树莓派

账号：pi

密码：raspberry

1.树莓派HDMI登陆

2.树莓派串口登陆：(13条消息) 树莓派--串口登录_*树莓派串口登录*_一只青木呀的博客-CSDN博客

3.网络登陆树莓派：

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

```
network={
        ssid="你的无线网名字"
        psk="密码"
        key_mgmt=WPA-PSK//这个是加密的方式，可以不写
}
```

(13条消息) 树莓派的几种登录方式及树莓派的网络配置_FHNCSDN的博客-CSDN博客

然后打开ssh功能

```
sudo raspi-config
```

最后重启一下

固定IP地址

```
sudo nano /etc/rc.local
```

在最后fi exit0之间加上

```
ifconfig wlan0 xxx.xxx.xx.xxx(ifconfig找到的ip)
```

然后重启

```
sudo reboot
```

ssh登陆，树莓派连接的WiFi要和电脑连接的WiFi是同一个（同一个局域网，这样mobaXterm才能用）

4.树莓派图形xrdp界面登陆

首先树莓派下载

```
sudo apt-get install xrdp
```

然后Windows中搜索　“远程桌面连接”输入树莓派IP就可以连接

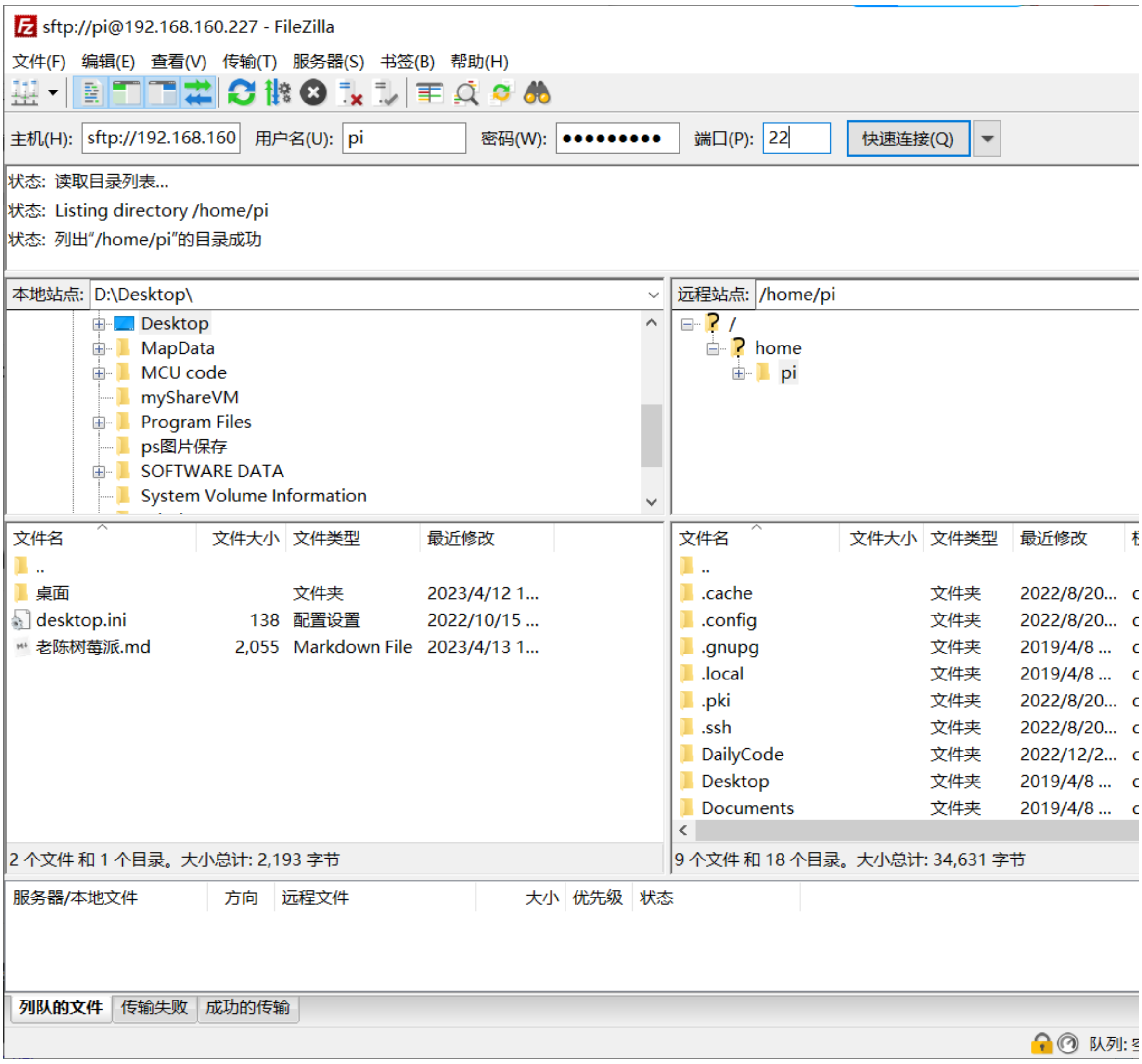
如果连不上就关闭防火墙

或者将树莓派的xrdp进行卸载再重新安装

卸载指令

```
sudo apt autoremove xxxxx
```

树莓派和Windows的FileZilla文件共享


sftp://pi@192.168.160.227 - FileZilla

主机(H): sftp://192.168.160　用户名(U): pi　密码(W): ●●●●●●●●●　端口(P): 22　快速连接(Q)

```
状态:  读取目录列表...
状态:  Listing directory /home/pi
状态:  列出"/home/pi"的目录成功
```

本地站点: D:\Desktop\

Desktop, MapData, MCU code, myShareVM, Program Files, ps图片保存, SOFTWARE DATA, System Volume Information

远程站点: /home/pi

/ → home → pi

文件名	文件大小	文件类型	最近修改
..			
桌面		文件夹	2023/4/12 1...
desktop.ini	138	配置设置	2022/10/15 ...
老陈树莓派.md	2,055	Markdown File	2023/4/13 1...

2 个文件 和 1 个目录。大小总计: 2,193 字节

文件名	文件大小	文件类型	最近修改
..			
.cache		文件夹	2022/8/20...
.config		文件夹	2022/8/20...
.gnupg		文件夹	2019/4/8 ...
.local		文件夹	2019/4/8 ...
.pki		文件夹	2022/8/20...
.ssh		文件夹	2022/8/20...
DailyCode		文件夹	2022/12/2...
Desktop		文件夹	2019/4/8 ...
Documents		文件夹	2019/4/8 ...

9 个文件 和 18 个目录。大小总计: 34,631 字节

文件(F) 编辑(E) 查看(V) 传输(T) 服务器(S) 书签(B) 帮助(H)
服务器/本地文件 方向 远程文件 大小 优先级 状态
列队的文件 传输失败 成功的传输
队列:

安装软件指令：

```
sudo apt-get install XXXX
```

树莓派换镜像源

```
sudo nano /etc/apt/sources.list
```

Linux中的库

[(13条消息) Linux中的库（静态库和动态库）详解*linux 静态库*石子君的博客-CSDN博客]()

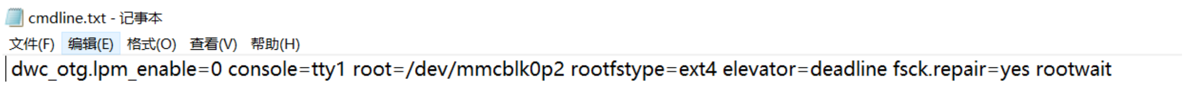
树莓派串口通信

首先：修改cmdline.txt文件

```
cd /boot

sudo vi cmdline.txt
```

然后删除以下内容（禁用串口调试功能，改成通信功能）

```
console=ttyAMA0,115200
```



```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait
```

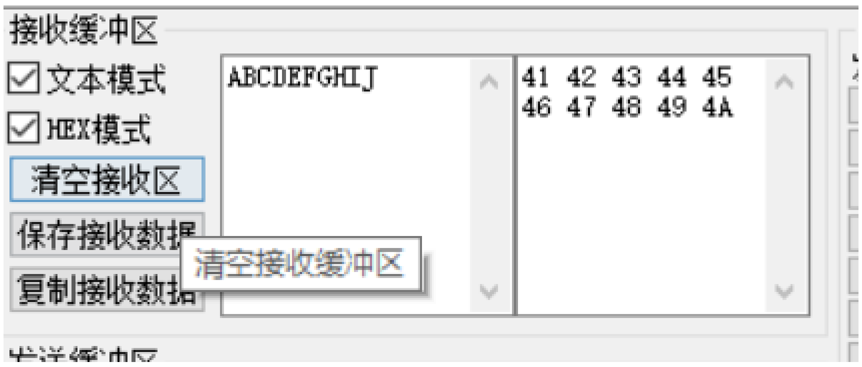
最后重启

```
sudo reboot
```

代码如下：

```c
#include <stdio.h>
#include <wiringSerial.h>
#include <wiringPi.h>
int main()
{
        int err_ret = wiringPiSetup ();
        if(err_ret == -1)
        {
                printf("setup failed\n");
                return -1;
        }
        int fd =  serialOpen ("/dev/ttyAMA0", 115200);
        if(fd == -1)
        {
                printf("serial open failed\n");
                return -1;
        }
        int cnt = 0;
        for(cnt = 0 ;cnt < 10 ;cnt ++)
        {
        serialPutchar (fd, 'A'+cnt);
        }

-- VISUAL --                                        26        1,1           Top
```

