基于 WEB 的嵌入式监控系统

之

概要设计

Dev_Team	西邮 linux 兴趣小组嵌入式 Linux 组
Change	许振文
Members	许振文, 武婷婷，贾二群

M_num	M_date	M_version	M_part	M_reason
001	2008-11-14	0.01	所有	创建文档
002	2008-12-2	0.02	第 3，4，5 章	增加功能详细说明和总体说明
003	2009-03-10	0.03	第 3 章	增加 CGI 协议说明

注：

M_num：修改编号，从**001** 开始，执行“+**1**”操作

M_dat：修改日期

M_version：修改后版本

M_part：修改了那部分

M_reason：修改说明

目 录

第1章 导言

1.1 目的

该文档是对《基于WEB的嵌入式监控系统》的概要设计，针对每个模块的具体功能需求，从技术的角度给出设计思路和解决方案。为《基于WEB的嵌入式监控系统》的软件开发人员提供指导参考资料。

1.2 范围

该文档是基于开发人员对该系统行业业务需求及初步调查写出的，目的为客户提供完整的业务需求，为开发人员提供系统开发基础要求。

本文档的预期读者是：

1） 设计人员

2） 开发人员

3） 测试人员

1.3 缩写说明

1） ARM：一种嵌入式处理器指令集,这里代指具有这种指令集的处理器。

2） Server：在嵌入式设备上提供监控服务的程序和进程。

3） Client：用户浏览器。

4） HttpServer：运行在嵌入式设备上和提供WEB服务。

5） OS：操作系统。

1.4 参考资料

1. 超文本传输协议-HTTP1.1 RFC文档（中文，英文）
2. 软件文档国家标准(GB8567-88)
3. 属于西邮linux兴趣小组已发表的其它文件

第2章 系统架构

2.1 系统架构设计

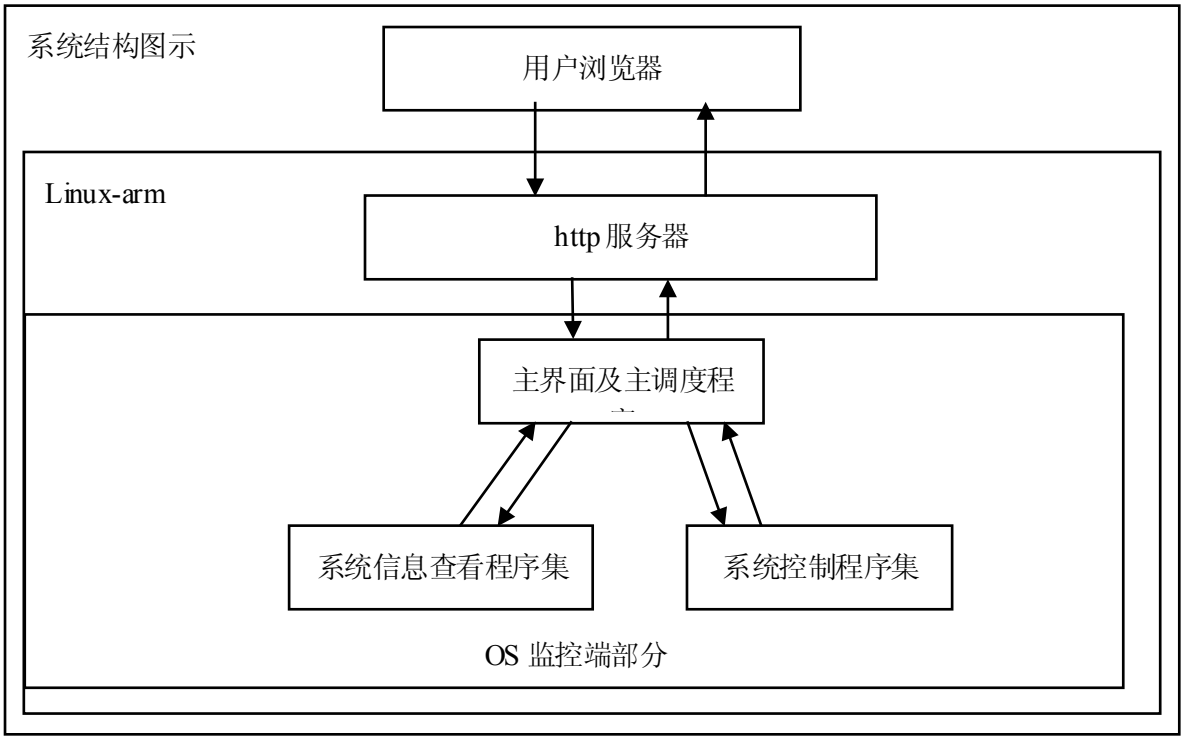
程序结构为B/S 模式，其中运行在开发主机上的是浏览器，命名为Client，运行在ARM开发板上的是服务器端，命名为 Server。所以整个监控系统的重点在于服务器端的开发。

