# [Ubuntu18.04安装FFmpeg](https://www.cnblogs.com/lonelamb/p/11620906.html)步骤

### 库的安装

### sudo apt-get install -y autoconf automake build-essential git libass-dev libfreetype6-dev libsdl2-dev libtheora-dev libtool libva-dev libvdpau-dev libvorbis-dev libxcb1-dev libxcb-shm0-dev libxcb-xfixes0-dev pkg-config texinfo wget zlib1g-dev

apt install libavformat-dev

apt install libavcodec-dev

apt install libswresample-dev

apt install libswscale-dev

apt install libavutil-dev

apt install libsdl1.2-dev

其中：

libavformat：用于各种音视频封装格式的生成和解析，包括获取解码所需信息以生成解码上下文结构  
和读取音视频帧等功能；  
libavcodec：用于各种类型声音/图像编解码；  
libavutil：包含一些公共的工具函数；  
libswscale：用于视频场景比例缩放、色彩映射转换；  
libpostproc：用于后期效果处理；  
ffmpeg：该项目提供的一个工具，可用于格式转换、解码或电视卡即时编码等；  
ffsever：一个 HTTP 多媒体即时广播串流服务器；  
ffplay：是一个简单的播放器，使用ffmpeg 库解析和解码，通过SDL显示；

### ffmpeg的安装

sudo apt-get install ffmpeg

### ffmpeg的测试

ffmpeg -version

出现以下结果安装成功！



# [FFmpeg](https://www.cnblogs.com/lonelamb/p/11620906.html)命令使用总结

1.推流命令

ffmpeg -re -i /Users/xu/Desktop/bangbangbang.mp4 -vcodec libx264 -acodec aac -f flv rtmp://localhost:1935/rtmplive/

2.从67以tcp方式拉流，并录像30s保存为flv格式。

ffmpeg -rtsp\_transport tcp -i rtsp://admin:jsepc123\!@192.168.1.67/h264/ch1/main/av\_stream -vcodec copy -acodec copy -t 30 -f flv rtsp-out.flv

参考文档：

1. <https://www.jianshu.com/p/d541b317f71c>
2. <https://www.linuxidc.com/Linux/2016-10/136545.htm>

# Linux（ubuntu 18.04）使用

#### g++编译：

（1）临时设置变量：export LD\_LIBRARY\_PATH=./lib

#### 查看端口占用：

netstat -apn | grep 2181

#### 修改IP地址：

##### 修改/etc/netplan/\*.yaml （文件名不一定）

network:

ethernets:

ens33:

addresses: [192.168.1.159/24]

gateway4: 192.168.1.1

dhcp4: yes

version: 2

##### 使得配置文件生效

netplan apply

#### 代码统计命令：

在代码根目录执行以下两命令

find . -name "\*.h" -print | xargs wc -l

find . -name "\*.cpp" -print | xargs wc -l

#### 两台linux服务器间文件或文件夹的传输(免密码)：

1. 在A机上生成公匙文件  
    ssh-keygen -t rsa  
   （下面的操作默认即可）  
    生成的公匙文件在/root/.ssh/目录下
2. 将文件拷贝到服务端B并改名  
    scp ./id\_dsa.pub  server\_ip:/root/.ssh/authorized\_keys
3. 在A端执行  
    ssh-agent
4. 使用scp命令进行传输

scp -r /home/uftp/ 192.168.1.111:/home/lcl/

其中：

/home/uftp/:源文件路径

192.168.1.111：目标服务器的IP地址

/home/lcl/：要传输到目标服务器的文件保存目录

注意：命令错误的时候就会有提示，如下

usage: scp [-12346BCpqrv] [-c cipher] [-F ssh\_config] [-i identity\_file]  
　　　　　　[-l limit] [-o ssh\_option] [-P port] [-S program]  
　　　　　　[[user@]host1:]file1 ... [[user@]host2:]file2

