# AGX & PC 环境配置

**中汽数据**

**中汽研智能网联技术（天津）有限公司**

**技术发展室**

目录

# AGX Xavier 环境配置

## 1 ibus 中文输入法安装

sudo apt-get install ibus-sunpinyin

## 2 Qt5 安装及配置

## 2.1 AGX ubuntu18.04 命令行安装 Qt

sudo apt-get install qt5-default qtcreator  -y

安装对串口类的支持：

sudo apt-get install libqt5serialport5-dev -y

安装 QWebEngineView 类：

sudo apt-get install qtwebengine5-dev qtpositioning5-dev -y

安装 multimedia 类：
sudo apt-get install qtmultimedia5-dev -y
sudo apt-get install qtmultimedia5-doc qtmultimedia5-doc-html qtmultimedia5-examples -y

安装示例帮助文档：
sudo apt-get install qt5-doc -y
sudo apt-get install qt5-doc-html qtbase5-doc-html -y
sudo apt-get install qtbase5-examples -y

卸载：
sudo apt-get remove qt5-default qtcreator

卸载连同依赖库：
sudo apt-get remove --auto-remove qt5-default qtcreator

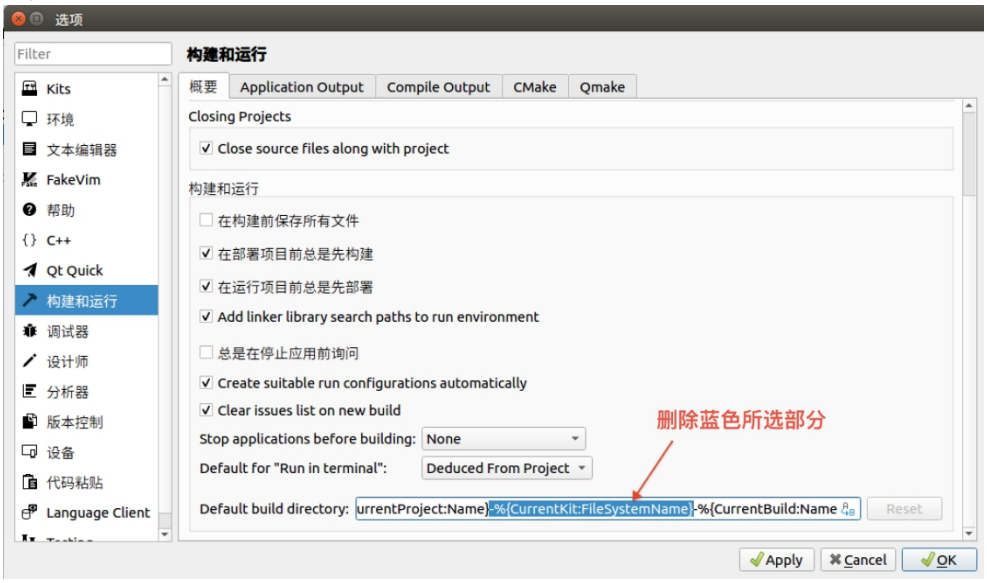
清空 qtcreator：
sudo apt-get purge qtcreator

清空 qtcreator 以及它的依赖：

sudo apt-get purge --auto-remove qtcreator

## 2.2 设置 Qt

设置 Qt 的 build 目录名称生成格式，删除图中蓝色所选部分。




## 2.3 QT Creator 配置多核编译

菜单栏->工具-> 选项->构建与运行->构建套件->点击自动检测内容->在同一页面找到 Environment ->点击 change ->

在弹出的窗口 添加 MAKEFLAGS=-j8

# 3 ROS 安装

备注：推荐使用脚本一键安装，步骤如下：

**$ cd ~**

$ git clone https://github.com/jetsonhacks/installROSXavier.git

$ cd installROSXavier

$ gedit  installROS.sh

将国外官方源：

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

更改为国内镜像源：

sudo sh -c 'echo "deb https://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

将 Key：

sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

更改为：

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key F42ED6FBAB17C654

$ ./installROS.sh -p ros-melodic-desktop-full

今后以下 3.1 ~ 3.2.7 步骤可以不再使用！！！

3.1 设置-->软件和更新：保证前四个选项都打钩


3.2 终端输入：

3.2.1 设置 sources.list（两个源选一个即可）

国外官方源：

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

国内镜像源：

sudo sh -c 'echo "deb https://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

3.2.2 设置 key

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key F42ED6FBAB17C654

3.2.3 更新 package

sudo apt-get update

3.2.4 安装 ROS 完整版（ubuntu18 对应版本为 melodic，ubuntu16 对应版本为 kinetic）

ubuntu18.04:

```
sudo apt-get install ros-melodic-desktop-full
```

ubuntu16.04:

```
sudo apt-get install ros-kinetic-desktop-full
```

3.2.5 初始化 rosdep （不翻墙可能会失败，不要紧，暂时不影响使用）

```
sudo rosdep init

rosdep update
```

3.2.6 配置 ROS 环境

ubuntu18.04:

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

ubuntu16.04:

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

3.2.7 使.bashrc 文件生效

```
source ~/.bashrc
```

# 4 VNC 远程连接配置

**注意：进行 1)~3)步骤前无法打开系统设置 -->桌面共享！！！**

1). 在 /usr/share/glib-2.0/schemas/org.gnome.Vino.gschema.xml 中添加如下键值

```
<key name='enabled' type='b'>
  <summary>Enable remote access to the desktop</summary>
  <description>
    If true, allows remote access to the desktop via the RFB
    protocol. Users on remote machines may then connect to
    the desktop using a VNC viewer.
  </description>
  <default>false</default>
```

</key>

2). 为 Gnome 编译修改后的式样

```
sudo glib-compile-schemas /usr/share/glib-2.0/schemas
```

3). 禁用 Vino 加密连接

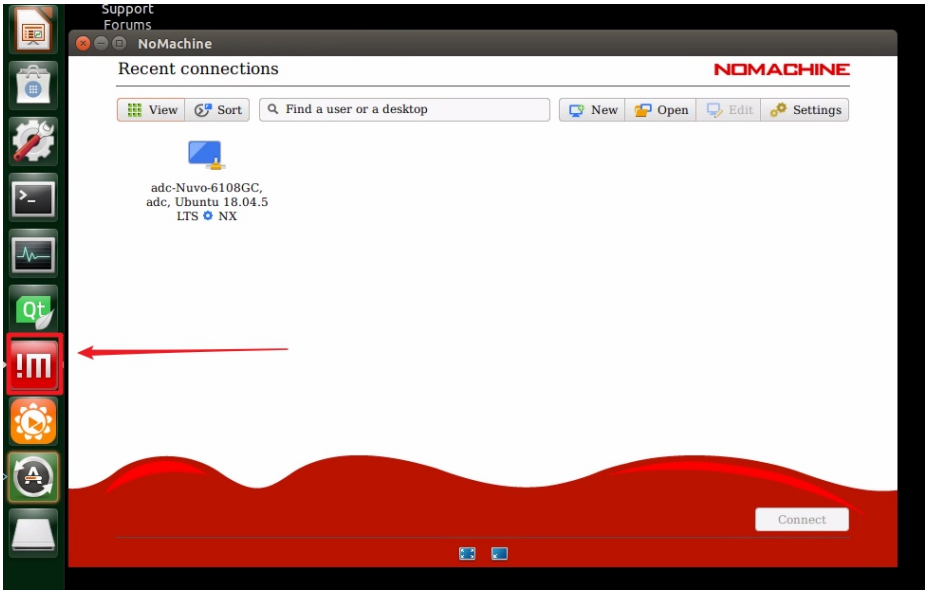
```
gsettings set org.gnome.Vino require-encryption false
```

