<https://blog.csdn.net/len_yue_mo_fu/article/details/75158686>

libghttp库下载地址：<http://lfs.linuxsir.org/htdocs/blfscvs/gnome/libghttp.html>   
官方API参考文档：<https://sourcecodebrowser.com/libghttp/1.0.9/ghttp_8h.html>   
将源码包下载下来，进入例行安装流程：   
先解压：

tar -xzvf libghttp-1.0.9.tar.gz

cd libghttp-1.0.9

安装：

./configure

make

make install

在安装的过程中可能会遇到一些问题：

checking host system type... Invalid configuration `x86\_64-unknown-linux-gnu': machine `x86\_64-unknown' not recognized

checking build system type... Invalid configuration `x86\_64-unknown-linux-gnu': machine `x86\_64-unknown' not recognized

checking for ranlib... ranlib

checking for ld used by GCC... /usr/bin/ld

checking if the linker (/usr/bin/ld) is GNU ld... yes

checking for BSD-compatible nm... /usr/bin/nm -B

checking whether ln -s works... yes

updating cache ./config.cache

ltconfig: you must specify a host type if you use `--no-verify'

Try `ltconfig --help' for more information.

configure: error: libtool configure failed

解决这个问题其实很简单：

# 首先查找一下config.guess和config.sub文件的目录

find / -name config.guess

find / -name config.sub

# 将查找出来的文件随便选择一个覆盖到软件目录

cp /usr/share/automake-1.11/config.guess .

cp /usr/share/automake-1.11/config.sub .

如果找不到则：

sudo apt-get install automake

下面是我自己封装的一个http客户端，拥有post和get方法：

头文件httpClient.h

/\*=========================================================

\*文件名称: httpClient.h

\*创建日期： 2017-7-6

\*创建者 ： 冷月莫负

\*修改记录：

\* 2017-7-6 created

===========================================================\*/

#ifndef HTTPCLIENT\_H

#define HTTPCLIENT\_H

#include <map>

#include <string>

using namespace std;

int Get(string url, string params, int timeout, string&result,map<string,string>&m) ;

int Post(string uri, string params, int timeout, string&result, map<string,string>&m) ;

#endif

cpp实现文件httpClient.cpp

#include "ghttp.h"

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include "httpClient.h"

#include "ulaneLog.h"

int Get(string url, string params, int timeout, string& result,map<string,string>&m)

{

ghttp\_status status;

char\* temresult = NULL;

ghttp\_request \*request = NULL;

request = ghttp\_request\_new();

char szUrl[256];

memset(szUrl,0x00,sizeof(szUrl));

strcpy(szUrl,url.c\_str());

if (ghttp\_set\_uri(request, szUrl) == -1)

{

ghttp\_clean(request);

return -1;

}

if (ghttp\_set\_type(request, ghttp\_type\_get) == -1)

{

ghttp\_clean(request);

return -1;

}

map<string,string>::iterator it = m.begin();

for(it;it!=m.end();it++)

{

ghttp\_set\_header(request, it->first.c\_str(),it->second.c\_str());

}

char timeout\_str[10];

sprintf(timeout\_str, "%d", timeout);

ghttp\_set\_header(request, http\_hdr\_Timeout, timeout\_str);

ghttp\_set\_sync(request, ghttp\_sync);

ghttp\_prepare(request);

status=ghttp\_process(request);

if (status == ghttp\_error)

{

ghttp\_request\_destroy(request);

return -1;

}

temresult = ghttp\_get\_body(request);

result += temresult;

ghttp\_request\_destroy(request);

return 0;

}

int Post(string uri, string params, int timeout, string& result, map<string,string>&m)

{

char\* temresult = NULL;

ghttp\_request \*request = NULL;

ghttp\_status status;

int len;

char szUri[256];

memset(szUri,0x00,sizeof(szUri));

strcpy(szUri,uri.c\_str());

request = ghttp\_request\_new();

if (ghttp\_set\_uri(request, szUri) == -1)

{

ghttp\_clean(request);

return -1;

}

if (ghttp\_set\_type(request, ghttp\_type\_post) == -1)

{

ghttp\_clean(request);

return -1;

}

map<string,string>::iterator it = m.begin();

for(it;it!=m.end();it++)

{

ghttp\_set\_header(request, it->first.c\_str(),it->second.c\_str());

}

ghttp\_set\_sync(request, ghttp\_sync);

char timeout\_str[10];

sprintf(timeout\_str, "%d", timeout);

ghttp\_set\_header(request, "Content-Type", "application/json");

ghttp\_set\_header(request, http\_hdr\_Timeout, timeout\_str);

char szParams[8192];

memset(szParams,0x00,sizeof(szParams));

strcpy(szParams,params.c\_str());

len = strlen(szParams);

ghttp\_set\_body(request, szParams, len); //

ghttp\_prepare(request);

status = ghttp\_process(request);

if (status == ghttp\_error)

{

return -1;

}

result = ghttp\_get\_body(request); //test

temresult = ghttp\_get\_body(request);

result += temresult;

ghttp\_request\_destroy(request);

return 0;

}

post和get方法中最后一个参数是一个map<string,string>类型的map容器，使用的时候只需要将你要设置的header都已key:value的形式存到容器中，就都会被设置。

参考文章：   
<http://blog.csdn.net/earbao/article/details/39007549>   
<http://www.phpjiayuan.com/109/270.html>