#### Pip3外网在线下载慢：

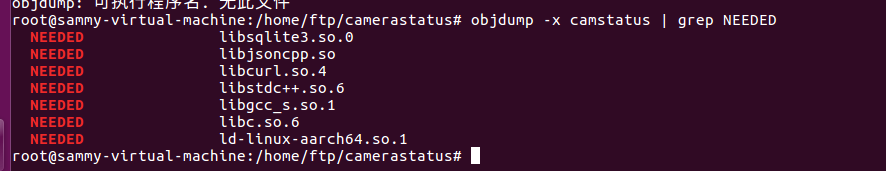
pip install --default-timeout=1000 --upgrade setuptools

#### ubuntu16.04允许root用户登陆ssh：

<https://jingyan.baidu.com/article/066074d615b8f4c3c31cb067.html>

#### 查看可执行文件的依赖库：

objdump -x camstatus | grep NEEDED



#### bash: ./cards\_main.py: /usr/bin/python3^M: 解释器错误: 没有那个文件或目录

sed -i 's/\r$//' cards\_main.py

# nginx + nginx-http-flv-module模块安装

#### 安装依赖包：

sudo apt-get update

sudo apt install build-essential

sudo apt install libtool

sudo apt install libpcre3 libpcre3-dev

sudo apt install zlib1g-dev

sudo apt-get install openssl libssl-dev

#### 安装稳定版nginx--目前是1.16.1

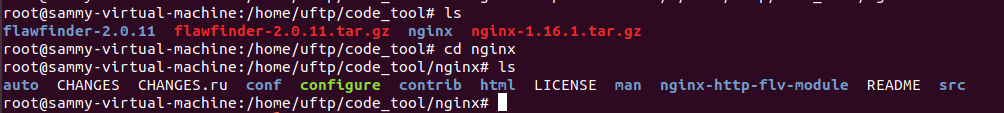
wget <https://nginx.org/download/nginx-1.16.1.tar.gz>

tar -zxvf nginx-1.16.1.tar.gz

mv nginx-1.16.1 nginx (改个名字)

