

uhttpd的实现框架

2013-05-14 21:58 by zmkeil, 4069 阅读, 0 评论, 收藏, 编辑

uhttpd是一个简单的web服务器程序，以前没怎么接触过，所以这里主要是对web服务器设计的一些学习总结。Openwrt系统中，真正用到的（需要了解的），其实不多，主要就是cgi的处理，包括与cgi程序的信息交互等，最后一节详细描述一下。

1.HTTP协议概述

HTTP协议是目前互联网使用最广泛的应用层协议。其协议框架很简单，在一个TCP连接中，以一问一答的方式进行信息交互。具体讲，就是客户端（如常见的浏览器）connect服务端的知名端口（通常是80），建立一个TCP连接，然后发送一个request；服务器端对该request解析后，发回相应的response应答，并关闭TCP连接。这就是一次交互，之后客户端再有请求，则重复上面的过程。

交互报文格式如下图所示：

request-line	Request-type	url	version	response-line	Version	code	phrase
	Data: Sat,2013...				Data Sat,2013...		
headers	Cookie: sysauth				Cookie: sysauth		
	.....				.....		
隔离行	Blank line				Blank line		
数据	Body-data				Body-data		

Request报文首行为request-line，其中type有GET、POST、HEAD三种方式，然后最重要的是url，它告诉服务器所请求的资源。Response报文首行为response-line，其中最重要的是code，它告知客户端响应情况（found、redirect、error等），然后跟一个简单的可读的短语。

两种报文后面具体的内容格式差不多，都是一些headers（其中冒号前的str指明header类型），然后以一个空行标识header结束，后面是数据。对于request，只有POST类型的请求需要提交数据，其它类型的是没有数据的。Response报文的数据就是url所指定的资源文件（html、doc、gif等）。

2.服务器架构

Uhttpd作为一个简单的web服务器，其代码量并不多，而且组织结构比较清楚。和其它网络服务器差不多，其main函数进行一些初始化（首先parse config-file，然后parse argv），然后进入一个循环，不断地监听，每当有一个客户请求到达时，则对它进行处理。

对于web服务器，所要做的处理主要就是分析url，判断出是file-request、cgi-request或lua-request，这主要是根据url的最前面的字符串（称为前缀prefix）得出的；然后就用相应的形式进行处理。如下图所示：

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

< 2013年5月 >						
日	一	二	三	四	五	六
28	29	30	<a href="#">1</a>	<a href="#">2</a>	3	4
5	6	7	8	9	10	11
12	13	<a href="#">14</a>	15	16	17	18
19	20	<a href="#">21</a>	22	23	24	<a href="#">25</a>
<a href="#">26</a>	27	28	29	30	31	1
2	3	4	5	6	7	8

随笔档案

- [2016年5月\(2\)](#)
- [2016年2月\(1\)](#)
- [2015年11月\(1\)](#)
- [2015年2月\(1\)](#)
- [2015年1月\(1\)](#)
- [2013年8月\(3\)](#)
- [2013年5月\(9\)](#)
- [2013年4月\(13\)](#)

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\(4069\)](#)

- 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)

```
uhttpd_main
init
.....
While(run) {
    // uh_mainloop
    select(max_fd+1, &read_fds,...)
    if(listenfd)
        listen new connect prepare new connection
    if(clientfd) {
        //已连接的client, 发来request data
        req= uh_http_header_rcv(cl) //获得request data
        if(uh_path_match(lua_prefix,url))
            lua_request(clreq,lua_state)
        else if(uh_path_match(cgi_prefix,url))
            uh_cgi_request(clreq,in,ipr)
        else
            uhfile_request(clreq,in)
        close(cur_fd) //关闭本次tcp连接
    }
    .....
}
```

采用select轮询模式，而非fork并发模式，适用于这种小型的服务器

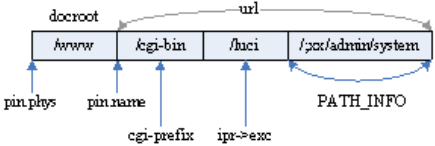
主要根据url\_prefix来判断采用什么方式来处理该request。uhttpd主要有三种：lua、cgi、file。最常见的就是file请求；lua是uhttpd为了使用lua语言开发而使用的一种特殊的处理方式，不具备通用性；cgi是指用另一个程序去处理该请求，并返回响应信息，由uhttpd回复给客户，这是web服务中的一种通用方式，openwrt中正是使用的这种方式。

3.cgi-response流程

前面已提到，openwrt系统中使用的uhttpd服务，主要是用cgi方式来回应客户请求的，下面就对这种方式详细阐述。

3.1url解析

由上图红色字所示，uh\_cgi\_request需要两个二外的参数pathinfo和interpreter，其中pin是一个struct，包含了路径中各种有用信息；ipr指明所用的cgi程序，因为一个服务器中可以有多个cgi程序。



