WEBNET 用户手册

RT-THREAD 文档中心

WWW.RT-THREAD.ORG

Wednesday 30th January, 2019

目录

第 1 章

软件包介绍

WebNet 软件包是 RT-Thread 自主研发的，基于 HTTP 协议的 Web 服务器实现，它不仅提供设备与 HTTP Client 通讯的基本功能，而且支持多种模块功能扩展，满足开发者对嵌入式设备服务器的功能需求。

1.1 软件包目录结构

WebClient 软件包目录结构如下所示：

名称	说明
docs	文档目录
inc	头文件目录
src	源文件目录
module	功能模块文件目录
samples	示例文件目录
LICENSE	许可证文件
README.md	软件包使用说明
SConscript	RT-Thread 默认的构建脚本

1.2 软件包功能特点

WebNet 软件包提供丰富多样的功能支持，大部分功能以模块的形式给出，每个功能模块提供单独的控制方式且不同功能模块之间互不干扰，模块与主程序之间配合形成了一套功能齐全、高性能、高并发且高度可裁剪的网络服务器框架体系。

WebNet 软件包功能特点：

- **支持 HTTP 1.0/1.1**

WebNet 软件包用于在嵌入式设备中运行网络服务器功能，其功能实现基于 HTTP 协议，且同时支持 HTTP 1.0 和 HTTP 1.1 协议，提高网络服务器兼容性。

- **支持 CGI 功能**

CGI（Common Gateway Interface ）功能，是服务器运行时的外部程序规范，可以用于扩展服务器的功能。CGI 功能可以完成浏览器和服务器的交互过程。WebNet 软件包中的 CGI 功能，用户可以**自定义注册 CGI 执行函数**，服务器通过判断浏览器请求的 CGI 类型，执行相应的函数，并给予相应反馈信息。

- **支持 ASP 变量替换功能**

ASP（Active Server Pages）功能，实现的网页中变量替换的功能，即当浏览器访问的页面代码中有 <% %> 标记出现时（**支持包含.asp 后缀的页面文件**），将自动替换成**自定义注册的 ASP 执行函数**，并且给与相应反馈信息。使用此功能可以很方便地独立修改每个变量执行函数的实现方式，完成整个页面不同部分的改动，方便前后台开发分工合作。

- **支持 AUTH 基本认证功能**

在使用 HTTP 协议进行通信的过程中，HTTP 协议定义了 Basic Authentication 基本认证过程以允许 HTTP 服务器对 WEB 浏览器进行用户身份认证。基本认证功能为服务器提供简单的用户验证方式，其认证方式快速、准确，适用于对服务器安全性能有一定要求的系统和设备。WebNet 软件包中基本认证功能设计成与 URL 中目录相挂钩，在使用时可以**根据目录划分权限**。

- **支持 INDEX 目录文件显示功能**

WebNet 服务器开启之后，在浏览器输入对应的设备 IP 地址和目录名，就可以访问该目录，并列出该目录下所有文件的信息，类似于简单的文件服务器，可以用于服务器文件的查询和下载。

- **支持 ALIAS 别名访问功能**

ALIAS 别名访问功能，可以给文件夹设置多个别名，可以用于**长路径的简化操作**，方便用户对资源的访问。

- **支持 SSI 文件嵌入功能**

SSI（Server Side Include）功能，一般用于页面执行服务器程序或者插入文本内容到网页中。WebNet 中的 SSI 功能目前只支持插入文本内容到网页功能，页面代码中有 或者 标记存在时（**支持包含.shtml 、shtm 后缀的页面文件**），将自动解析成对应文件中的信息。通过 SSI 文件嵌入功能，可以使整个页面处理模块化，从而实现修改指定文件就可以完成页面的改动。

- **支持文件上传功能**

支持浏览器上传文件到 WebNet 服务器，可以自定义文件处理函数，以实现把上传的文件存储在文件系统中或直接写入目标存储器。利用此功能可以实现：

1. 上传资源文件到文件系统，如图片和网页；

2. 上传存储器镜像以烧写系统中的存储器，如 SPI FLASH；
3. 上传配置文件到系统以更新系统设置；
4. 上传固件并直接写入 FLASH 以实现固件更新功能。

- **支持预压缩功能**

预压缩功能用于提前压缩网页资源文件（**支持包含.gz 后缀的压缩文件**），降低对存储空间的需求，缩短文件系统读取时间，减少网络流量，提高网页加载速度。一般来讲，使用预压缩功能后，WebNet 服务器占用的存储空间仅为原来的 30%~40%。

- **支持缓存功能**

WebNet 缓存机制可以判断浏览器请求文件是否修改，从而决定是否发送完整文件内容到浏览器。WebNet 软件包缓冲机制分为如下三个级别：

1. level 0：关闭缓存功能，即浏览器每次都会从 WebNet 完整的读取文件内容；
2. level 1：开启缓存功能，WebNet 通过读取请求文件的最后修改的时间，如果和本地文件相同，则返回 304 通知浏览器文件并无更新，不会发送文件；
3. level 2：开启缓存功能，在原来判断修改时间的基础上，添加缓存文件有效时间支持，操作有效时间浏览器可重新访问该文件。

- **支持断点续传功能**

WebNet 服务器支持断点续传功能，即客户端在下载文件中途出现错误，再次下载时只需要提供给 WebNet 服务器前一次文件下载的偏移量，服务器将从指定文件偏移量发送文件内容给客户端，断点续传功能可以确保文件上传快速、准确，提高服务器运行效率。

1.3 软件包性能测试

测试环境：AM1808，主频 400M，DDR2 内存

测试环境	页面文件在 SD 卡上，DM9000AEP 外置网卡
静态页面请求	142 pages/s
CGI 页面请求	429 pages/s

测试环境	页面文件在 SD 卡上，EMAC 外置网卡
静态页面请求	190 pages/s
CGI 页面请求	567 pages/s

1.4 软件包资源占用

ROM：16 Kbytes

RAM：1.5 Kbytes + 1.1 Kbytes x 连接数

NOTE：以上数据为参考值，示例平台是 Cortex-M3，因为指令长度和框架不同，软件包资源占用有所不同。WebNet 在处理页面时，文件系统和模块内部也会有部分资源消耗。

