

更新于2020.10.8

写在开始：以下所有操作默认使用root账号完成

挂载新硬盘并迁移home目录

一般服务器是为了节省成本的同时提升性能，将系统安装在固态硬盘上，将数据放在机械硬盘上，这里是将大小为8T的硬盘挂载到home目录下，以解决服务器内存不够的问题

查找磁盘信息

查看硬盘信息：使用命令 `lsblk -C disk`

其中8T的硬盘就是需要挂载的。

```
1  *-disk:0
2      description: SCSI Disk
3      product: MR9361-8i
4      vendor: AVAGO
5      physical id: 2.0.0
6      bus info: scsi@6:2.0.0
7      logical name: /dev/sda
8      version: 4.68
9      serial: 003bd7df0683ae9b24b06edd0eb00506
10     size: 223GiB (239GB)
11     capabilities: partitioned partitioned:dos
12     configuration: ansiversion=5 logicalsectorsize=512 sectorsize=4096
13     signature=2d550c58
14  *-disk:1
15     description: SCSI Disk
16     product: MR9361-8i
17     vendor: AVAGO
18     physical id: 2.1.0
19     bus info: scsi@6:2.1.0
20     logical name: /dev/sdb
21     version: 4.68
22     serial: 0049b4430789ae9b24b06edd0eb00506
23     size: 7451GiB (8TB)
24     capabilities: gpt-1.00 partitioned partitioned:gpt
25     configuration: ansiversion=5 guid=57d31fee-70a0-41f4-bcb9-b7b72f94fac4
26     logicalsectorsize=512 sectorsize=512
```

查看分区情况：使用命令 `fdisk -l | grep sd`

```
1  Partition 2 does not start on physical sector boundary.
2  Disk /dev/sda: 223.1 GiB, 239511535616 bytes, 467795968 sectors
3  /dev/sda1 *          2048  1499135  1497088  731M 83 Linux
4  /dev/sda2          1501182 467793919 466292738 222.4G 5 Extended
5  /dev/sda5          1501184 467793919 466292736 222.4G 8e Linux LVM
6  Disk /dev/sdb: 7.3 TiB, 8000450330624 bytes, 15625879552 sectors
7  /dev/sdb1  2048 15625879518 15625877471 7.3T Linux filesystem
```

可以看到所有的存储设备，这里的sdb就是需要挂载到home的硬盘，这是已经挂载后的信息，其中sdb1就是将sdb中所有的内存划分出来的。

挂载前设置硬盘

硬盘小于2T设置方法

使用命令：`fdisk /dev/sdb`

得到

```
1  welcome to fdisk (util-linux 2.27.1).
2  Changes will remain in memory only, until you decide to write them.
3  Be careful before using the write command.
4
5
6  Command (m for help):
```

接着就是一些列命令行交互输入，输入 m 可看到所有的可用输入。大概知道了怎么操作之后，跟着下文的提示操作即可。

- 输入 p 查看 `/dev/sdb` 分区的状态
- 输入 n 创建 `sdb` 这块硬盘的分区
- 选 `p primary` => 输入 `p`
- Partition number => 全部存储分一个区，所以输入 `1``
- 接下来选项默认即可
- 最后一步输入 `w` 保存并退出 Command 状态

硬盘大于2T设置方法

使用命令：`parted /dev/sdb`

得到：

```
1  GNU Parted 3.2
2  Using /dev/sdb
3  welcome to GNU Parted! Type 'help' to view a list of commands.
4  (parted)
```

- 输入查看 `/dev/sdb` 分区的状态
- 输入 `mklabel gpt` 选择为 `yes`，将硬盘转化为gpt格式，从而分区能大于2T
- 接下来输入 `mkpart` 进行分区
- 分别需要输入分区名，这里分一个区为 `sdb1`
- 接下来是类型 `ext4/ext3` 都可以（我也不是很清楚区别）
- 再然后是内存大小从0开始，到最大内存
- 出现 `Ignore/Cancel` 选择 `Ignore`

这时用p查看可以看到一个 `sdb1` 空间



分区成功选择格式进行格式化

```
1  mkfs.ext3 /dev/sdb1 或 mkfs.ext4 /dev/sdb1
```

迁移旧的 home 目录文件到新硬盘

首先，你得挂载已经分区好的硬盘，然后把 `home` 目录下的全部文件拷贝到硬盘挂载的目录下。然后删除 `home` 目录，最后把第一步挂载好的新硬盘重新挂载在 `home` 目录下。具体步骤如下：

这部分的内容得使用 root 用户登录主机，因为涉及到把 home 目录删除，所有的非 root 用户都会失效。

挂载设置好的硬盘

```
1 | mkdir /mnt/tmp
2 | mount /dev/sdb1 /mnt/tmp
```

同步 home 目录所有文件，删除之前的 home 目录下的所有文件

```
1 | rsync -avx /home/ /mnt/tmp
```

确定同步成功之后，删除旧 home 目录

```
1 | rm -rf /home/*
2 | umount -l /home
```

重新挂载新硬盘并设置启动挂载

```
1 | mount /dev/sdb1 /home
```

设置系统启动挂载需要得到硬盘的信息：使用命令 `blkid` 得到

```
1 | /dev/sda1: UUID="74c4ba93-7b9b-44d0-877c-acb174742db1" TYPE="ext2"
PARTUUID="2d550c58-01"
2 | /dev/sda5: UUID="jFepng-8fBb-40CJ-JwOv-y9On-uVfY-xPakte" TYPE="LVM2_member"
PARTUUID="2d550c58-05"
3 | /dev/sdb1: UUID="ef3d534d-b536-4e8f-b0b6-c44e3f004839" TYPE="ext3"
PARTUUID="8a7f400b-8dbf-4eb4-b9a9-16beb2dfdcfd"
4 | /dev/mapper/IPL--120--vg-root: UUID="f48a9fa5-16c9-4603-899c-179281aea5f8"
TYPE="ext4"
5 | /dev/mapper/IPL--120--vg-swap_1: UUID="4b96ad21-63d4-41d0-8f4a-d28f53206db1"
TYPE="swap"
```

获取到的 sdb1 的 UUID 和 TYPE 在下文会用到。

接着编辑 `/etc/fstab` 文件,把以下代码添加到最后

```
1 | UUID=ef3d534d-b536-4e8f-b0b6-c44e3f004839 /home ext3 defaults
0 2
```

结束

运行 `df -h` 即可看到我们新挂载在 home 目录的硬盘设备

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	63G	0	63G	0%	/dev
tmpfs	13G	22M	13G	1%	/run
/dev/mapper/IPL--120--vg-root	218G	110G	97G	54%	/
tmpfs	63G	1.2M	63G	1%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	63G	0	63G	0%	/sys/fs/cgroup
/dev/sda1	720M	111M	573M	17%	/boot
/dev/sdb1	7.3T	218G	6.7T	4%	/home

网络

服务器外网配置

参考教程：[服务器外网IP设置](#) [ubuntu实现SSH外网连接内网](#)

在服务器外网配置开始前要准备一台有公网IP的电脑，这里采用的是购买阿里云的云服务器ECS，学生机有很大的优惠，预装Ubuntu16.04系统

阿里云服务器配置出站

首先，登录阿里云，进入控制台



在更多中选择网络与安全组中的安全组配置



接着选择配置规则



操作

修改 | 克隆 | 还原规则

管理实例 | 配置规则 | 管理弹性网卡

共有1条, 每页显示: 10条

<<

<

1

>

>>

再添加安全组规则

费用 工单 备案 企业 支持与服务 消息 通知 购物车 帮助 首页 简体中文

教我设置

返回

添加安全组规则

快速创建规则

导入规则

导出全部规则

描述	优先级	创建时间	操作
-	1	2019年7月7日 16:10	<div>修改 克隆 删除</div>

网卡类型:

内网

规则方向:

入方向

授权策略:

允许

协议类型:

全部

* 端口范围:

-1/-1

?

优先级:

1

?

授权类型:

IPv4地址段访问

* 授权对象:

0.0.0.0/0

?

教我设置

描述:

长度为2-256个字符，不能以http://或https://开头。

确定

取消

配置外网

方法一

实验室之前使用的主要方法，但是由于Ubuntu14.04版本过于旧的原因，无法使用所以找到另一种方法
现在先介绍第一种方法

1. 本地服务配置

```
1 apt install golang-go
2 git clone https://github.com/huanghailiang/popub-byma.git
3 cd popub-byma/
4 ls
5 make
6 ls
7 make install
8 systemctl status popub-relay@config
9 cd /etc/popub/local/
10 ls
11 cp example.conf srt.conf
```

然后编辑本地服务器上的配置文件 `srt.conf`（文件保存位置随意），添加：

```
1 LOCAL_ADDR=:22
2 RELAY_ADDR=my.server.addr:中继服务器端口号
3 AUTH_KEY=SomePassword
```

`my.server.addr` 改成中继服务器即阿里云服务器的公网 IP 地址，由于需要设置对台机器，冒号后面的端口设置时需检查是否与其他服务器上的配置冲突；

`SomePassword` 任意设置，尽量复杂一些，长一些，并复制下来，将阿里云上的 `SomePassword` 设置为相同。

2. 阿里云服务器设置

使用命令

```
1 cd /etc/popub/relay
2 cp example.conf srt-xxx.conf #xxx自己设置，用于分辨不同服务器
```

编辑文件

```
1 RELAY_ADDR=: 此处应与本地服务器上设置的端口号一样
2 PUBLIC_ADDR=: 给服务器设置一个没有被占用的端口号
3 AUTH_KEY=将原本设置的密码黏贴上来即可
```

3. 激活外网IP

本地服务器

```
1 systemctl start popub-local@srt.service
```

阿里云服务器

```
1 systemctl start popub-relay@srt-xxx.service
```

接着使用

```
1 systemctl enable popub-local@srt.service
2 #以及
3 systemctl enable popub-relay@srt-xxx.service
```

使 Popub 在每次服务器重启后自动运行。

方法二

1. 准备

一台内网机器 A

- IP: 192.168.1.120
- SSH端口: 22
- 用户名: a
- 密码: passworda
- 内网配置端口: 22 (即配置 SSH 端口的反向隧道)

带有公网ip的机器 B

- IP: 106.15.187.212
- SSH端口: 22
- 用户名: b
- 密码: passwordb
- 公网端口: 22001 (即用 B 的 22001 端口连到 A 的 SSH 22 端口)

端口可以更改, 从而实现多台机器的外网配置

2. 配置SSH密钥

在 A 主机上生成 SSH 密钥, 和 B 用 SSH 建立认证。

```
1 | ssh-keygen -t rsa -C "your@email.com"
```

接着直接连按3次enter就可以了, 什么都不用输入, 密钥自动保存在默认的地方, 方便下一步操作。

你会获得一长串 `SHA256:.....` 注意不要清屏了, 这个等下有用

然后利用如下命令将 A 的 SSH 密钥即 `SHA256:.....` 里面的字符串, 添加到 B 的 `authorized_keys` 里面:

```
1 | ssh-copy-id b@106.15.187.212
```