4) 设置桌面共享
系统设置 -->桌面共享（对应选项打钩，密码设置成 123456）


5) 运行 vino-server （需要设置该文件自启动，请按照下面的“5 设置自启动项” 进行设置）

```
/usr/lib/vino/vino-server
```

Ubuntu 下建议使用 Remmina 进行远程登陆访问；
Windows 下建议使用免费开源的 VNC-viewer 进行远程登陆访问。
如果追求访问速度，还可以尝试安装和使用 NoMachine。




# 5 设置自启动项

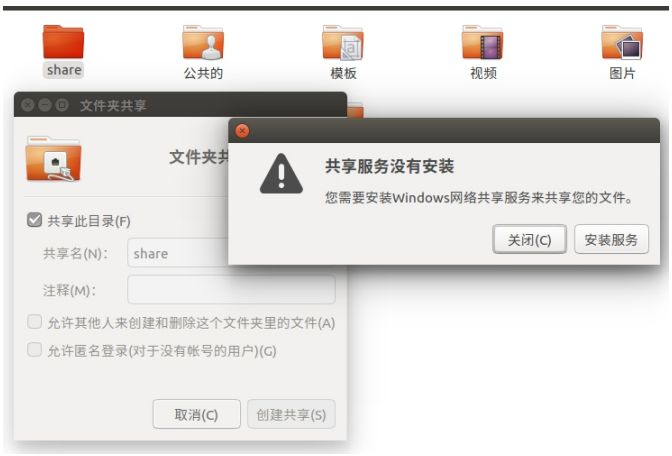
1)终端输入

gnome-session-properties

显示如下界面，点击添加（add），设置名称，浏览选择需要自启动的文件。下次开机就会自启动。（vino-server 需要自启动）


# 6 建立共享文件夹

新建一个空文件夹（取名 share），鼠标右键点击该文件夹选择“本地网络共享”，出现如下界面，选择安装服务。三个选项均要打钩。




# 7 设置 Xavier 固定 IP 地址（192.168.1.102）

点击屏幕右上角网络图标，选择编辑连接，编辑有线连接 1，IPv4 设置，方法选择手动。点击增加。填入 IP 信息，选择保存。重新连接网络 IP 设置完成。


# 8 串口、CAN 驱动

添加用户

```
sudo gpasswd --add nvidia dialout
```

将 rc.local 文件复制到/etc 目录下实现自启动

```
sudo cp rc.local /etc/
```

添加权限

```
sudo chmod +777 /etc/rc.local
```

**rc.local 文件内容：**

```
sudo modprobe can
```

```
sudo modprobe can-raw
sudo modprobe can-dev
sudo modprobe mttcan

sudo busybox devmem 0x0c303000 32 0x0000C400
sudo busybox devmem 0x0c303008 32 0x0000C458
sudo busybox devmem 0x0c303010 32 0x0000C400
sudo busybox devmem 0x0c303018 32 0x0000C458

sudo ip link set can0 type can bitrate 500000 dbitrate 2000000 berr-reporting on fd on
sudo ip link set can1 type can bitrate 500000 dbitrate 2000000 berr-reporting on fd on

sudo ip link set up can0
sudo ip link set up can1

sudo modprobe usbserial
sudo modprobe pl2303

#sudo /usr/bin/jetson_clocks
sudo jetson_clocks
sudo nvpmodel -m 0

exit 0
```

# 9.其他配置项

## 9.1 安装 busybox

```
sudo apt-get install busybox
```

## 9.2 安装支持 windows 远程桌面 mstsc 登录的软件

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

## 9.3 在 home 目录下创建一个 map 目录

用来存放地图文件，因采集地图的软件不会主动生成 map 目录，故需手动创建。

## 9.4 安装 protoc

sudo apt install protobuf-compiler

## 9.5 安装 telnet

sudo apt-get install openbsd-inetd
sudo apt-get install telnetd
sudo /etc/init.d/openbsd-inetd restart

## 9.6 安装 patchelf

sudo apt install patchelf

## 9.7 安装和配置 SVN

sudo apt install subversion

mkdir -p ~/code/ADS

cd ~/code/ADS

svn checkout https://60.247.58.121:5819/svn/adc_svn/adc_auto_driving/trunk/ADS_modularization/modularization

输入 p 永久接受，然后输入登陆用户的密码、SVN 账号及密码。

## 9.8 安装串口调试助手

sudo apt-get install cutecom

## 9.9 安装 eigen-3.3.7

**下载解压**
wget https://gitlab.com/libeigen/eigen/-/archive/3.3.7/eigen-3.3.7.tar.gz
tar -zxvf eigen-3.3.7.tar.gz
**编译安装 eigen3.3.7**
cd eigen-3.3.7/
mkdir build
cd build
cmake ..
sudo make
sudo make install

**复制 Eigen 库到 /usr/local/include 中**

sudo cp -r /usr/local/include/eigen3/Eigen /usr/local/include

## 9.10 固定内核版本

sudo apt-mark hold linux-image-generic linux-headers-generic

## 9.11 禁用 ROS 的 sudo apt-get update 更新

sudo rm /etc/apt/sources.list.d/ros-latest.list

**到此 AGX Xavier 环境配置即完成！！！！**

**以下内容如果没有需求可以不用管！！！！**

## 9.12 单独安装 boost

### 安装方法

A 使用命令安装

- sudo apt-get install libboost-dev

B 文件安装

- boost 官网下载 https://www.boost.org/
- 解压并进入文件夹 tar -zxvf 压缩文件名.tar.gz
- sudo ./bootstrap.sh
- sudo ./b2 install

## 9.13 安装 yaml-cpp

安装 yaml-cpp 依赖于 boost 库，确保 Boost library 已经存在，否则会报错。
(yaml-cpp 0.6.0 has been released! This release requires C++11, and no longer depends on Boost.)