整个系统服务器端的设计分为两部分：

A）HttpServer 部分

B）OS 监控端部分

2.2 系统整体结构划分图示



系统结构图示（图2-1）

第3章 HttpServer 部分

3.1 HttpServer 部分

该部分作为整个系统的核心部分，负责系统的整体调度，从上面的系统结构图中可以看出，这一部分是整个系统的中心调度部分。负责 HTTP 协议的解析和处理，会话管理，并且调用监控程序完成系统的监控。是和用户浏览器直接交互的一块。

首先能够处理一般的静态页面的处理，支持基本的get 和post 请求。接受浏览器的请求发送相关的 html 代码和图片到浏览器。

能够支持类似于 CGI 一样的处理，这样能够调用可执行程序，从而实现对系统的查看和控制。系统实现调用可执行的 c 程序或是 shell 程序，并将执行结果发送到客户端。

要内够支持简单的会话管理，识别是系统管理员还是一般用户，还是非法用户。

具备日志记录功能，能够将用户登录信息和操作信息以日志的形式记录到指定的日志文件。为系统管理提供依据。

在设计实现上可划分为一下几个模块：

1. 系统配置初始化模块
2. 系统服务连接初始化模块
3. HTTP 协议头处理模块
4. 监听/接受服务模块
5. 静态页面处理模块
6. CGI 程序处理模块

3.2 HttpServer 功能设计

HttpServer 部分主要是一个小巧的支持HTTP 协议的服务器。目前设计主要有以下几部分功能：

1） 一般静态页面处理

首先能够处理一般的静态页面的处理，支持基本的get 和post 请求。接受浏览器的请求发送相关的 html 代码和图片到浏览器。

2） 简单 CGI 支持

能够支持类似于 CGI 一样的处理，这样能够调用可执行程序，从而实现对系统的查看和控制。系统实现调用可执行的 c 程序或是 shell 程序，并将执行结果发送到客户端。

3） 会话管理

要内够支持简单的会话管理，识别是系统管理员还是一般用户，还是非法用户。

4） 日志记录

具备日志记录功能，能够将用户登录信息和操作信息以日志的形式记录到指定的日志文件。为系统管理提供依据。可以设定每个一定的时间(比如 10 天)删除记录的日志或者存储，以节省 ARM 系统资源。

5） 在线升级

能够在线进行软件人升级。设计是在有升级服务器的前提条件之下进行设升级。也可以进行文件上传的形式升级系统。

3.3 Http 协议分析

什么是HTTP协议？最简单的例子，就是你的浏览器与网页服务器之间使用的应用层协议。虽然官方文档说HTTP协议可以建立在任何可靠传输的协议之上，但是就我们所见到的，HTTP 还是建立在 TCP 之上的。

httpd最简单的 response 是返回静态的 HTML 页面。在该设计中不只要实现对静态网页的响应，还需要实现 CGI 特性。

首先要分析 HTPP协议。关于 HTTP协议的详细文档，可以参看RFC 文档rfc2616：http://www.w3.org/Protocols/rfc2616/rfc2616.html 。还有一个简单 HTTP 协议<HTTP Made Really Easy>：http://jmarshall.com/easy/http/ 。一下的分析和设计的依据都来自于此。