注意换成你自己的ip, 执行后会提示输入主机 B 的密码, 执行完毕之后, 我们登录到 B, 就发现 `authorized_keys` 里面就多了 A 的 SSH 公钥了, 成功建立 SSH 认证。

这个步骤主要是完成了从机子A到B `ssh` 的免密登录, 具体可以看 `~/.ssh/` 下面的一些密钥更改情况

在公网上配置

编辑 `/etc/ssh/sshd_config`, 添加:

```
1 | GatewayPorts yes
```

这句话的意思是监听端口可以绑定到任意其他ip, 不然只有本机127.0.0.1可以访问

重启一下 `sshd` 服务

```
1 | service sshd restart
```

A的配置

安装 `autossh`:

```
1 | apt install autossh
```

开启 `autossh`

```
1 | autossh -M 22002 -Nfr 0.0.0.0:22001:localhost:22 b@106.15.187.212
```

这里 `-M` 后面任意填写一个可用端口即可, `-N` 代表只建立连接, 不打开shell, `-f` 代表建立成功后在后台运行, `-R` 代表指定端口映射。

这里是将 A 主机的 22 端口映射到 B 主机的 22001 端口, 这样就完成了配置。

主要我们再访问 B 主机的 22001 端口, 就会自动转发到 A 主机的 22 端口了, 即可以公网访问了。

另外，为了方便，可以考虑将 `autossh` 设置为开机自启动

```
1 vi /etc/rc.local
```

添加下面上面开启 `autossh` 代码

```
1 autossh -M 22002 -Nfr 0.0.0.0:22001:localhost:22 b@106.15.187.212
```

终端翻墙到国外

代理的途径有两种情况：

1. 若走本地电脑的ss，IP地址默认为127.0.0.1
2. 若走局域网内其他电脑的ss，并且那台电脑的ss软件开启了“允许来自局域网的连接”的功能，则IP地址为那台电脑的内网IP地址。例如10.168.1.176

端口号默认都是1080。

这里使用的是xzf的windows开启的代理。如果换电脑了需要更改。

```
1 ##### Linux #####
2 # 将如下内容添加到/etc/bash.bashrc
3 IP=10.168.1.182
4 Port=7890
5 proxyon(){
6     export http_proxy="http://${IP}:${Port}"
7     export https_proxy="https://${IP}:${Port}"
8     # export http_proxy="socks5://${IP}:${Port}"
9     # export https_proxy="socks5://${IP}:${Port}"
10    echo "proxy on, and IP is $(curl ip.sb)"
11 }
12 proxyoff(){
13     unset http_proxy
14     unset https_proxy
15     echo "proxy off"
16 }
17
18 ##### windows #####
19 set http_proxy=http://10.168.1.182:7890
20 set https_proxy=https://10.168.1.182:7890
21 curl.exe ip.sb
22 # windows的我不知道怎么设置变量
23 # 取消为
24 set http_proxy=
25 set https_proxy=
```

Linux，可以在终端中运行命令 `proxyon` 打开代理，只对当前终端有效。想要取消的话运行命令 `proxyoff`

Windows则直接在终端中输入上面的命令，同样只对当前终端有效。打开代理后

打开代理后，会输出一个ip地址，可以自己去网上查一下这个IP地址的位置，如果不是在大陆的话说明代理成功。

参考教程：[让 Zsh 终端走代理](#)

终端翻墙到国内

参考教程: [小白也能轻松在 VPS 搭建 Shadowsocks 出海](#)

为方便实验室出国访学同学, 连接国内服务器, 不需要阿里云服务器, 只需要实验室服务器即可下载shadowsocks

```
1 wget --no-check-certificate -O shadowsocks-all.sh
  https://raw.githubusercontent.com/teddysun/shadowsocks_install/master/shadowsocks-all.sh
```

给文件权限并运行

```
1 chmod +x shadowsocks-all.sh
2 ./shadowsocks-all.sh 2>&1 | tee shadowsocks-all.log
```

选择脚本 (Python、R、Go、libev) , 任选一个:

```
1 which shadowsocks server you'd select:
2 1.Shadowsocks-Python
3 2.ShadowsocksR
4 3.Shadowsocks-Go
5 4.Shadowsocks-libev
6 Please enter a number (default 1):
```

我选择Shadowsocks-Go, 输入3.....然后, 输入密码和端口,

```
1 You choose = Shadowsocks-Go
2
3 # 这里设置密码, 直接回车的话就默认teddysun.com, 强烈建议更改, 这也是后面的Shadowsocks客户端登录密码。
4 Please enter password for Shadowsocks-Go
5 (default password: teddysun.com):
6
7 # 这里设置端口, 默认的是8989,
8 Please enter a port for Shadowsocks-Go [1-65535]
9 (default port: 8989):
10 port = 8989
11
12 Please select stream cipher for Shadowsocks-Python:
13
14 1) aes-256-gcm
15 2) aes-192-gcm
16 3) aes-128-gcm
17 4) aes-256-ctr
18 5) aes-192-ctr
19 6) aes-128-ctr
20 7) aes-256-cfb
21 8) aes-192-cfb
22 9) aes-128-cfb
23 10) camellia-128-cfb
24 11) camellia-192-cfb
25 12) camellia-256-cfb
26 13) xchacha20-ietf-poly1305
27 14) chacha20-ietf-poly1305
```

```
28 15) chacha20-ietf
29 16) chacha20
30 17) salsa20
31 18) rc4-md5
32 which cipher you'd select(Default: aes-256-gcm):
```

选择通信加密方式，千万不要选择默认的，强烈建议使用18) rc4-md5，默认的话很可能被封。

安装成功后，命令行出现：

```
1 Congratulations, Shadowsocks-Go server install completed!
2 Your Server IP      : 45.32.73.59
3 Your Server Port    : 8989
4 Your Password       : teddysun.com
5 Your Encryption Method: aes-256-cfb
6
7 Welcome to visit: https://teddysun.com/486.html
8 Enjoy it!
```

如果安装失败，请尝试其他脚本

下载Shadowsocks客户端：windows客户端下载：<https://github.com/shadowsocks/shadowsocks-windows/releases>

安卓手机以及mac都可以下载该客户端

iPhone上可以下载 Potatso Lite 但要求 app store 要在非大陆地区，切换方法<https://support.apple.com/HT201389>

或者花钱借个账号下载 <http://16bing.com/2017/12/24/iphone-potatso-lite/>

固定IP地址

到路由器云管理后台，网络设置 -> DHCP设置 -> DHCP静态分配。

开启SSH服务远程登陆

```
1 apt install openssh-server
```

SSH禁用密码登陆，使用密钥登录

准备：用xshell远程连接，然后切换到root账户

编辑 /etc/ssh/sshd_config 文件

```
1 #启用密钥登陆
2 RSAAuthentication yes
3 PubkeyAuthentication yes
4 #禁止root远程登陆
5 PermitRootLogin no
6 #禁止密码登陆
7 PasswordAuthentication no
```

然后登录到打算使用私钥登录的账户（接下来就不要换到root了），执行命令 `ssh-keygen`，在选择路径的时候直接按 Enter

接着输入密钥锁码，或直接按 Enter 留空（建议留空，实现无密码登录）。然后进入前面保存密钥的路径。

```
1 #在服务器上安装公钥
2 cat id_rsa.pub >> authorized_keys
3 #赋予文件权限
4 chmod 600 authorized_keys
5 chmod 700 ~/.ssh
6 #使用xshell的可以使用sz命令将私钥传输到本地，格式为sz 文件名
7 sz id_rsa
8 #或者使用scp命令传输到其他服务器，比如
9 scp id_rsa xzf@192.168.1.102:~/id_rsa
```

弄好以后长这样

```
xzf@srt-102:~/.ssh$ ll
total 20
drwx----- 2 xzf xzf 4096 Feb 21 02:32 ./
drwxr-xr-x 12 xzf xzf 4096 Feb 21 02:29 ../
-rw----- 1 xzf xzf 393 Feb 21 02:32 authorized_keys
-rw----- 1 xzf xzf 1679 Feb 21 02:29 id_rsa_xzf
-rw-r--r-- 1 xzf xzf 393 Feb 21 02:29 id_rsa_xzf.pub
```

私钥、公钥可以导出来后重命名也可也在里面重命名

参考教程：[设置 SSH 通过密钥登录](#)、[ssh修改登录端口禁止密码登录并免密登录](#)

命令行美化

在个人环境变量 `~/.bashrc` 的最后添加如下代码：

```
1 #To beautify the bash.
2 #Added by xzf
3 #2019.1.31
4 export PS1="\[\e[36;1m\]\u\[\e[0m\]@\[\e[33;1m\]\h\[\e[0m\]:\[\e[31;1m\]\w\[\e[0m\]\$ "
```

如果要加换行的话加转义字符
添加好后重新注入环境变量

```
1 source ~/.bashrc
```

建议谁要用给谁弄，不要添加到系统环境变量
参考教程：[Bash美化](#)、[Linux终端bash美化教程](#)

中文支持

1. 安装中文语言包

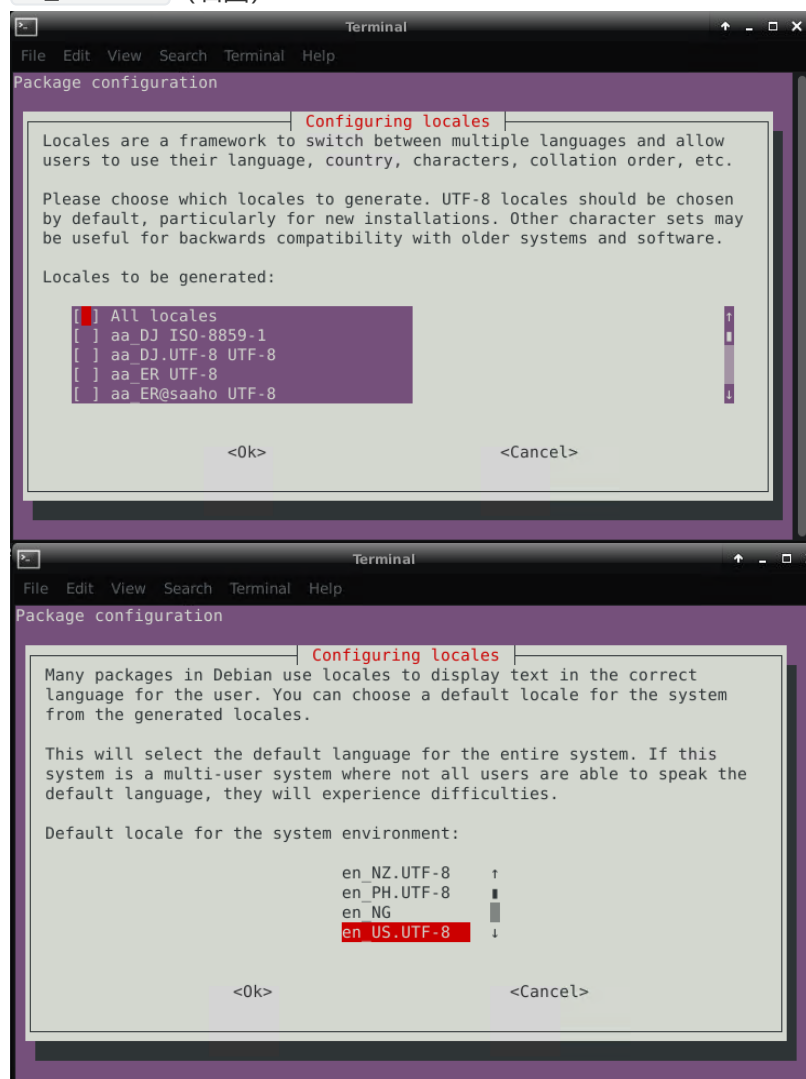
```
1 apt install language-pack-zh-hans*
```