结果如下：



```
ABCDEFGHIJ
```

```
41 42 43 44 45
46 47 48 49 4A
```

树莓派交叉编译工具链的安装

树莓派mjpg-streamer实现监控功能调试

[(9条消息) 树莓派安装MJPG-streamer_mjpg-streamer还维护吗_冷月枫啊的博客-CSDN博客]

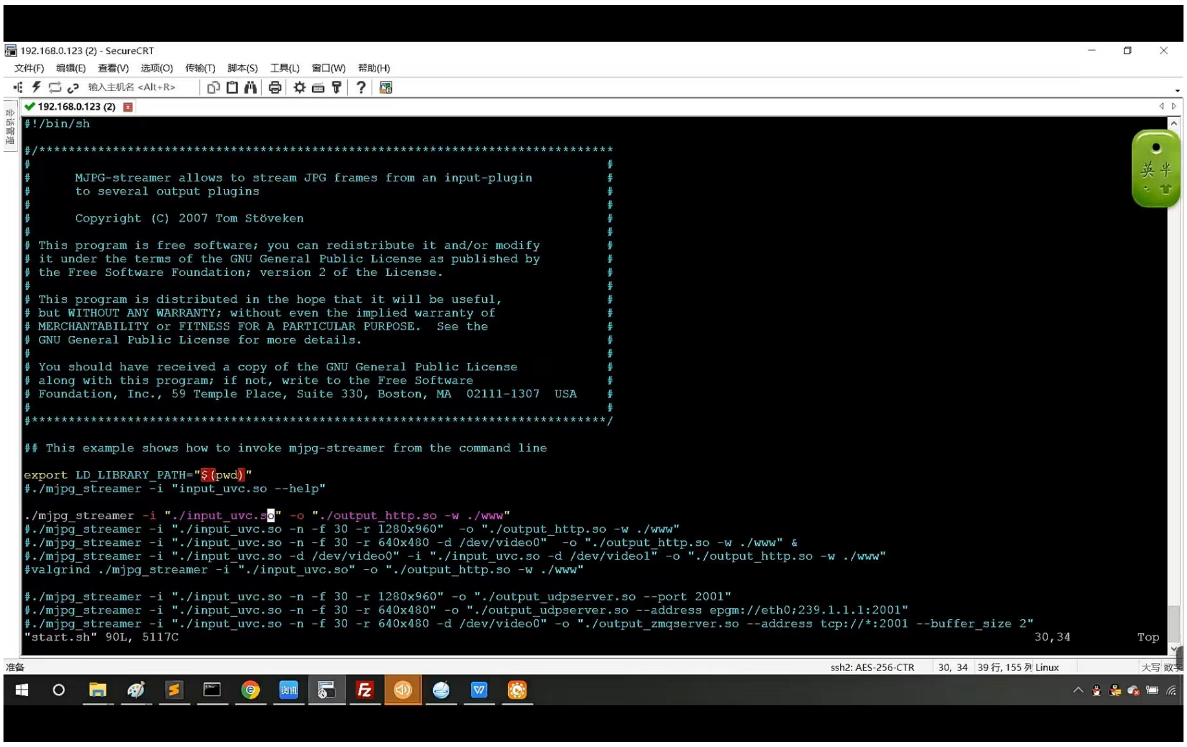
[(9条消息) 树莓派（raspberry pi）mjpg——streamer 远程识别监控（换源+csl摄像头+usb摄像头零基础指导以及一些过程问题的解决）_放小孩的羊的博客-CSDN博客]

[树莓派3B + Pi摄像头+mjpg-streamer-百度经验 (baidu.com)]

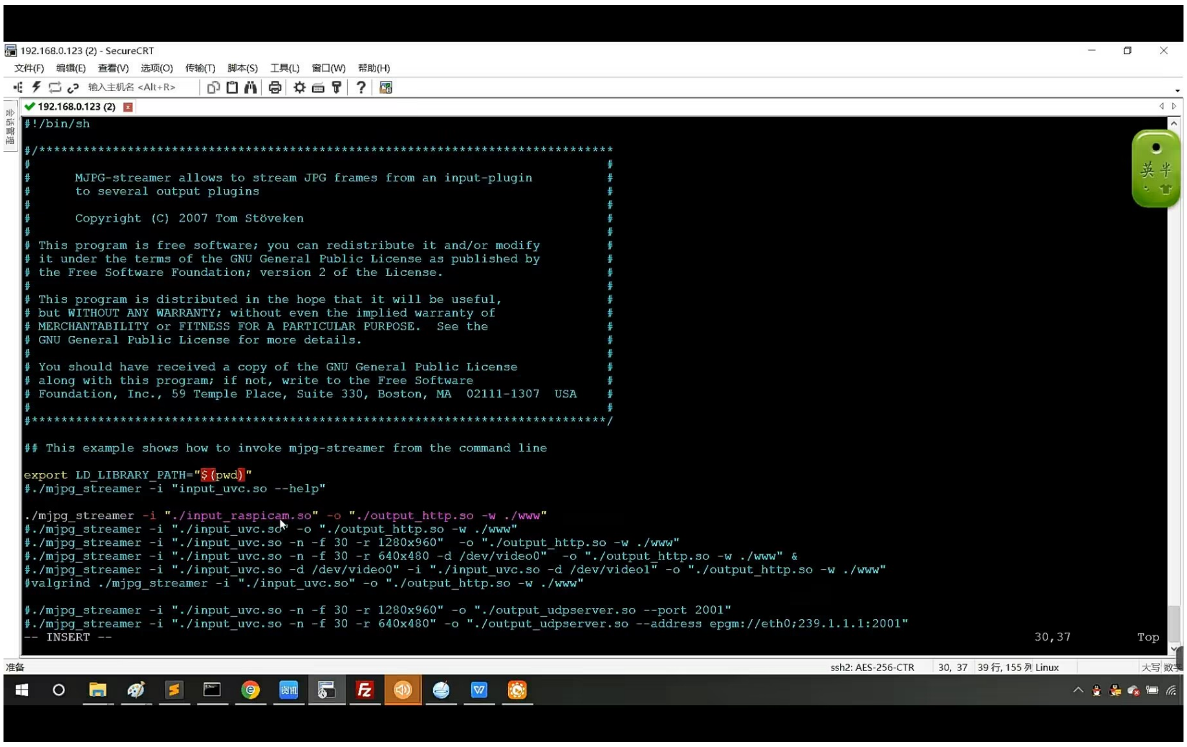
input_uvc.so 是usb摄像头

input_raspicam.so是树莓派摄像头

将start.sh中input_uvc.so改为input_raspicam.so



```
#!/bin/sh

#/******************************************************************************
#                                                                              #
#      MJPG-streamer allows to stream JPG frames from an input-plugin         #
#      to several output plugins                                              #
#                                                                              #
#      Copyright (C) 2007 Tom Stöveken                                        #
#                                                                              #
# This program is free software; you can redistribute it and/or modify        #
# it under the terms of the GNU General Public License as published by        #
# the Free Software Foundation; version 2 of the License.                     #
#                                                                              #
# This program is distributed in the hope that it will be useful,             #
# but WITHOUT ANY WARRANTY; without even the implied warranty of              #
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               #
# GNU General Public License for more details.                                #
#                                                                              #
# You should have received a copy of the GNU General Public License           #
# along with this program; if not, write to the Free Software                 #
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA   #
#                                                                              #
#******************************************************************************/

## This example shows how to invoke mjpg-streamer from the command line

export LD_LIBRARY_PATH="$(pwd)"
#./mjpg_streamer -i "input_uvc.so --help"

./mjpg_streamer -i "./input_uvc.so" -o "./output_http.so -w ./www"
#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 1280x960"  -o "./output_http.so -w ./www"
#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 640x480 -d /dev/video0"  -o "./output_http.so -w ./www" &
#./mjpg_streamer -i "./input_uvc.so -d /dev/video0" -i "./input_uvc.so -d /dev/video1" -o "./output_http.so -w ./www"
#valgrind ./mjpg_streamer -i "./input_uvc.so" -o "./output_http.so -w ./www"

#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 1280x960" -o "./output_udpserver.so --port 2001"
#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 640x480" -o "./output_udpserver.so --address epgm://eth0;239.1.1.1:2001"
#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 640x480 -d /dev/video0" -o "./output_zmqserver.so --address tcp://*:2001 --buffer_size 2"
"start.sh" 90L, 5117C                                                                         30,34          Top
```



```
#!/bin/sh

#/******************************************************************************
#                                                                              #
#      MJPG-streamer allows to stream JPG frames from an input-plugin         #
#      to several output plugins                                              #
#                                                                              #
#      Copyright (C) 2007 Tom Stöveken                                        #
#                                                                              #
# This program is free software; you can redistribute it and/or modify        #
# it under the terms of the GNU General Public License as published by        #
# the Free Software Foundation; version 2 of the License.                     #
#                                                                              #
# This program is distributed in the hope that it will be useful,             #
# but WITHOUT ANY WARRANTY; without even the implied warranty of              #
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               #
# GNU General Public License for more details.                                #
#                                                                              #
# You should have received a copy of the GNU General Public License           #
# along with this program; if not, write to the Free Software                 #
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA   #
#                                                                              #
#******************************************************************************/

## This example shows how to invoke mjpg-streamer from the command line

export LD_LIBRARY_PATH="$(pwd)"
#./mjpg_streamer -i "input_uvc.so --help"

./mjpg_streamer -i "./input_raspicam.so" -o "./output_http.so -w ./www"
#./mjpg_streamer -i "./input_uvc.so" -o "./output_http.so -w ./www"
#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 1280x960"  -o "./output_http.so -w ./www"
#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 640x480 -d /dev/video0"  -o "./output_http.so -w ./www" &
#./mjpg_streamer -i "./input_uvc.so -d /dev/video0" -i "./input_uvc.so -d /dev/video1" -o "./output_http.so -w ./www"
#valgrind ./mjpg_streamer -i "./input_uvc.so" -o "./output_http.so -w ./www"

#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 1280x960" -o "./output_udpserver.so --port 2001"
#./mjpg_streamer -i "./input_uvc.so -n -f 30 -r 640x480" -o "./output_udpserver.so --address epgm://eth0;239.1.1.1:2001"
-- INSERT --                                                                                  30,37          Top
```