第 2 章

示例程序

WebNet 软件包提供了一个综合的示例页面用于展示软件包的多项功能，包括：AUTH、CGI、ASP、SSI、INDEX、ALIAS、Upload 等功能。本章节主要介绍 WebNet 软件包中各个功能模块示例的使用方式。

示例文件

示例程序路径	说明
samples/wn_sample.c	综合示例代码
samples/wn_sample_upload.c	上传文件示例代码
samples/index.html	综合示例页面
samples/index.shtml	SSI 功能示例页面
samples/version.asp	ASP 功能示例页面

2.1 准备工作

2.1.1 软件包获取

- menuconfig 配置获取软件包和示例代码

打开 RT-Thread 提供的 Env 工具，使用 **menuconfig** 配置软件包。启用 WebNet 软件包，并配置使能测试例程配置（**Enable webnet samples**），如下所示：

```
RT-Thread online packages
    IoT - internet of things  --->
        [*] WebNet: A HTTP Server for RT-Thread
            (80)  Server listen port                    ## 服务器监听套接字端口号
            (16)  Maximum number of server connections  ## 服务器最大支持的连接数
            (/webnet)   Server root directory           ## 服务器根目录
                  Select supported modules  --->        ## 默认开启使用的功能模块
                     [ ] LOG: Enanle output log support
                     -*- AUTH: Enanle basic HTTP authentication support
```

```
                    -*- CGI: Enanle Common Gateway Interface support
                    -*- ASP: Enanle Active Server Pages support
                    -*- SSI: Enanle Server Side Includes support
                    -*- INDEX: Enanle list all the file in the directory support
                    -*- ALIAS: Enanle alias support
                    [ ] DAV: Enanle Web-based Distributed Authoring and Versioning
                        support
                    -*- UPLOAD: Enanle upload file support
                    [ ] GZIP: Enable compressed file support by GZIP
                    (0) CACHE: Configure cache level
            [*]   Enable webnet samples                ## 开启测试例程
                  Version (latest)  --->
```

- 使用 `pkgs --update` 命令下载软件包
- 编译下载

2.1.2 页面文件准备

WebNet 软件包示例中需要获取本地静态页面，需要文件系统的支持（FAT 文件系统，ROMFS 文件系统等，只需要支持 RT-Thread 的设备虚拟文件系统）。

静态页面需要上传到文件系统中服务器根目录下（示例中使用根目录为 /webnet）。设备挂载文件系统成功，需要依次执行下面操作：

1. 使用 `mkdir webnet` 命令创建 WebNet 软件包根目录 **/webnet**，并使用 `cd webnet` 命令进入该目录；
2. 使用 `mkdir admin` 和 `mkdir upload` 命令创建 **/webnet/admin** 和 **/webnet/upload** ，用于 AUTH 功能和 Upload 功能测试；
3. 将 WebNet 软件包 /sample 目录下的：**index.html**、**index.shtml**、**version.asp** 三个文件依次上传到设备 **/webnet** 目录（WebNet 根目录）中。（可以使用 TFTP 工具上传文件，具体操作方式参考 TFTP 使用说明)

创建目录和上传文件成功之后，就可以启动例程，测试 WebNet 软件功能。

2.2 启动例程

本例程参数和环境配置如下：

- 监听端口号：80
- 根目录地址：/webnet
- 文件系统：FAT 文件系统

设备启动，连接网络成功之后，在 Shell 命令行输入 `webnet_test` 命令启动 WebNet 服务器。查看 Shell 命令行，显示如下日志信息，说明 WebNet 服务器初始化成功：

```
msh />webnet_test
[I/wn] RT-Thread webnet package (V2.0.0) initialize success.
```

然后在 Shell 命令行中使用 `ifconfig` 命令获取本设备 IP 地址为 **192.168.12.29**。

```
msh />ifconfig
network interface: w0 (Default)
MTU: 1500
MAC: 44 32 c4 75 e0 59
FLAGS: UP LINK_UP ETHARP BROADCAST IGMP
ip address: 192.168.12.29
gw address: 192.168.10.1
net mask  : 255.255.0.0
dns server #0: 192.168.10.1
dns server #1: 223.5.5.5
```

接着在浏览器（这里使用谷歌浏览器）中输入设备 IP 地址，将默认访问设备根目录下 **/index.html** 文件，如下图所示，页面文件正常显示：

图 2.1: *例程主页面*

该页面上显示了 WebNet 软件包的基本功能介绍，并根据不同的功能给出相应的演示示例，下面将按顺序介绍如下几个例程：

- AUTH 基本认证例程
- CGI 事件处理例程
- ASP 变量替换例程
- SSI 文件嵌套例程
- INDEX 目录显示例程
- ALIAS 别名访问例程
- Upload 文件上传例程

2.2.1 AUTH 基本认证例程

在例程主页（/index.html）AUTH Test 模块下点击 基本认证功能测试：用户名及密码为 admin:admin 按键，弹出基本认证对话框，输入用户名 **admin**，密码 **admin**。成功输入用户名和密码后，会进入根目录下 /admin 目录，如下图所示流程：