A 使用命令安装

- sudo apt-get install libyaml-cpp-dev

B 文件安装

- git 下载源文件　　https://github.com/jbeder/yaml-cpp
- 解压文件并进入文件夹 tar -zxvf 压缩文件名.tar.gz
- mkdir build
- cd build
- cmake [-G generator] [-DYAML_BUILD_SHARED_LIBS=ON|OFF] ..
- make
- sudo make install

# 10 AGX 使用过程中遇到问题攻略

## 10.1 没有公钥

sudo apt-get update 由于没有公钥无法验证下列签名： NO_PUBKEY F42ED6FBAB17C654

解决方法：安装公钥

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys F42ED6FBAB17C654

## 10.2 关于 AGX 日志文件过大造成系统无法开机问题解决方法

### 10.2.1 编辑脚本文件

nvidia@nvidia-desktop:~$ mkdir ～/adc
nvidia@nvidia-desktop:~$ cd ～/adc
nvidia@nvidia-desktop:~$ touch auto_del_log.sh
nvidia@nvidia-desktop:~$ chmod +x auto_del_log.sh
nvidia@nvidia-desktop:~$ gedit auto_del_log.sh

文件内容：
#!/bin/sh
cat /dev/null > /var/log/syslog
cat /dev/null > /var/log/wtmp
保存，关闭退出　　注：如果有其他大的日志文件，可以追加脚本文件内容。

### 10.2.2 添加到自动更新脚本

nvidia@nvidia-desktop:~$ crontab -e　　//编辑 nvidia 用户的 cron 服务

按下 Enter 键后，如果是第一次使用终端将要求您选择一个编辑器以打开此文件。可以输入 2 选择 **nano** 编辑器，如果想每隔 1 小时执行一次脚本，可以追加输入：

* */1 * * * ~/adc/auto_del_log.sh

使用 nano 的话 Ctrl + O 输入文件名保存，Ctrl + X 退出；使用 vi 的话:wq 保存退出。
然后重启 cron 服务：
nvidia@nvidia-desktop:~$ service cron restart

### 10.2.3 备注说明：

crontab 样式注释：

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

以上非命令字段中还可以使用以下特殊字符：
星号（*）：代表所有可能的值
逗号（,）:可以用逗号隔开的值指定一个列表范围，例如：“1,2,3”
中杠（-）：表示一个整数范围，例如："2-5"表示“2,3,4,5”
正斜杠（/）:表示指定时间的间隔频率，例如“0-23/2”表示每两小时执行一次；*/10：在 minute 字段中表示每 10 分钟执行一次。

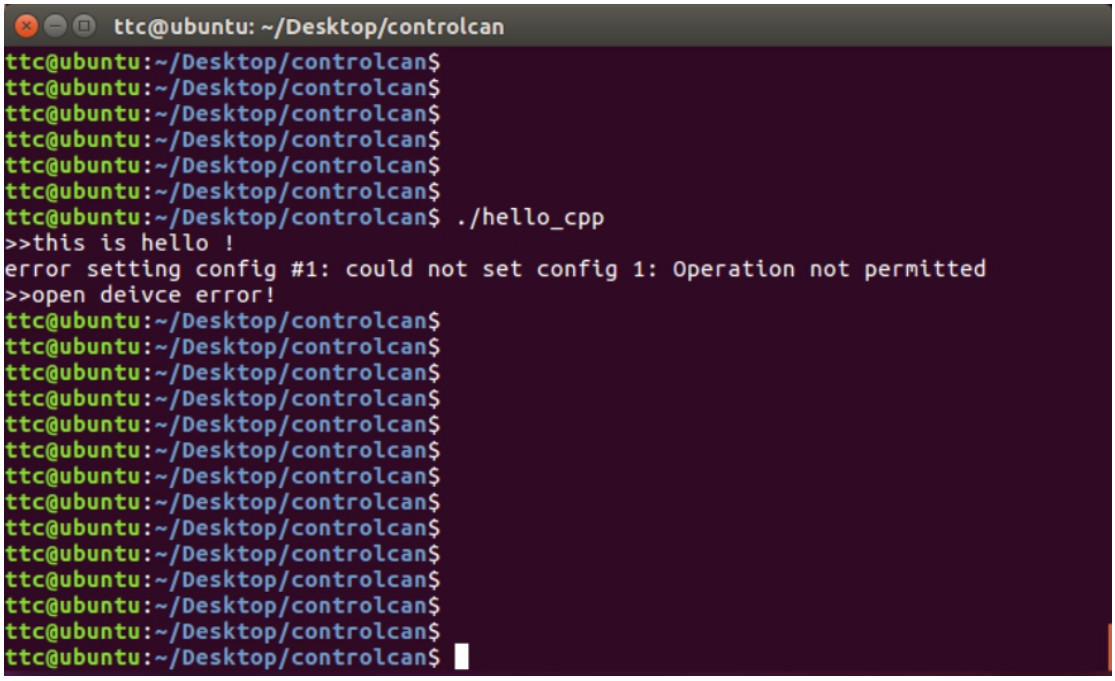
# 10.3 Kvaser 不需要签名的 Linux 驱动安装

# sudo apt-get install build-essential
# sudo apt-get install linux-headers-`uname -r`
# wget --content-disposition "https://www.kvaser.com/downloads-kvaser/?utm_source=software&utm_ean=7330130980754&utm_status=latest"
# tar xvzf linuxcan.tar.gz
# cd linuxcan
# make
# sudo make install

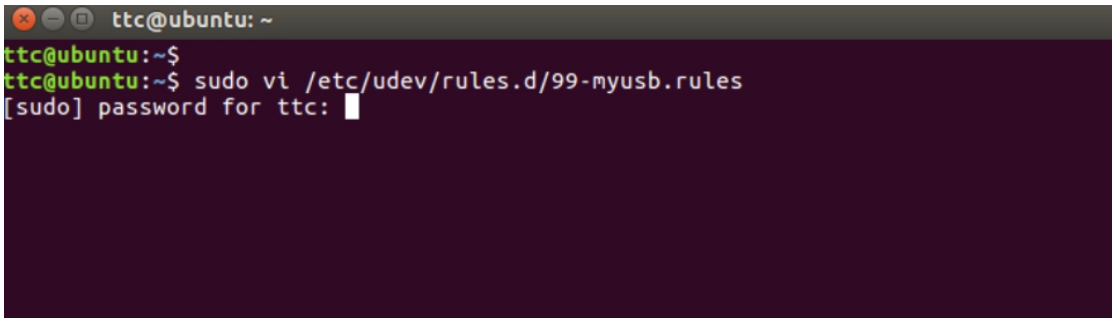
# 10.4 VCI USB 权限设置

因 Linux 系统下将涉及到 usb 底层驱动的调用,运行时,一定要加 sudo 获取权限运行,否则 USB 设备没有权限操作。 现提供一种 USB 权限设置,配置后,可以不加权限运行。

