

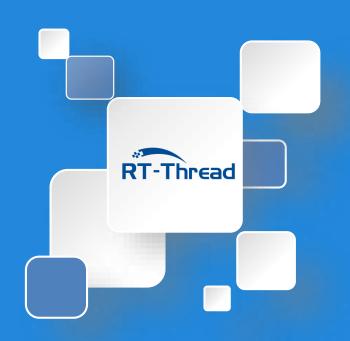
网络编程提升篇

HTTP: 利用 HTTP 协议获取天气

目录

- 背景介绍
- 运行示例
- 注意事项
- 示例代码讲解





背景介绍

背景介绍

- HTTP 协议是互联网上应用最为广泛的一种网络协议,越来越多的应用程序需要直接通过 HTTP 协议来访问网络资源。
- webclient 是 RT-Thread 上实现的一个 HTTP 客户端,用来提供高效且功能丰富的 HTTP 客户端编程工具包。
- 这个教程展示了如何利用 HTTP 协议获取天气,我们是使用 webclient 这个工具包实现的。





运行示例

开启 WebClient 软件包

- <u>Webclient</u> 软件包是一个 HTTP 协议的客户端工具 ,利用这个工具可以完成与 HTTP 服务器的通信。
- 打开 env 工具输入 menuconfig 按照下面的路径开启 WebClient 软件包

```
RT-Thread online packages

IoT - internet of things --->

[*] WebClient: A webclient package for rt-thread --->
```

WebClient 软件包的配置如下所示

```
--- WebClient: A HTTP/HTTPS Client for RT-Thread

[ ] Enable support tls protocol

[ ] Enable webclient GET/POST samples

    Version (latest) --->
```



WebClient 配置介绍

- Eable support tls protocol: 开启加密传输
- Enable webclient GET/POST samples: 开启 webclient 的 GET/POST 示例程序
- version (latest) --->: 选择版本



开启 cJSON 软件包

- 因为服务器返回的天气信息是 JSON 格式的,所以我们需要利用 cJSON 软件 包来解析返回的数据。
- 打开 env 工具输入 menuconfig 按照下面的路径开启 cJSON 软件包

```
RT-Thread online packages

IoT - internet of things --->

[*] cJSON: Ultralightweight JSON parser in ANSI C --->
```

• cJSON 软件包的配置保持默认即可

```
--- cJSON: Ultralightweight JSON parser in ANSI C
Version (v1.0.2) --->
```



获取示例代码

• 打开 env 工具输入 menuconfig 按照下面的路径获取 httpclient.c 示例代码

```
RT-Thread online packages --->
miscellaneous packages --->
samples: RT-Thread kernel and components samples --->
a network_samples package for rt-thread --->
[*] [network] http client
```