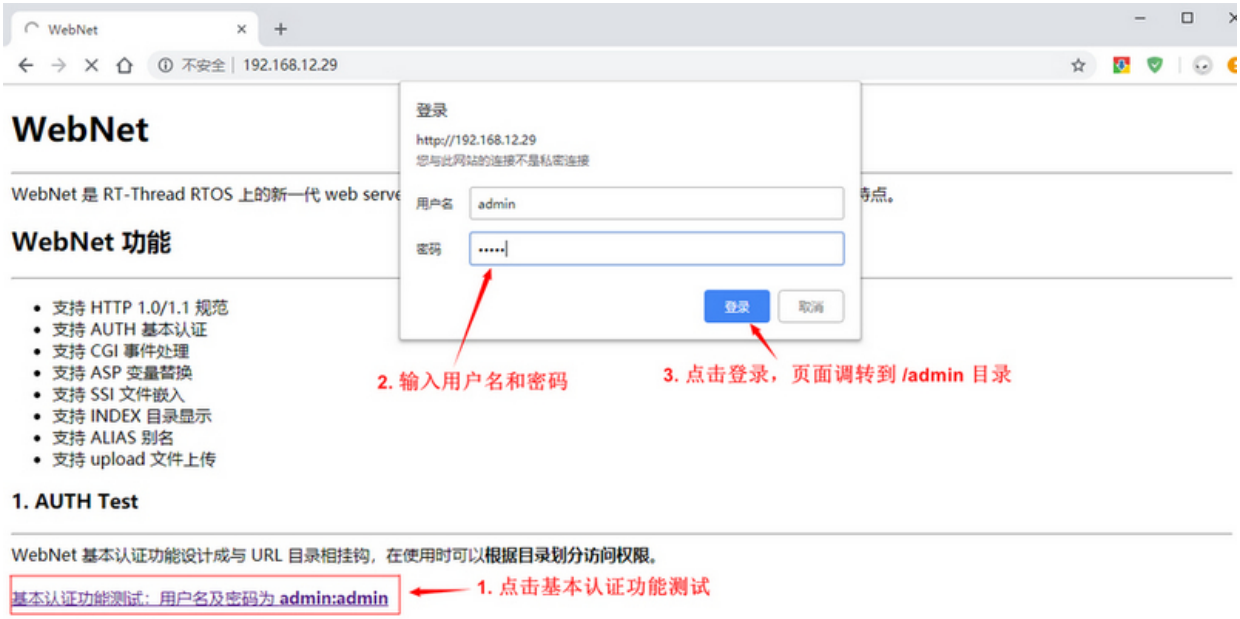

图 2.2: *基本认证例程*

2.2.2 CGI 事件处理例程

本例程提供两个 CGI 示例：**hello world 例程**和 **calc 例程**，用于演示 CGI 事件处理的基本功能。

- **hello world 例程**

hello world 例程演示了在页面演示文本内容功能，在例程主页 CGI Test 模块下点击 > hello world 按键，会跳转到新页面显示日志信息，新页面显示了 hello world 说明 CGI 事件处理成功，然后在新页面点击 Go back to root 按键回到例程主页面，如下图所示：

图 2.3: *hello world 例程*

- **calc 例程**

calc 例程中使用 CGI 功能在页面上展示了简单的加法计算器功能，在例程主页 CGI Test 模块下点击 `> calc` 按键，会跳转到新的页面，输入两个数字点击 `计算` 按键，页面会显示两数字相加的结果。在新页面点击 `Go back to root` 按键可以回到例程主页面，如下图所示：

图 2.4: *calc 例程*

2.2.3 ASP 变量替换例程

ASP 例程演示了页面变量替换的功能。在例程主页 ASP Test 模块下点击 `ASP 功能测试：访问 version.asp 文件` 按键，会跳转到根目录下 version.asp 页面，该页面显示当前使用 RT-Thread 系统的版本号。在新页面点击 `Go back to root` 按键回到例程主页面，如下图所示：

图 2.5: *ASP 例程*

2.2.4 SSI 文件嵌套例程

SSI 例程演示在一个页面中嵌套另一个页面的功能。在例程主页 SSI Test 模块下点击 `SSI 功能测试：访问 /version.shtml 页面` 按键，会跳转到根目录下 index.shtml 页面，该页面中嵌套显示了主页面 index.html 内容。在新页面点击 `Go back to root` 按键回到例程主页面。

图 2.6: *SSI 例程*

2.2.5 INDEX 目录显示例程

INDEX 例程演示页面文件列表功能。

首先需要任意上传一个文件到根目录的 /admin 目录中，例程中已上传了 admin.txt 文件到该目录中。

然后在例程主页 INDEX Test 模块下点击 `INDEX 功能测试：访问/admin 目录` 按键，会跳转到根目录下 /admin 目录，并且列出该目录下所有文件名和文件长度，如下图所示：

图 2.7: *INDEX 例程*

2.2.6 ALIAS 别名访问例程

ALIAS 例程演示了使用目录别名访问该目录的功能。该例程代码中已经将 /test 目录设置别名为 /admin，在例程主页 ALIAS Test 模块下点击 `ALIAS 功能测试：访问 /test 目录会跳转到 /admin 目录` 按键，实际我们访问的是 /test 目录，但会跳转访问 /admin 目录，并列出该目录下所有文件信息，如下图所示：

图 2.8: *ALIAS 例程*

2.2.7 Upload 文件上传例程

Upload 例程实现上传文件到 WebNet 服务器固定目录功能。在例程主页上 Upload File Test 模块下点击 选择文件 按键，选取需要上传的文件（本例程是用 upload.txt 文件），点击 上传，可以将文件上传到根

目录下的 /upload 目录，如下图所示：

图 2.9: *上传文件*

文件上传成功之后，返回例程主页，点击 浏览上传文件的目录 按键，可以访问 /upload 文件，查看刚才上传的文件信息。

图 2.10: *查看上传文件*

第 3 章

工作原理

WebNet 软件包主要用于在嵌入式设备上实现 HTTP 服务器，软件包的主要工作原理基于 HTTP 协议实现。

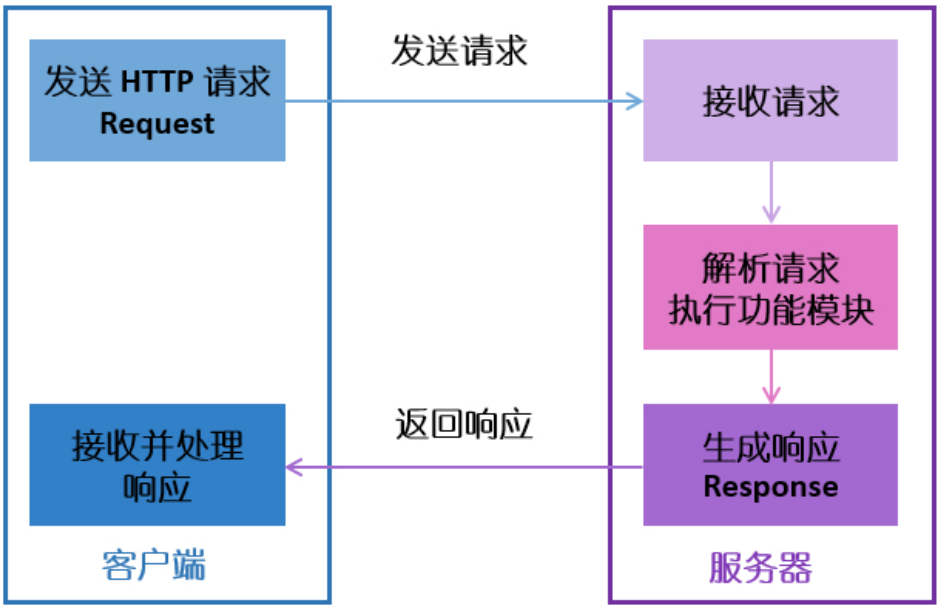

图 3.1: *WebNet 软件包工作原理*

HTTP 协议定义了客户端如何从服务器请求数据，以及服务器如何把数据传送给客户端的方式。HTTP 协议采用了请求/响应模型。客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器通过解析请求头部信息，执行相应的功能模块，并且给客户端发送响应数据，响应数据的内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。