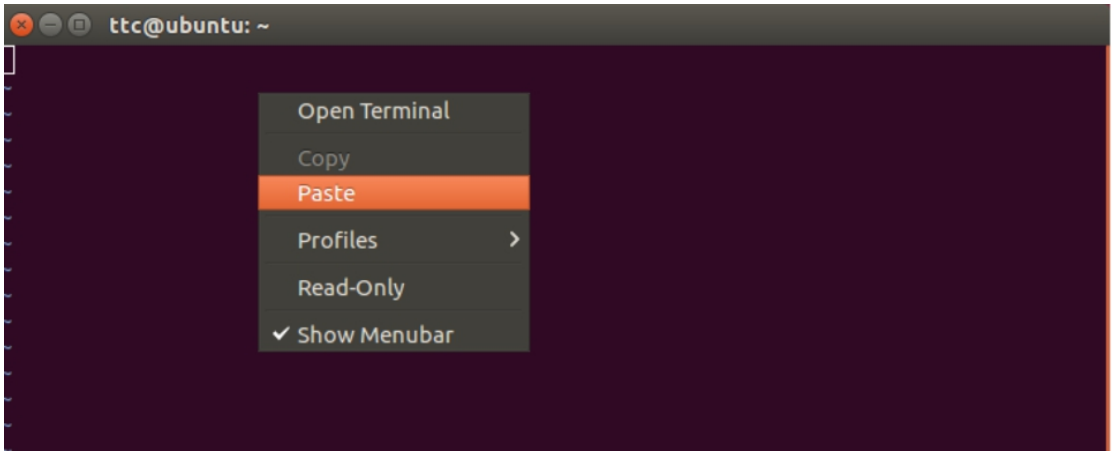
10.4.1 不加 sudo 权限运行,将会报如下错误:




10.4.2 下面创建一个新的 udev 规则。名称取为:99-myusb.rules
sudo vi /etc/udev/rules.d/99-myusb.rules
注意:
1、 数字 99 最好不要改动,否则可能设置失败
2、 要加 sudo

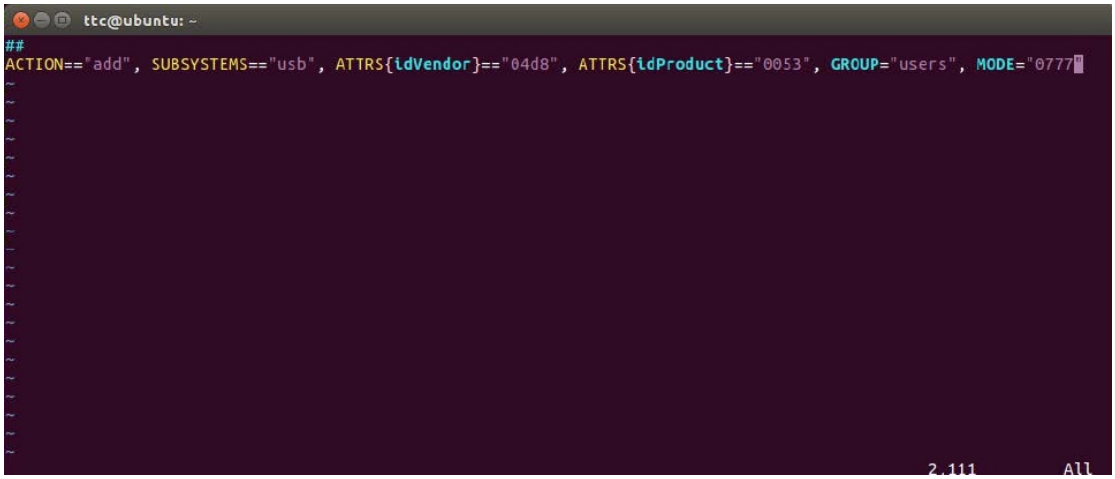



10.4.3 把以下两行代码复制到新建的 99-myusb.rules 文件中
注意:按键盘上 Insert 键切换到“REPLACE”输入模式
##
ACTION=="add",SUBSYSTEMS=="usb", ATTRS{idVendor}=="04d8",GROUP="users",
MODE="0777"
ATTRS{idProduct}=="0053",

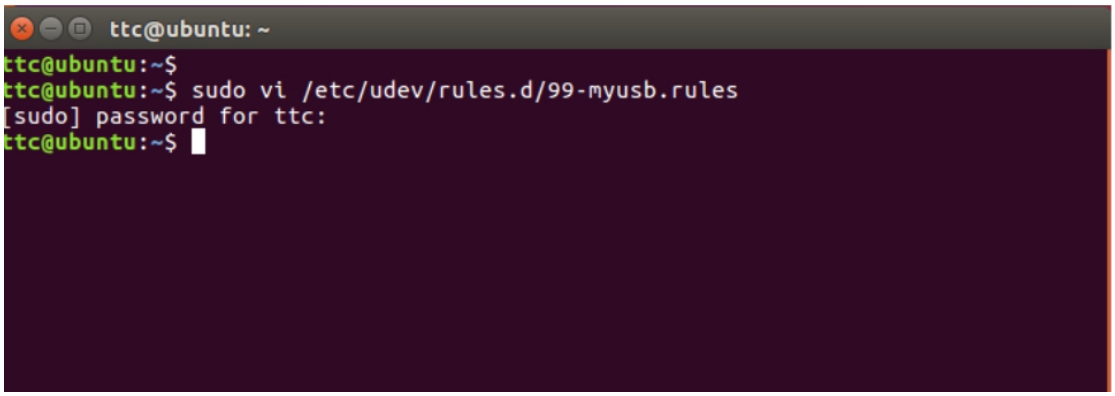




10.4.4 按“Esc”键一次




10.4.5 直接输入“:wq”回车,即保存退出


10.4.6 插拔一下 USBCAN 设备或重启一下电脑后,即可不加 sudo 权限运行程序了。

# 10.5 远程桌面登陆不允许配置网络

在使用远程桌面操系统过程中，不允许关闭“启用联网”选项，否则远程登陆会立即失效。


# 10.6 ubuntu 为 USB 串口绑定固定的设备名

ubuntu USB 设备号为从零开始依次累加，多个设备每次开机后设备号不固定.
udev 的规则，可以参考博客：http://blog.csdn.net/cokewei/article/details/8281239
将端口重映射到固定的名字，并且设置其权限为可读。使用对应的 id 设备映射到固定的名字上。