socket,tcp,udp,http,之间的区别和原理

[socket、tcp、udp、http 的认识及区别 - 叫我程某某 - 博客园 (cnblogs.com)]()

http,https,libcurl协议相关

[Http协议之libcurl实现 - 谢呈勖 - 博客园 (cnblogs.com)]()

url就是网址

libcurl等第三方库的通用编译方法

首先从GitHub上下载源码

```
https://github.com/curl/curl/releases/tag/curl-7_71_1
```

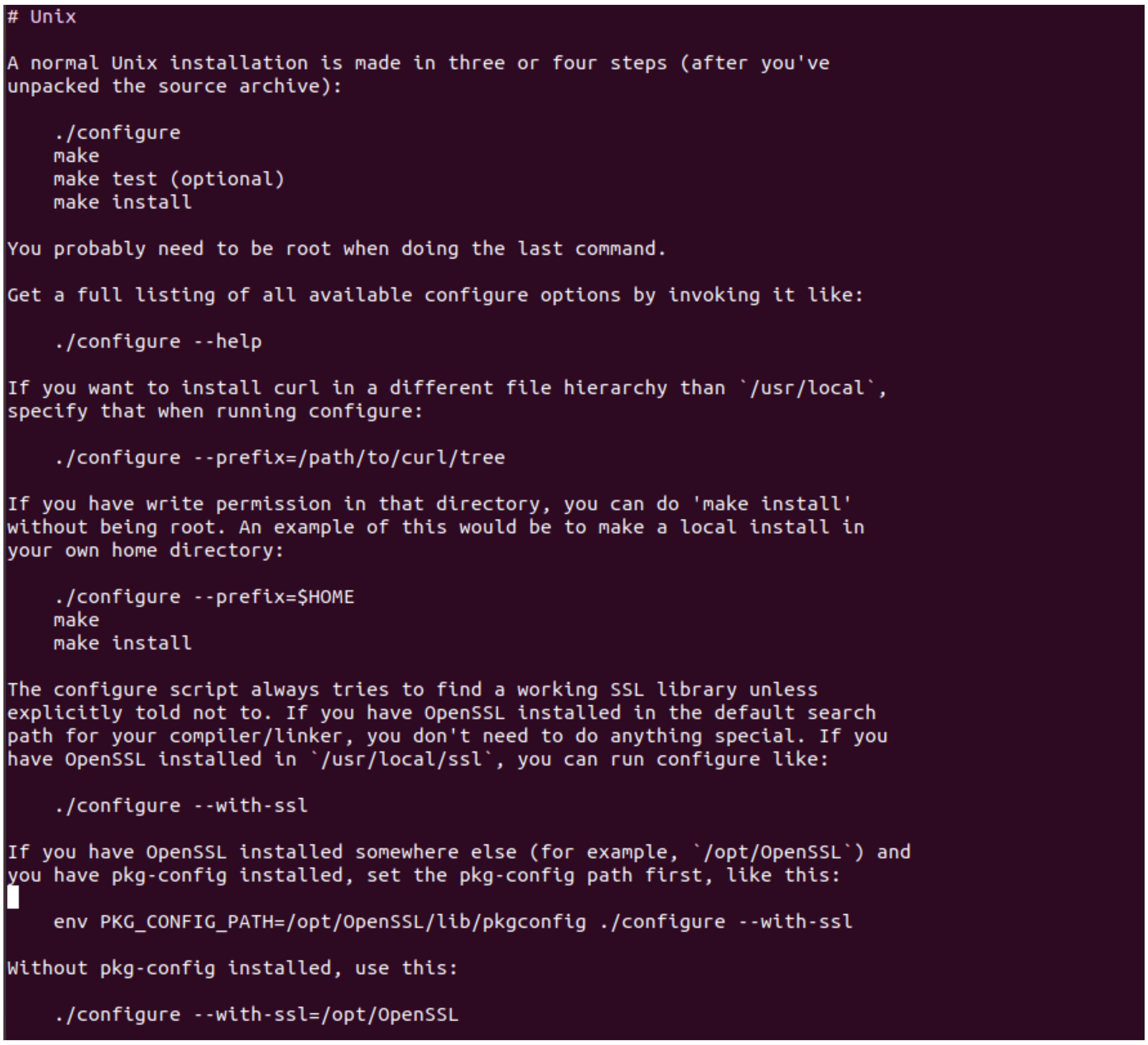
下载后放到Ubuntu中进行解压

```
tar -xjf curl-7.71.1.tar.bz2
```

进入解压好的文件夹

```
cd curl-7.71.1/
```

可以看看docs中的INSTALL.md，里面有各种环境的安装方法，我们当前的环境是linux



```
# Unix

A normal Unix installation is made in three or four steps (after you've
unpacked the source archive):

    ./configure
    make
    make test (optional)
    make install

You probably need to be root when doing the last command.

Get a full listing of all available configure options by invoking it like:

    ./configure --help

If you want to install curl in a different file hierarchy than `/usr/local`,
specify that when running configure:

    ./configure --prefix=/path/to/curl/tree

If you have write permission in that directory, you can do 'make install'
without being root. An example of this would be to make a local install in
your own home directory:

    ./configure --prefix=$HOME
    make
    make install

The configure script always tries to find a working SSL library unless
explicitly told not to. If you have OpenSSL installed in the default search
path for your compiler/linker, you don't need to do anything special. If you
have OpenSSL installed in `/usr/local/ssl`, you can run configure like:

    ./configure --with-ssl

If you have OpenSSL installed somewhere else (for example, `/opt/OpenSSL`) and
you have pkg-config installed, set the pkg-config path first, like this:

    env PKG_CONFIG_PATH=/opt/OpenSSL/lib/pkgconfig ./configure --with-ssl

Without pkg-config installed, use this:

    ./configure --with-ssl=/opt/OpenSSL
```

```
cd curl-7.71.1/
```

进入curl-7.71.1这个文件夹，现在开始进行编译

```
./configure --prefix=$PWD/_install
```

将编译后生成的东西放到当前工作路径的_install文件夹底下

也可以指定交叉编译工具

```
./configure --prefix=$PWD/_install --host=arm-linux-gnueabihf-gcc
```

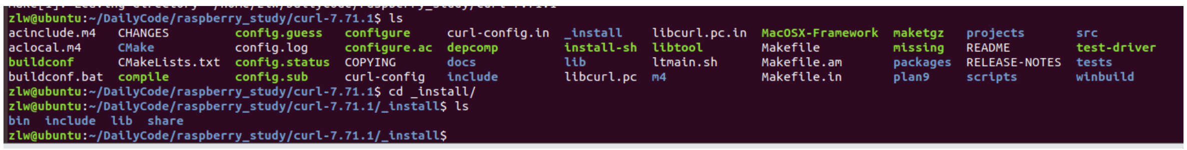
然后

```
make
```

最后

```
make install
```

进入到_install文件夹查看一下编译生成的文件



```
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$ ls
acinclude.m4   CHANGES          config.guess   configure      curl-config.in  _install     libcurl.pc.in  MacOSX-Framework  maketgz   projects       src
aclocal.m4     CMake            config.log     configure.ac   depcomp         install-sh   libtool        Makefile          missing   README         test-driver
buildconf      CMakeLists.txt   config.status  COPYING        docs            lib          ltmain.sh      Makefile.am       packages  RELEASE-NOTES  tests
buildconf.bat  compile          config.sub     curl-config    include         libcurl.pc   m4             Makefile.in       plan9     scripts        winbuild
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$ cd _install/
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1/_install$ ls
bin  include  lib  share
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1/_install$
```