3.3.1 HTTP 协议结构

HTTP 协议无论是请求报文(request message)还是回应报文(response message)都分为四部分：

* 报文头 (initial line)
* 0个或多个 header line
* 空行(作为 header lines 的结束)
* 可选 body

HTTP协议是基于行的协议，每一行以\r\n作为分隔符。报文头通常表明报文的类型(例如请求类型)，报文头只占一行；header line

附带一些特殊信息，每一个header line 占一行，其格式为name:value，即以分号作为分隔；空行也就是一个\r\n；可选 body 通常

包含数据，例如服务器返回的某个静态 HTML 文件的内容。举个例子，以下是一个很常见的请求报文，你可以截获浏览器发送的数据包而获得：

```
 1 GET / HTTP/1.1
 2 Host: 127.0.0.1:8080
 3 User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.3) Gecko/2008092816 Icew      easel/3.0.3
(Debian-3.0.3-3)
 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
 5 Accept-Language: en-us,en;q=0.5
 6 Accept-Encoding: gzip,deflate
 7 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
 8 Keep-Alive: 300
 9 Connection: keep-alive
10
```

第 1 行就是 initial line，2-9 行是 header lines，10 行是一个 header line 的结束符，没有显示出来。

以下是一个回应报文：

```
1  HTTP/1.1 200 OK
2  Server: xiyouhttpd/0.1.0
3  Content-Type: text/html
4  Content-Length: 72
```

```
5
6  <head><head><title>index.html</title></head><body>index.html</body>
```

第6行就是可选的body，这里是index.html这个文件的内容。

3.3.2 HTTP request method

首先重点分析请求报文。先看initial line，该行包含几个字段，每个字段用空格分开，例如以上的GET /index.html HTTP/1.1就可以分为三部分：GET、/index.html、HTTP/1.1。其中第一个字段GET就是所谓的request method。它表明请求类型，HTTP有很多method，例如：GET、POST、HEAD等。

就针对于该系统的目标而言，只需要实现对GET、HEAD等几个常用的请求做响应即可。

GET是最普遍的method，表示请求一个资源。什么是资源？诸如HTML网页、图片、声音文件等都是资源。顺便提一句，HTTP协议中为每一个资源设置一个唯一的标识符，就是所谓的URI(更宽泛的URL)。

HEAD与GET一样，不过它不请求资源内容，而是请求资源信息，例如文件长度等信息。

initial line后面的内容：

对应于GET和HEAD两个method，紧接着的字段就是资源名，其实从这里可以看出，也就是文件名(相对于你服务器的资源目录)，例

如这里的/index.html；最后一个字段表明HTTP协议版本号。目前我们只需要支持HTTP1.1和1.0，没有多大的技术差别。

然后是header line。我们并不需要关注每一个header line。我只列出常用和必须的几个header line：

- Host：对于HTTP1.1而言，请求报文中必须包含此header，如果没有包含，服务器需要返回bad request错误信息。

- Date：用于回应报文，用于客户端缓存数据用。

- Content-Type：用于回应报文，表示回应资源的文件类型，以MIME形式给出。什么是MIME？它们都有自己的格式，例如：

  text/html, image/jpg, image/gif等。

- Content-Length：用于回应报文，表示回应资源的文件长度。

body域很简单，你只需要将一个文件全部读入内存，然后附加到回应报文段后发送即可，即使是二进制数据。

- 回应报文

上面已经给出了一个回应报文的例子，以其为例说明。首先是initial line，第一个字段表明HTTP协议版本，可以直接以请求报文为准(即请求报文版本是多少这里就是多少)；第二个字段是一个status code，也就是回应状态，相当于请求结果，请求结果被HTTP官方事先定义，例如200表示成功、404表示资源不存在等；最后一个字段为status code的可读字符串，你随便给吧。