使用命令 lsusb 查看对应的 usb 端口信息

```
apollo@apollo-ThinkPad-X390:~$ lsusb
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 0bda:0411 Realtek Semiconductor Corp.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 04ca:7070 Lite-On Technology Corp.
Bus 001 Device 007: ID 0bda:8179 Realtek Semiconductor Corp. RTL8188EUS 802.11n
Wireless Network Adapter
Bus 001 Device 011: ID 0403:6001 Future Technology Devices International, Ltd FT
232 USB-Serial (UART) IC
Bus 001 Device 003: ID 0bda:5411 Realtek Semiconductor Corp.
Bus 001 Device 002: ID 046d:c534 Logitech, Inc. Unifying Receiver
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
apollo@apollo-ThinkPad-X390:~$
```

udev 的规则

$kernel, %k：设备的内核设备名称，例如：sda、cdrom。

ID 0403:6001 表示 usb 设备的 ID(这个 ID 由芯片制造商设置，可以唯一表示该设备)

0403 usb_device_descriptor.idVendor --VID

6001 usb_device_descriptor.idProduct --PID

依据上面信息写 udev 文件

串口设备信息

Bus 001 Device 011: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) IC

sudo vim /etc/udev/rules.d/p900.rules

```
KERNEL=="ttyUSB*", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", MODE:="0777", SYMLINK+="p900"
```

创建生效后**重新插拔 USB**(也可以用命令实现)

service udev reload

service udev restart

ls -l /dev | grep ttyUSB*

显示

```
apollo@apollo-ThinkPad-X390:~$ ls -l /dev | grep ttyUSB*
lrwxrwxrwx  1 root   root             7 Sep  7 10:40 p900 -> ttyUSB0
crwxrwxrwx  1 root   dialout 188,     0 Sep  7 10:40 ttyUSB0
apollo@apollo-ThinkPad-X390:~$
```

多个不同型号设备可使用这种方法来区分。

打开设备时，用（/dev/p900）即可。

# 10.7 刷机 Jetpack4.4 后找不到 OpenCV

Xavier刷机完成之后，查看OpenCV版本，会报错No package 'opencv' found。

报这个错是因为没有找到opencv.pc，而刷机时是安装了OpenCV的，不过是OpenCV4，我在/usr/lib/aarch64-linux-gnu/pkgconfig中找到了opencv4.pc，把它复制到/usr/lib/pkgconfig下，并改名为opencv.pc就好了。

# 10.8 AGX 更新源失败问题

默认的 Ubuntu 的源下载速度太慢，可以通过替换

/etc/lib/apt/sources.list

文件改为国内源（注意修改之前先将原文件备份），还有一点要注意，换源需要换 ARM 的源，不要换成了 pc 平台的软件源。这里推荐两个国内源，将原来文件里面的内容全部替换成下面两个源之一就可以：

清华源：

```
#清华源
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-updates main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-security main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-security main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic-backports main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic main universe restricted
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports/ bionic main universe restricted
```

和科大的源：

```
#中科大源
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-updates main restricted universe multiverse
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-security main restricted universe multiverse
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-security main restricted universe multiverse
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic-backports main restricted universe multiverse
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic main universe restricted
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ bionic main universe restricted
```

替换之后，执行

sudo apt-get update

如果遇到如下错误

```
已下载 49.2 MB，耗时 2分 4秒 (399 kB/s)
正在读取软件包列表... 完成
E: 无法下载 http://mirrors.ustc.edu.cn/ubuntu-ports/dists/bionic-updates/main/binary-i386/Packages  404  Not Found [IP: 202.141.176.110 80]
E: 无法下载 http://mirrors.ustc.edu.cn/ubuntu-ports/dists/bionic-security/main/binary-i386/Packages  404  Not Found [IP: 202.141.176.110 80]
E: 无法下载 http://mirrors.ustc.edu.cn/ubuntu-ports/dists/bionic-backports/main/binary-i386/Packages  404  Not Found [IP: 202.141.176.110 80]
E: 无法下载 http://mirrors.ustc.edu.cn/ubuntu-ports/dists/bionic/main/binary-i386/Packages  404  Not Found [IP: 202.141.176.110 80]
E: 部分索引文件下载失败。如果忽略它们，那将转而使用旧的索引文件。
```

执行

dpkg --print-architecture

查看当前 CPU 的架构，然后打印如下：

```
nvidia@nvidia-desktop:~$ dpkg --print-architecture
arm64
```

终端执行

dpkg --print-foreign-architectures

查看设置的多架构支持，打印如下

```
nvidia@nvidia-desktop:~$ dpkg --print-foreign-architectures
i386
```

发现多架构支持被设置了 i386，然而我们的平台是 ARM 架构的，不需要 x86 平台的支持，先前的问题可能就是这个原因导致的，于是尝试删除这个设置

sudo dpkg --remove-architecture i386

然后更新源

sudo apt-get update

成功!

# 11 Jetson AGX Xavier 系统备份与克隆系统到新 Xavier 板

## 11.1 方法一（DD 方式备份系统）

**准备：**

- 需要有刷机包，通过刷机教程完成安装之后就有。
- 安装有刷机包的 PC 机

**备份：**

- xavier 正常进入系统
- 标记系统为只读

```
sudo echo "u" | sudo dd of=/proc/sysrq-trigger
```

- 通过 DD 方式备份系统

```
sudo dd if=/dev/mmcblk0p1 | ssh ubu@192.168.0.85 dd of=/home/ubu/xavier-image.raw
```

- ubu 替换为自己的主机
- 其中生成的 xavier-image.raw 就是镜像

**还原：**

- 进入刷机包

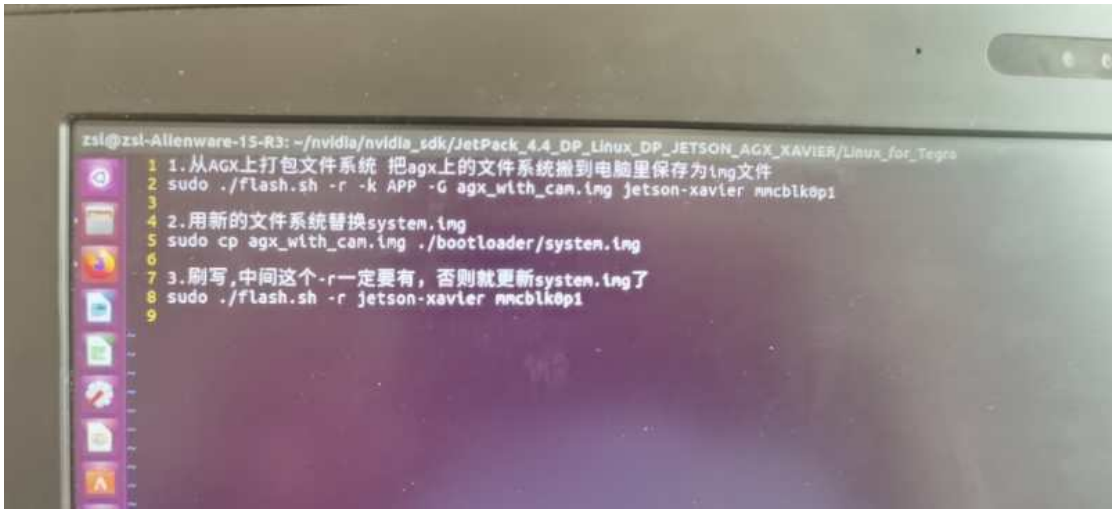
```
cd /home/nvidia/Downloads/Xavier/Linux_for_Tegra/bootloader/
mv system.img system.img.bk
ln -s /home/ubu/xavier-image.raw system.img
```