2. 勾选中文编码：

方法一：

```
1 dpkg-reconfigure locales
```

在 zh_CN.GBK GBK 和 zh_CN.UTF-8 UTF-8 前面打勾将其添加到系统（左图），然后系统语言还是选择 en_US.UTF-8（右图）



方法二：

编辑 `/etc/locale.gen`，将 zh_CN.GBK GBK 和 zh_CN.UTF-8 UTF-8 的注释去掉，然后运行

```
1 | locale-gen
```

3. 安装中文字体：

```
1 | #都是网上查的
2 | apt install fonts-droid-fallback fonts-wqy-zenhei fonts-wqy-microhei fonts-
   | arphic-ukai fonts-arphic-uming fonts-noto fonts-noto-mono
```

必装软件

更换系统镜像源

将系统镜像源换为[清华镜像源](#)或[阿里镜像源](#)

或者可以使用[apt-smart](#)来检测链接状态最好的镜像源，不过需要python的支持。使用方法如下

```
1 # 安装
2 pip install apt-smart
3 # 使用
4 apt-smart -l
```

然后更新软件列表

必装软件集合

软件名称	说明	安装方法
vs code	代码编辑器	官网 下载.deb格式安装包 dpkg -i xxxx.deb
tmux	终端分屏	apt install tmux
xfce4	GUI桌面	apt install xfce4
nethogs	查看网速	apt install nethogs slurm
cmake ccmake	工程构建	apt install cmake cmake-curses-gui
make	编译	apt install make
gcc	编译	各个版本都装一遍，需要啥版本就把软连接指向啥版本！ apt install g++-4.8 g++-5 g++-6 g++-7 g++-8
g++	编译	各个版本都装一遍，需要啥版本就把软连接指向啥版本！ apt install gcc-4.8 gcc-5 gcc-6 gcc-7 gcc-8
zip	压缩	apt install zip unzip
dos2unix	DOS格式的文本文件转成UNIX格式	apt install dos2unix
gdb	程序调试器（可以查看段错误）	apt install gdb
build-essential	包含了很多开发必要的软件	apt install build-essential
git	分布式版本控制	apt install git

整合如下：

```
1 apt install xfce4 nethogs slurm dos2unix tmux cmake make g++-4.8 g++-5 g++-6
g++-7 g++-8 gcc-4.8 gcc-5 gcc-6 gcc-7 gcc-8 unzip zip rar unrar gdb build-
essential git cmake-curses-gui chromium-browser gedit vim clang http
```

Miniconda

最新版本的教程[参考这里](#)

推荐安装Miniconda而不是Anaconda。Anaconda里面有很多用不上的包，而且切换镜像源后容易报warning。

安装

到[官网](#)或[清华镜像](#)上下载安装包

下载好了后，启动安装程序。例如

```
1 | bash Miniconda3-latest-Linux-x86_64.sh
```

接下来就进入安装界面了。下图为欢迎界面，按回车（ENTER）

```
Welcome to Miniconda3 py38_4.8.3

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>> |
```

然后是许可协议，按q直接略过

```
=====
End User License Agreement - Anaconda Individual Edition
=====

Copyright 2015-2020, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

This End User License Agreement (the "Agreement") is a legal agreement between y
ou and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Individual E
dition (which was formerly known as Anaconda Distribution).

Subject to the terms of this Agreement, Anaconda hereby grants you a non-exclusi
ve, non-transferable license to:

    * Install and use the Anaconda Individual Edition (which was formerly known as
    Anaconda Distribution),
    * Modify and create derivative works of sample source code delivered in Anacon
da Individual Edition; and
    * Redistribute code files in source (if provided to you by Anaconda as source)
    and binary forms, with or without modification subject to the requirements set
    forth below.

--更多--
```

协议看完了问你是否同意，输入yes

```
Do you accept the license terms? [yes|no]
[no] >>> yes
```

然后最重要的一步来了——**安装路径的选择**。默认的安装路径是当前操作用户的个人文件夹。因为我是用root进行操作，而root的个人文件夹是根目录下的/root，所以这里的默认路径是/root/miniconda3。因为服务器是很多人都要用的，所以建议不要安装到某个用户的个人文件夹里面。可以安装到根目录下的 /opt。（/opt 的用途可以参见[这篇博客](#)）。

当然，也可以安装到默认路径下后，把整个文件夹挪到 /opt。只不过需要到环境变量里面改一下miniconda的路径

所以，安装路径选择 `/opt/miniconda3`

```
Miniconda3 will now be installed into this location:
/root/miniconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[/root/miniconda3] >>> /opt/miniconda3
```

如果你下载的安装包是python 2.x的，那你路径中应该是 `miniconda2` 而不是 `miniconda3`。当然也可以是 `miniconda456`，只不过是文件夹的名字罢了。

接下来就是一大段自动执行的安装代码。

接着让你选择是否要让安装器修改你个人的bash的环境变量文件。这个选择 `yes` or `no` 都无所谓，因为待会儿要修改系统的环境变量来让所有用户都可以用。

```
Preparing transaction: done
Executing transaction: done
installation finished.
Do you wish the installer to initialize Miniconda3
by running conda init? [yes|no]
[no] >>> no
```

安装结束！

配置系统环境变量

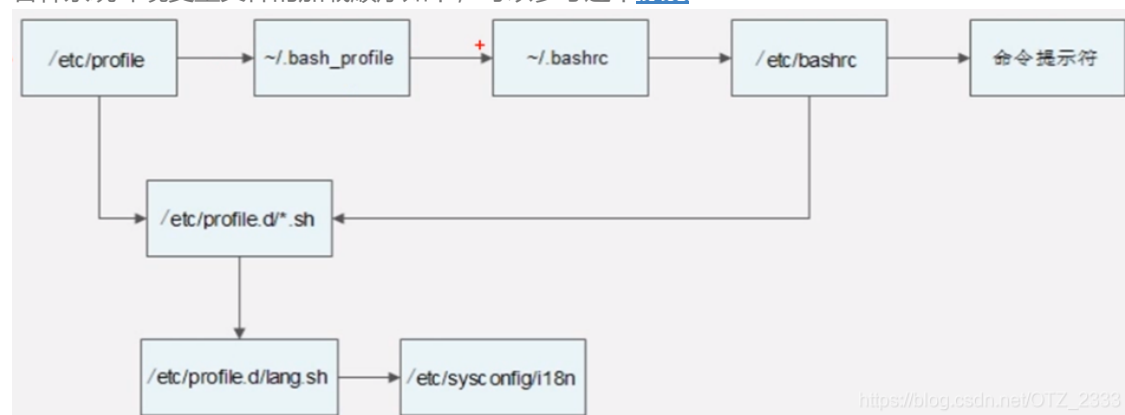
为了让所有用户在登录以后可以直接使用miniconda，需要将miniconda的路径添加到系统环境变量中。

编辑系统环境变量文件 `/etc/bash.bashrc`

也有可能文件名为 `/etc/bashrc`。

此外，你也可以选择编辑另一个系统环境变量文件 `/etc/profile`。如果你选择这个文件的话，切换了账号，需要重新注入环境变量，会比较麻烦！

各种系统环境变量文件的加载顺序如下，可以参考这个[视频](#)



https://blog.csdn.net/OTZ_2333

在最后一行加入如下代码。其中 `/opt/miniconda3` 是你的安装路径

```
1 export PATH=/opt/miniconda3/bin:$PATH
```

如果你在上面安装的时候，在“让你选择是否要让安装器修改你个人的bash的环境变量文件”选择了 `yes`，那你可以把 `~/.bashrc` 中关于miniconda的一大段代码都复制出来，添加到环境变量文件中。

我也不知道那一大段代码跟上面那一行 `export ...` 有什么区别:)

保存退出，重新注入环境变量

```
1 source /etc/bash.bashrc
```


注意：

1. 注入变量的时候最好不要用fish等其他shell，而是用默认的shell。如果你看不懂前面那句话的话就不要管它啦

还可以将现有的python的版本改成3.6（因为3.7太新了！！）

```
1 | conda install python=3.6
```

更换镜像源

我一般都使用[清华镜像源](#)。建议进入它们的官网查看最新的更换镜像源命令，因为可能会有变动。注意：只能修改用户自己的镜像源，即便root用户也是如此（全局的方法我目前还不知道）

为了方便大家，我搬运过来了：编辑 `~/.condarc` 文件，内容替换为如下

```
1 | channels:
2 |   - defaults
3 | show_channel_urls: true
4 | channel_alias: https://mirrors.tuna.tsinghua.edu.cn/anaconda
5 | default_channels:
6 |   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
7 |   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
8 |   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
9 |   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/pro
10 |  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2
11 | custom_channels:
12 |   conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
13 |   msys2: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
14 |   bioconda: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
15 |   menpo: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
16 |   pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
17 |   simpleitk: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

运行命令 `conda clean -i` 清除索引缓存，保证用的是镜像站提供的索引。

运行命令 `conda info` 查看是否更换成功

创建公用环境

创建**公用**环境的时候要把路径选择到安装路径/opt下面，使用 `root` 账号

```
1 | #在/opt/miniconda3/envs/里创建一个名为py27的环境，其使用的python版本为2.7
2 | conda create -p /opt/miniconda3/envs/py27 python=2.7
```

原因如下：添加的环境要求**所有人**都可以用。但是如果用 `conda create -n python27 python=2.7` 这个命令将会在用户的个人文件下创建环境（root账户也算是“个人账户”），导致其他人不可以用这个环境。比如下图，我用xzf的账号创建了一个叫python36的环境，用test账号就不能检测到那个环境（但是root好像可以检测到）。

```
test@srt-102:~$ conda info --envs
# conda environments:
#
base                                * /opt/anaconda3

test@srt-102:~$ su xzf
Password:
xzf@srt-102:/home/test$ cd /home/xzf/
xzf@srt-102:~$ conda info --envs
conda: command not found
xzf@srt-102:~$ conda info --envs
# conda environments:
#
python36                            /home/xzf/.conda/envs/python36
base                                * /opt/anaconda3
```

如此做的**优点**：

- 1. 普通账户只能在自己的个人文件下创建环境，不能在/opt/miniconda3/envs/里创建
- 2. 普通账户可以使用创建在/opt/miniconda3/envs/里的所有环境，但是不能对其进行修改（安装or删除软件等）

环境配置

计划中.....

环境名称	base	py3_old	py2_new	py2_old
opencv(contrib))	4.x	3.x	4.x	3.x
pytorch	1.4.x			
tensorflow				
datatoolkit	9.0			
cuda				
caffe				

其他必装库：

英伟达显卡驱动安装

最新内容参考我的[这篇博客](#)

推荐博客：[How to install Nvidia drivers and cuda-10.0 for RTX 2080 Ti GPU on Ubuntu-16.04/18.04](#)

0. 卸载Nvidia残余文件。

如果系统是新装的，还没有安装过任何nvidia的东西，可以跳过这一步直接到后面。如果电脑已经安装过nvidia的驱动了，**推荐**清理一下已经存在的nvidia文件