#### Git下载nginx-http-flv-module模块到nginx目录下。

git clone <https://github.com/winshining/nginx-http-flv-module.git>

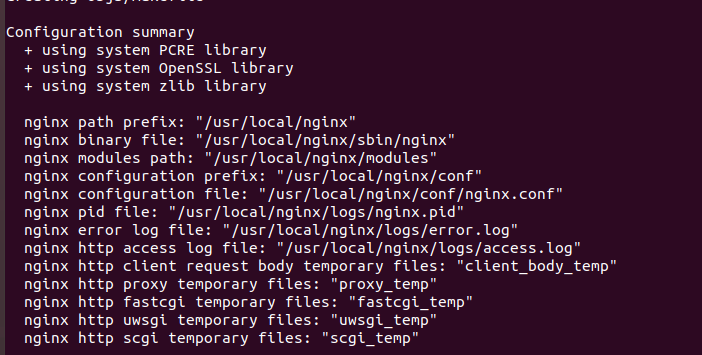


#### 带http-flv模板重新编译

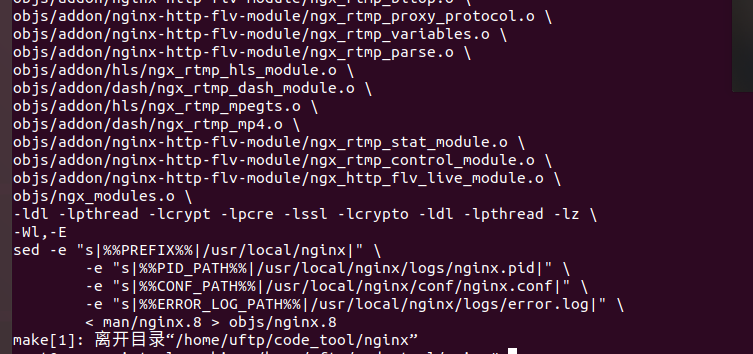
(1)cd nginx/

(2)./configure --add-module=/home/uftp/code\_tool/nginx/nginx-http-flv-module

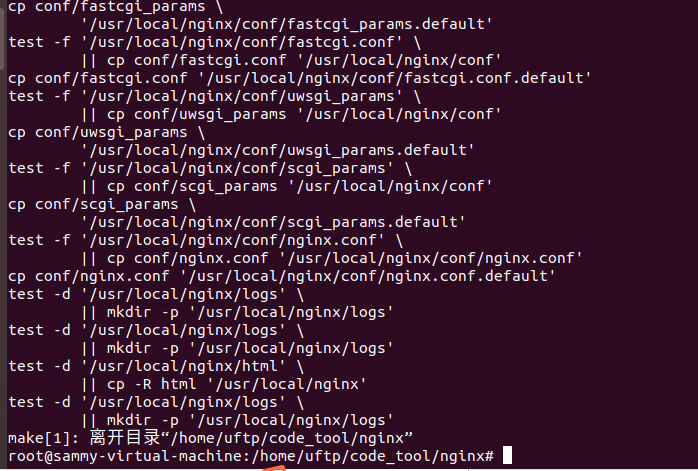
成功了会出现以下截图：



(3)make



(4)make install



#### 安装完成后 目录应该是 /usr/local/nginx 下；

其中 执行文件 ：/usr/local/nginx/sbin/nginx

默认配置文件 ：/usr/local/nginx/conf/nginx.conf

ps：如果是apt-get 方式 安装，应该是 /usr/sbin/nginx

#### 加入全局变量，修改"/etc/profile"文件：

export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/nginx/sbin

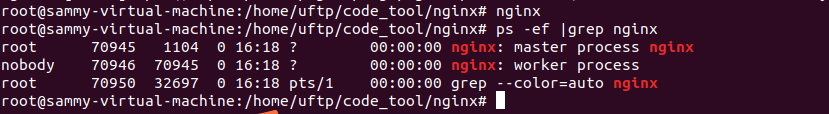
执行如下命令，使全局变量生效：

source /etc/profile

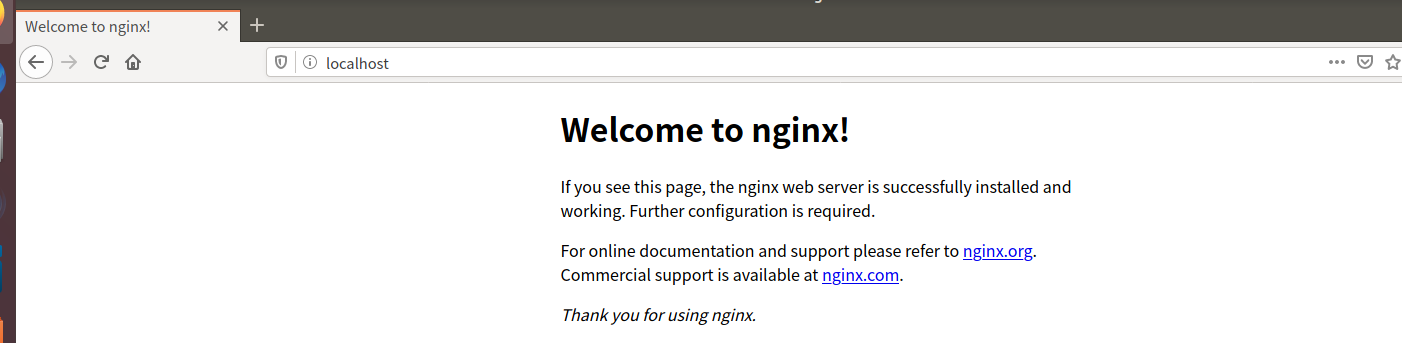
#### 启动nginx 并查看是否运行

nginx

ps -ef | grep nginx



#### 浏览器打开http://localhost,有如下页面表面nginx正在运行。



#### 修改nginx配置文件（很重要很容易出问题）

如5.所说默认配置文件路径 ：/usr/local/nginx/conf/nginx.conf

注意：编码需要改成utf-8

worker\_processes 1;

error\_log logs/error.log error;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

location /live {

flv\_live on; #open flv live streaming (subscribe)

chunked\_transfer\_encoding on; #open 'Transfer-Encoding: chunked' response

add\_header 'Access-Control-Allow-Origin' '\*'; #add additional HTTP header

add\_header 'Access-Control-Allow-Credentials' 'true'; #add additional HTTP header