- 保存并更新软件包 pkgs --update
- 编译工程 scons
- · 然后运行 qemu



运行示例代码

• 示例代码中已经将 weather 命令导出到了 msh 命令列表中,因此系统运行起来后,在 msh 命令行下输入 weather 命令即可让示例代码运行。

msh> weather

• 终端会打印出一些天气信息

cityName:浦东

temp :25°C

wd :东风

ws :2级

sd :49%

date :20131012

time :15:00

• 注:由于 env 编码格式不支持 UTF-8, 会出现中文显示乱码的问题。



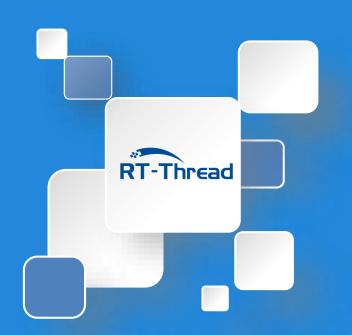


注意事项

注意事项

- 示例代码中的 AREA_ID 是城市的代码 , 更换城市代码可以获取不同城市的天气。
- 由于 env 编码格式不支持 UTF-8,会出现中文显示乱码的问题。
- 此程序仅为 http client 的示例程序,其中获取天气的 API 已经被弃用,实际使用时更换为其他的 API 才可获取最新的天气。
- 电脑需要关闭防火墙





```
*程序清单:利用 http client 获取天气
* 这是一个利用 http client 获取天气的例程
* 导出 weather 命令到控制终端
* 命令调用格式: weather
*程序功能:作为一个 http 客户端,通过 GET 方法与远方服务器通信,获取到服务器传来的天气信息*/
#include <webclient.h> /* 使用 HTTP 协议与服务器通信需要包含此头文件 */ #include <sys/socket.h> /* 使用BSD socket,需要包含socket.h头文件 */
#include <netdb.h>
#include <cJSON.h>
#include <finsh.h>
                                     //头部大小
#define GET HEADER BUFSZ
                             1024
                                      //响应缓冲区大小 //网址最大长度
#define GET_RESP_BUFSZ
                             1024
#define GET URL LEN MAX
                             256
                             "http://mobile.weather.com.cn/data/sk/%s.html" //获取天气的 API
#define GET URI
                             "101021300" //上海浦东地区 ID
#define AREA ID
```



```
/* 天气数据解析 */
void weather_data_parse(rt_uint8_t *data)
  cJSON *root = RT NULL, *object = RT NULL, *item = RT NULL;
  root = cJSON_Parse((const char *)data);
  if (!root)
    rt kprintf("No memory for cJSON root!\n");
    return;
  object = cJSON GetObjectItem(root, "sk info");
  item = cJSON GetObjectItem(object, "cityName");
  rt kprintf("\ncityName:%s ", item->valuestring);
  item = cJSON GetObjectItem(object, "temp");
  rt kprintf("\ntemp :%s", item->valuestring);
  item = cJSON_GetObjectItem(object, "wd");
  rt kprintf("\nwd :%s ", item->valuestring);
```



```
item = cJSON GetObjectItem(object, "ws");
  rt kprintf("\nws :%s ", item->valuestring);
  item = cJSON GetObjectItem(object, "sd");
  rt kprintf("\nsd :%s", item->valuestring);
  item = cJSON GetObjectItem(object, "date");
  rt kprintf("\ndate :%s", item->valuestring);
  item = cJSON GetObjectItem(object, "time");
  rt kprintf("\ntime :%s \n", item->valuestring);
  if (root != RT NULL)
    cJSON Delete(root);
void weather(int argc, char **argv)
  rt uint8 t *buffer = RT NULL;
  int resp status;
  struct webclient session *session = RT NULL;
  char *weather_url = RT_NULL;
  int content length = -1, bytes read = 0;
  int content pos = 0;
```



```
/* 为 weather url 分配空间 */
 weather url = rt calloc(1, GET URL LEN MAX);
 if (weather_url == RT_NULL)
   rt kprintf("No memory for weather url!\n");
   goto exit;
 /* 拼接 GET 网址 */
 rt snprintf(weather url, GET URL LEN MAX, GET URI, AREA ID);
 /* 创建会话并且设置头部的大小 */
 session = webclient session create(GET HEADER BUFSZ);
 if (session == RT NULL)
   rt_kprintf("No memory for get header!\n");
   goto exit;
 /* 发送 GET 请求使用默认的头部 */
 if ((resp status = webclient get(session, weather url)) != 200)
   rt kprintf("webclient GET request failed, response(%d) error.\n", resp status);
   goto __exit;
```



```
/* 分配用于存放接收数据的缓冲 */
  buffer = rt_calloc(1, GET_RESP_BUFSZ);
  if(buffer == RT_NULL)
    rt_kprintf("No memory for data receive buffer!\n");
   goto exit;
  content_length = webclient_content_length_get(session);
  if (content length < 0)
    /* 返回的数据是分块传输的. */
    do
      bytes_read = webclient_read(session, buffer, GET_RESP_BUFSZ);
      if (bytes read <= 0)
        break;
    } while (1);
  } else{
    do
```



```
bytes read = webclient read(session, buffer,
          content length - content pos > GET RESP BUFSZ ?
               GET RESP BUFSZ : content length - content pos);
      if (bytes read <= 0)
        break;
      content pos += bytes read;
    } while (content pos < content length);</pre>
  /* 天气数据解析 */
  weather data parse(buffer);
  exit:
  if (weather_url != RT_NULL)
 rt_free(weather_url);
/* 关闭会话 */
  if (session != RT NULL)
 webclient_close(session);
/*释放缓冲区空间 */
  if (buffer != RT NULL)
    rt free(buffer);
MSH CMD EXPORT(weather, Get weather by webclient);
```