在 HTTP 协议的实际使用过程中，一般遵循以下流程：

1. 客户端连接到服务器

   通常是通过 TCP 三次握手建立 TCP 连接，WebNet 中默认的连接端口号为 80。

2. 客户端发送 HTTP 请求

   通过 TCP 套接字，客户端向 WebNet 发送一个请求报文，一个请求报文由请求行、请求头部、空行和请求数据四部分组成

3. 服务器接收请求并解析数据信息

   服务器接收到客户端的请求后，开启解析请求信息，定位服务器上请求的资源文件，然后向客户端发送响应数据，由客户端读取。一个响应数据由状态行、响应头部、空行和响应数据四部分组成。

4. 客户端和服务器断开连接

   若客户端和服务器之间连接模式为普通模式，则服务器主动关闭 TCP 连接，客户端被动关闭连接，释放 TCP 连接。若连接模式为 keepalive 模式，则该连接保持一段时间，在该时间内可以继续接收数据。

第 4 章

使用指南

本节主要介绍 WebNet 软包的基本使用流程，并针对使用过程中经常涉及到的结构体和重要 API 进行简要说明。

4.1 准备工作

4.1.1 Env 配置说明

首先需要下载 WebNet 软件包，并将软件包加入到项目中。在 BSP 目录下使用 menuconfig 命令打开 Env 配置界面，在 `RT-Thread online packages → IoT - internet of things` 中选择 WebNet 软件包，具体路径如下：

```
RT-Thread online packages
    IoT - internet of things  --->
        [*] WebNet: A HTTP Server for RT-Thread
            (80)  Server listen port
            (16)  Maximum number of server connections
            (/webnet)   Server root directory
                  Select supported modules  --->
                     [*] LOG: Enanle output log support
                     [*] AUTH: Enanle basic HTTP authentication support
                     [*] CGI: Enanle Common Gateway Interface support
                     [*] ASP: Enanle Active Server Pages support
                     [*] SSI: Enanle Server Side Includes support
                     [*] INDEX: Enanle list all the file in the directory support
                     [*] ALIAS: Enanle alias support
                     [*] DAV: Enanle Web-based Distributed Authoring and Versioning
                         support
                     [*] UPLOAD: Enanle upload file support
                     [*] GZIP: Enable compressed file support by GZIP
                     (2) CACHE: Configure cache level
                        (1800) Cache-Control time in seconds
            [*]   Enable webnet samples
                  Version (latest)  --->
```

Server listen port：配置服务器监听端口号；

Maximum number of server connections：配置服务器最大支持连接数量；

Server root directory：配置服务器根目录路径；

Select supported modules：选择服务器支持的功能模块，默认开启全部功能模块支持；

- **LOG**：配置开启 WebNet 软件包日志功能支持；
- **AUTH**：配置开启 Basic Authentication 基本认证功能支持；
- **CGI**：配置开启 CGI 事件处理功能支持；
- **ASP**：配置开启 ASP 变量替换功能支持；
- **SSI**：配置开启文件嵌入功能支持；
- **INDEX**：配置开启显示当前页面文件列表功能支持；
- **ALIAS**：配置开启别名访问功能支持；
- **DAV**：配置开启基于 Web 的分布式创作和版本控制支持；
- **Upload**：配置开启文件上传功能支持；
- **GZIP**：配置开启服务器压缩文件访问支持；
- **CHCHE**：配置开启缓存功能支持（可选级别 0,1,2）；
- **Cache-Control time**：配置缓存功能有效时间，需要缓存功能等级为 2；

Enable webnet samples ：配置添加服务器示例文件；

Version：配置软件包版本。

这里我们默认开启 WebNet 软件包**全部功能支持**，版本号选择 **latest** 最新版本。选择合适的配置项和版本后，使用 `pkgs --update` 命令下载软件包并更新用户配置。

4.2 文件系统使用说明

WebNet 软件包使用，需要文件系统的支持（FAT 文件系统，ROMFS 文件系统等，支持 RT-Thread 的设备虚拟文件系统），用于 WebNet 软件包中访问的静态页面的存储、上传下载文件的存储等功能。

4.3 工作流程

使用 WebNet 软件包之前，可以先了解 WebNet 软件包基本工作流程，如下所示：

- 初始化 WebNet 软件包，启动监听连接请求；
- 接收连接请求，创建连接会话；
- 接收 HTTP 请求数据，解析请求信息；
- 判断请求的功能模块，执行对应的功能；
- 返回 HTTP 响应数据；

- 关闭连接会话。

WebNet 软件包可以实现**在浏览器访问设备端保存的页面，并且上传和下载设备端文件**。WebNet 软件包使用流程基于 HTTP 协议的请求和响应过程，通过解析 HTTP 请求的类型和参数进行判断，执行相应的功能模块，并且正确返回 HTTP 响应数据，完成整个流程。