```

1 # 上来就删！
2 # apt remove *cuda*
3 apt remove *nvidia*
4 /usr/bin/nvidia-uninstall
5 # 下面这句可要可不要吧。不过我一般都加上的，删的干干净净的不好嘛
6 apt autoremove
7
8 # 再看看dpkg里面还有没有nvidia相关的软件
9 dpkg -l | grep nvidia
10 # 如果还有的话，需要一个个手动删。软件名称是不包括冒号之后的哦。
11 dpkg --purge <软件名称>
12 # 其实我也不知道dpkg这里要不要删，反正我删了，试了几台电脑也都成功了。

```

还可以用locate命令定位一下nvidia文件，比如用命令 `locate nvidia`（先用命令 `updatedb` 更新一下数据库）。不过可能找到各种地方的路径，推荐可以手动删除 `/usr` 下面的文件，其他地方我也不敢乱删呀。这一步我觉得可以不做，把上面的做好了应该就ok了。

1. 禁用Nouveau的驱动

在 `/etc/modprobe.d/blacklist.conf` 最后添加如下代码：
(用来禁用nouveau第三方驱动，之后也不需要改回来)

```

1 blacklist nouveau
2 options nouveau modeset=0

```

然后执行

```
1 update-initramfs -u
```

重启后，执行以下代码，若没有显示则禁用成功

```
1 lsmod | grep nouveau
```

如果遇到这个问题 `perl: warning: Falling back to a fallback locale ("en_US.UTF-8")`

```
1 apt install locales-all
```

2. 检测NVIDIA显卡型号：

命令行有三种方法

```

1 #方法一：使用ubuntu-drivers-common这个软件`
2 apt install ubuntu-drivers-common
3 ubuntu-drivers devices
4 #方法二
5 lshw -numeric -C display //最好用sudo
6 #方法三
7 lspci -vnn | grep VGA

```

也可以用其他方法。

然后下载[官方驱动](#)，我比较喜欢最新的版本。推荐下载英文版的驱动，防止中文乱码。例如

```
1 | wget http://us.download.nvidia.com/XFree86/Linux-x86_64/410.93/NVIDIA-Linux-x86_64-410.93.run
```

3. 关闭图形界面

如果系统已经有图形界面（比如ubuntu desktop），最好**关闭一下图形界面**。关闭之前要先切换到**命令行界面**，使用快捷键 **Ctrl+Alt+F1**（F1不行的话，就换成 F2、F3...）。然后sudo权限运行命令

```
1 | service lightdm stop
2 | # 如果提示Failed to stop lightdm.service: Unit lightdm.service not loaded,
    可以不用管它，继续
```

4. 安装

给驱动run文件赋予执行权限，然后运行

```
1 | chmod +x NVIDIA-Linux-x86_64-410.93.run
2 | ./NVIDIA-Linux-x86_64-410.93.run
```

也可以直接使用bash运行

```
1 | bash NVIDIA-Linux-x86_64-410.93.run
```

启动安装程序，可以加参数（安装驱动前可能需要先安装gcc和make）。不过我一般都不加任何参数，如果想要加的话，可以加一个 **--no-x-check**

参数有：

--no-opengl-files：表示只安装驱动文件，不安装OpenGL文件。这个参数不可省略，否则会- 导致登陆界面死循环，英语一般称为“login loop”或者“stuck in login”。**然而我并没有用**

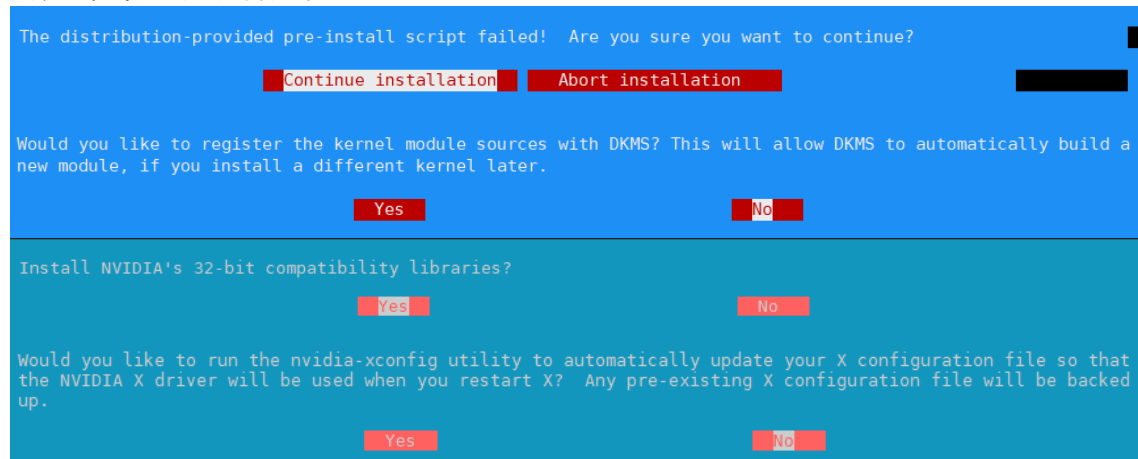
--no-x-check：表示安装驱动时不检查X服务。如果没有这个参数，可能会出现“X-Server needs to be disabled before installing the drivers”的错误。**然而我并没有用**

--no-nouveau-check：表示安装驱动时不检查nouveau，非必需。

-Z, --disable-nouveau：禁用nouveau。此参数非必需，因为之前已经手动禁用了nouveau。

-A：查看更多高级选项。

安装过程中遇到的选择如下：



如果出现如下界面，需要终止驱动安装（**Ctrl+C**），然后到bios里关闭安全启动，然后从第四步开始。

The target kernel has CONFIG_MODULE_SIG set, which means that it supports cryptographic signatures on kernel modules. On some systems, the kernel may refuse to load modules without a valid signature from a trusted key. This system also has UEFI Secure Boot enabled; many distributions enforce module signature verification on UEFI systems when Secure Boot is enabled. Would you like to sign the NVIDIA kernel module?

Sign the kernel module

Install without signing

5. 若遇到关于kernel的Error，应该是驱动版本和系统内核版本不匹配（图后面有机会补上吧）。

方法一：下载更新版本（or更旧，一般都是更新吧）的驱动。推荐这个，因为我都成功了。

方法二：更改内核的版本。具体应该切换到什么版本的内核，可以参考其他已经成功安装了显卡驱动的电脑的内核版本。查看当前使用的内核 `uname -r`，查看grub版本 `grub-install -V`，切换内核可以参考这个[博客](#)

6. 如果系统原本就有图形界面，并且执行了第3步，那安装完驱动后可以打开图形界面

```
1 | service lightdm start
```

如果报错 `Failed to stop lightdm.service: Unit lightdm.service not loaded`，需要重装一下 `lightdm`

```
1 | apt install lightdm
```

如果重启后，系统默认进入的是命令行界面而不是图形界面，

```
1 | # 查看当前启动模式。如果输出为multi-user.target，表示默认是命令行界面
2 | systemctl get-default
3 |
4 | # 将命令行模式更改为图形界面
5 | systemctl set-default graphical.target
```

cuda、cudnn

python版本

用conda在指定环境下安装。可以直接安装cudnn，这样可以自动安装cuda，但要选择cuda版本对应的cudnn。比如这里在miniconda环境中安装cuda8，选择cudnn7.0.5（**自行选择适合的版本**）
例如

```
1 | #python2.7, cudnn=7.0.5, cuda=8.0
2 | conda install cudnn=7.0.5
```

c++版本

考虑到用conda安装的cuda在/usr/local下没有cuda的文件夹，对于一些c++的代码或者需要cuda路径的代码，方法一就不行了。步骤如下：

1. 准备:更换gcc和g++到合适版本，这里以4.8为例

```
1 | cd /usr/bin/
2 | rm gcc
3 | ln -s gcc-4.8 gcc
4 | rm g++
5 | ln -s g++-4.8 g++
```

2. 安装cuda8.0: 到[cuda官网](#)上下载.deb格式安装包。

最好选择"local"而不是"network", 因为我在用服务器安装的时候死活连不上nvidia的服务器! 推荐在windows上用IDM下载好了传过去。(这句话是对我自己说的:)

Select Target Platform ⓘ

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows	Linux	Mac OSX
Architecture ⓘ	x86_64	ppc64le	
Distribution	Fedora	OpenSUSE	RHEL
	SLES	Ubuntu	CentOS
Version	16.04	14.04	
Installer Type ⓘ	runfile (local)	deb (local)	deb (network)
	cluster (local)		

https://blog.csdn.net/OTZ_2333

> Base Installer

Download (1.9 GB) 

Installation Instructions:

1. ``sudo dpkg -i cuda-repo-ubuntu1604-8-0-local-ga2_8.0.61-1_amd64.deb``
2. ``sudo apt-get update``
3. ``sudo apt-get install cuda``

https://blog.csdn.net/OTZ_2333

执行

```
1 dpkg -i cuda-repo-ubuntu1604-8-0-local-ga2_8.0.61-1_amd64.deb
2 apt update
3 apt install cuda
```

之所以选择.deb格式而不是.run, 是因为我在用.run格式安装的时候一直遇到"error: cannot find toolkit in /usr/local/cuda-8.0"的错误, 怎么也解决不了。

3. 安装cuda10.0: 到[cuda官网](#)上下载安装包

Select Target Platform ⓘ

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System

Windows Linux Mac OSX

Architecture ⓘ

x86_64 ppc64le

Distribution

Fedora OpenSUSE RHEL CentOS SLES Ubuntu

Version

18.04 16.04 14.04

Installer Type ⓘ

runfile (local) deb (local) deb (network) cluster (local)

Download Installers for Linux Ubuntu 16.04 x86_64

The base installer is available for download below.
There is 1 patch available. This patch requires the base installer to be installed first.

> Base Installer

Download (2.0 GB) ⬇

Installation Instructions:
1. Run ``sudo sh cuda_10.0.130_410.48_linux.run``
2. Follow the command-line prompts

> Patch 1 (Released May 10, 2019)

Download (3.3 MB) ⬇

In this patch we introduce new APIs for JPEG stream parsing and device and pinned memory control as well as a new hybrid decode API that decouples decoding process into pure host and device stages enabling more flexible control flow. The new APIs also support ROI decoding and 4 channel jpeg bitstreams.

The CUDA Toolkit contains Open-Source Software. The source code can be found [here](#).
The checksums for the installer and patches can be found in [Installer Checksums](#).
For further information, see the [Installation Guide for Linux](#) and the [CUDA Quick Start Guide](#).

按命令安装

```
accept/decline/quit: accept

Install NVIDIA Accelerated Graphics Driver for Linux-x86_64 410.48?
(y)es/(n)o/(q)uit: n (这里不要再安装驱动了!!!)