如图所示，docroot是服务器的资源目录，是为了os准确定位资源位置，由uhttpd的config文件设定，如openwrt中为/www。后面的是client传来的url，开头的为cgi-prefix，也是有uhttpd的config文件设定的，它指明serv端采用cgi处理方式，如openwrt中的为/www/cgi-bin；紧接着的是cgi的程序名，它指明了使用哪个cgi程序；再后面就是实际的path信息了，在cgi方式中，它会被当成参数供cgi程序使用。

3.2cgi处理框架

要运行cgi程序，首先意味着需fork出一个子进程，并通过execl函数替换进程空间为cgi程序；其次，数据传递，子进程替换了进程空间后，怎么获得原信息，有怎么把回馈数据传输给父进程（即uhttpd），父进程又怎么接收这些数据。

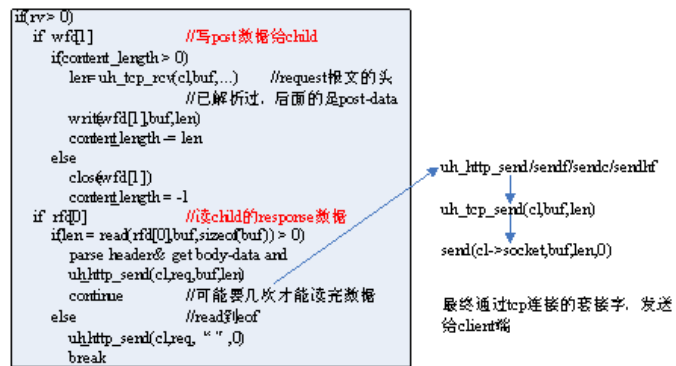
```
uh_cgi_request(clreq,in,ipr)
{
    int rfd[2],wfd[2];
    pipe(rfd);
    pipe(wfd);
    switch(kchild = fork())
    {
        case 0: //kchild
            close(rfd[0],wfd[1]);
            dup2(rfd[1],1) //stdout
            dup2(wfd[0],0) //stdin
            解析reqpinipr
            setenv(PATH_INFO)
            setenv(.....)
            exec(ipr->extu,ipr->extu,NULL)
        default: //parent
            close(rfd[1],wfd[0])
            if(req->method == POST)
                get_content_length_from_headers
                while(1) {
                    FD_SET(rfd[0],&reader)
                    FD_SET(wfd[1],&writer)
                    rv= select(fd_max+1,&reader,
                        (content_length>0)? &writer:null,null,&timeout)
                    if(rv == 0) //timeout
                        kill(kchild,0) //only kill,进入下次循环
                    else if(rv > 0)
                        break or again //要看处理完没
                    else //rv<0,waked up by signal
                        uhhttp_send(clreq, "", 0)
                        break
                }
    }
}
```

首先创建了两个pipe，这实际上是利用AF\_UNIX协议域，创建两个相连的socket\_unix，那么它们映射的文件描述符（即这里的fd[0]、fd[1]）就构成了一个pipe，且这种关系即使fork后也仍然存在，因为fork仅是增加文件的引用次数，而os维护的file结构和socket结构都没变，这就是父子进程间传递数据的方式。然后fork出一个子进程。

子进程中首先把两个管道的一端close，注意这仅是使得文件引用次数变为1。由于子进程待会要exec替换，替换后rfd、wfd就不存在了，因此先把它dup2给知名的stdin、stdout，这样即使exec替换后，ipt->extu程序可以以此来和父进程传递数据。另外，exec替换后，cgi程序仍需要之前的一些参数信息，如PATH\_INFO等，这种情况下，最简单的办法就是setenv，把需要的参数设为环境变量。

为什么要两个pipe，因为子进程向父进程传递回馈数据需要一个out-pipe，而若有post数据，子进程还需要一个in-pipe，从父进程读取post数据。

父进程中首先也是close，同上所述。若有post数据，先从httprequest-header中得到content-length，为后面传递给子进程做准备。然后进入一个循环（为什么要循环，什么时候退出，后面讲），通过select轮询io，超时、中断的情况就不看了，轮询的io一个是reader，即从子进程读取回馈数据，而若有post数据的话，还要另一个io，writer，向子进程写post数据。主要的处理就是上图中红色字所示，具体如下：



好文要顶

关注我

收藏该文





zmkeil

关注 - 0

粉丝 - 37

+ 加关注

2

0

- « 上一篇: [PPP协议体系的实现](#)
- » 下一篇: [Luci实现框架](#)

分类: [网络相关](#)

[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...



**90%的开发者选择极光推送**  
**不仅是集成简单、24小时一对一技术支持**

**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...