下面以浏览器访问 WebNet 服务器根目录下主页面为例，介绍 WebNet 基本工作流程：

- **初始化 WebNet 软件包**

```
int webnet_init(void);
```

WebNet 软件包使用之前需要先初始化，初始化函数中创建了 webnet 线程。该线程用于初始化开启的功能模块，完成创建服务器监听套接字，并使用监听套接字等待客户端连接和数据交互。如下图为线程函数主要操作：

```
/* WebNet 服务器线程处理函数 */
static void webnet_thread(void *parameter)
{
    ....
    /* 创建监听套接字 */
    listenfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    ...
    /* 初始化开启的功能模块 */
    webnet_module_handle_event(RT_NULL, WEBNET_EVENT_INIT);

    /* 等待连接请求，等待接收数据 */
    for (;;)
    {
        ...
        sock_fd = select(maxfdp1, &tempfds, &tempwrtfds, 0, 0);
        if (sock_fd == 0)
            continue;

        if (FD_ISSET(listenfd, &tempfds))
        {
            /* 处理新的连接请求 */
        }

        /* 处理接收或发送的数据 */
        webnet_sessions_handle_fds(&tempfds, &writeset);
    }
    ...
}
```

- **接收连接请求，创建连接会话**

WebNet 初始化线程创建成功之后，当有新的连接请求产生时，会创建一个连接会话结构体，结构体定义如下：

```
struct webnet_session
{
    struct webnet_session *next;                    // 会话结构体链表

    int socket;
    struct sockaddr_in cliaddr;
    struct webnet_request* request;                 // 会话的请求相关信息

    rt_uint16_t buffer_length;
    rt_uint16_t buffer_offset;
    rt_uint8_t  buffer[WEBNET_SESSION_BUFSZ];       // 会话缓冲区数据，用于接收请求数
        据
    rt_uint32_t  session_phase;                     // 当前会话状态
    rt_uint32_t  session_event_mask;
    const struct webnet_session_ops* session_ops; // 会话事件执行函数 read、write、
        close等
    rt_uint32_t user_data;
};
```

webnet_session 结构体用于存放当前建立的连接会话的部分信息，可用与当前会话连接的整个流程。在进行 HTTP 数据交互之前，需要先创建并初始化该结构体，**新会话的创建已经在 webnet 线程中完成**，如下所示：

```
struct webnet_session* accept_session;
accept_session = webnet_session_create(listenfd);
if (accept_session == RT_NULL)
{
    /* 创建失败，关闭连接 */
}
```

- **接收 HTTP 请求数据，解析请求信息**

创建会话结构体成功之后，当连接会话接收到 HTTP 请求后，会对接收的 HTTP 请求进行处理，顺序地解析请求的类型、头部信息及附加参数。大致解析请求信息的流程如下所示：

```
/* 该函数用于解析当前会话连接的请求模式、头部信息和参数 */
static void _webnet_session_handle_read(struct webnet_session* session)
{
    /* 读取当前会话 HTTP 连接会话数据 */
    ....

    if (session->buffer_offset)
    {
        /* 解析 HTTP 请求模式（GET、POST 等）*/
        if (session->session_phase == WEB_PHASE_METHOD)
        {
            webnet_request_parse_method(...);
        }
```

```
        /* 解析 HTTP 请求头部信息 */
        if (session->session_phase == WEB_PHASE_HEADER)
        {
            webnet_request_parse_header(...);
        }

        /* 解析 HTTP URL 中附带请求参数 */
        if (session->session_phase == WEB_PHASE_QUERY)
        {
            webnet_request_parse_post(...);
        }
    }
}
```

- 判断请求的功能模块，执行对应的功能

通过对请求模式和头部信息的解析，得到当前连接会话请求的基本信息，然后继续判断使用的功能模块的类型，并且执行对应的模块，判断的大致流程如下：

```
/* 该函数为 WebNet 中用于判断和执行当前会话请求的功能，如日志功能、CGI 事件处理功能
    等 */
static int _webnet_module_system_uri_physical(struct webnet_session* session, int
    event)
{
    /* 如果开启 LOG 功能模块，使用 LOG 日志输出功能 */
#ifdef WEBNET_USING_LOG
    webnet_module_log(session, event);
#endif

     /* 如果开启 ALIAS 功能模块，判断当前请求是否是别名请求 */
#ifdef WEBNET_USING_ALIAS
    result = webnet_module_alias(session, event);
    if (result == WEBNET_MODULE_FINISHED) return result;
#endif

     /* 如果开启 AUTH 功能模块，判断当前请求是否需要执行基本认证操作 */
#ifdef WEBNET_USING_AUTH
    result = webnet_module_auth(session, event);
    if (result == WEBNET_MODULE_FINISHED) return result;
#endif

    /* 如果开启 CGI 功能模块，判断当前请求是否需要执行 CGI 操作 */
#ifdef WEBNET_USING_CGI
    result = webnet_module_cgi(session, event);
    if (result == WEBNET_MODULE_FINISHED) return result;
#endif
     ...
}
```

- **返回 HTTP 响应数据**

判断功能模块类型成功，并且正确执对应功能之后，WebNet 服务器会对当前会话连接的请求给予响应，如 CGI 功能执行之后，在 CGI 执行函数中可以使用 webnet_session_printf 或 webnet_session_write 函数发送响应数据到客户端。

```
/* 该函数 CGI 功能执行函数，当浏览器访问当前 CGI 事件时，执行该函数返回响应头部信息
    和数据 */
static void cgi_hello_handler(struct webnet_session* session)
{
     /* 拼接需要发送的页面数据 */
     ....

    /* 发送响应头部信息 */
    webnet_session_set_header(session, mime_get_type(".html"), 200, "Ok", strlen(
        status));

    /* 发送响应数据 */
    webnet_session_write(session, (const rt_uint8_t*)status, rt_strlen(status));
}
```

- **关闭连接会话**

当前会话连接请求解析成功、功能模块执行完成、响应数据发送完成之后，会关闭当前连接会话，释放会话结构体，完成整个 HTTP 数据数据交互过程，实现在浏览器上访问设备端提供的网页文件，或者完成上传、下载服务器上文件的操作。

4.4 使用方式

对于 WebNet 服务器的多种功能模块，有些功能模块在使用之前需要先设置对应配置参数，部分功能模块需要配合页面代码实现功能，接下来将介绍 WebNet 服务器不同功能模块的使用方式。

- **LOG 日志显示功能**

开启之后可以显示会话请求的基本信息，比如连接设置的 IP 地址和端口号，HTTP 请求类型、访问地址等信息，建议调试代码时开启。

- **AUTH 基本认证功能**

Basic Authentication 基础认证功能可以按目录划分访问权限。需要在 WebNet 服务器初始化之前调用 webnet_auth_set 函数设置目录的用户名和密码（设置的格式为 **用户名：密码**），浏览器中访问该目录时需要输入正确的用户名和密码才能访问目录。相关函数定义如下：

```
/* 设置目录基本认证信息 */
void webnet_auth_set(const char* path, const char* username_password);
```