回应报文中最好跟上Content-Type、Content-Length等header。

3.4 CGI的设计与实现

该部分主要是规定在本项目中的嵌入式web服务器下CGI 程序的开发规范和标准。主要是方便CGI 程序开发者更好，更方便的开发应用程序。应用CGI 程序均在程序文件夹“www”目录中。该部分主要叙述表中CGI 程序的编程规范。

CGI规定了Web 服务器调用其他可执行程序(CGI 程序)的接口协议标准。Web服务器通过调用CGI 程序实现和Web浏览器的交互。CGI 程序可以用任何程序设计语言编写,如Shell脚本语言、 Perl、Fortran、Pascal、C 语言等。但是用C 语言编写的CGI程序具有执行速度快、安全性高等特点。本小节主要分析用 C语言进行CGI 程序设计的方法、过程和技巧。

3.4.1 相关内容

CGI 接口标准包括标准输入、环境变量、标准输出三部分。

1） 环境变量

2） 标准输入

3） 标准输出

3.4.2 环境变量

操作系统提供了许多环境变量,它们定义了程序的执行环境,应用程序可以存取它们。Web服务器和CGI接口又另外设置了自己的一些环境变量,用来向CGI 程序传递一些重要的参数。CGI 的GET 方法还通过 环境变量QUERY-STRING向CGI 程序传递 Form 中的数据。

环境变量是文本串(名字/值对),可以被OS Shell 或其他程序设置 ,也可以被其他程序访问。它们是Web 服务器传递数据给CGI 程序的简单手段,之所以称为环境变量是因为它们是全局变量,任何程序都可以存取它们。

下面是CGI 程序设计中常常要用到的一些环境变量。

HTTP-REFERER:调用该CGI 程序的网页的URL。

REMOTE-HOST:调用该CGI 程序的 Web 浏览器的机器名和域名。

REQUEST-METHOD:指的是当 Web 服务器传递数据给 CGI 程序时所采用的方法,分为GET 和 POST 两种方法。GET 方法仅通过环境变量(如 QUERY-STRING)传递数据给 CGI 程序,而 POST 方法通过环境变量和标准输入传递数据给 CGI 程序,因此POST方法可较方便地传递较多的数据给 CGI 程序。

SCRIPT-NAME:该CGI 程序的名称。

QUERY-STRING:当使用 POST 方法时,Form 中的数据最后放在 QUERY-STRING 中,传递给 CGI 程序。

CONTENT-TYPE:传递给CGI程序数据的MIME类型,通常为″applica tion /x-www-form-url encodede″,它是从 HTML Form中以 POST方法传递数据给CGI 程序的数据编码类型,称为URL 编码类型。

CONTENT-LENGTH:传递给CGI程序的数据字符数(字节数)。

在C 语言程序中,要访向环境变量,可使用getenv()库函数。例如:

```
if (getenv (″CONTENT-LENGTH″))
n=atoi(getenv (″CONTENT-LENGTH″));
```

请注意程序中最好调用两次getenv():第一次检查是否存在该环境变量,第二次再使用该环境变量。这是因为函数getenv()在给定的环境变量名不存在时,返回一个NULL(空)指针,如

果你不首先检查而直接引用它,当该环境变量不存在时会引起 CGI 程序崩溃。

当用户提交一个HTML Form时,Web浏览器首先对Form中的数据以名字/值对的形式进行编码,并发送给 Web服务器,然后由 Web服务器传递给 CGI 程序。其格式如下:

name1=value1&name2=value2&name3=value3&name4=value4&...

其中名字是Form 中定义的INPUT、SELECT或TEXTAREA 等标置(Tag)名字,值是用户输入或选择的标置值。这种格式即为URL编码,程序中需要对其进行分析和解码。要分析这种数据流,CGI 程序必须首先将数据流分解成一组组的名字/值对。这可以通过在输入流中查找下面的两个字符来完成。

每当找到字符=,标志着一个Form变量名字的结束;每当找到字符& ,标志着一个Form变量值的结束。请注意输入数据的最后一个变量的值不以&结束。

一旦名字/值对分解后,还必须将输入中的一些特殊字符转换成相应的ASCII字符。这些特殊字符是:

+:将+转换成空格符;