Install the CUDA 10.0 Toolkit? (是否安装CUDA 10, 这里必须要安装)
(y)es/(n)o/(q)uit: y

Enter Toolkit Location (安装路径, 使用默认, 直接回车就行)
[ default is /usr/local/cuda-10.0 ]:

Do you want to install a symbolic link at /usr/local/cuda? (同意创建软链接)
(y)es/(n)o/(q)uit: y

Install the CUDA 10.0 Samples?
(y)es/(n)o/(q)uit: y

Installing the CUDA Toolkit in /usr/local/cuda-10.0 ... (开始安装)
```

4. 安装cudnn7.1.3: 从[cudnn官网](#)上下载cudnn的时候选择library for linux, 如下图

Download cuDNN v7.1.3 (April 17, 2018), for CUDA 8.0

cuDNN v7.1.3 Library for Linux

cuDNN v7.1.3 Library for Windows 7

cuDNN v7.1.3 Library for Windows 10

cuDNN v7.1.3 Runtime Library for Ubuntu16.04 (Deb)

cuDNN v7.1.3 Developer Library for Ubuntu16.04 (Deb)

cuDNN v7.1.3 Code Samples and User Guide for Ubuntu16.04 (Deb)

cuDNN v7.1.3 Runtime Library for Ubuntu14.04 (Deb)

cuDNN v7.1.3 Developer Library for Ubuntu14.04 (Deb)

cuDNN v7.1.3 Code Samples and User Guide for Ubuntu14.04 (Deb)

执行

```
1 tar -xvzf cudnn-8.0-linux-x64-v7.1.tgz
2 cp cuda/include/cudnn.h /usr/local/cuda-8.0/include
3 cp cuda/lib64/libcudnn* /usr/local/cuda-8.0/lib64
4 chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda-8.0/lib64/libcudnn*
```

对应cuda10.0的cudnn的安装方法大同小异, 就是把路径中的8改成10

5. 后续问题

5.1 安装完后, 我发现驱动好像崩了:(

```
root@srt-102:/home/xzf$ nvidia-smi
Failed to initialize NVML: Driver/library version mismatch
```

重启一下, 发现不行

```
xzf@srt-102:~$ nvidia-smi
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Make sure that the latest NVIDIA driver is installed and running.
```


重新安装一遍驱动就好了

```
root@srt-102:~/Package/Nvidia$ nvidia-smi
Mon Apr 22 15:14:23 2019

+-----+
| NVIDIA-SMI 410.93          Driver Version: 410.93          CUDA Version: 10.0          |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|    0   TITAN X (Pascal)         Off | 00000000:04:00:0 | Off           N/A   |
| 35%    50C    P0      57W / 250W |  0MiB / 12196MiB |    0%      Default  |
+-----+-----+-----+-----+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type    Process name                     Usage      |
+-----+-----+-----+-----+-----+
| No running processes found                                     |
+-----+
https://blog.csdn.net/OTZ_2333
```

5.2 如果需要测试cuda是否安装成功，可以跑一下示例代码，这就是上面安装samples的作用。如果没有./deviceQuery，但是有deviceQuery.cpp的话，自己编译make一下就好了。

```
1 | cd /usr/local/cuda-10.0/samples/1_Uutilities/deviceQuery
2 | ./deviceQuery
```

出现如下文字表示成功

```
./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 4 CUDA Capable device(s)

Device 0: "GeForce RTX 2080 Ti"
  CUDA Driver Version / Runtime Version      10.0 / 10.0
  CUDA Capability Major/Minor version number: 7.5
  Total amount of global memory:             10989 MBytes (11523260416 bytes)
  (68) Multiprocessors, ( 64) CUDA Cores/MP: 4352 CUDA Cores
  GPU Max Clock rate:                       1545 MHz (1.54 GHz)
  Memory Clock rate:                        7000 Mhz
  Memory Bus Width:                         352-bit
  L2 Cache Size:                           5767168 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:           65536 bytes
  Total amount of shared memory per block:   49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                32
  Maximum number of threads per multiprocessor: 1024
  Maximum number of threads per block:       1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                     2147483647 bytes
  Texture alignment:                        512 bytes
  Concurrent copy and kernel execution:      Yes with 3 copy engine(s)
  Run time limit on kernels:                 No
  Integrated GPU sharing Host Memory:         No
  Support host page-locked memory mapping:    Yes
  Alignment requirement for Surfaces:         Yes
  Device has ECC support:                    Disabled
  Device supports Unified Addressing (UVA):    Yes
```

5.3 有一些垃圾教程说可以使用 `nvcc -v` 来测试。确实，安装完cuda后，路径下的bin里面会有nvcc，但是有时候系统并不会将它添加到系统bin里面 或者 添加到环境变量里面（反正我还没有遇到系统会自动添加的）。所以如果使用 `nvcc -v` 报错并不一定是因为没有安装成功，可以自己添加一个软连接：

```
1 | ln -s /usr/local/cuda/bin/nvcc /usr/bin/nvcc
```

此外，如果运行 `nvcc -v` 成功了，也可能运行的是 apt 安装的 `nvidia-cuda-toolkit`，因为这个会在 `/usr/bin/` 下面弄一个 `nvcc` 的，只不过跟你自己装的没有关系

5.4 由于我手里的服务器同时使用了方法一和方法二，用conda跑代码（比如pytorch）使用的是用conda安装的cudatoolkit，见下图，所以我不知道只用方法二的话，conda能不能用cuda，有兴趣的话你可以试一下，反正我是不敢试的，毕竟搞崩了我可承受不起:)

```
(cuda90) root@srt-102:/usr/local$ conda list | grep cuda
# packages in environment at /opt/anaconda3/envs/cuda90:
cuda90          1.0             h6433d27_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch
cudatoolkit     9.0             h13b8566_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/main
cudnn           7.3.1          cuda9.0_0     https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/main

(cuda90) root@srt-102:/usr/local/cuda$ cd ..
(cuda90) root@srt-102:/usr/local$ ll
total 52
drwxr-xr-x 13 root root 4096 Apr 22 15:34 ./
drwxr-xr-x 10 root root 4096 Jul 26 2018 ../
drwxr-xr-x  2 root root 4096 Apr 16 22:06 bin/
lrwxrwxrwx  1 root root    8 Apr 22 13:33 cuda -> cuda-8.0/
drwxr-xr-x 15 root root 4096 Apr 22 13:32 cuda-8.0/
drwxr-xr-x 18 root root 4096 Apr  2 16:54 cuda-9.0/
drwxr-xr-x  2 root root 4096 Jul 26 2018 etc/
drwxr-xr-x  2 root root 4096 Jul 26 2018 games/
drwxr-xr-x  2 root root 4096 Apr 16 22:06 include/
drwxr-xr-x  5 root root 4096 Apr 16 22:06 lib/
lrwxrwxrwx  1 root root    9 Jul 26 2018 man -> share/man/
drwxr-xr-x  6 root root 4096 Apr  9 14:09 Matlab/
drwxr-xr-x  2 root root 4096 Jul 26 2018 sbin/
drwxr-xr-x  8 root root 4096 Apr 16 22:06 share/
drwxr-xr-x  2 root root 4096 Jul 26 2018 src/
(cuda90) root@srt-102:/usr/local$ ipython
Python 3.6.8 [Anaconda, Inc.] (default, Dec 30 2018, 01:22:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import torch

In [2]: torch.version.cuda
Out[2]: '9.0.176'

In [3]:
```

https://blog.csdn.net/OTZ_2333

看了很多的教程（官方的和个人的、中文的和英文的），cuda、cudatoolkit、toolkit这些的名词的区别我还是不知道。

安装的时候可以参考英伟达官网上给出的[cuda文档](#)和[cudnn文档](#)
cuda和cudnn的版本关系可以参考这篇博客：[TensorFlow版本配套关系表（cudnn、cuda、Python的配套关系，包含所有操作系统）](#)
[参考博客](#)

Matlab

根据本人在多台服务器上安装过多个版本的Matlab的经验，此教程适用于R2014a、R2016b、R2018a版本的Matlab。

这里以 MatlabR2018a 命令行安装为例

安装包

[Matlab2018a百度网盘链接](#)

提取码: kndg

由于Linux下没有好用的可以下载百度网盘文件的工具，推荐再windows上使用[Pandownload](#)下载好后传到linux上。

安装

解压破解文件压缩包

```
1 | tar -xvf Matlab2018aLinux64Crack.tar.gz
```

由于安装包被分成了两个镜像，先挂载第一个

```
1 | mkdir /iso
2 | mount -o loop R2018a_glnxa64_dvd1.iso /iso
```

转移配置文件。其中license_standalone.lic来自于破解压缩包

压缩包里面可能提供不止一个.lic文件，选择license_standalone.lic。但是也有可能破解压缩包只提供了一个.lic文件，只是这个文件可能不叫这个名字，可能叫license_405329_R2014a.lic，反正就是把.lic结尾的那个文件复制到安装目录下的install文件夹里面，然后在编辑activate.ini的时候注意一下文件名字

```
1 | #/usr/local/Matlab/R2018a/install用来存放安装配置文件
2 | mkdir /usr/local/Matlab/R2018a/install
3 | cp license_standalone.lic /usr/local/Matlab/R2018a/install/
4 | cp /iso/installer_input.txt /usr/local/Matlab/R2018a/install/
5 | cp /iso/activate.ini /usr/local/Matlab/R2018a/install/
```

编辑 /usr/local/Matlab/R2018a/install/installer_input.txt，用来配置安装选项

```
1 | #这里所有R2018a按版本填写
2 | #安装目录
3 | destinationFolder=/usr/local/Matlab/R2018a
4 | #你的序列号，不同版本的序列号可以去网上查
5 | fileInstallationKey=09806-07443-53955-64350-21751-41297
6 | #同意协议
7 | agreeToLicense=yes
8 | #安装日志（Optional）
9 | outputFile=/tmp/mathwork_install.log
10 | #开启无人值守安装
11 | mode=silent
12 | #选择激活文件
13 | activationPropertiesFile=/usr/local/Matlab/R2018a/install/activate.ini
```

编辑 /usr/local/Matlab/R2018a/install/activate.ini，用来配置激活选项

(不能直接复制粘贴到文件的最开始，要填到相应的位置，否则破解 应该会报如下错误)

```
root@art-102:/usr/local/Matlab/R2018a/bin$ /usr/local/Matlab/R2018a/bin/activate_matlab.sh -propertiesFile /usr/local/Matlab/etc/activate.ini
Silent activation failed. Please see /tmp/aws_root.log for more information.
```

```
1 | #开启silent模式
2 | issilent=true
3 | #设置激活方式，离线激活 无需联网
4 | activateCommand=activateOffline
5 | #license文件位置（来自于破解文件）
6 | licenseFile=/usr/local/Matlab/R2018a/install/license_standalone.lic
```

执行安装命令

```
1 | /iso/install -inputFile /usr/local/Matlab/R2018a/install/installer_input.txt
```

提示如下文字后需要 重开一个终端 来卸载第一个镜像然后挂载第二个镜像

```
(Feb 24, 2019 13:27:54) Info: Eject DVD 1 and insert DVD 2 to continue.
(Feb 24, 2019 13:27:54) Info: Eject DVD 1 and insert DVD 2 to continue.
(Feb 24, 2019 13:27:54) Info: Eject DVD 1 and insert DVD 2 to continue.
(Feb 24, 2019 13:27:54) Info: Eject DVD 1 and insert DVD 2 to continue.
(Feb 24, 2019 13:27:54) Info: Eject DVD 1 and insert DVD 2 to continue.
(Feb 24, 2019 13:27:54) Info: Eject DVD 1 and insert DVD 2 to continue.
(Feb 24, 2019 13:27:54) Info: Eject DVD 1 and insert DVD 2 to continue.
```

```
1 umount /iso
2 mount -t auto -O loop R2018a_glnxa64_dvd2.iso /iso
```

安装完成后提示还要有一大堆的步骤要操作，不过先不管了，安装成功了再说，那些操作后面有空弄。

```
(Feb 24, 2019 13:35:12) Notes:
Your installation may require additional configuration steps.