}

location /hls {

types {

application/vnd.apple.mpegurl m3u8;

video/mp2t ts;

}

root /tmp;

add\_header 'Cache-Control' 'no-cache';

}

location /dash {

root /tmp;

add\_header 'Cache-Control' 'no-cache';

}

location /stat {

#configuration of push & pull status

rtmp\_stat all;

rtmp\_stat\_stylesheet stat.xsl;

}

location /stat.xsl {

root /var/www/rtmp; #specify in where stat.xsl located

}

location /control {

rtmp\_control all; #configuration of control module of rtmp

}

}

# another virtual host using mix of IP-, name-, and port-based configuration

#

#server {

# listen 8000;

# listen somename:8080;

# server\_name somename alias another.alias;

# location / {

# root html;

# index index.html index.htm;

# }

#}

# HTTPS server

#

#server {

# listen 443 ssl;

# server\_name localhost;

# ssl\_certificate cert.pem;

# ssl\_certificate\_key cert.key;

# ssl\_session\_cache shared:SSL:1m;

# ssl\_session\_timeout 5m;

# ssl\_ciphers HIGH:!aNULL:!MD5;

# ssl\_prefer\_server\_ciphers on;

# location / {

# root html;

# index index.html index.htm;

# }

#}

}

rtmp\_auto\_push on;

rtmp\_auto\_push\_reconnect 1s;

rtmp\_socket\_dir /tmp;

rtmp {

out\_queue 4096;

out\_cork 8;

max\_streams 128;

timeout 15s;

drop\_idle\_publisher 15s;

log\_interval 5s; #interval used by log module to log in access.log, it is very useful for debug

log\_size 1m; #buffer size used by log module to log in access.log

server {

listen 1935;

server\_name www.test.\*; #for suffix wildcard matching of virtual host name

application myapp {

live on;

gop\_cache on; #open GOP cache for reducing the wating time for the first picture of video

}

application hls {

live on;

hls on;

hls\_path /tmp/hls;

}

application dash {

live on;

dash on;

dash\_path /tmp/dash;

}

}

server {

listen 1935;

server\_name \*.test.com; #for prefix wildcard matching of virtual host name

application myapp {

live on;

gop\_cache on; #open GOP cache for reducing the wating time for the first picture of video

}

}

server {

listen 1935;

server\_name www.test.com; #for completely matching of virtual host name

application myapp {

live on;

gop\_cache on; #open GOP cache for reducing the wating time for the first picture of video

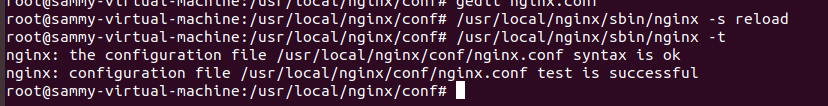
}

}

}

#### 重新加载：/usr/local/nginx/sbin/nginx -s reload

测试配置文件修改是否正常：/usr/local/nginx/sbin/nginx -t



#### 利用ffmpeg推流

ffmpeg -rtsp\_transport tcp -i "rtsp://admin:jsepc123@192.168.1.67/h264/ch1/main/av\_stream" -vcodec copy -acodec aac -ar 44100 -strict -2 -ac 1 -f flv -s 1920x1080 -q 10 -f flv "rtmp://192.168.229.129:1935/myapp/mystream"

（注意：目前利用ffmpeg推rtsp流发现密码中不能包含特殊字符，否则会报401错误.）

#### 利用VLC可查看视频是否正常播放

VLC用以下两种方式查看视频流均可：

（1）<http://192.168.229.129/live?port=1935&app=myapp&stream=mystream>

（2）rtmp://192.168.229.129:1935/myapp/mystream



# Ngnix常见问题

#### nginx重启时报错问题：

nginx: [error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)错误；

解决方法： /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf

# Ssdb（类似可替换redis）安装说明

具体可参照<https://www.jianshu.com/p/c90a9994e872>

注意：在make之前依赖autoconf

sudo apt-get install autoconf

# fusion director（FD）

华为软件升级和模型分发平台

https://support.huawei.com/enterprise/zh/management-software/fusiondirector-pid-23015464/doc