- 还原

```
cd /home/nvidia/Downloads/Xavier/Linux_for_Tegra/
sudo ./flash.sh -r jetson-xavier mmcblk0p1
```

- 大概要 20 分钟左右完成刷机，即可恢复系统。

## 11.2 方法二（直接使用 flash.sh 备份系统）




# 12 查看系统和软件版本参数状态

## 12.1 Jetpack4.2 版本信息

说明：

- 介绍如何查看 Xavier 系统和软件版本和参数
- 测试 Xavier 版本：Jetpack4.2.3

步骤：

- 查看架构

```
sudo apt-cache show nvidia-jetpack
```

- 查看 Jetson Xavier L4T 版本

```
ubuntu@AiROS:~$ head -n 1 /etc/nv_tegra_release
# R32 (release), REVISION: 2.3, GCID: 17644089, BOARD: t186ref, EABI: aarch64, DATE: Tue Nov  5 21:48:17 UTC 2019
```

- 查看 TensorRT 的版本

```
ubuntu@AiROS:~$ dpkg -l | grep TensorRT
ii  graphsurgeon-tf      5.1.6-1+cuda10.0              arm64        GraphSurgeon for TensorRT package
ii  libnvinfer-dev       5.1.6-1+cuda10.0              arm64        TensorRT development libraries and headers
ii  libnvinfer-samples   5.1.6-1+cuda10.0              all          TensorRT samples and documentation
ii  libnvinfer5          5.1.6-1+cuda10.0              arm64        TensorRT runtime libraries
ii  python-libnvinfer    5.1.6-1+cuda10.0              arm64        Python bindings for TensorRT
ii  python-libnvinfer-dev    5.1.6-1+cuda10.0          arm64        Python development package for TensorRT
ii  python3-libnvinfer       5.1.6-1+cuda10.0          arm64        Python 3 bindings for TensorRT
ii  python3-libnvinfer-dev   5.1.6-1+cuda10.0          arm64        Python 3 development package for TensorRT
ii  tensorrt                 5.1.6.1-1+cuda10.0        arm64        Meta package of TensorRT
ii  uff-converter-tf         5.1.6-1+cuda10.0          arm64        UFF converter for TensorRT package
```

- 查看系统版本

```
ubuntu@AiROS:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.3 LTS"
```

- 查看系统内核：

```
ubuntu@AiROS:~$ uname -a
Linux AiROS 4.9.140-tegra #1 SMP PREEMPT Tue Nov 5 13:37:19 PST 2019 aarch64 aarch64 aarch64 GNU/Linux
```

- 查看内存：

```
ubuntu@AiROS:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:          15690        2630        3852          82        9207       12921
Swap:          7845           0        7845
```

- 查看 CPU

```
ubuntu@AiROS:~$ lscpu
Architecture:        aarch64
Byte Order:          Little Endian
CPU(s):              8
On-line CPU(s) list: 0-3
Off-line CPU(s) list: 4-7
Thread(s) per core:  1
Core(s) per socket:  2
Socket(s):           2
Vendor ID:           Nvidia
Model:               0
Model name:          ARMv8 Processor rev 0 (v8l)
Stepping:            0x0
CPU max MHz:         2265.6001
CPU min MHz:         115.2000
BogoMIPS:            62.50
L1d cache:           64K
L1i cache:           128K
L2 cache:            2048K
L3 cache:            4096K
Flags:               fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp
```

- 查看硬盘空间

```
ubuntu@AiROS:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p1   28G   12G   15G  44% /
none            7.7G     0  7.7G   0% /dev
tmpfs           7.7G   10M  7.7G   1% /dev/shm
tmpfs           7.7G   56M  7.7G   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           7.7G     0  7.7G   0% /sys/fs/cgroup
tmpfs           1.6G  148K  1.6G   1% /run/user/1000
```

- 查看 cuda 版本

```
ubuntu@AiROS:~$ /usr/local/cuda/bin/nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Mon_Mar_11_22:13:24_CDT_2019
Cuda compilation tools, release 10.0, V10.0.326
```

- 查看 cudnn 版本

ubuntu@AiROS:~$ cat /usr/include/cudnn.h | grep CUDNN_MAJOR -A 2

#define CUDNN_MAJOR 7

#define CUDNN_MINOR 5

#define CUDNN_PATCHLEVEL 0

--

#define CUDNN_VERSION (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)

#include "driver_types.h"

- 查看 opencv 版本

ubuntu@AiROS:~$ pkg-config --modversion opencv

3.3.1

- 查看 python 版本

ubuntu@AiROS:~$ python -V

Python 2.7.15+

- 查看 python3 版本

ubuntu@AiROS:~$ python3 -V

Python 3.6.9

# 12.2 JetPack 4.4 版本信息

JetPack 4.4 Highlights:

- Support for the new Jetson Xavier NX module [1] and Jetson Xavier NX Developer Kit
- Support for CUDA 10.2, and TensorRT 7.1.3 and cuDNN 8.0.0
- Support for Vulkan 1.2 [1] and VPI 0.3 [1] (Developer Preview)
- Support for upgrading JetPack and L4T using Debian package management tool
- Support for Generic Timestamping Engine (GTE) for Jetson AGX Xavier and Jetson Xavier NX
- Support for Dynamic Frequency Scaling (DFS) for Video Image Compositor (VIC) using actmon
- SE (Security Engine) samples to demonstrate hardware backed authentication and encryption capabilities of Jetson TX2 series, Jetson AGX Xavier and Jetson Xavier NX modules.
- Utility to fuse multiple Jetson modules simultaneously
- Option to specify APP partition size on the microSD card during initial configuration at first boot of Jetson Nano Developer Kit

JetPack 4.4 components:

- L4T R32.4.3
- CUDA 10.2
- cuDNN 8.0.0
- TensorRT 7.1.3
- VisionWorks 1.6
- OpenCV 4.1
- Vulkan 1.2
- VPI 0.3 (Developer Preview)
- Nsight Systems 2020.2
- Nsight Graphics 2020.1
- Nsight Compute 2019.3

