpd的实现框架\(4069\)](#)

端联合认证。

2.luci程序流程

前面已经说明了，luci作为web服务器的cgi程序，是通过execl函数替换到进程空间的，并且详细说明了它与其它进程的交互方法。另外上一节给出了初始阶段http报文，可以看到从第2次交互开始，所有request都是cgi方式（除一些css、js等资源文件外），且执行的cgi程序都是luci，只是带的参数不同，且即使所带参数相同（如都是'/」），由于需要认证，执行的过程也是不同的。

正是由于多种情况的存在，使得luci中需要多个判断分支，代码多少看起来有点乱，但openwrt还是把这些分支都糅合在了一个流程线中。下面首先给出整体流程，首先介绍一下lua语言中一个执行方式coroutine，它可以创造出另一个执行体，但却没有并行性，如下图所示，每一时刻只有一个执行体在执行，通过resume、yield来传递数据，且数据可以是任意类型，任意多个的。



Luci正是利用了这种方式，它首先执行的是running()函数，其中create出另一个执行体httpdispatch，每次httpdispatch执行yield返回一些数据时，running()函数就读取这些数据，做相应处理，然后再次执行resume(httpdispath)，.....如此直到httpdispatch执行完毕，如下图所示：

```
luci_running()
r= luci_http.Request(
    luci_sys.getenv      //获得环境变量
    limitsource(io.stdin) //读取post数据
    ....)
x= coroutine.create(httpdispatch,r)
local hcache= ""
local active=true
while(status(x)~=dead) {
    resid,data1,data2 = resume(x,r)
    if active then
        if #1 then
            io.write(status,data1,data2) //http res-line
        if #2 then
            hcache += data1:data2 //准备header
        if #3 then
            io.write(hcache) //写header、blank
            io.write( 'r\n' ) //默认到tdout
        if #4 then
            io.write(data1) //写body
        if #5 then
            io.flush()
            io.close()
            active = false
        if #6 then
            ...
        }
    }
}
```

```
httpdispatch() //参数都放在context
pathinfo = getenv(PATH_INFO)
//形如 /stok=xxxx/admin/network
get_stok,put_into_context
dispatch(pathinfo)

dispatch()
local c = context.tree
1 if not c then
    c = createtree()
    parse the pathinfo, and get node
2 if node.index then //需要显示的node
    init template //初始化模板
3 if tracksysauth
    authsession 认证
4 get node target
    copcal(targetargs)
```

如上图所示，其实luci真正的主体部分正是dispatch，该函数中有多个判断分支，全部糅合在一起，代码比较烦，总体上有4个部分，下面对它们进行一些描述。

首先说明一下代码组成，在openwrt文件系统中，lua语言的代码不要编译，类似一种脚本语言被执行，还有一些uhttpd服务器的主目录，它们是：

- /www/index.html
- /cgi-bin/luci
- /luci-static/xxx/xx.css、js、gif
- /usr/lib/lua/nixio.so、uci.so
- /luci/http.lua、dispatcher.lua、core...

3. Linux下的虚拟Bridge实现(3963)

4. OpenWRT平台搭建及简单应用(3162)

5. Linux下VLAN功能的实现(1967)

```
/controller/xxx.lua

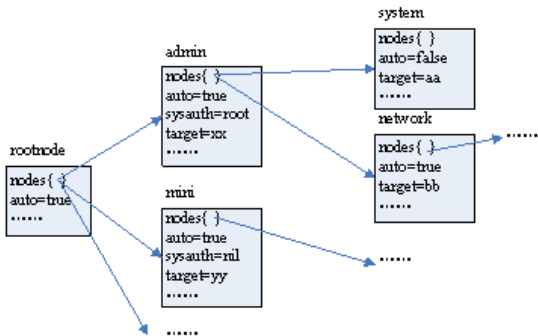
/model/xxx.lua

/view/xxx.lua
```