调用libcurl编程访问百度主页

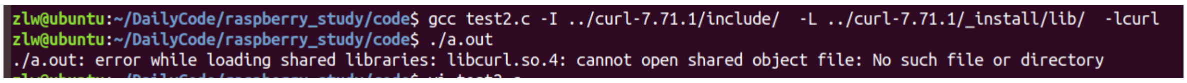
代码如下：

```
#include <stdio.h>
#include <curl/curl.h>

#define true 1
#define false 0
typedef unsigned int bool;

bool getUrl(char *filename)
{
        CURL *curl;
        CURLcode res;
        FILE *fp;
        if ((fp = fopen(filename, "w")) == NULL)  // 返回结果用文件存储
                return false;
        struct curl_slist *headers = NULL;
        headers = curl_slist_append(headers, "Accept: Agent-007");
        curl = curl_easy_init();    // 初始化
        if (curl)
        {
                //curl_easy_setopt(curl, CURLOPT_PROXY, "10.99.60.201:8080");// 代理
                curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);// 改协议头
                curl_easy_setopt(curl, CURLOPT_URL,"http://www.baidu.com");
                curl_easy_setopt(curl, CURLOPT_WRITEDATA, fp); //将返回的http头输出到fp指向的文件
                curl_easy_setopt(curl, CURLOPT_HEADERDATA, fp); //将返回的html主体
数据输出到fp指向的文件
                res = curl_easy_perform(curl);   // 执行
                if (res != 0) {

                        curl_slist_free_all(headers);
                        curl_easy_cleanup(curl);
                }
                fclose(fp);
                return true;
        }
}
bool postUrl(char *filename)
{
        CURL *curl;
        CURLcode res;
        FILE *fp;
        if ((fp = fopen(filename, "w")) == NULL)
                return false;
        curl = curl_easy_init();
        if (curl)
        {
                curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "/tmp/cookie.txt");
// 指定cookie文件
                curl_easy_setopt(curl, CURLOPT_POSTFIELDS,
"&logintype=uid&u=xieyan&psw=xxx86");    // 指定post内容
                //curl_easy_setopt(curl, CURLOPT_PROXY, "10.99.60.201:8080");
                curl_easy_setopt(curl, CURLOPT_URL, "
http://mail.sina.com.cn/cgi-bin/login.cgi ");   // 指定url
                curl_easy_setopt(curl, CURLOPT_WRITEDATA, fp);
                res = curl_easy_perform(curl);
                curl_easy_cleanup(curl);
        }
        fclose(fp);
        return true;
}
int main(void)
{
        getUrl("/tmp/get.html");
        postUrl("/tmp/post.html");
}
```

编译

```
gcc test2.c -I ../curl-7.71.1/include/  -L ../curl-7.71.1/_install/lib/  -lcurl
```

有报错



```
zlw@ubuntu:~/DailyCode/raspberry_study/code$ gcc test2.c -I ../curl-7.71.1/include/  -L ../curl-7.71.1/_install/lib/  -lcurl
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./a.out
./a.out: error while loading shared libraries: libcurl.so.4: cannot open shared object file: No such file or directory
```

需要配置环境变量

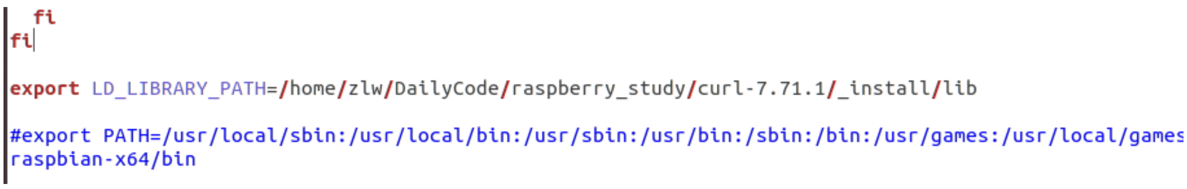
```
export LD_LIBRARY_PATH=/home/zlw/DailyCode/raspberry_study/curl-
7.71.1/_install/lib
```

也可以永久配置环境：

```
gedit ~/.bashrc
```

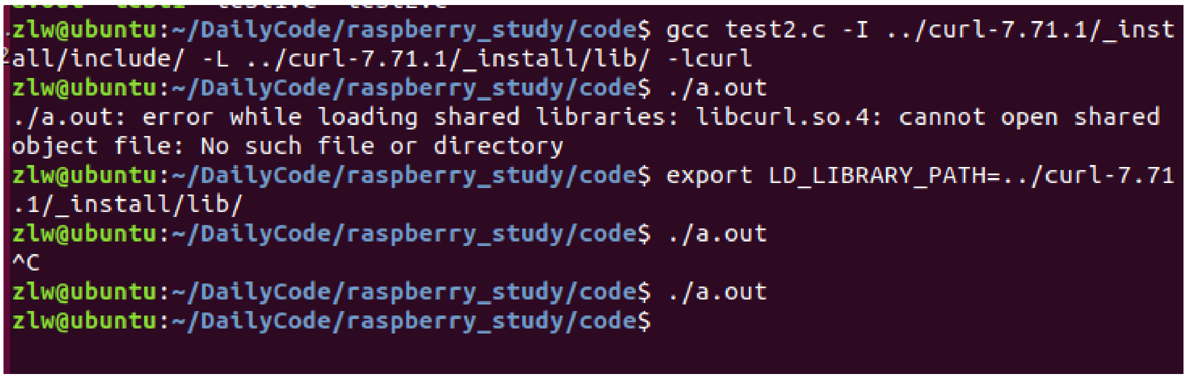
在最后添加以下内容

```
export LD_LIBRARY_PATH=/home/zlw/DailyCode/raspberry_study/curl-
7.71.1/_install/lib
```



```
  fi
fi

export LD_LIBRARY_PATH=/home/zlw/DailyCode/raspberry_study/curl-7.71.1/_install/lib

#export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
raspbian-x64/bin
```

然后重启虚拟机就OK了



```
zlw@ubuntu:~/DailyCode/raspberry_study/code$ gcc test2.c -I ../curl-7.71.1/_inst
all/include/ -L ../curl-7.71.1/_install/lib/ -lcurl
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./a.out
./a.out: error while loading shared libraries: libcurl.so.4: cannot open shared
object file: No such file or directory
zlw@ubuntu:~/DailyCode/raspberry_study/code$ export LD_LIBRARY_PATH=../curl-7.71
.1/_install/lib/
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./a.out
^C
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./a.out
zlw@ubuntu:~/DailyCode/raspberry_study/code$
```

这样子就可以运行了

libcurl函数库常用字段解读并设置数据读取回调函数

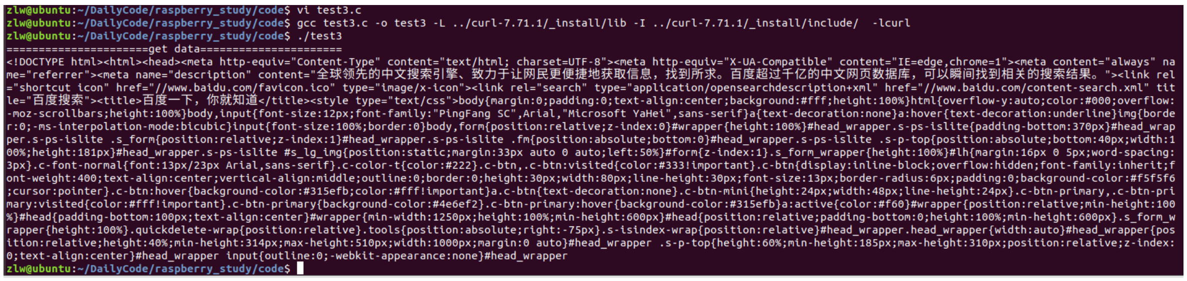
read_func是回调函数

代码：

```c
#include <stdio.h>
#include <curl/curl.h>
#include <string.h>
#define true 1
#define false 0
typedef unsigned int bool;


size_t read_func( void *ptr, size_t size, size_t nmemb, void *stream)
{
        char buf[10240] = {0};
        strncpy(buf, ptr, 10240);
        printf("======================get data======================\n");
        printf("%s\n",buf);

}
```

```
bool getUrl(char *filename)
{
        CURL *curl;
        CURLcode res;
        FILE *fp;
        if ((fp = fopen(filename, "w")) == NULL)  // 返回结果用文件存储
                return false;
        struct curl_slist *headers = NULL;
        headers = curl_slist_append(headers, "Accept: Agent-007");
        curl = curl_easy_init();    // 初始化
        if (curl)
        {
                //curl_easy_setopt(curl, CURLOPT_PROXY, "10.99.60.201:8080");//
代理
                curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);// 改协议头
                curl_easy_setopt(curl, CURLOPT_URL,"http://www.baidu.com");
                curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, read_func); //通过
回调函数存储数据
                //curl_easy_setopt(curl, CURLOPT_WRITEDATA, fp); //将返回的http头
输出到fp指向的文件
                //curl_easy_setopt(curl, CURLOPT_HEADERDATA, fp); //将返回的html主
体数据输出到fp指向的文件
                res = curl_easy_perform(curl);   // 执行
                if (res != 0) {

                        curl_slist_free_all(headers);
                        curl_easy_cleanup(curl);
                }
                fclose(fp);
                return true;
        }
}
bool postUrl(char *filename)
{
        CURL *curl;
        CURLcode res;
        FILE *fp;
        if ((fp = fopen(filename, "w")) == NULL)
                return false;
        curl = curl_easy_init();
        if (curl)
        {
                curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "/tmp/cookie.txt");
// 指定cookie文件
                curl_easy_setopt(curl, CURLOPT_POSTFIELDS,
"&logintype=uid&u=xieyan&psw=xxx86");    // 指定post内容
                //curl_easy_setopt(curl, CURLOPT_PROXY, "10.99.60.201:8080");
                curl_easy_setopt(curl, CURLOPT_URL, "
http://mail.sina.com.cn/cgi-bin/login.cgi ");   // 指定url
                curl_easy_setopt(curl, CURLOPT_WRITEDATA, fp);
                res = curl_easy_perform(curl);
                curl_easy_cleanup(curl);
        }
        fclose(fp);
        return true;
}
int main(void)
{
        getUrl("/tmp/get.html");
        postUrl("/tmp/post.html");
}
```