# 13 系统使用小技巧

## 13.1 查看系统对外 IP

#curl cip.cc

## 13.2 安装和使用比 tree 更好用的文件查看工具

#sudo apt-get install ranger
#ranger

## 13.3 监视指定网络接口的数据包

#sudo apt-get install tcpdump
#tcpdump -i eth0

## 13.4 监视系统端口

#sudo apt-get install lsof
#lsof -i

## 13.5 每次上电同步系统时间防止编译报错

#sudo apt-get install ntpdate
然后将如下命令添加到/etc/rc.local 最后一行

sudo ntpdate time.windows.com

## 13.6 使用向日葵远程控制软件

软件安装包放在了 SVN 云端，需要同步到本地
#svn checkout https://60.247.58.121:5819/svn/adc_svn/adc_auto_driving/trunk/ADS_modularization/modularization/thirdpartylib/Sunlogin
切换到 Sunlogin 目录下，执行安装命令
#sudo dpkg -i sunlogin.deb
安装并切换 lightdm 图形界面并重启系统,否则回向日葵软件会闪退。
$ sudo apt install lightdm
$ sudo dpkg-reconfigure lightdm
$ reboot

## 13.7 安装*.deb 包时缺少依赖

```
sudo dpkg -i *.deb
```

如果出现依赖问题，执行：

```
sudo apt install -f
```

然后再次安装*.deb。

## 13.8 ubuntu 快速查找文件

```
sudo apt-get install mlocate
sudo updatedb
locate xxx  #xxx 是文件名
```

## 13.9 Jetson AGX Xavier 安装 jtop

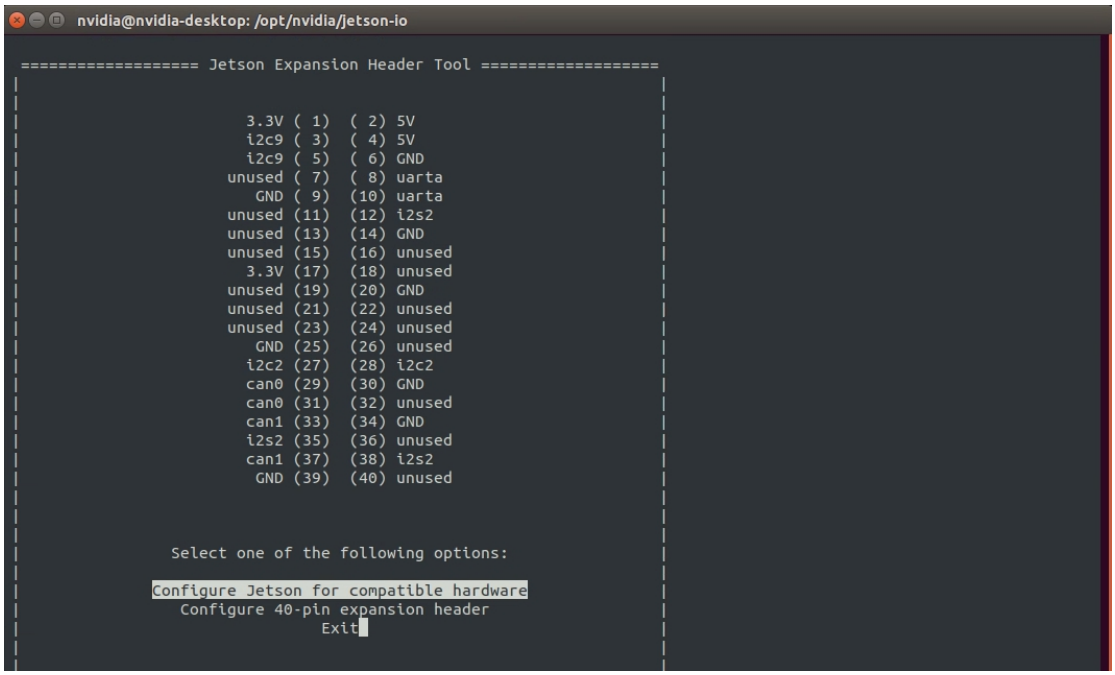
```
sudo apt install python3-pip
sudo -H pip3 install -U jetson-stats
```

运行：sudo jtop

有五个页面显示设备状态

## 13.10 Jetson AGX Xavier 查看 40pin 引脚定义

```
sudo /opt/nvidia/jetson-io/jetson-io.py
```

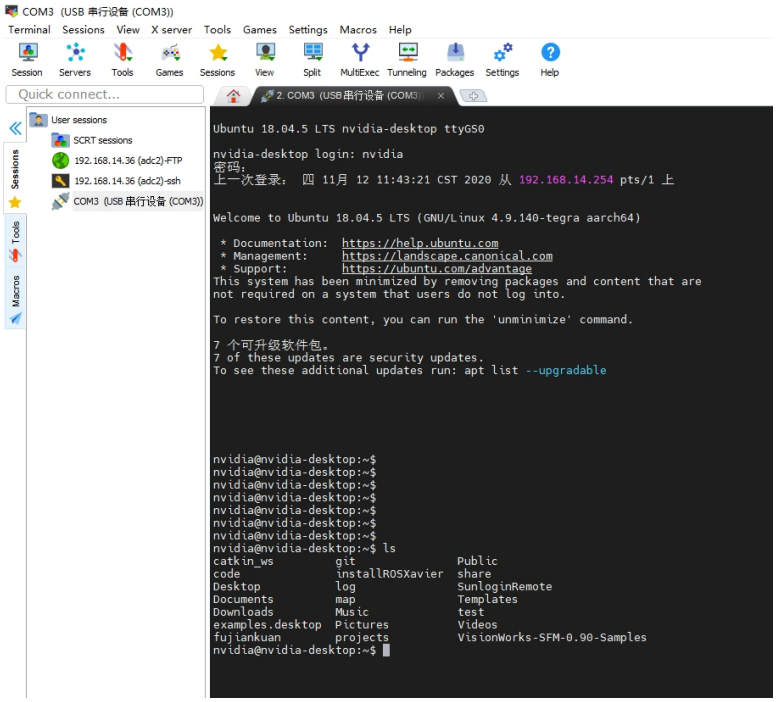



按 Ctrl+C 退出

# 13.11 Jetson AGX Xavier USB 串口和网卡

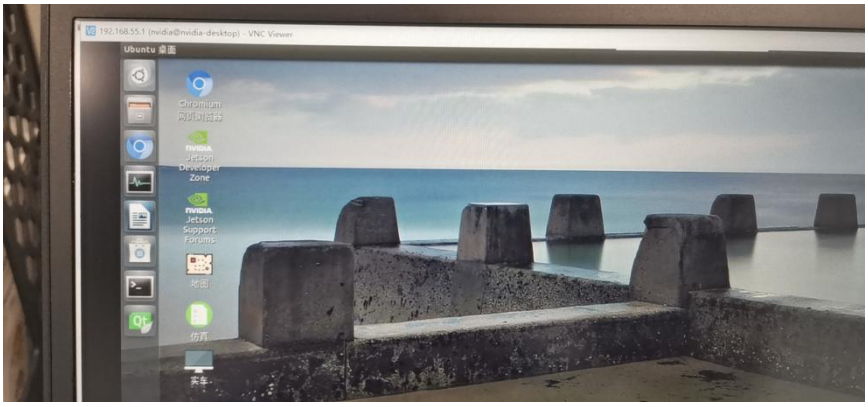
AGX 有个调试串口，引出来接到电脑上，在没有接屏幕的 AGX 可以用来进系统执行用户命令。