2.1 节点树node-tree

在controller目录下，每个.lua文件中，都有一个index()函数，其中主要调用entry()函数，形如entry(path,target,title,order)，path形如{admin,network,wireless}，entry()函数根据这些创建一个node，并把它放在全局node-tree的相应位置，后面的参数都是该node的属性，还可以有其他的参数。其中最重要的就是target。

Createtree()函数就是要找到controller目录下所有的.lua文件，并找到其中的index()函数执行，从而生成一个node-tree。这样做的io操作太多，为了效率，第一次执行后，把生成的node-tree放在/tmp/treecache文件中，以后只要没有更新（一般情况下，服务器里的.lua文件是不会变的），直接读该文件即可。生成的node-tree如下：



这里要注意的是，每次dispatch()会根据path\_info逐层索引，且每一层都把找到的节点信息放在一个变量track中，这样做使得上层node的信息会影响下层node，而下层node的信息又会覆盖上层node。比如{/admin/system}，最后的auto=false，target=aa，而由于admin有sysauth值，它会遗传给它的子节点，也即所有admin下的节点都需要认证。

2.2target简介

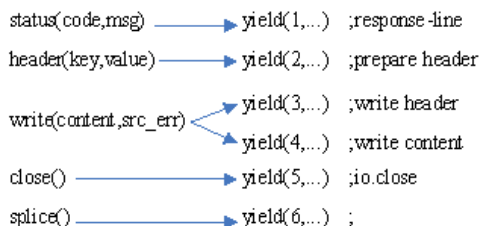
对每个节点，最重要的属性当然是target，这也是dispatch()流程最后要执行的方法。target主要有：alise、firstchild、call、cbi、form、template。这几个总体上可以分成两类，前两种主要用于链接其它node，后一个则是主要的操作、以及页面生成。下面分别描述。

链接方法：在介绍初始登录流程时，已经讲到了这种方法。比如初始登录时，url中的path\_info仅为 '/'，这应该会索引到rootnode节点。而该节点本身是没有内容显示的，所以它用alias('admin')方法，自动链接到admin节点。再比如，admin节点本身也没有内容显示，它用firstchild()方法，自动链接到它的第一个子节点/admin/status。

操作方法：这种方法一般用于一个路径的叶节点leaf，它们会去执行相应的操作，如修改interface参数等，并且动态生成页面html文件，传递给client。这里实际上是利用了所谓的MVC架构，这在后面再描述，这里主要描述luci怎么把生成的html发送给client端。

Call、cbi、form、template这几种方法，执行的原理各不相同，但最终都会生成完整的http-response报文（包括html文件），并调用luci.template.render()，luci.http.redirect()等函数，它们会调用几个特殊的

函数，把报文内容返回给luci.running()流程。



如上图所示，再联系luci.running()流程，就很容易看出，生成的完整的http-response报文会通过io.write()写在stdout上，而uhttpd架构已决定了，这些数据将传递给父进程，并通过tcp连接返回给client端。

### 3.sysauth用户认证

2.1节已描述了，由于节点是由上而下逐层索引的，所以只要一个节点有sysauth值，那么它所有的子节点都需要认证。不难想象，/admin节点有sysauth值，它下的所有子节点都是需要认证才能查看、操作的；/mini节点没有sysauth值，那么它下的所有子节点都不需要认证。

luci中关于登陆密码，用到的几个函数为：

```

user.setpasswd(username,passwd)  → os.execute(passwd username)
user.getpasswd(username)         → getspnam()/getpwnam()
user.checkpasswd(username,passwd) → crypt(pass,pw1)
  
```

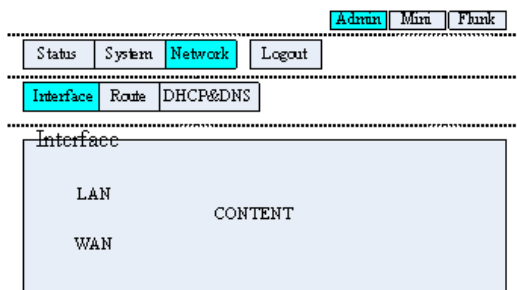
可以看出它的密码是用的linux的密码，而openwrt的精简内核没有实现多用户机制，只有一个root用户，且开机时自动以root用户登录。要实现多用户，必须在web层面上，实现另外一套（user、passwd）系统。