输出结果：



```
zlw@ubuntu:~/DailyCode/raspberry_study/code$ vi test3.c
zlw@ubuntu:~/DailyCode/raspberry_study/code$ gcc test3.c -o test3 -L ../curl-7.71.1/_install/lib -I ../curl-7.71.1/_install/include/  -lcurl
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./test3
======================get data======================
<!DOCTYPE html><html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><meta content="always" na
me="referrer"><meta name="description" content="全球领先的中文搜索引擎、致力于让网民更便捷地获取信息，找到所求。百度超过千亿的中文网页数据库，可以瞬间找到相关的搜索结果。"><link rel
="shortcut icon" href="//www.baidu.com/favicon.ico" type="image/x-icon"><link rel="search" type="application/opensearchdescription+xml" href="//www.baidu.com/content-search.xml" tit
le="百度搜索"><title>百度一下，你就知道</title><style type="text/css">body{margin:0;padding:0;text-align:center;background:#fff;height:100%}html{overflow-y:auto;color:#000;overflow:
-moz-scrollbars;height:100%}body,input{font-size:12px;font-family:"PingFang SC",Arial,"Microsoft YaHei",sans-serif}a{text-decoration:none}a:hover{text-decoration:underline}img{borde
r:0;-ms-interpolation-mode:bicubic}input{font-size:100%;border:0}body,form{position:relative;z-index:0}#wrapper{height:100%}#head_wrapper.s-ps-islite{padding-bottom:370px}#head_wrap
per.s-ps-islite .s_form{position:relative;z-index:1}#head_wrapper.s-ps-islite .fm{position:absolute;bottom:0}#head_wrapper.s-ps-islite .s-p-top{position:absolute;bottom:40px;width:1
00%;height:181px}#head_wrapper.s-ps-islite #s_lg_img{position:static;margin:33px auto 0 auto;left:50%}#form{z-index:1}.s_form_wrapper{height:100%}#lh{margin:16px 0 5px;word-spacing:
3px}.c-font-normal{font:13px/23px Arial,sans-serif}.c-color-t{color:#222}.c-btn,.c-btn:visited{color:#333!important}.c-btn{display:inline-block;overflow:hidden;font-family:inherit;f
ont-weight:400;text-align:center;vertical-align:middle;outline:0;border:0;height:30px;width:80px;line-height:30px;font-size:13px;border-radius:6px;padding:0;background-color:#f5f5f6
;cursor:pointer}.c-btn:hover{background-color:#315efb;color:#fff!important}a.c-btn{text-decoration:none}.c-btn-mini{height:24px;width:48px;line-height:24px}.c-btn-primary,.c-btn-pri
mary:visited{color:#fff!important}.c-btn-primary{background-color:#4e6ef2}.c-btn-primary:hover{background-color:#315efb}a:active{color:#f60}#wrapper{position:relative;min-height:100
%}#head{padding-bottom:100px;text-align:center}#wrapper{min-width:1250px;height:100%;min-height:600px}#head{position:relative;padding-bottom:0;height:100%;min-height:600px}.s_form_w
rapper{height:100%}.quickdelete-wrap{position:relative}.tools{position:absolute;right:-75px}.s-isindex-wrap{position:relative}#head_wrapper.head_wrapper{width:auto}#head_wrapper{pos
ition:relative;height:40%;min-height:314px;max-height:510px;width:1000px;margin:0 auto}#head_wrapper .s-p-top{height:60%;min-height:185px;max-height:310px;position:relative;z-index:
0;text-align:center}#head_wrapper input{outline:0;-webkit-appearance:none}#head_wrapper
zlw@ubuntu:~/DailyCode/raspberry_study/code$ 
```

(9条消息) Ubuntu连接不上网络问题的解决方法*ubuntu连接不了网络的解决方法*菜鸡什么时候能变强的博客-CSDN博客

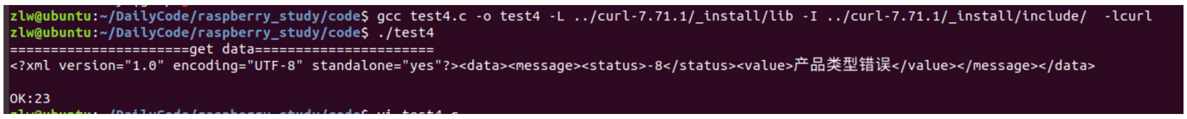
编程实现人脸识别第一次

平台：***翔云人工智能*API_服务平台 (netocr.com)**

代码：

```
#include <stdio.h>
#include <curl/curl.h>
#include <string.h>
#include <stdlib.h>
#define true 1
#define false 0
typedef unsigned int bool;


size_t read_func( void *ptr, size_t size, size_t nmemb, void *stream)
{
        char buf[1024] = {0};
        strncpy(buf, ptr, 1024);
        printf("======================get data======================\n");
        printf("%s\n",buf);

}
bool postUrl()
{
        CURL *curl;
        CURLcode res;

        char *postString;
        char img1[12];
        char img2[12];
        char *key = "MPWeajiYjAtrYDfZweE8vk";
        char *secret = " 9e300b424e7e4a20a7dabd3799410c56";
        int typeID = 21;
        char *formate = "xml";

        postString = (char*)malloc(strlen(key)+strlen(secret)+2048);


        //字符串拼接

 sprintf(postString,"&img1=%s&img2=%s&key=%s&secret=%s&typeID=%d&formate=%s",
                                          "",      "",     key,     secret,    typeID,
 formate);

        curl = curl_easy_init();
        if (curl)
        {
                curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "/tmp/cookie.txt");
// 指定cookie文件
                curl_easy_setopt(curl, CURLOPT_POSTFIELDS, postString);    // 指
定post内容
                curl_easy_setopt(curl, CURLOPT_URL,
"https://netocr.com/api/faceliu.do");   // 指定url
                curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, read_func); //通过
回调函数存储数据
                res = curl_easy_perform(curl);

                printf("OK:%d\n",res);


                curl_easy_cleanup(curl);
        }
        return true;
}
int main(void)c
{
        postUrl();
}
```

运行结果：



```
zlw@ubuntu:~/DailyCode/raspberry_study/code$ gcc test4.c -o test4 -L ../curl-7.71.1/_install/lib -I ../curl-7.71.1/_install/include/  -lcurl
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./test4
======================get data======================
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><data><message><status>-8</status><value>产品类型错误</value></message></data>

OK:23
```

原因：没有传递两张图片参数

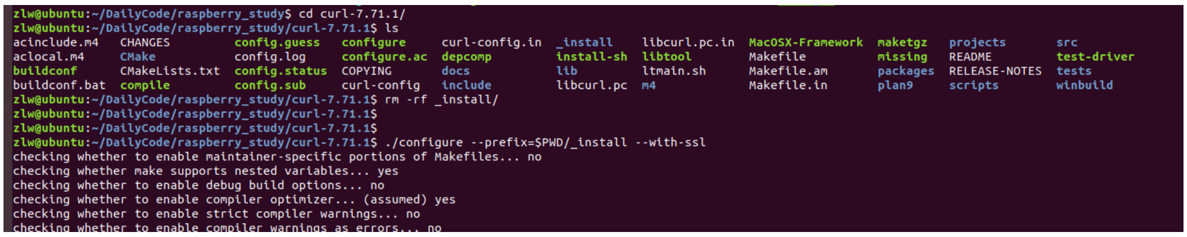
编译openssl支持libcurl的https访问

首先删除之前编译后生成的文件（因为翔云这个平台需要ssl）

```
rm -rf _install
```

然后重新配置

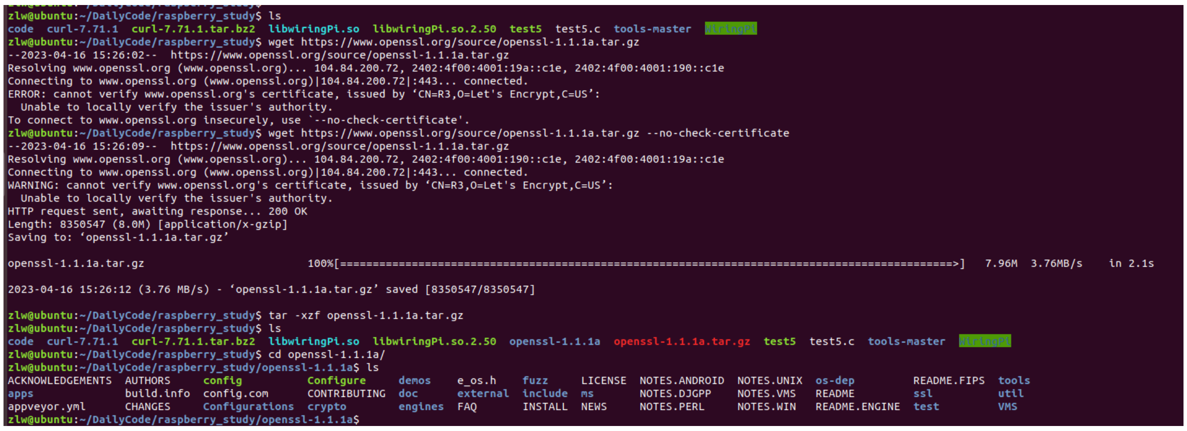
```
./configure --prefix=$PWD/_install --with-ssl
```



```
zlw@ubuntu:~/DailyCode/raspberry_study$ cd curl-7.71.1/
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$ ls
acinclude.m4   CHANGES         config.guess   configure     curl-config.in  _install     libcurl.pc.in  MacOSX-Framework  maketgz   projects       src
aclocal.m4     CMake           config.log     configure.ac  depcomp         install-sh   libtool        Makefile          missing   README         test-driver
buildconf      CMakeLists.txt  config.status  COPYING       docs            lib          ltmain.sh      Makefile.am       packages  RELEASE-NOTES  tests
buildconf.bat  compile         config.sub     curl-config   include         libcurl.pc   m4             Makefile.in       plan9     scripts        winbuild
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$ rm -rf _install/
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$ ./configure --prefix=$PWD/_install --with-ssl
checking whether to enable maintainer-specific portions of Makefiles... no
checking whether make supports nested variables... yes
checking whether to enable debug build options... no
checking whether to enable compiler optimizer... (assumed) yes
checking whether to enable strict compiler warnings... no
checking whether to enable compiler warnings as errors... no
```