User： jsshengtong

Passport：st\*87853776

# Huawei技术文档

HuaweiAtlas/samples：

<https://gitee.com/HuaweiAtlas/samples>

**https://support.huawei.com/enterprise**

# aarch64-linux-gnu-gcc交叉编译sqlite3

1.下载sqlite3源码：<https://sqlite.org/2018/sqlite-autoconf-3260000.tar.gz>

2.用aarch64-linux-gnu-gcc交叉编译，命令如下：

tar -vxzf sqlite-autoconf-3320000.tar.gz

cd sqlite-autoconf-3320000/

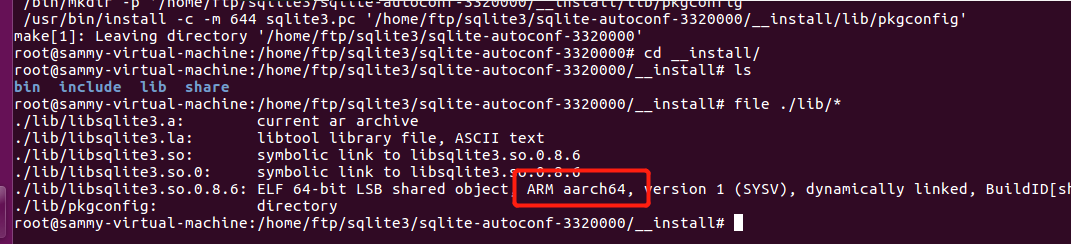
mkdir \_\_install

./configure --host=arm-linux --prefix=$PWD/\_\_install CC=aarch64-linux-gnu-gcc

make

make install

3.查看交叉编译出来的库：file \_\_install/lib/\*



4.拷贝库到交叉编译链：

cp -af ./lib/\* /usr/aarch64-linux-gnu/lib/

cp -af ./include/\* /usr/aarch64-linux-gnu/include/

# aarch64-linux-gnu-gcc交叉编译curl

1.下载sqlite3源码:https://curl.haxx.se/download.html

2.用aarch64-linux-gnu-gcc交叉编译，命令如下：

tar -vxf curl-7.70.0.tar.xz

cd curl-7.70.0/

mkdir \_\_install

./configure --with-ssl\

--disable-shared\

--enable-static\

--disable-dict\

--disable-ftp\

--disable-imap\

--disable-ldap\

--disable-ldaps\

--disable-pop3\

--disable-proxy\

--disable-rtsp\

--disable-smtp\

--disable-telnet\

--disable-tftp\

--disable-zlib\

--without-ca-bundle\

--without-gnutls\

--without-libidn\

--without-librtmp\

--without-libssh2\

--without-nss\

--without-zlib\

--enable-shared\

--enable-static\

--host=arm-linux\

CC=aarch64-linux-gnu-gcc --prefix=$PWD/\_\_install

make && make install

3.查看交叉编译出来的库：file \_\_install/lib/\*

4.拷贝库到交叉编译链：

cp -af ./lib/\* /usr/aarch64-linux-gnu/lib/

cp -af ./include/\* /usr/aarch64-linux-gnu/include/

# A500相关

1.显卡信息查询：npu-smi info

2.

# Mysql表设计demo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 中文名 | Key否 | 备注 |
| **[Province](https://cn.bing.com/dict/search?q=province&FORM=BDVSP6&mkt=zh-cn)ID** | **tinyint** | 省份代码 |  | 为以后扩展用。可把江苏省写为如：33 |
| **CityID** | **tinyint** | 市代码 |  | 13市分别定义为1~13 |
| **NetID** | **int** | 营业厅编号 |  |  |
| **CameraID** | **int** | 摄像机编号 |  |  |
| **EventType** | **tinyint** | 事件类型 |  |  |
| **IPAddress** | **bigint** |  |  |  |
| **TimeStamp** | **VARCHAR(20)** | 时间戳 |  |  |
|  |  |  |  |  |