XRADIO SHTTPD

Developer Guide

Revision 1.0

Oct 22, 2019

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY (“XRADIO”). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Data	Summary of Changes
1.0	2019-10-22	Initial Version

Contents

1 模块概要

1.1 功能介绍

Shttpd 提供了实现轻量级 web server 的应用接口，内部功能、特性丰富，支持 CGI, SSL, cookie, MD5 认证等。SDK 中移植版本为 1.42，裁剪并保留了部分功能，例如单链接、http 1.0/ 1.1、get/ post、ssl、ssi(CALL)、网页支持 html/shtml 等，通过上述的所支持的功能，并利用其接口，可以快速的在 XR871 上开发出 web 服务器，基于 web 服务器用户可以实现不同的功能，比如设备控制、信息设置等。（源码中提供了建立 web server 的例程）

1.2 代码位置

模块	文件类型	代码位置
SHTTPD	source	sdk/src/net/shttpd-1.42/
	header	sdk/include/net/HTTPClient/
	demo	sdk/src/net/shttpd-1.42/examples/web_server_demo.c
		sdk/project/common/cmd/cmd_httpd.c

表 1- 1 SHTTPD 文件结构

2 模块接口

void _shttpd_init_local_file(struct usr_file *list, int count)		备注
功能	初始化本地 server 资源，用户通过该函数向系统注册文件。	
参数	list：本地文件链表 count：资源数目	
返回值	void	

struct shttpd_ctx *shttpd_init(int argc, char *argv[])		备注
功能	申请并初始化 shttpd 上下文。	
参数	(NULL, NULL)	
返回值	返回 shttpd_ctx 指针，供后续调用。	

void shttpd_fini(struct shttpd_ctx *ctx)		备注
功能	释放申请的上下文资源，关闭所有链接。	
参数	ctx: shttpd 上下文指针	
返回值	void	

void shttpd_poll(struct shttpd_ctx *ctx, int milliseconds)		备注
功能	处理连接请求。	
参数	ctx：shttpd 上下文指针 milliseconds：超时时间（ms）	
返回值	void	

int shttpd_set_ssl_cert(void *cert)		备注
功能	向系统注册证书。	
参数	cert：证书信息，传入符合 tls 的 security_server 结构	
返回值	0：成功，-1：失败	

int shttpd_set_option(struct shttpd_ctx *ctx, const char *opt, const char *val)		备注
功能	设置服务器选项新的参数（如端口）。	
参数	ctx：shttpd 上下文指针 opt：参数名称指针(“ssl_cert”，“ports”) val：参数值指针	
返回值	1：成功，0：失败	

int shttpd_get_var(const char *var, const char *buf, int buf_len, char *value, int value_len)		备注
功能	通过名称获取 POST / GET 变量值。	
参数	var：变量名字 buf：输入 buffer buf_Len：buffer 的长度 value：输出 buffer value_len：输出 buffer 的长度	
返回值	返回变量值的长度	

const char * shttpd_get_env(struct shttpd_arg *arg, const char *env_name)		备注
功能	通过变量名获取对应的参数（变量名：REQUEST_METHOD, REQUEST_URI, REMOTE_USER, REMOTE_ADDR）。	
参数	arg：用户的 callback 参数 env_name：参数名称指针	
返回值	返回参数指针	

const char *shttpd_get_header(struct shttpd_arg *arg, const char *header_name)		备注
功能	指定 HTTP 头部关键字获取相应的值。	
参数	arg：用户 callback 的参数 header_nam：指定的头关键字	

返回值	指定 HTTP 头的返回值	

void shttpd_register_ssi_func(struct shttpd_ctx *ctx, const char *name, shttpd_callback_t func, void *user_data)		备注
功能	指定相应 name 字段的回调函数（回调用于解析动态网页）。	
参数	ctx：shttpd 上下文指针 name：指定 name 字符串指针 func：用户回调函数 user_data：参数用于回调函数	
返回值	void	

void shttpd_register_uri(struct shttpd_ctx *ctx, const char *uri, shttpd_callback_t callback, void *data)		备注
功能	设置指定 URL 的回调函数。	
参数	ctx：服务器上下文指针 uri：指定的资源标志符 callback：回调函数指针 data：回调函数的参数	
返回值	void	

char *shttpd_version(void)		备注
功能	返回 shttpd 的版本指针。	
参数	void	
返回值	版本指针	

3 模块接口例程

3.1 流程描述（webserver）

本节提供上节介绍接口的示例，描述接口的使用方法及流程，文中代码皆为参考代码，不能直接运行，运行代码可直接参考 sdk 中源码文件 web_server_demo.c，web_server_demo.c 中基本上包含了常用接口的使用。