另外，认证后，serv端会发给client一个session值，且它要一直以cookie的形式存在于request报文中，供serv端来识别用户。这是web服务器的一般做法，这里就不多讲了。

### 4.MVC界面生成

这其实是luci的精华所在，第二节开始介绍/usr/lib/lua/luci/下有三个目录model、view、controller，它们对应M、V、C。第2.2节介绍了生成的界面怎么传递给client，下面简单介绍生成界面的方法。

Call()方法会调用controller里的函数，主要通过openwrt系统的uci、network、inconfig等工具对系统进行设置，如果需要还会生成新界面。动态生成界面的方法有两种，一是通过cbi()/form()方法，它们利用model中定义的模板map，生成html文件；另一种是通过template()方法，利用view中定义的htm（一种类似html的文件），直接生成界面。



上面的标题是由node-tree生成的，下面的内容由每个node通过上面的方法来动态生成。这套系统是很复杂的，但只要定义好了，使用起来就非常方

法，增加页面，修改页面某个内容等操作都非常简单。

好文要顶

关注我

收藏该文

zmkeil

关注 - 0

粉丝 - 37

2

0

+ 加关注

« 上一篇: [uhttpd的实现框架](#)  
» 下一篇: [关于uC/OS的简单学习](#)

分类: [网络相关](#)

#1楼 H浪花

2013-07-09 19:42

[ADD YOUR COMMENT](#)

还关于这个的文章吗? 我要做一个socket通信的程序进去.

支持(0) 反对(0)

#2楼 [楼主] zmkeil

2013-07-30 16:45

@ H浪花

不好意思! 之前一段时间忙外出. 没有了, 这个方案公司最终未采用, 所以没去深究.

支持(0) 反对(0)

#3楼 H浪花

2013-07-31 09:28

@ zmkeil

可惜了,国内难找到探究LuCI的人....不容易

支持(1) 反对(0)

#4楼 [楼主] zmkeil

2013-07-31 14:36

@ H浪花

恩, 可能因为luci是即写即用的, 不符合商业产品安全性的要求, 一般公司里面不会用这套架构

支持(0) 反对(0)

#5楼 石斋

2014-03-07 09:27

@ H浪花

是呀, 这段时间也在搞这个东西, 感觉里面的代码特别乱, 没有方向感觉.

支持(0) 反对(0)

#6楼 H浪花

2014-03-07 10:07

@ 石斋

复杂的功能好像很难实现, 比如socket, ....

支持(0) 反对(0)

#7楼 zyzferrari

2015-09-30 15:14

您好, 想请教一个问题, 我想将Luci的admin-full下面的syslog显示功能移植到admin-mini, 请问怎么实现?

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问网站首页](#)。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库  
【推荐】融云即时通讯云—豆果美食、Faceu等亿级APP都在用

【推荐】报表开发有捷径：快速设计轻松集成，数据可视化和交互

【推荐】一个月仅用630元赚取15000元，学会投资

【推荐】阿里舆情首次开放，69元限量秒杀



最新IT新闻：

- 华为企业云发布一年考
  - 大老板的焦虑、寂寞和人才困境
  - 穷游网十二年，一个老社区的演变和它的新生意
  - 微软推出Android测试版Flow自动化事务处理应用
  - IM企业热衷推出实体商品：Slack开售美式纹身贴纸
- » 更多新闻...



最新知识库文章：

- 程序猿媳妇儿注意事项
  - 可是姑娘，你为什么要编程呢？
  - 知其所以然（以算法学习为例）
  - 如何给变量取个简短且无歧义的名字
  - 编程的智慧
- » 更多知识库文章...