安装openssl

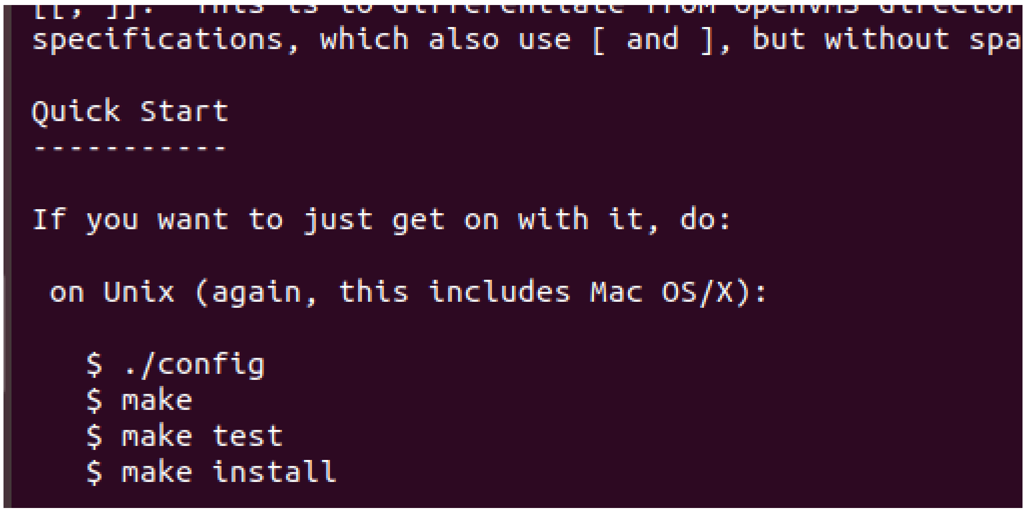
[(9条消息) 交叉编译：OpenSSL*使wget支持https*钱德勒宾的博客-CSDN博客]



```
zlw@ubuntu:~/DailyCode/raspberry_study$ ls
code  curl-7.71.1  curl-7.71.1.tar.bz2  libwiringPi.so  libwiringPi.so.2.50  test5  test5.c  tools-master  WiringPi
zlw@ubuntu:~/DailyCode/raspberry_study$ wget https://www.openssl.org/source/openssl-1.1.1a.tar.gz
--2023-04-16 15:26:02--  https://www.openssl.org/source/openssl-1.1.1a.tar.gz
Resolving www.openssl.org (www.openssl.org)... 104.84.200.72, 2402:4f00:4001:19a::c1e, 2402:4f00:4001:190::c1e
Connecting to www.openssl.org (www.openssl.org)|104.84.200.72|:443... connected.
ERROR: cannot verify www.openssl.org's certificate, issued by ‘CN=R3,O=Let's Encrypt,C=US’:
  Unable to locally verify the issuer's authority.
To connect to www.openssl.org insecurely, use `--no-check-certificate'.
zlw@ubuntu:~/DailyCode/raspberry_study$ wget https://www.openssl.org/source/openssl-1.1.1a.tar.gz --no-check-certificate
--2023-04-16 15:26:09--  https://www.openssl.org/source/openssl-1.1.1a.tar.gz
Resolving www.openssl.org (www.openssl.org)... 104.84.200.72, 2402:4f00:4001:190::c1e, 2402:4f00:4001:19a::c1e
Connecting to www.openssl.org (www.openssl.org)|104.84.200.72|:443... connected.
WARNING: cannot verify www.openssl.org's certificate, issued by ‘CN=R3,O=Let's Encrypt,C=US’:
  Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 200 OK
Length: 8350547 (8.0M) [application/x-gzip]
Saving to: ‘openssl-1.1.1a.tar.gz’

openssl-1.1.1a.tar.gz                   100%[==========================================================================================>]   7.96M  3.76MB/s    in 2.1s

2023-04-16 15:26:12 (3.76 MB/s) - ‘openssl-1.1.1a.tar.gz’ saved [8350547/8350547]

zlw@ubuntu:~/DailyCode/raspberry_study$ tar -xzf openssl-1.1.1a.tar.gz
zlw@ubuntu:~/DailyCode/raspberry_study$ ls
code  curl-7.71.1  curl-7.71.1.tar.bz2  libwiringPi.so  libwiringPi.so.2.50  openssl-1.1.1a  openssl-1.1.1a.tar.gz  test5  test5.c  tools-master  WiringPi
zlw@ubuntu:~/DailyCode/raspberry_study$ cd openssl-1.1.1a/
zlw@ubuntu:~/DailyCode/raspberry_study/openssl-1.1.1a$ ls
ACKNOWLEDGEMENTS  AUTHORS     config          Configure     demos    e_os.h    fuzz     LICENSE  NOTES.ANDROID  NOTES.UNIX  os-dep         README.FIPS  tools
apps              build.info  config.com      CONTRIBUTING  doc      external  include  ms       NOTES.DJGPP    NOTES.VMS   README         ssl          util
appveyor.yml      CHANGES     Configurations  crypto        engines  FAQ       INSTALL  NEWS     NOTES.PERL     NOTES.WIN   README.ENGINE  test         VMS
zlw@ubuntu:~/DailyCode/raspberry_study/openssl-1.1.1a$
```

看README，了解怎么安装

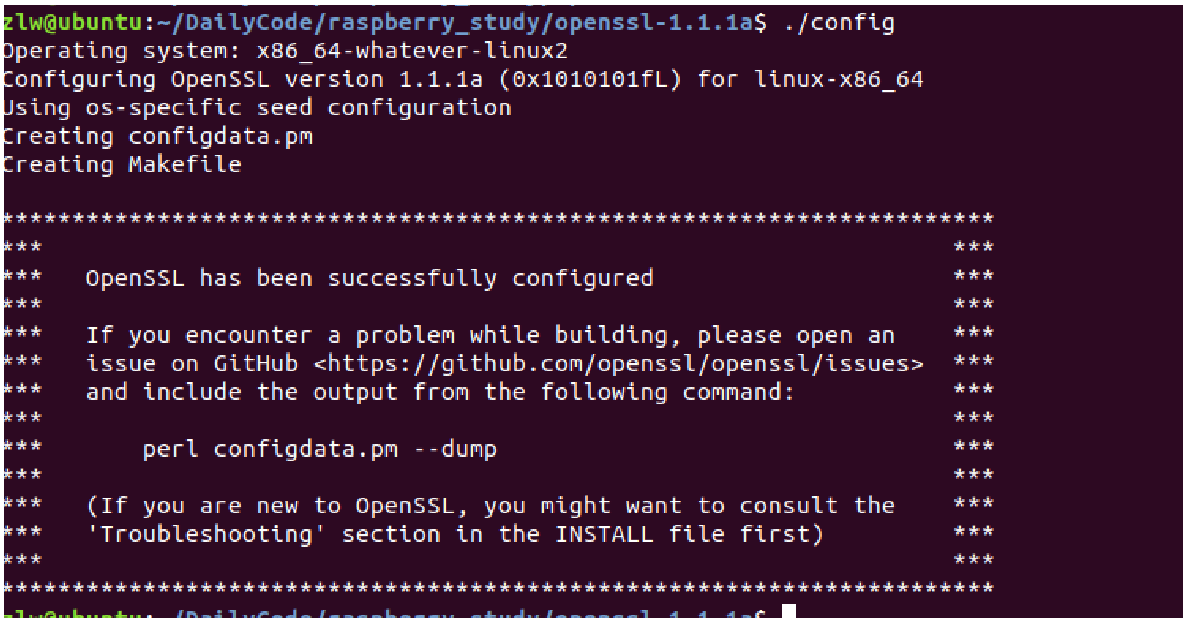
再去看INSTALL,看看怎么安装



```
specifications, which also use [ and ], but without spa

Quick Start
-----------

If you want to just get on with it, do:

 on Unix (again, this includes Mac OS/X):

   $ ./config
   $ make
   $ make test
   $ make install
```

安装到系统路径底下去

```
./config
```



```
zlw@ubuntu:~/DailyCode/raspberry_study/openssl-1.1.1a$ ./config
Operating system: x86_64-whatever-linux2
Configuring OpenSSL version 1.1.1a (0x1010101fL) for linux-x86_64
Using os-specific seed configuration
Creating configdata.pm
Creating Makefile

**********************************************************************
***                                                                ***
***   OpenSSL has been successfully configured                     ***
***                                                                ***
***   If you encounter a problem while building, please open an    ***
***   issue on GitHub <https://github.com/openssl/openssl/issues>  ***
***   and include the output from the following command:           ***
***                                                                ***
***       perl configdata.pm --dump                                ***
***                                                                ***
***   (If you are new to OpenSSL, you might want to consult the    ***
***   'Troubleshooting' section in the INSTALL file first)         ***
***                                                                ***
**********************************************************************
```