第一步：注册服务器文件资源

```
_shttpd_init_local_file(file_list,ARRAY_SIZE(file_list));

//file_list 为本地文件资源，通过该函数将本地文件资源注册到服务器中。

struct usr_file file_list[] = {
    {"./",              ""},
    {"./index.html",    index_html},
    {"./config.shtml",  config_shtml},
#if !defined(NO_CHECKSUM)
    {"./checksum1.txt",  checksum1_txt},
    {"./checksum2.txt",  checksum2_txt},
#endif
    {NULL},
};
```

第二步：服务器初始化

```
if ((ctx = shttpd_init(argc, argv)) == NULL) {
        _shttpd_elog(E_FATAL, NULL, "Cannot initialize SHTTPD context.");
        return -1;
}
```

第三步：设置服务器选项及参数

```
#if defined(SHTTPD_SSL)
ca_param.nCa = mbedtls_test_cas_pem_len;
ca_param.pCa = (char *)mbedtls_test_cas_pem;
ca_param.nCert = mbedtls_test_srv_crt_len;
ca_param.pCert = (char *)mbedtls_test_srv_crt;
ca_param.nKey = mbedtls_test_srv_key_len;
ca_param.pKey = (char *)mbedtls_test_srv_key;

shttpd_set_ssl_cert((void*) &ca_param);
shttpd_set_option(ctx, "ssl_cert", NULL);
shttpd_set_option(ctx, "ports", "443s");
#else
shttpd_set_option(ctx, "ports", "80");
#endif
```

代码中通过宏定义区分两类情况：安全、非安全。安全条件下，需向系统注册证书，证书的相关细节请参考 tls 相关章节内容。另外需要注册端口，通常非安全使用 80 端口，安全使用 443s 端口（web 服务器）。

第四步：如果使用 ssi（提供 CALL，用于解析 shtml 文件）机制，需要向系统注册对应关键字的回调函数

```
shttpd_register_ssi_func(ctx, "DeviceName", ssi_get_ssid, ap_info.g_ssid);
shttpd_register_ssi_func(ctx, "DevicePasswd", ssi_get_passwd,
                         ap_info.g_passwd);
```

第五步： 如果实现 post 机制，需要向系统注册对应 url 的回调函数。

```
shttpd_register_uri(ctx, "/devicename", set_ap_info, (void *)&ap_info);

#if !defined(NO_CHECKSUM)
shttpd_register_uri(ctx, "/post/key/checksum1_txt", set_post_info, NULL);
shttpd_register_uri(ctx, "/post/key/checksum2_txt", set_post_info, NULL);
shttpd_register_uri(ctx, "/get/key/checksum1_txt", set_post_info, NULL);
#endif
```

第五步： 调用服务器运行处理函数，处理相应的连接。

```
_shttpd_exit_flag = 0;
while (_shttpd_exit_flag == 0)
       shttpd_poll(ctx, 1000);
```

第六步： 释放资源，服务器退出。

```
shttpd_fini(ctx);
```

3.2 流程注意

上节描述的步骤中，第三、四步不是必须的，客户按照自己的需要定义功能。

4 其他配置

1. 使能 SSL 功能

```
文件：Sdk/include/net/shttpd/compat_rtos.h

//#define SHTTPD_SSL
```

2. 请求 buffer 大小

```
文件：Sdk/include/net/shttpd/config.h

define URI_MAX       4096
```

3. 索引文件格式

```
文件：Sdk/include/net/shttpd/config.h

#define    INDEX_FILES   "index.html,index.htm,index.php,index.cgi"
```

4. SSI 支持格式

```
文件：Sdk/include/net/shttpd/config.h

#define    SSI_EXT       "shtml,shtm"
```

5. 超时时间（用于长连接）

```
文件：Sdk/include/net/shttpd/config.h

#define    EXPIRE_TIME   10
```

无特别需求 config.h 保持默认参数，超时时间可以根据需求来定单位为秒。