AUTH 基本认证功能示例代码如下：

```
void webnet_test(void)
{
    /* 设置 /admin 目录用户名为 admin 密码为 admin */
     webnet_auth_set("/admin", "admin:admin");
     webnet_init();
}
```

/admin 目录设置基本认证功能之后，在浏览器中访问 /admin 目录时，需要输入设置的用户名和密码才能访问，如图所示：

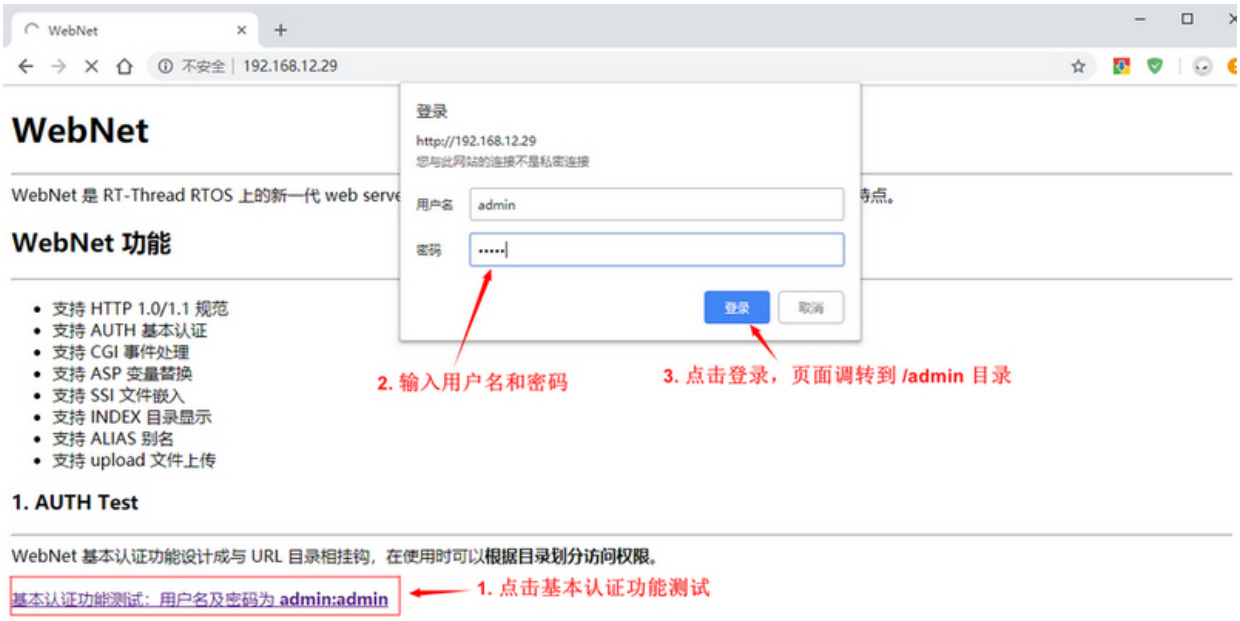

图 4.1: *基本认证功能*

- **CGI 功能**

CGI 功能可以自定义事件的执行函数，当浏览器发送对应该事件的请求时，WebNet 服务器可以执行相应的操作。需要在 WebNet 服务器初始化之前调用 `webnet_cgi_register` 函数注册 CGI 执行函数，相关函数定义如下：

```
/* 设置 CGI 事件根目录 */
void webnet_cgi_set_root(const char* root);
/* 设置 CGI 事件执行函数 */
void webnet_cgi_register(const char* name, void (*handler)(struct webnet_session*
    session));

/* 发送头部信息到 WebNet 连接的客户端，用于 CGI 事件执行函数中 */
void webnet_session_set_header(struct webnet_session *session, const char* mimetype,
     int code, const char* status, int length);
/* 发送固定格式数据到 WebNet 连接的客户端，用于 CGI 事件执行函数中 */
void webnet_session_printf(struct webnet_session *session, const char* fmt, ...);
/* 发送数据到 WebNet 连接的客户端，用于 CGI 事件执行函数中 */
```

```
int  webnet_session_write(struct webnet_session *session, const rt_uint8_t* data,
    rt_size_t size);
```

CGI 功能使用的示例代码如下：

```
static void cgi_hello_handler(struct webnet_session* session)
{
     const char* hello = "Hello World\n";
     webnet_session_set_header(session, mime_get_type(".html"), 200, "Ok", strlen(
         status));
     webnet_session_write(session, hello, rt_strlen(hello));
     webnet_session_printf(session, "%s", hello);
}
void webnet_test(void)
{
    /* 设置 CGI 事件执行函数*/
   webnet_cgi_register("hello", cgi_hello_handler);
    webnet_init();
}
```

对应的页面代码如下，浏览器上点击 **hello world** 按键将发送对应 CGI 请求给服务器。

```
<html>
  <body>
    <hr>
    <h3> CGI Test</h3>
    WebNet CGI 功能可以让用户执行指定的函数，CGI测试：
    <br/><br/>

    <br/>
  </body>
</html>
```

- **ASP 变量替换功能**

ASP 变量替换功能，可以匹配网页代码中 <% %> 标记中包含的变量，替换成代码中注册的执行函数。所以在 WebNet 初始化之前需要调用 webnet_asp_add_var 设置 ASP 变量替换函数，相关函数定义如下：

```
/* 设置 ASP 变量执行函数 */
void webnet_asp_add_var(const char* name, void (*handler)(struct webnet_session*
    session));
```

ASP 功能示例代码如下：

```
static void asp_var_version(struct webnet_session* session)
{
    webnet_session_printf(session, "RT-Thread: %d.%d.%d\n", RT_VERSION,
        RT_SUBVERSION, RT_REVISION);
```

```
}

void webnet_test(void)
{
    /* 设置 ASP 变量执行函数*/
    webnet_asp_add_var("version", asp_var_version);
    webnet_init();
}
```