%xx:用其十六进制ASCII码值表示的特殊字符。根据值xx将其转换成相应的ASCII字符。

对Form变量名和变量值都要进行这种转换。

3.4.3 标准输入

CGI 程序像其他可执行程序一样,可通过标准输入(stdin)从 Web服务器得到输入信息,如 Form 中的数据,这就是所谓的向 CGI 程序传递数据的 POST 方法。这意味着在操作系统命令行状态可执行 CGI 程序,对 CGI 程序进行调试。POST方法是常用的方法,本文将以此方法为例,分析 CGI 程序设计的方法、过程和技巧。

3.4.4 标准输出

CGI 程序通过标准输出(stdout)将输出信息传送给 Web服务器。传送给 Web服务器的信息可以用各种格式,通常是以纯文本或者 HTML 文本的形式,这样我们就可以在命令行状态调试CGI 程序,并且得到它们的输出。

下面是一个简单的CGI 程序,它将 HTML 中 Form的信息直接输出到Web 浏览器。

```
#include <stdio.h>
#include <stdib.h>
main()
{
   int,i,n;
printf ("Contenttype:text/html\r\n\r\n");
n=0;
if(getenv("CONTENT-LENGTH"))
n=atoi(getenv(CONTENT-LENGTH"));
for (i=0;i<n;i++)
putchar(getchar());
putchar ('n');
```

```
fflush(stdout);
}
```

下面对此程序作一下简要的分析。

```
prinft ("Contenttype:text/html\r\n\r\n");
```

此行通过标准输出将字符串"Contenttype:text/html\r\n\r\n"传送给 Web 服务器。它是一个 MIME 头信息,它告诉 Web 服务器随后的输出是以纯 ASCII 文本的形式。请注意在这个头信息中有两个新行符,这是因为 Web 服务器需要在实际的文本信息开始之前先看见一个空行。

```
if (getenv("CONTENT-LENGTH"))
n=atoi (getenv("CONTENT-LENGTH"));
```

此行首先检查环境变量 CONTENT-LENGTH 是否存在。Web 服务器在调用使用 POST 方法的 CGI 程序时设置此环境变量,它的文本值表示 Web 服务器传送给 CGI 程序的输入中的字符数目,因此我们使用函数atoi() 将此环境变量的值转换成整数,并赋给变量n。请注意 Web 服务器并不以文件结束符来终止它的输出,所以如果不检查环境变量 CONTENT-LENGTH, CGI 程序就无法知道什么时候输入结束了。

```
for (i=0;i<n;i++)
putchar(getchar());
```

此行从 0 循环到(CONTENT-LENGTH-1)次将标准输入中读到的每一个字符直接拷贝到标准输出,也就是将所有的输入以 ASCII 的形式回送给 Web 服务器。

通过此例,我们可将 CGI 程序的一般工作过程总结为如下几点。

1.通过检查环境变量 CONTENT-LENGTH,确定有多少输入;

2.循环使用 getchar()或者其他文件读函数得到所有的输入;

3.以相应的方法处理输入;