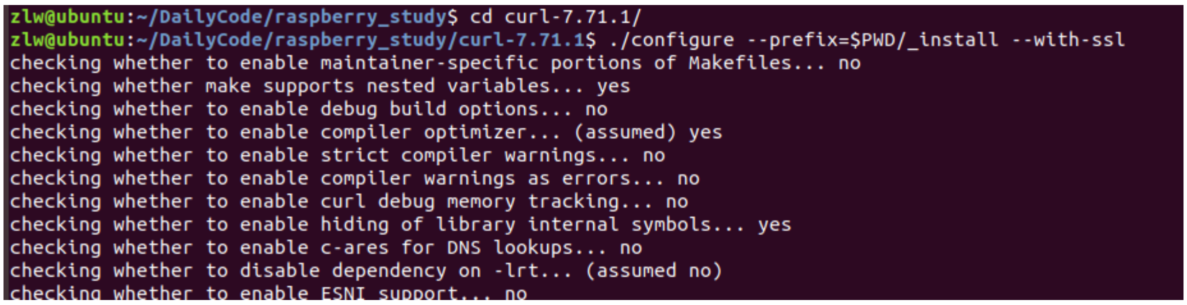
然后

```
make
```

最后

```
sudo make install
```

然后进入到cd curl-7.71.1/目录，重新进行编译curl

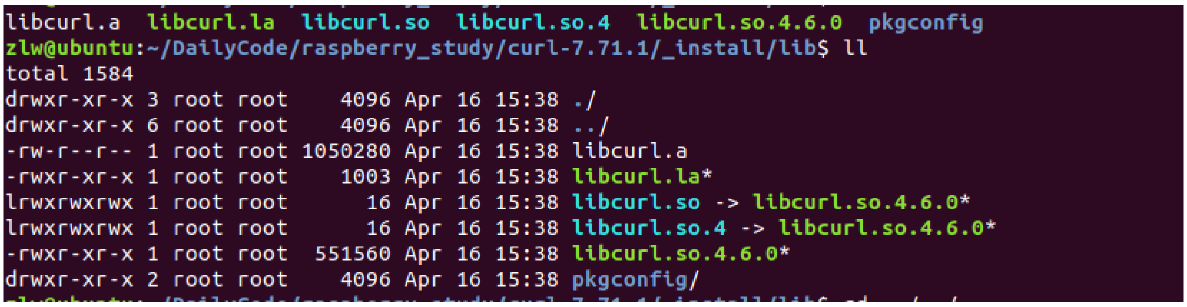


```
zlw@ubuntu:~/DailyCode/raspberry_study$ cd curl-7.71.1/
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1$ ./configure --prefix=$PWD/_install --with-ssl
checking whether to enable maintainer-specific portions of Makefiles... no
checking whether make supports nested variables... yes
checking whether to enable debug build options... no
checking whether to enable compiler optimizer... (assumed) yes
checking whether to enable strict compiler warnings... no
checking whether to enable compiler warnings as errors... no
checking whether to enable curl debug memory tracking... no
checking whether to enable hiding of library internal symbols... yes
checking whether to enable c-ares for DNS lookups... no
checking whether to disable dependency on -lrt... (assumed no)
checking whether to enable ESNI support... no
```

```
./configure --prefix=$PWD/_install --with-ssl
```

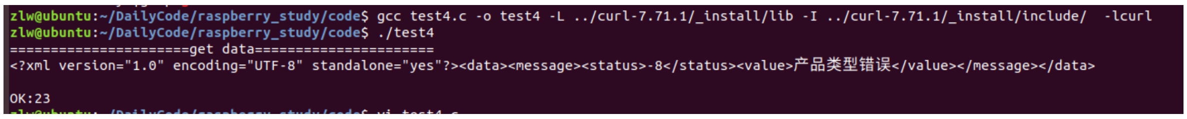
```
make
```

```
sudo make install
```



```
libcurl.a  libcurl.la  libcurl.so  libcurl.so.4  libcurl.so.4.6.0  pkgconfig
zlw@ubuntu:~/DailyCode/raspberry_study/curl-7.71.1/_install/lib$ ll
total 1584
drwxr-xr-x 3 root root    4096 Apr 16 15:38 ./
drwxr-xr-x 6 root root    4096 Apr 16 15:38 ../
-rw-r--r-- 1 root root 1050280 Apr 16 15:38 libcurl.a
-rwxr-xr-x 1 root root    1003 Apr 16 15:38 libcurl.la*
lrwxrwxrwx 1 root root      16 Apr 16 15:38 libcurl.so -> libcurl.so.4.6.0*
lrwxrwxrwx 1 root root      16 Apr 16 15:38 libcurl.so.4 -> libcurl.so.4.6.0*
-rwxr-xr-x 1 root root  551560 Apr 16 15:38 libcurl.so.4.6.0*
drwxr-xr-x 2 root root    4096 Apr 16 15:38 pkgconfig/
```

然后就可以运行代码来访问翔云平台（将代码的typeID改为typeId这样就解决了产品类型错误）



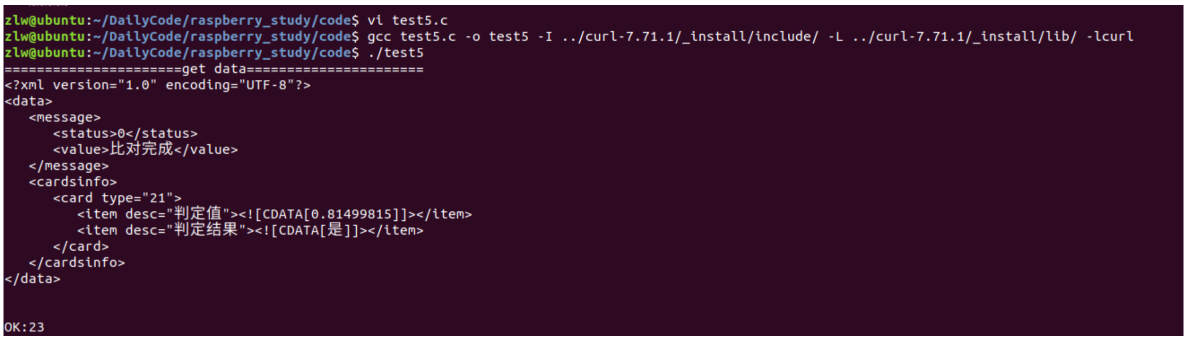
```
zlw@ubuntu:~/DailyCode/raspberry_study/code$ gcc test4.c -o test4 -L ../curl-7.71.1/_install/lib -I ../curl-7.71.1/_install/include/  -lcurl
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./test4
======================get data======================
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><data><message><status>-8</status><value>产品类型错误</value></message></data>

OK:23
```

编程实现人脸识别第二次加入图片base64编码

代码如下：

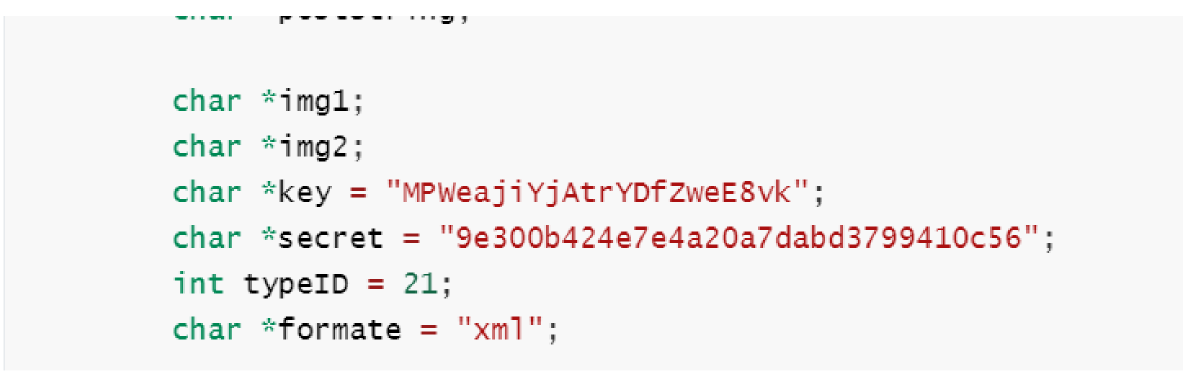
```c
#include <stdio.h>
#include <curl/curl.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>


#define true 1
#define false 0
typedef unsigned int bool;


size_t read_func( void *ptr, size_t size, size_t nmemb, void *stream)
{
        char buf[1024] = {0};
        strncpy(buf, ptr, 1024);
        printf("======================get data======================\n");
        printf("%s\n",buf);
}
bool postUrl()
{
        CURL *curl;
        CURLcode res;

        char *postString;

        char *img1;
        char *img2;
        char *key = "MPWeajiYjAtrYDfZweE8vk";
        char *secret = "9e300b424e7e4a20a7dabd3799410c56";
        int typeID = 21;
        char *formate = "xml";

        system("base64 pyy1.jpg > tmpFile1");
        system("base64 pyy2.jpg > tmpFile2");

        int fd1 = open("./tmpFile1",O_RDWR);
        int fd2 = open("./tmpFile2",O_RDWR);

        int fd1_len = lseek(fd1,0,SEEK_END);
        lseek(fd1,0,SEEK_SET);
        int fd2_len = lseek(fd2,0,SEEK_END);
        lseek(fd2,0,SEEK_SET);

        img1 = (char*)malloc(fd1_len+20);
        img2 = (char*)malloc(fd2_len+20);

        memset(img1,'\0',fd1_len+20);
        memset(img2,'\0',fd2_len+20);

        read(fd1,img1,fd1_len);
        read(fd2,img2,fd2_len);
        postString =
(char*)malloc(strlen(key)+strlen(secret)+fd1_len+fd2_len+1024);

        memset(postString,'\0',strlen(postString));

        //字符串拼接

 sprintf(postString,"&img1=%s&img2=%s&key=%s&secret=%s&typeId=%d&formate=%s",
                        img1,     img2,    key,    secret,   typeID,
 formate);

        curl = curl_easy_init();
        if (curl)
        {
                curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "/tmp/cookie.txt");
// 指定cookie文件
                curl_easy_setopt(curl, CURLOPT_POSTFIELDS, postString);    // 指
定post内容
                curl_easy_setopt(curl,
CURLOPT_URL,"https://netocr.com/api/faceliu.do");   // 指定url
                curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, read_func); //通过
回调函数存储数据
                res = curl_easy_perform(curl);

                printf("OK:%d\n",res);



                curl_easy_cleanup(curl);
        }
        free(img1);
        free(img2);
        free(postString);
        close(fd1);
        close(fd2);
        return true;
}
int main(void)
{
        postUrl();
}
```