1. The following products require a supported compiler:

Stateflow 9.1
Simulink Coder 8.14
MATLAB Coder 4.0
Simulink Test 2.4

2. Simulink requires a C compiler for simulation acceleration, model reference, and MATLAB Function Block capabilities. It is recommended that you install a supported compiler on your machine.

3. To accelerate computations with the following products, a supported compiler is required:

SimBiology 5.8
Fixed-Point Designer 6.1

4. After this installation is complete, you should continue with your configuration of the MATLAB Distributed Computing Server as outlined in the instructions obtained from www.mathworks.com/distconfig.

5. MATLAB Compiler SDK 6.5 requires the following:

• a supported compiler for creation of C and C++ Shared libraries
• a Java JDK for creation of Java packages

(Feb 24, 2019 13:35:13) Exiting with status 0
(Feb 24, 2019 13:35:16) End - Successful.
Finished
```

https://blog.csdn.net/OTZ_2333

激活

先将破解文件中提供的所有.so文件都拷贝到安装路径下。

如果破解文件提供的不是这样的一个文件夹而是很多的.so文件的话，则把所有的.so文件拷贝到/usr/local/Matlab/R2018a/bin/glnxa64/覆盖。可以提前把安装目录下对应的.so文件备份一下。

```
1 #比如
2 #破解文件提供了一个名为R2018a的文件夹，这个文件夹下面的文件结构跟安装路径里的一样，里面有.so的文件。
3 #直接拷贝那个文件夹到安装路径覆盖
4 cp -r ./Matlab2018aLinux64Crack/R2018a /usr/local/Matlab/
```

然后通过前面已经配置好的激活文件来进行激活

```
1 /usr/local/Matlab/R2018a/bin/activate_matlab.sh -propertiesFile
   /usr/local/Matlab/R2018a/install/activate.ini
```

出现 silent activation succeeded 表示破解成功

```
root@~: /usr/local/Matlab/R2018a/bin$ /usr/local/Matlab/R2018a/bin/activate_matlab.sh -propertiesFile /usr/local/Matlab/etc/activate.ini
Silent activation succeeded.
```

添加环境变量

编辑 `/etc/profile`

```
1 export PATH=/usr/local/Matlab/R2018a/bin:$PATH
```

备注：当安装完所需几个版本matlab后，注意环境变量只能选择一个，其他版本注释，否则又冲突，需要什么版本再修改环境变量

启动测试

```
1 matlab -nodesktop -nodisplay
```

出现下图的情况表示成功了！

```
root@srt-102:~$ matlab -nodesktop -nodisplay

                                     < M A T L A B (R) >
                                     Copyright 1984-2018 The MathWorks, Inc.
                                     R2018a (9.4.0.813654) 64-bit (glnxa64)
                                     February 23, 2018

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> https://blog.csdn.net/OTZ_2333
```

如果报错

```
1 Fatal Internal Error: Unexpected exception:
  'N9Mathworks6System15SimpleExceptionE: Dynamic exception type:
  St13runtime_error
2 std::exception::what: Bundle#1 start failed: libXt.so.6: cannot open shared
  object file: No such file or directory
3 ' in createMVMAndCallParser phase 'Creating local MVM'
```

执行

```
1 apt install x11-xserver-utils
2 xhost si:localuser:root
```

参考教程：

[Ubuntu 16.04安装MATLAB R2018a](#)

[服务器远程安装Matlab2015](#)

[linux安装MATLAB R2018a步骤](#)

[linux命令行模式下安装matlab](#)

拓展

其他版本的Matlab安装过程大同小异。

这里附赠Matlab 2014a的安装包链接：[百度网盘](#)，提取码：529r

Samba：Windows共享Linux文件夹

安装：[官网](#)

```
1 apt install samba
```

官网给出的教程还要安装 `attr winbind libpam-winbind libnss-winbind libpam-krb5 krb5-config krb5-user`，但是ta好像是为了进行域管理，我们暂时不需要，就不用安装。

配置：在 `/etc/samba/smb.conf` 末尾添加

```
1 [xzf]
2 path = /home/xzf
3 valid users = xzf
4 force user = xzf
5 create mask = 0755
6 browseable = yes
7 available = yes
8 read only = no
9 writeable = yes
10 public = yes
```

参数说明：

[\[xzf\]](#)：共享目录的名称

`path`：共享的目录

[force user](#)：使得操作的时候和服务端用户一样

`create mask`：创建文件属性

`browseable`：在浏览资源中显示共享目录，若为否则必须指定共享路径才能存取

`read only`：加上只是为了以防万一

`writeable`：不以只读方式共享当与`read only`发生冲突时，无视`read only`

`public`：公开共享，若为否则进行身份验证(只有当`security = share` 时此项才起作用)

添加一个普通账户，用于访问这个目录

```
1 #需要root权限
2 smbpasswd -a xzf
```

重启

```
1 #我在ubuntu 18.04和16.04测试的都是smbd，根网上的教程中用的smb或samba都不一样，可能是软件更新后命令不一样了吧
2 service smbd restart
```

Windows端，进入文件资源管理器，左侧列表里，对“网络”右键，添加“映射网络驱动器”，输入“\\192.168.1.102\xzf”，输入前面用 `smbpasswd` 创建的用户名和密码。

Netdata：状态监控软件

前提：终端访问外网

脚本安装[Netdata](#)

```
1 bash <(curl -ss https://my-netdata.io/kickstart-static64.sh)
```

不要使用`sudo`!!! 脚本里用到的时候会让你输入密码的

安装路径为`/opt/netdata/`

安装[第三方Nvidia GPU插件](#)

```
1 #先安装英伟达驱动和这个
2 pip install nvidia-ml-py      #最好在root下安装
3
4 #然后安装插件
5 git clone https://github.com/Splo0sh/netdata_nv_plugin --depth 1
6 cp netdata_nv_plugin/nv.chart.py /opt/netdata/usr/libexec/netdata/python.d/
7 cp netdata_nv_plugin/python_modules/pynvml.py
  /opt/netdata/usr/libexec/netdata/python.d/python_modules/
8 cp netdata_nv_plugin/nv.conf /opt/netdata/etc/netdata/python.d/
```

因为安装的是static版本的，所以复制过去的路径和github中的不一样

重启Netdata

```
1 service netdata restart
```

VS Code

去[官网](#)下载安装包

部分设置：

1. [自动换行](#)

推荐扩展：

1. [Markdown All In One](#)：Markdown快捷键、预览等功能
2. [Markdown PDF](#)：将Markdown转化为PDF格式
3. [Markdown TOC](#)：自动生成目录
4. [TabNine](#)：超级好用智能的自动补齐插件，适用于所有编程语言！
5. [Python](#)：在vs code里运行python代码。
6. [Anaconda Extension Pack](#)：切换conda环境
7. [C/C++](#)、[Code Runner](#)：在vs code里运行C/C++代码，[参考教程](#)

若出现“检测到#include错误。请更新您的includePath”的错误，按 `ctrl + shift + p`，键入 `C/C++: Edit Configurations (UI)`，在 `includePath` 中输入 `D:/Library/mingw64/**`，虽然其实这个问题没有关系的，因为是用 `Code Runner` 跑代码而不是 `C/C++`

库

各种库

库名称	说明	安装方法
Eigen	矩阵处理	apt install libeigen3-dev
Pangolin	可视化	依赖: apt install libgl1-mesa-dev libglew-dev git后用cmake编译安装
Sophus	李代数	git后用cmake编译 (无需安装)
Ceres	求解最小二乘问题	依赖: apt install liblapack-dev libsuitesparse-dev libxcxsparse3 libgflags-dev libgoogle-glog-dev libgtest-dev git后用cmake编译安装
g2o	基于图优化	依赖: apt install cmake libeigen3-dev libsuitesparse-dev qtdeclarative5-dev qt5-qmake qt5-default libqglviewer- dev-qt5 libxcxsparse3 libcholmod3 git后用cmake编译安装
OpenCV	计算机视觉库	见下面OpenCV专栏
PCL	点云库	见下面PCL专栏
FLANN	最完整的 (近似) 最近邻开源库	见下面PCL专栏
Boost	为C++语言标准库 提供扩展的一些 C++程序库的总称	见下面PCL专栏
VTK	可视化工具库	见下面PCL专栏
OpenNI	开放自然交互。。。懂的都懂	见下面PCL专栏
QHull	计算几何库	见下面PCL专栏

整理成.sh如下:

```

1  #安装这些依赖的时候好像会安装python2.7，我也不知道为啥。而且安装完后运行python会自动运行
   #python2.7，不过重新注入环境变量了以后再运行python用的就是conda里面的python，所以我也就
   #没有管它了。
2  apt install libeigen3-dev liblapack-dev libxcxsparse3 libgflags-dev
   libgoogle-glog-dev libgtest-dev cmake libsuitesparse-dev qtdeclarative5-dev
   qt5-qmake qt5-default libqglviewer-dev-qt5 libxcxsparse3 libcholmod3 libgl1-
   mesa-dev libglew-dev liblz4-dev
3  #安装Pangolin出现‘No package ‘xkbcommon’ found’
4  apt install libxkbcommon-x11-dev
5
6  # Pangolin
7  git clone https://github.com/stevenlovegrove/Pangolin.git
8  cd Pangolin
9  mkdir build
10 cd build
11 cmake ..
12 make -j7
13 make install
14 cd ../..

```