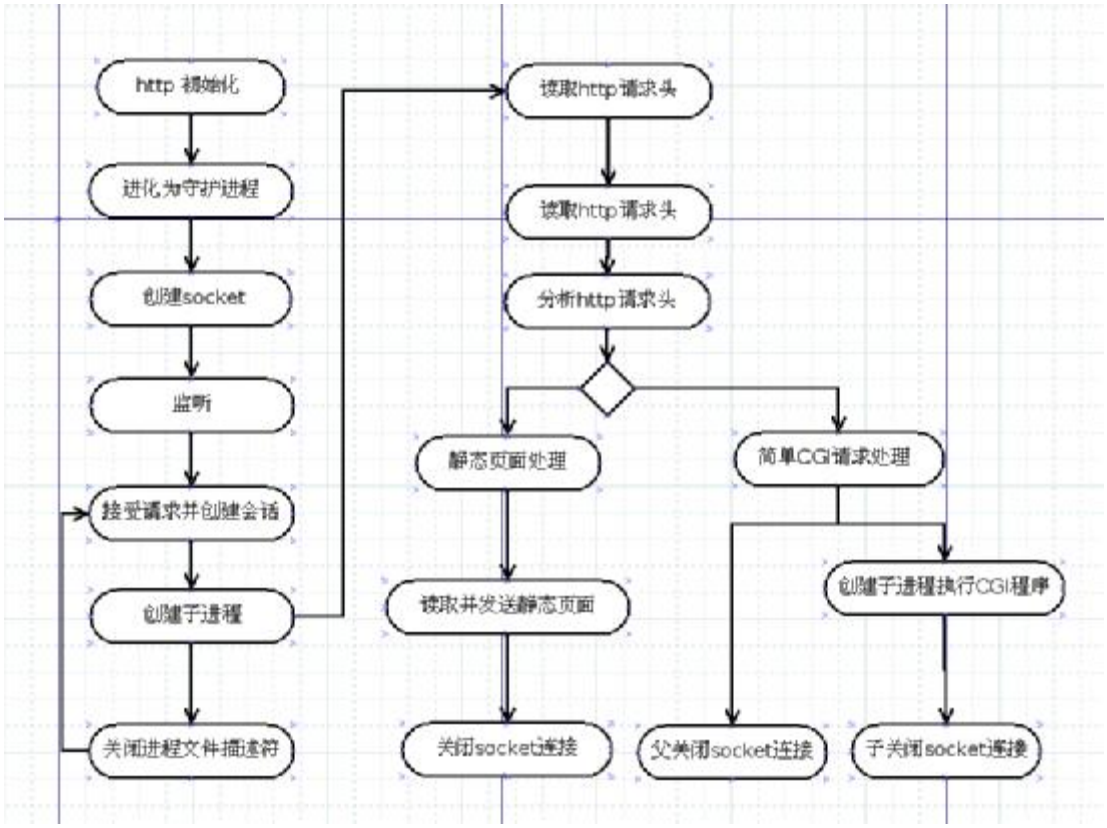
4.通过"Contenttype:"头信息,将输出信息的格式告诉 Web 服务器;

5.通过使用 printf()或者 putchar()或者其他的文件写函数,将输出传送给 Web 服务器。

总之,CGI 程序的主要任务就是从 Web 服务器得到输入信息,进行处理,然后将输出结果再送回给 Web 服务器。

3.5 Http 协议的设计实现

下面画出了 http 系统的内部调度过程。



HTTPD 系统的内部调度过程图（3-1）

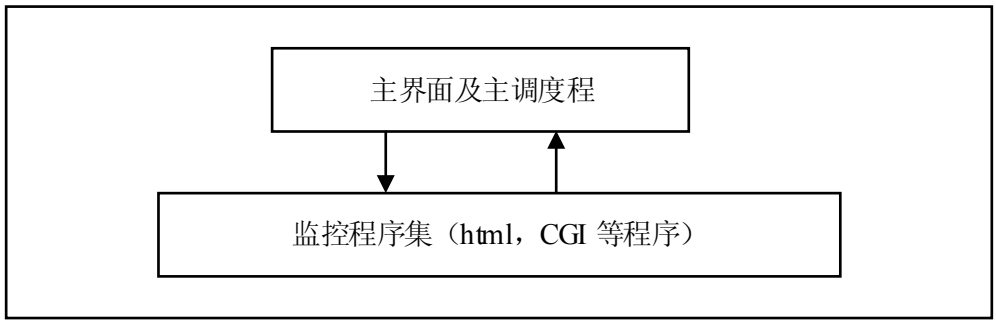
其中日志的记录贯穿于整个系统调度过程中，故没有单独列出来。

第4章 OS 监控端部分

该部分负责系统信息的收集和对系统实施控制,该部分程序现对比较庞大,也比较复杂。该部分正真实现系统信息的收集和系统控制命令的执行。是整个监控系统的最底层的部分,这部分相对分的比较细，根据具体功能实现的不同来写具体的模块