结果:

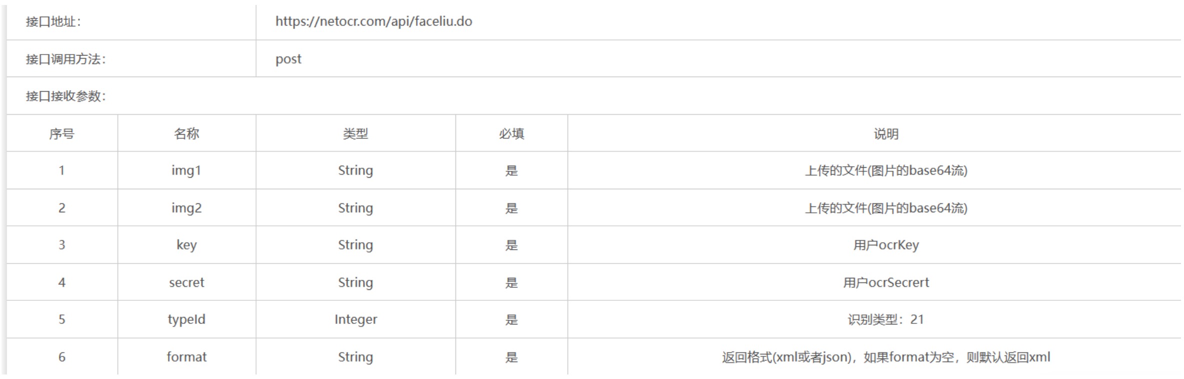


```
zlw@ubuntu:~/DailyCode/raspberry_study/code$ vi test5.c
zlw@ubuntu:~/DailyCode/raspberry_study/code$ gcc test5.c -o test5 -I ../curl-7.71.1/_install/include/ -L ../curl-7.71.1/_install/lib/ -lcurl
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./test5
======================get data======================
<?xml version="1.0" encoding="UTF-8"?>
<data>
   <message>
      <status>0</status>
      <value>比对完成</value>
   </message>
   <cardsinfo>
      <card type="21">
         <item desc="判定值"><![CDATA[0.81499815]]></item>
         <item desc="判定结果"><![CDATA[是]]></item>
      </card>
   </cardsinfo>
</data>


OK:23
```

代码中的这些内容要与网页的相对应



```
char *img1;
char *img2;
char *key = "MPWeajiYjAtrYDfZweE8vk";
char *secret = "9e300b424e7e4a20a7dabd3799410c56";
int typeID = 21;
char *formate = "xml";
```



接口地址:	https://netocr.com/api/faceliu.do			
接口调用方法:	post			
接口接收参数:				
序号	名称	类型	必填	说明
1	img1	String	是	上传的文件(图片的base64流)
2	img2	String	是	上传的文件(图片的base64流)
3	key	String	是	用户ocrKey
4	secret	String	是	用户ocrSecrert
5	typeId	Integer	是	识别类型: 21
6	format	String	是	返回格式(xml或者json), 如果format为空, 则默认返回xml

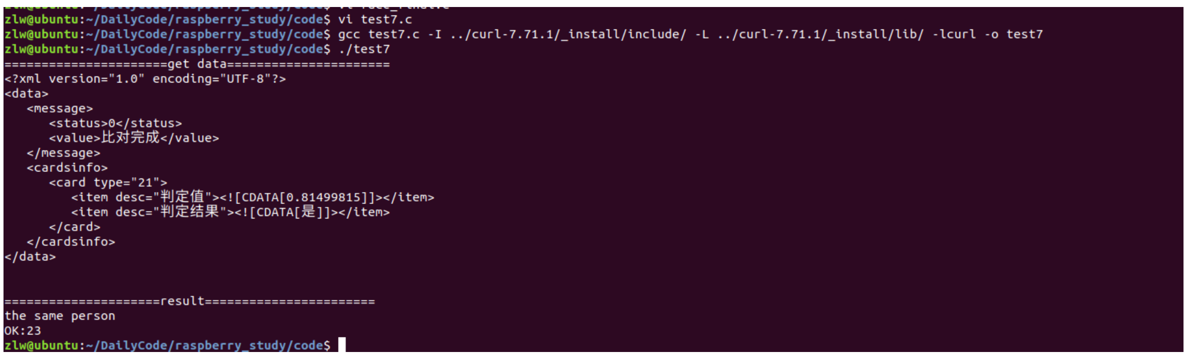
人脸识别成功并封装Base64编码函数

代码如下:

```
#include <stdio.h>
#include <curl/curl.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>


#define true 1
#define false 0
typedef unsigned int bool;


size_t read_func( void *ptr, size_t size, size_t nmemb, void *stream)
```

```
{
        char buf[1024] = {0};
        strncpy(buf, ptr, 1024);
        printf("======================get data======================\n");
        printf("%s\n",buf);

        printf("=====================result=======================\n");
        if(strstr(buf,"是") != NULL)
        {
                printf("the same person\n");
        }
        else
        {
                printf("different person\n");
        }

}
char* getPicBase64FromFile(char *filePath)
{

        char *bufPic;
        char cmd[128]={'\0'};

        sprintf(cmd,"base64 %s >tmpFile",filePath);

        system(cmd);

        int fd = open("./tmpFile",O_RDWR);
        int fd_len = lseek(fd,0,SEEK_END);
        lseek(fd,0,SEEK_SET);

        bufPic = (char*)malloc(fd_len+2);

        memset(bufPic,'\0',fd_len+2);

        read(fd,bufPic,fd_len);

        close(fd);

        system("rm -rf tmpFile");

        return bufPic;

}

bool postUrl()
{
        CURL *curl;
        CURLcode res;

        char *postString;

        char *img1;
        char *img2;
        char *key = "MPWeajiYjAtrYDfZweE8vk";
        char *secret = "9e300b424e7e4a20a7dabd3799410c56";
        int typeID = 21;
        char *formate = "xml";
        //图片转字符流

        img1 = getPicBase64FromFile("./pyy1.jpg");

        img2 = getPicBase64FromFile("./pyy2.jpg");


        postString =
(char*)malloc(strlen(key)+strlen(secret)+strlen(img1)+strlen(img2)+1024);
        memset(postString,'\0',strlen(postString));

        //字符串拼接

 sprintf(postString,"&img1=%s&img2=%s&key=%s&secret=%s&typeId=%d&formate=%s",
                        img1,     img2,     key,     secret,    typeID,
 formate);

        curl = curl_easy_init();
        if (curl)
        {
                curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "/tmp/cookie.txt");
// 指定cookie文件
                curl_easy_setopt(curl, CURLOPT_POSTFIELDS, postString);    // 指
定post内容
                curl_easy_setopt(curl,
CURLOPT_URL,"https://netocr.com/api/faceliu.do");   // 指定url
                curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, read_func); //通过
回调函数存储数据
                res = curl_easy_perform(curl);

                printf("OK:%d\n",res);



                curl_easy_cleanup(curl);
        }
        free(img1);
        free(img2);
        free(postString);
        return true;
}
int main(void)
{
        postUrl();
}
```

结果如下:



```
zlw@ubuntu:~/DailyCode/raspberry_study/code$ vi test7.c
zlw@ubuntu:~/DailyCode/raspberry_study/code$ gcc test7.c -I ../curl-7.71.1/_install/include/ -L ../curl-7.71.1/_install/lib/ -lcurl -o test7
zlw@ubuntu:~/DailyCode/raspberry_study/code$ ./test7
=====================get data=====================
<?xml version="1.0" encoding="UTF-8"?>
<data>
   <message>
      <status>0</status>
      <value>比对完成</value>
   </message>
   <cardsinfo>
      <card type="21">
         <item desc="判定值"><![CDATA[0.81499815]]></item>
         <item desc="判定结果"><![CDATA[是]]></item>
      </card>
   </cardsinfo>
</data>


====================result======================
the same person
OK:23
zlw@ubuntu:~/DailyCode/raspberry_study/code$ 
```