```

15
16 # Sophus
17 git clone https://github.com/strasdat/Sophus.git
18 cd Sophus
19 mkdir build
20 cd build
21 cmake ..
22 make -j7
23 make install #可以不安装，但是我还是装了
24 cd ../../
25
26 git clone https://github.com/ceres-solver/ceres-solver.git
27 cd ceres-solver
28 mkdir build
29 cd build
30 cmake ..
31 make -j7
32 make install
33 cd ../../
34
35 git clone https://github.com/RainerKuemmerle/g2o.git
36 cd g2o
37 mkdir build
38 cd build
39 cmake ..
40 make -j7
41 make install
42 cd ../../
43
44 apt install libpcl-dev

```

PCL

- 方法一： `apt install libpcl-dev`。一般版本都比较老。

注：网上的一些教程在命令行安装之前，会添加一个仓库 `add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl`，但是我试了一下，使用这个仓库会报错（没有 Release 文件），而且貌似里面的文件都比较老了（2015年之前）

- 方法二：源码编译安装（推荐）

1. 安装依赖以及第三方库：Boost, Eigen, FLANN, VTK, (OpenNI, QHull)

```

1 # 必装：其中eigen和vtk一直在更新，安装名称中的数字可能会发生变化
2 apt install build-essential libboost-all-dev libeigen3-dev libvtk7-dev
3 # FLANN
4 git clone https://github.com/mariusmuja/flann.git
5 cd flann
6 mkdir build
7 cd build
8 cmake ..
9 make -j7
10 make install
11 cd ../../
12
13 # 可选
14 apt install libopenni-dev libqhull-dev

```

2. **下载源码**，然后解压缩，例如 `tar xvfj pcl-pcl-1.7.2.tar.gz`，或者从[GitHub](#)克隆。

```
1 cd pcl-pcl-1.7.2 && mkdir build && cd build
2 cmake ..
3 # 如果想要安装Release版本，运行命令cmake -DCMAKE_BUILD_TYPE=Release ..
4 make -j6
5 make -j6 install
```

OpenCV

python

```
1 #只安装opencv
2 pip install opencv_python
3 #安装opencv + opencv_contrib
4 pip install opencv-contrib-python
```

查看可以安装的版本，在命令后面加上 `==`，例如 `pip install opencv_python==`

C++

以下不保证最新，最新的内容可以看[我的博客](#)

安装前一定先看一遍[官方教程](#) ([Installation in Linux](#), [opencv_contrib](#)) 和[以下全文](#)，尤其是最后的问题

以opencv 4.2.0版本为例，我 home 下的 Downloads 文件夹里有 opencv-4.2.0、opencv_contrib-master 和 opencv_need 三个文件夹，分别存放着opencv 4.2.0的源码、opencv contrib的源码和问题三中自己手动下载的所有文件

```
1 #安装所有必须的软件和依赖项。如果显示E: Unable to locate package xxxx，把镜像源更换为清华的应该就能解决。
2 apt install libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
3 #可选项。若libjasper-dev不能安装,参考问题一。除了python的两个，其他的我全装了（都是处理各种图片格式的库）
4 apt install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
5
6 #获取源文件，git或者用IDM直接下载后传给linux解压
7 git clone https://github.com/opencv/opencv.git
8 git clone https://github.com/opencv/opencv_contrib.git
9
10 #进入opencv的文件夹
11 cd opencv-4.2.0/
12 mkdir build
13 cd build
14
15 #如果报错，在-D后加个空格；
16 #-DOPENCV_EXTRA_MODULES_PATH=后面跟的是opencv_contrib的路径,因为我的opencv_contrib-master和opencv-4.2.0两个文件夹在同一个文件夹下
17 #-DBUILD_opencv_java和-DBUILD_opencv_python是用来选择是否要java和python的
18 cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local -DOPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-master/modules -DBUILD_opencv_java=OFF -DBUILD_opencv_python=OFF ..
```

```

19 #若显示    -- Configuring done
20 #          -- Generating done
21 #则进行下一步, -j7表示多线程
22 make -j7
23 make install

```

PS: 如果cmake的时候, 输出说 Could NOT find xxx 之类的, 不要担心, 只要不是让cmake终止的 error 都没问题。cmake成功后会显示 Configuring done 和 Generating done

问题一: 安装可选依赖包libjasper-dev的时候, 显示E: Unable to locate package libjasper-dev

```

1 add-apt-repository "deb http://security.ubuntu.com/ubuntu xenial-security
  main"
2 apt update
3 apt install libjasper1 libjasper-dev
4 add-apt-repository --remove "deb http://security.ubuntu.com/ubuntu xenial-
  security main"
5 #其中libjasper1是libjasper-dev的依赖包

```

参考教程: [Ubuntu18.04下安装OpenCv依赖包libjasper-dev无法安装的问题](#)

问题二: 如果在 cmake编译 的时候, 显示No package 'gtk+-3.0' found

```

orz@orz-HP: /Downloads/opencv-4.2.0/build$ cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local -DOPENCV_E
XTRA_MODULES_PATH=../../opencv_contrib-master/modules ..
-- Detected processor: x86_64
-- Looking for ccache - not found
-- Found ZLIB: /usr/lib/x86_64-linux-gnu/libz.so (found suitable version "1.2.11", minimum required is "1.2.3")
-- Could NOT find JPEG (missing: JPEG_LIBRARY JPEG_INCLUDE_DIR)
-- libjpeg-turbo: VERSION = 2.0.2, BUILD = opencv-4.2.0-libjpeg-turbo
-- Could NOT find TIFF (missing: TIFF_LIBRARY TIFF_INCLUDE_DIR)
-- Could NOT find Jasper (missing: JASPER_LIBRARIES JASPER_INCLUDE_DIR)
-- Found ZLIB: /usr/lib/x86_64-linux-gnu/libz.so (found version "1.2.11")
-- Checking for module 'gtk+-3.0'
-- No package 'gtk+-3.0' found
-- IPPTCV: Download: ipptcv_2019_lnx_intel64_general_20180723.tgz

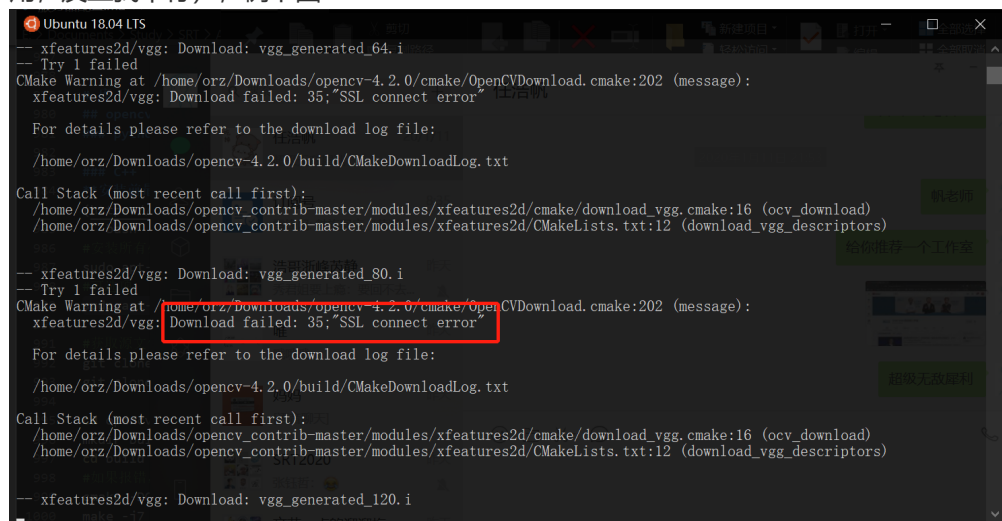
```

```

1 #以防万一, 我觉得还是装了比较好
2 apt install libgtk-3-dev

```

问题三: 如果在 cmake编译 的时候, 需要到Github下载一些文件, 但是下载不下来 (终端翻墙可能也没用, 反正我不行), 例下图



```

Ubuntu 18.04 LTS
-- xfeatures2d/vgg: Download: vgg_generated_64.i
-- Try 1 failed
CMake Warning at /home/orz/Downloads/opencv-4.2.0/cmake/OpenCVDDownload.cmake:202 (message):
  xfeatures2d/vgg: Download failed: 35; "SSL connect error"
For details please refer to the download log file:
/home/orz/Downloads/opencv-4.2.0/build/CMakeDownloadLog.txt
Call Stack (most recent call first):
/home/orz/Downloads/opencv_contrib-master/modules/xfeatures2d/cmake/download_vgg.cmake:16 (ocv_download)
/home/orz/Downloads/opencv_contrib-master/modules/xfeatures2d/CMakeLists.txt:12 (download_vgg_descriptors)
-- xfeatures2d/vgg: Download: vgg_generated_80.i
-- Try 1 failed
CMake Warning at /home/orz/Downloads/opencv-4.2.0/cmake/OpenCVDDownload.cmake:202 (message):
  xfeatures2d/vgg: Download failed: 35; "SSL connect error"
For details please refer to the download log file:
/home/orz/Downloads/opencv-4.2.0/build/CMakeDownloadLog.txt
Call Stack (most recent call first):
/home/orz/Downloads/opencv_contrib-master/modules/xfeatures2d/cmake/download_vgg.cmake:16 (ocv_download)
/home/orz/Downloads/opencv_contrib-master/modules/xfeatures2d/CMakeLists.txt:12 (download_vgg_descriptors)
-- xfeatures2d/vgg: Download: vgg_generated_120.i

```

- 方法一: 需要手动下载每一个文件, 并修改cmake文件中的文件路径。步骤如下:

1. 打开下载所需文件对应的cmake文件。文件路径一般在报错下面的 call stack (most recent call first) 紧跟的第一行。例如

```
-- IPPICV: Download: ippicv_2019_lnx_intel64_general_20180723.tgz
-- Try 1 failed
CMake Warning at cmake/OpenCVDDownload.cmake:202 (message):
  IPPICV: Download failed: 35;"SSL connect error"

For details please refer to the download log file:

/home/orz/Downloads/opencv-4.2.0/build/CMakeDownloadLog.txt

Call Stack (most recent call first):
  3rdparty/ippicv/ippicv.cmake:42 (ocv_download)
  cmake/OpenCVFindIPP.cmake:240 (download_ippicv)
  cmake/OpenCVFindLibsPerf.cmake:12 (include)
  CMakeLists.txt:674 (include)
```

2. 找到cmake文件中的需要下载文件的URL，例

如 "https://raw.githubusercontent.com/opencv/opencv_3rdparty/\${IPPICV_COMMIT}/ippicv/"。其中的 \${IPPICV_COMMIT} 在同一个文件中有定义，为一长串的字母数字组合的字符串。最后，在URL的最后加上文件名，同样也有定义，针对不同的系统与环境会有所不同；或者在cmake编译的时候看它的输出也行。

得到最终的下载连接为

"https://raw.githubusercontent.com/opencv/opencv_3rdparty/32e315a5b106a7b89dbed51c28f8120a48b368b4/ippicv/ippicv_2019_lnx_intel64_general_20180723.tgz"