4.1 主控程序

根据“系统结构图示”，可以将系统 http 下面部分设计为可扩展的系统。让系统功能以插件的形式添加，并且开放系统插件开发协议与标准，最大限度的让更多的人来开发和丰富插件功能。



可扩展框架图示**(4-1)**

主要完成功能:

1） 系统主控程序初始化

该部分功能主要完成系统主控程序环境变量的初始化，

2） 系统功能插件查找

该部分功能主要实现系统插件的查找和识别。

3） 对系统插件分类识别

该部分功能主要是实现已找到插件的分类，类别会在扩展框架标准中给出。

4） 系统界面生成

5） 根据插件分类和功能需求生成相应的用户界面。

4.2 系统信息的查看

主要是完成对系统信息的收集，并将其返回给 HttpServer，再由 HttpServer 发送到请求的浏览器端。对应系统功能定义中的《系统信息浏览功能》部分。(见需求分析之 2.2-A）

根据查看信息的不同分为如下几个方面的设计:

6） 对系统文件信息的查看

该部分目的在于能够实现对系统文件系统的浏览，和文件属性的获取。是使用简单的CGI程序来完成。利用简单CGI 程序来封装成html 链接的形式，使的在浏览器上就像在本机一样的点击就可以查看文件信息。

1，在主界面上点击文件系统浏览后，浏览器会向 http 服务器发出请求并且传递一定的参数。

2，服务器先将参数设置为进程的环境变量，在执行相应的 CGI 程序。

3，CGI 程序会根据参数的不同显示不同目录中的相关文件的相关信息。

主要参考到的 shell 命令有：ls，cd，file 等。

主要用到的文件有：

7） 对系统磁盘信息的查看

对磁盘信息的查看也是用简单 CGI 来完成。主要是对磁盘分区情况，使用情况，挂载情况的查看。一图形化的形式来显示系统磁盘信息。

主要参考到的 shell 命令有：du，df，fdisk 等。

主要用到的文件有：

8） 对系统进程信息的浏览

对当前系统进程信息的查看，进程的运行情况。该部分的实现还是使用简单 CGI 来完成。

主要参考到的 shell 命令有：ps，top 等。

主要用到的文件有：

9） 对系统网络配置的查看

该部分主要是查看网络的配置信息，如 ip 信息，路由信息，dns 信息，网络服务信息。使用简单的 shell 程序来往成，最好只用一个 CGI 程序完成不同的需求。内够实现简单的网络状况诊断和当前网络服务情况的查看。

主要参考到的 shell 命令有：ifconfig，route，netstat，ping 等。

主要用到的文件有：

10） 对系统用户信息的查看

对用户信息的查看包括：系统当前的用户有那些，他们都有怎样的权限。当前有那些用户在登录，从什么地方登录，做什么事，使用了那些程序和服务。该部分的实现还是使用简单 shell 来完成。

主要参考到的 shell 命令有：w，who，whoami， groups，users，usermod 等。

主要用到的文件有：passwd，shadow，group，gshadow，

4.3 系统控制部分

该部分主要完成系统控制命令的执行，HttpServer 在接受浏览器端有效控制命令之后，HttpServer会调用该部分相应的控制模块去执行相应的控制操作。对应系统功能定义中的《系统控制功能》部分。(见需求分析之 2.2-B)

1） 对系统用户的管理

能够完成系统中用户的添加，删除，修改；用户组的添加，删除，修改。

主要参考到的 shell 命令有：useradd，userdel，groupdel，groupmod

主要用到的文件有：

2） 对系统进程的管理

进车管理主要是对进程的杀死和从起。

主要参考到的 shell 命令有：kill 等

主要用到的文件有：

3） 对系统文件系统的管理

对文件或是文件夹的创建，删除，拷贝，从命名等操作。还有文件的上传下载。

主要参考到的shell命令有：rm，rmdir，mv，cp等

主要用到的文件有：

4） 对其它系统设备管理

设备的启用，关闭，配置。

5） 对系统的其它管理

4.4 用户界面部分

Client端界面由于是浏览器，所以在服务端写程序时要求要写成以html代码输出，再由浏览器解释为图形界面。

该部分是显示页面的部分，该部分完成系统界面的生成。使用C程序或是shell程序编写，或者使用静态的html编写，对应系统功能定义中的《提供比较友好的用户界面》部分(见2.2-C-1)。