对应的页面代码如下（文件名为 version.asp），访问该页面代码将 ASP 替换显示 RT-Thread 最新版本信息：

```
<html>
  <head>
    <title> ASP Test </title>
  </head>
  <body>
    <% version %>               /* ASP 变量替换成 RT_Thread 版本号显示 */
  </body>
</html>
```

- **SSI 文件嵌套功能**

WebNet 中支持嵌入文本文件到网页中，页面中需要有 <!--#include virtual="/index.html" --> 或者 <!--#include file="/index.html" --> 标记存在将替换成对应的文件内容，SSI 功能页面一般以 **.shtml**、**.stm** 结尾，如下为示例页面代码（文件名为 index.shtml）：

```
<html>
  <head>
    <title> SSI Test </title>
  </head>
  <body>
    <h1> SSI Test</h1>
    <font size=\"+2\">The index.html page embedded in the following page</font>
    <hr>
    <!--#include virtual="/index.html" -->   /* 该页面这嵌入index.html 文件内容 */
  </body>
</html>
```

- **INDEX 目录文件显示功能**

WebNet 服务器初始化成功之后，直接在浏览器中输入设置 IP 地址和要访问的目录，可以列出当前目录下所有的文件，如下图所示，访问服务器 /admin 目录，列出该目录下所有文件：

图 4.2: *目录显示功能*

- **ALIAS 别名访问功能**

ALIAS 别名访问功能可以给文件夹设置别名访问。需要在 WebNet 服务器初始化之前设置该文件夹的别名，如下代码所示，调用 `webnet_alias_set` 函数设置 /test 目录的别名为 /admin，浏览器访问 /test 时会跳转访问到 /admin 目录：

```
void webnet_test(void)
{
    /* 设置 /test 目录的别名为 /admin */
    webnet_alias_set("/test", "/admin");
    webnet_init();
}
```

图 4.3: *别名访问功能*

- **Upload 文件上传功能**

Upload 文件上传功能用于上传本地文件到 WebNet 服务器指定目录中，上传文件之前需要创建并实现上传文件结构体，如下所示：

```
struct webnet_module_upload_entry
{
    const char* url;                                        /* 文件上传的目录名 */

    int (*upload_open) (struct webnet_session* session);    /* 打开文件 */
    int (*upload_close)(struct webnet_session* session);    /* 关闭文件 */
    int (*upload_write)(struct webnet_session* session, const void* data, rt_size_t
        length);    /* 写数据到文件 */
    int (*upload_done) (struct webnet_session* session);    /* 下载完成 */
};
```

该结构体定义上传文件的目录文件和需要使用的事件回调函数，如：打开文件、关闭文件、写数据到文件等。

用户需要根据实际情况完成回调函数的实现，各回调函数中大致操作如下：

- upload_open：通过解析的文件名称，在指定的目录创建和打开文件；
- upload_close：关闭文件；
- upload_write：写数据到打开为文件中；
- upload_done：文件上传成功之后，对浏览器返回响应数据。

在回调函数实现的过程中，可能用到的获取当前上传文件会话信息的函数定义如下：

```
/* 获取当前上传文件名称 */
const char* webnet_upload_get_filename(struct webnet_session* session);
/* 获取当前上传文件类型 */
const char* webnet_upload_get_content_type(struct webnet_session* session);
/* 获取当前上传文件打开之后的文件描述符 */
const void* webnet_upload_get_userdata(struct webnet_session* session);
```

具体实现方式可以参考软件包 /samples/wn_sample_upload.c 中各个函数的实现方式。

最后，在 WebNet 初始化之前需要调用 webnet_upload_add 函数设置上传文件的信息，如下代码所示：

```
static int upload_open (struct webnet_session* session)
{
    /* 打开或新建文件 */
}
static int upload_close (struct webnet_session* session)
{
    /* 关闭文件 */
}
static int upload_write (struct webnet_session* session)
{
    /* 写数据到文件 */
}
static int upload_done (struct webnet_session* session)
{
    /* 下载完成，返回响应数据 */
}
const struct webnet_module_upload_entry upload_entry_upload =
{
    "/upload",
    upload_open,
    upload_close,
    upload_write,
    upload_done
};

void webnet_test(void)
{
```

```
    /* 注册文件上传执行函数 */
     webnet_upload_add(&upload_entry_upload);
     webnet_init();
}
```

对应页面上传文件的代码如下：

```
<html>
  <body>
    <h3>Upload File Test</h3>
    文件上传模块可以用于上传文件到指定的目录，这里上传到根目录下的 /upload 目录。
    <br/><br/>
    <form name="upload" method="POST" enctype="multipart/form-data" action="/upload"
        >
      <input type="file" name="file1" >
      <input type="submit" name="submit" value="上传">
    </form>
    <br/>

    <br/><br/>
  </body>
</html>
```

- 预压缩功能

WebNet 服务器预压缩功能，需要在服务器端提前压缩页面资源文件，生成以.gz 后缀的压缩文件。以根目录下 index.html 页面为例，浏览器访问该页面文件时，如果存在名称为 **index.html.gz** 的压缩文件，将发送该压缩文件内容到浏览器，浏览器自动解析后显示原页面文件。

使用预压缩功能时，需要先将压缩后的 **index.html.gz** 文件上传到文件系统中，如下图所示：

图 4.4: *预压缩功能*

4.5 常见问题

4.5.1 Q: 浏览器访问设备 IP 地址不显示页面信息。

A:

- 原因：设置的根目录地址错误。
- 解决方法：确定设置的根目录地址和设备文件系统上创建的目录地址一致，确定根目录下有页面文件。

4.5.2 Q: 设备出现 `out of pbuf` 错误情况。

A:

- 原因：设备内存不足。
- 解决方式：WebNet 软件包上传文件等功能需要额外占用资源空间，建议在资源空间充足的设备上运行，或者在 qemu 上使用。

第 5 章

API 说明

为了方便用户使用，这里列出了常用的 API，并给出了相关的使用说明。

5.1 初始化函数

```
int webnet_init(void);
```

用于初始化 WebNet 服务器，包括创建线程用于监听客户端连接事件、初始化开启的功能模块等功能；

参数	描述
无	无
返回	–
= 0	初始化成功
< 0	初始化失败

5.2 设置监听套接字端口

```
void webnet_set_port(int port);
```

用于设置当前 WebNet 服务器监听端口号，WebNet 服务器默认监听端口号是 80，这也是 HTTP 协议默认端口号。使用默认端口号访问 URL 地址时可以不输入端口号直接访问，当使用非默认端口号时，需要在 URL 地址上指明端口号，如：`http://host:8080/index.html` 。该函数只能**用于 WebNet 服务器初始化之前**。