```
3rdparty > ippicv > ippicv.cmake
1 function(download_ippicv root_var)
2   set(${root_var} "" PARENT_SCOPE)
3
4   # Commit SHA in the opencv 3rdparty repo
5   set(IPPICV_COMMIT "32e315a5b106a7b89dbed51c28f8120a48b368b4")
6   # Define actual ICV versions
7   if(APPLE) ...
17 elseif((UNIX AND NOT ANDROID) OR (UNIX AND ANDROID_ABI MATCHES "x86"))
18   set(OPENCV_ICV_PLATFORM "linux")
19   set(OPENCV_ICV_PACKAGE_SUBDIR "ippicv_lnx")
20   if(X86_64)
21     set(OPENCV_ICV_NAME "ippicv_2019_lnx_intel64_general_20180723.tgz")
22     set(OPENCV_ICV_HASH "c0bd78adb4156bbf552c1dfe90599607")
23   else()
24     set(OPENCV_ICV_NAME "ippicv_2019_lnx_ia32_general_20180723.tgz")
25     set(OPENCV_ICV_HASH "4f38432c30bfd6423164b7a24bbc98a0")
26   endif()
27 elseif(WIN32 AND NOT ARM) ...
37 else()
38   return()
39 endif()
40
41 set(THE_ROOT "${OpenCV_BINARY_DIR}/3rdparty/ippicv")
42 ocv_download(FILENAME ${OPENCV_ICV_NAME}
43             HASH ${OPENCV_ICV_HASH}
44             URL
45               "${OPENCV_IPPICV_URL}"
46               "${ENV{OPENCV_IPPICV_URL}}"
47               "https://raw.githubusercontent.com/opencv/opencv_3rdparty/32e315a5b106a7b89dbed51c28f8120a48b368b4/ippicv/ippicv_2019_lnx_intel64_general_20180723.tgz"
48             DESTINATION_DIR "${THE_ROOT}"
49             ID IPPICV
50             STATUS res
51             UNPACK RELATIVE_URL)
```

3. 用IDM下载完后传给Linux，把cmake文件里下载文件的URL改成

1 | "file://{刚刚手动下载的IPP文件的上一级目录}/"

最后，我将自己所需手动下载的所有东西整理成了一个表格，仅供参考：

文件名	.cmake文件位置	下载连接
IPPICV	/home/orz/Downloads/opencv-4.2.0/3rdparty/ippicv/ippicv.cmake	https://raw.githubusercontent.com/opencv/opencv_3rdparty/32e315a5b106a7b89dbed51c28f8120a48b368b4/ippicv/ippicv_2019_Intel64_general_20180723.tgz
boostdesc_bgm.i	/home/orz/Downloads/opencv_contrib-master/modules/xfeatures2d/cmake/download_boostdesc.cmake	https://raw.githubusercontent.com/opencv/opencv_3rdparty/34e4206aef44d50e6bbcd0ab06354b52e7466d26/boostdesc_bgm.i
boostdesc_bgm_bi.i	同上	https://raw.githubusercontent.com/opencv/opencv_3rdparty/34e4206aef44d50e6bbcd0ab06354b52e7466d26/boostdesc_bgm_bi.i
boostdesc_bgm_hd.i	同上	https://raw.githubusercontent.com/opencv/opencv_3rdparty/34e4206aef44d50e6bbcd0ab06354b52e7466d26/boostdesc_bgm_hd.i
boostdesc_binboost_064.i	同上	https://raw.githubusercontent.com/opencv/opencv_3rdparty/34e4206aef44d50e6bbcd0ab06354b52e7466d26/boostdesc_binboost_064.i

文件名	.cmake文件位置	下载连接
boostdesc_binboost_128.i	同上	https://raw.githubusercontent.com/opencv3rdparty/34e4206aef44d50e6bbcd0ab06354b52e7466d26/boostdesc_binboost_128.i
boostdesc_binboost_256.i	同上	https://raw.githubusercontent.com/opencv3rdparty/34e4206aef44d50e6bbcd0ab06354b52e7466d26/boostdesc_binboost_256.i
boostdesc_lbgm.i	同上	https://raw.githubusercontent.com/opencv3rdparty/34e4206aef44d50e6bbcd0ab06354b52e7466d26/boostdesc_lbgm.i
vgg_generated_48.i	/home/orz/Downloads/opencv_contrib-master/modules/xfeatures2d/cmake/download_vgg.cmake	https://raw.githubusercontent.com/opencv3rdparty/fccf7cd6a4b12079f73bbfb21745f9babcd4eb1d/vgg_generated_48.i
vgg_generated_64.i	同上	https://raw.githubusercontent.com/opencv3rdparty/fccf7cd6a4b12079f73bbfb21745f9babcd4eb1d/vgg_generated_64.i

文件名	.cmake文件位置	下载连接
vgg_generated_80.i	同上	https://raw.githubusercontent.com/opencv/opencv_3rdparty/fccf7cd6a4b12079f73bbfb21745f9babcd4eb1d/vgg_generated_80.i
vgg_generated_120.i	同上	https://raw.githubusercontent.com/opencv/opencv_3rdparty/fccf7cd6a4b12079f73bbfb21745f9babcd4eb1d/vgg_generated_120.i
face_landmark_model.dat	/home/orz/Downloads/opencv_contrib-master/modules/face/CMakeLists.txt	https://raw.githubusercontent.com/opencv/opencv_3rdparty/8afa57abc8229d611c4937165d20e2a2d9fc5a12/face_landmark_model.dat

然后，所有 .cmake 文件的 URL 改成 "file:///home/orz/Downloads/opencv_need/"

PS: 尝试了 opencv 3.4.x 对应的 opencv_contrib，发现所有需要手动下载的文件跟 opencv 4.x 的一致，所以 opencv_need 文件夹直接共用就好了。

参考教程：[手动安装OpenCV下的IPP加速库](#)

- 方法二：对于不需要的 dnn 和 vgg，可以在 cmake 编译的时候，在命令后面加上 BUILD_opencv_* 参数，比如：

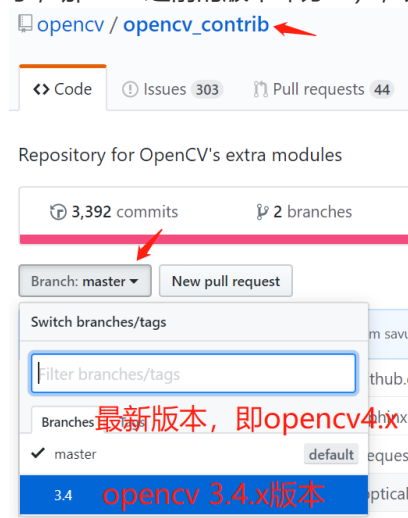
```
1 #我没用过这个方法，不知道行不行得通
2 cmake -DOPENCV_EXTRA_MODULES_PATH=<opencv_contrib>/modules -
  DBUILD_opencv_legacy=OFF <opencv_source_directory>
```

参考教程：[opencv_contrib](#)

问题四：报错 Duplicated modules NAMES has been found，如下图

```
> videostab
> viz
xfeatures2d
ximgproc
xobjdetect
xphoto
CMake Error at cmake/OpenCVModule.cmake:353 (message):github.com/opencv/opencv_c
Duplicated modules NAMES has been found
Call Stack (most recent call first):
  cmake/OpenCVModule.cmake:371 (_assert_uniqueness)
  modules/CMakeLists.txt:71(ocv_glob_modules) Merge pull request #2410 from savuor:fix
-- Configuring incomplete, errors occurred!
See also "/home/orz/Downloads/opencv-3.4.9/build/CMakeFiles/CMakeOutput.log".
See also "/home/orz/Downloads/opencv-3.4.9/build/CMakeFiles/CMakeError.log".
```

因为版本不匹配!!! opencv contrib也是分版本的!!! 在从github上下载opencv contrib的时候, 需要选择branch。master 指的是最新的版本, 即opencv 4.x, 3.4 应该指的是opencv 3.4.x的(不懂了, 那3.4.x之前的版本咋办?), 如图:



可选操作

可选软件集合

软件名称	说明	安装方法
transmission-cli	BT下载工具	<code>apt install transmission-cli</code>
aria2	下载工具	<code>apt install aria2</code>
The fuch	自动纠错软件	<code>pip install thefuck</code>
hwinfo	查看电脑配置	<code>apt install hwinfo</code> 使用时运行命令 <code>hwinfo</code> 或者 <code>hwinfo --short</code>
LS	小火车	<code>apt install sl</code>

参考教程

[transmission命令行工具集中文使用说明](#)

[aria2.conf配置文件](#)

[The Fuck — 敲错命令? 没关系, 自动纠正你的终端命令](#)

shadowsocks

推荐使用[electron-ssr](#)

安装

```
1 pip3 install https://github.com/shadowsocks/shadowsocks/archive/master.zip
2 #安装完后检查是否为3.0.0版本,若显示Shadowsocks 3.0.0则进行下一步
3 ssserver --version
```

配置: 创建shadowsocks.json, 然后复制粘贴一下内容并适当修改:

```
1 {
2     "server": "服务器ip",
3     "server_port": 6666,
4     "local_address": "127.0.0.1",
5     "local_port": 1080,
6     "password": "连接密码",
7     "timeout": 300,
8     "method": "aes-256-cfb"
9 }
```

启动Shadowsocks

```
1 ssserver -c /etc/shadowsocks.json -d start
```

[参考教程](#)

扩展Swap分区

注: 以下操作使用root身份完成

查看SWAP 分区大小

```
1 # 方法一: 只能查看大小和使用情况
2 htop
3 # 方法二: 只能查看大小和使用情况
4 free -h
5 # 方法三: 推荐, 可以查看大小、使用情况和swap文件名
6 swapon --show
```

所谓的扩展Swap分区其实就是改变Swap文件的大小。在不同系统下面, Swap文件名不一。比如, 在ubuntu18.04下, 为 /swapfile or /swap.img, 类型为file; 在ubuntu16.04下, 为 /dev/dm-1, 类型为partition

```
(planercnn) xzf@112-Linux:~/Projects/graduation$ swapon --show
NAME      TYPE      SIZE USED  PRIO
/dev/dm-1 partition 980M 961M   -1
xzf@srt-102:/snap$ swapon --show
NAME      TYPE      SIZE USED  PRIO
/swap.img file      8G    8G    -2
```

file类型的swap

注: file类型的swap就是用文件模拟分区创建swap

此处以名为 /swapfile 的swap文件做示例, 其他文件名的话将命令中的文件名修改成自己的就好了
首先停止swap分区的使用, 防止报错 Text file busy, 执行:

```
1 swapoff -a
```

这个时候再用 `swapon --show` 命令，不会输出任何东西。然后更改swap的文件大小，这里改成了16G，具体大小可以参考[这篇博客](#)。执行：

```
1 | fallocate -l 16G /swapfile
```

如果报错 `fallocate failed: operation not supported`，可以使用如下命令创建交换文件，执行

```
1 | # swap文件大小=bs*count
2 | dd if=/dev/zero of=/swapfile bs=1G count=16
```

此时需保证 `/swapfile` 的权限为600。**如果不是的话**，执行：

```
1 | chmod 600 /swapfile
```

再在文件上设置 Linux SWAP 区域，执行：

```
1 | mkswap /swapfile
```

激活 swap 文件，执行：

```
1 | swapon /swapfile
```

最后确保文件 `/etc/fstab` 有如下内容，让开机自动识别上面设置的swap文件

```
1 | /swapfile none swap sw 0 0
```

参考：[Ubuntu 18.04 swap分区扩展](#)

partition类型的swap

先占个坑

参考：[swap分区扩容](#)