主功能界面设计采用经典的”工“字形结构设计。如下所示：

<table>
<tr><td colspan="2">logo 图标</td></tr>
<tr><td colspan="2">banner 主菜单</td></tr>
<tr><td>二级菜单
显示主菜单
下的二级菜单</td><td>显示菜单项对应的系统信息
显示控制系统的具体操作面板</td></tr>
<tr><td colspan="2">版权信息</td></tr>
</table>

4.5 系统其它管理与控制

1） 能够在线进行软件人升级。设计是在有升级服务器的前提条件之下进行设升级。也可以进行文件上传的形式升级系统。

第5章 Net-snmp 接口实现

5.1 Net-snmp 介绍

Simple Network Management Protocol (SNMP) 是一个被广泛使用的协议，可以监控网络设备（比如路由器）、计算机设备甚至是 UPS。NET-SNMP 是一种开放源代码的简单网络管理协议（Simple Network Management Protocol）软件。Net-SNMP 是用于实施 SNMP v1，SNMP v2，SNMPv3 的应用程序套件，可以使用在 IPv4、IPv6 的环境中。这个套件包括：

命令行程序包括：

从支持 SNMP 的设备中检索信息的命令。用于执行单个请求（snmpget,snmpgetnext），或者执行多个请求（snmpwalk,snmptable,snmpdelta）。

可以用于手动设置信息的命令（snmpset）。

检索一套固定信息的命令（snmpdf，snmpnetstat，snmpstatus）。

可以把MIB oid的信息在“数字”形式和“字符”形式之间进行转换的命令（snmptranslate），它还能显示 MIB 的内容和结构。

使用 Tk/perl 来提供一个图形化的 MIB 浏览器（tkmib）。

一个接收 SNMPtrap 信息的daemon。经过选择的snmp 通知信息可以被日志记录（记录在 syslog，或者 NT 的日志，或者文本文件），转发到另一个 SNMP 管理系统，或者传递到其它的程序。

一个可扩展的代理程序（snmpd），用于对管理系统提出的 SNMP 请求做出响应。这包括了内建的多种支持性：

支持广泛的 MIB 信息模块,可以使用动态加载的模块进行扩展,可以使用外部的脚本和命令进行扩展,对多路复用 SNMP（SMUX）和代理可扩展性协议（AgentX）的支持。

*包括一个库，用于支持对新的 SNMP 开发，支持 C 和 Perl API。

Net-SNMP 对于许多的 UNIX 和类 UNIX 操作系统都是支持的，也支持 windows。注意：对于不同的系统功能会有所变化。请阅读你所在平台的 README 文件。

Net-SNMP 是什么?

包含了以下多个支持 SNMP 协议的工具：

* 一个可扩展的代理
* 一个 SNMP 库
* 用于对 SNMP 代理进行查询操作和设置操作的工具
* 用于生成和处理 SNMP traps 的工具
* 一个支持 SNMP 的‘netstat'命令
* 一个基于 Perl/Tk/SNMP 的 MIB 信息浏览器

这个包最初基于卡耐基梅隆大学的 SNMP 开发工程（version 2.1.2.1）。

5.2 Net-snmp 移植

HTTP 服务器的设计思路为：

简单实用即可。对协议尽可能的简化，但也要符合标准。

系统控制部分的设计思路为：

设计为框架型，具体的功能实现以添加插件的方式来添加。

5.3 Net-snmp 扩展

5.4 httpserver 与 Net-snmp 接口实现

第6章 设计总结

6.1 设计总体思路

HTTP 服务器的设计思路为：

简单实用即可。对协议尽可能的简化，但也要符合标准。

系统控制部分的设计思路为：

设计为框架型，具体的功能实现以添加插件的方式来添加。

6.2 设计实现中的问题