参数	描述
port	设置的监听套接字端口
返回	–

参数	描述
无	无

5.3 获取监听套接字端口

```
int webnet_get_port(void);
```

用于获取当前 WebNet 服务器监听套接字端口号。

参数	**** 描述 ****
无	无
返回	–
>=0	监听套接字端口号

5.4 设置服务器根目录

```
void webnet_set_root(const char* webroot_path);
```

用于设置当前 WebNet 服务器根目录路径，WebNet 服务器默认根目录为 `/webnet`，浏览器和 WebNet 函数中使用或访问的路径都是基于根目录路径。当浏览器访问 `http://host/index.html` 时，会把文件系统中的 `/webnet/index.html` 返回给浏览器。

参数	描述
webroot_path	设置的根目录地址
返回	–
无	无

5.5 获取服务器根目录

```
const char* webnet_get_root(void);
```

用于获取当前 WebNet 服务器根目录地址。

参数	描述
无	无
返回	–

参数	描述
!= NULL	根目录地址

5.6 获取请求链接的类型

```
const char* mime_get_type(const char* url);
```

用于获取当前请求 URL 链接的类型，如：网页、图片、文本等。

参数	描述
url	请求链接的地址
返回	–
!= NULL	请求链接的类型

5.7 添加 ASP 变量处理方式

```
void webnet_asp_add_var(const char* name, void (*handler)(struct webnet_session*
    session));
```

该函数用于添加一个 ASP 变量处理方式，当 ASP 文件中出现添加的 `name` 变量名时，会执行对应的 `handle` 操作。

参数	描述
name	ASP 变量名称
void *(handler)(struct webnet_session* session)	ASP 变量处理方式
返回	–
无	无

5.8 添加 CGI 事件处理方式

```
void webnet_cgi_register(const char* name, void (*handler)(struct webnet_session*
    session));
```

该函数用于注册一个 CGI 事件处理方式，当浏览器请求带有 `name` 名称的 URL 时，会执行相应的 `handle` 操作。

参数	描述
name	CGI 事件名称
void (*handler)(struct webnet_session* session)	CGI 事件处理方式
返回	–
无	无

5.9 设置 CGI 事件根目录

```
void webnet_cgi_set_root(const char* root);
```

WebNet 服务器默认的 CGI 事件根目录为/cgi-bin，当浏览器请求 http://host/cgi-bin/test 地址时，会执行 test 名称对应的 CGI 事件处理函数。

该函数用于设置新的 CGI 事件根目录，设置成功之前的 CGI 根目录将不再起作用。

参数	描述
root	CGI 事件根目录
返回	–
无	无

5.10 设置基本认证信息

```
void webnet_auth_set(const char* path, const char* username_password);
```

用于设置目录访问时的基本认证信息，包括用户名和密码。

参数	描述
path	需要设置基本认证信息的目录
username_password	设置的用户名和密码，格式为 username:password
返回	–
无	无

5.11 设置目录别名

```
void webnet_alias_set(char* old_path, char* new_path);
```

用于设置目录的别名，设置成功之后可以使用目录别名访问该目录。

参数	描述
old_path	需要设置别名的目录
new_path	设置的目录别名，一般为服务器中存在的目录
返回	–
无	无

5.12 发送 HTTP 请求头部

```
void webnet_session_set_header(struct webnet_session* session, const char* mimetype,
    int code, const char* title, int length);
```

用于拼接并发送头部信息到连接的客户端，一般用于 ASP 变量处理函数和 CGI 事件处理函数中。

参数	描述
session	当前服务器连接的会话
mimetype	需要发送的响应文件类型（Content-Type），可以使用 `mime_get_type` 函数获取
code	发送的响应状态码，正常为 200
title	发送的响应状态类型，正常为 OK
length	需要发送的响应文件长度（Content-Length）
返回	–
无	无

5.13 发送 HTTP 响应数据

```
int  webnet_session_write(struct webnet_session* session, const rt_uint8_t* data,
    rt_size_t size);
```

用于发送响应数据到客户端，一般用于 ASP 变量处理函数和 CGI 事件处理函数中。

参数	描述
session	当前服务器连接的会话
data	发送的数据指针
size	发送的数据长度
返回	–
无	无

参数	描述

5.14 发送 HTTP 固定格式响应数据

```
void webnet_session_printf(struct webnet_session* session, const char* fmt, ...);
```

用于发送固定格式的响应数据到客户端，一般用于 ASP 变量处理函数和 CGI 事件处理函数中。

参数	描述
session	当前服务器连接的会话
fmt	自定义的输入数据的表达式
...	输入的参数
返回	–
无	无

5.15 获取上传文件的名称

```
const char* webnet_upload_get_filename(struct webnet_session* session);
```

获取当前上传文件的名称，用于打开或创建文件。

参数	描述
session	当前服务器连接的会话
返回	–
!= NULL	当前上传文件的名称

5.16 获取上传文件的类型

```
const char* webnet_upload_get_content_type(struct webnet_session* session);
```

获取当前上传文件的类型。

参数	描述
session	当前服务器连接的会话
返回	–

参数	描述
!= NULL	当前上传文件的类型

5.17 获取上传文件参数

```
const char* webnet_upload_get_nameentry(struct webnet_session* session, const char*
    name);
```

获取注册的上传文件的分隔符（HTTP 请求 boundary 参数）。

参数	描述
session	当前服务器连接的会话
name	上传文件的目录路径
返回	–
!= NULL	当前上传文件的类型

5.18 获取上传文件打开的文件描述符

```
const void* webnet_upload_get_userdata(struct webnet_session* session);
```

获取当前上传文件打开之后生成的文件描述符，用于读写数据到文件中。

参数	描述
session	当前服务器连接的会话
返回	–
!= NULL	上传文件打开的文件描述符