使用上位机连接 USB 串口，登录用户名 nvidia。




这个 usb 也可以虚拟出来一个网卡，直接用电脑远程桌面连接，agx 的 ip 为 192.168.55.1。

```
l4tbr0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.55.1  netmask 255.255.255.0  broadcast 192.168.55.255
        inet6 fe80::1  prefixlen 128  scopeid 0x20<link>
        inet6 fe80::e030:f7ff:fe28:df15  prefixlen 64  scopeid 0x20<link>
        ether e2:30:f7:28:df:15  txqueuelen 1000  (以太网)
        RX packets 10  bytes 1297 (1.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 6936 (6.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



# 13.12 Jetson AGX Xavier CAN 状态查看

ip -details -statistics link show can0

ip -details -statistics link show can1

```
nvidia@nvidia-desktop:~$ ip  -details -statistics link show can0
7: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 72 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 10
    link/can  promiscuity 0
    can <BERR-REPORTING,FD> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0
          bitrate 500000 sample-point 0.870
          tq 20 prop-seg 43 phase-seg1 43 phase-seg2 13 sjw 1
          mttcan: tseg1 2..255 tseg2 0..127 sjw 1..127 brp 1..511 brp-inc 1
          dbitrate 2000000 dsample-point 0.720
          dtq 20 dprop-seg 8 dphase-seg1 9 dphase-seg2 7 dsjw 1
          mttcan: dtseg1 1..31 dtseg2 0..15 dsjw 1..15 dbrp 1..15 dbrp-inc 1
          clock 50000000
          re-started bus-errors arbit-lost error-warn error-pass bus-off
          0          0          0          0          0          0          numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
    RX: bytes  packets  errors  dropped overrun mcast
    0          0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    0          0        0       0       0       0
nvidia@nvidia-desktop:~$
```

# 13.13 Jetson AGX Xavier 扩展 SSD

查看硬盘信息:

sudo fdisk -lu

```
Disk /dev/mmcblk0boot1: 8 MiB, 8388608 bytes, 16384 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mmcblk0boot0: 8 MiB, 8388608 bytes, 16384 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/nvme0n1: 232.9 GiB, 250059350016 bytes, 488397168 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xd80dc8f5
nvidia@miivii-tegra:~$
```

格式化硬盘:

sudo mkfs -t ext4 /dev/nvme0n1

```
nvidia@miivii-tegra:~$ sudo mkfs -t ext4 /dev/nvme0n1
mke2fs 1.42.13 (17-May-2015)
Found a dos partition table in /dev/nvme0n1
Proceed anyway? (y,n) y
```

查看硬盘 UUID:

sudo blkid /dev/nvme0n1

```
nvidia@miivii-tegra:~$ sudo blkid /dev/nvme0n1
/dev/nvme0n1: UUID="6e643050-77bb-40d3-97b4-7835fc016afb" TYPE="ext4"
nvidia@miivii-tegra:~$
```

开机自动挂载硬盘的设置方法： **在/etc/systemd/system 路径下创建一个 systemd 服务，用来开机自动执行挂载硬盘，如 ：agx_mount_ssd.service**

```
#创建服务 agx_mount_ssd.service
vim agx_mount_ssd.service
[Unit]
Description=AGX specific script
After=udev.service

[Service]
ExecStart=/etc/systemd/agx_mount_ssd.sh

[Install]
WantedBy=multi-user.target
```

**在/etc/systemd/路径下创建一个脚本，用来挂载硬盘，如： agx_mount_ssd.sh**

```
#创建服务脚本 agx_mount_ssd.sh
vim agx_mount_ssd.sh
#!/bin/bash
mount -o rw /dev/nvme0n1 /home/nvidia/workspace
```

**为创建的脚本文件添加可执行权限**

```
sudo chmod +x agx_mount_ssd.sh
```

**将挂载硬盘的服务设置为开机自启动**

```
sudo systemctl enable agx_mount_ssd.service
```

# 13.14 UART 接口配置方法

打开/dev/(folder)下面对应的设备节点，设置波特率，停止位，奇偶校验位，数据位等。可以使用 stty 命令配置串口的波特率，停止位，奇偶校验位，数据位等，详细见 stty 命令说明。

命令示例，请将< >中的信息修改为想要调整的串口节点号，具体对应关系请参考【接口说明】部分

```
sudo stty -F /dev/<UART_XXX> speed 115200 cs8 -parenb -cstopb -echo
```

输出数据测试

```
sudo echo “agx tty debug” > /dev/<UART_XXX>
```

使用下面命令接收输入数据

```
sudo cat /dev/<UART_XXX>
```

# 13.15 CAN 口配置方法

CAN 设备配置使用方法，参考 https://github.com/linux-can/can-utils 中的 cansend.c 和 candump.c

测试命令：

```
sudo modprobe can
sudo modprobe can_raw
sudo modprobe mttcan
sudo ip link set can0 type can bitrate 1000000 dbitrate 1000000 berr-reporting on fd on
sudo ip link set up can0
sudo cansend can0 123#abcdabcd
sudo candump can0
sudo ip -details -statistics link show can0
sudo ifconfig can0 down
```

CAN fd 配置使用方法：

```
sudo modprobe can
sudo modprobe can_raw
sudo modprobe mttcan
sudo ip link set can0 type can bitrate 500000 dbitrate 2000000 berr-reporting on fd on
sudo ip link set up can0
sudo cansend can0 213##011
```

CAN fd 和 CAN 2.0 的区别:

1)

```
sudo ip link set can0 type can bitrate 500000 dbitrate 2000000 berr-reporting on fd on
```

其中 bitrate 为 can2.0 模式下的波特率； dbitrate 为 can fd 模式下的波特率，根据官方文档，这个值最大可配置为 5M，一般应用最好采用 2M；

2)

```
sudo cansend can0 213##011
```

发送命令中，id 与数据之间多了一个#，并且##后的第一个字节(0)为 canfd_frame.flags 的值，范围为 0~F； canfd_frame.flags 后面的字节(11)为第一个数据，一次最多可以传输 64 个字节。

https://www.coderdbc.com/codegen/uploaddbc

.dbc 文件在线直接生成 C 代码，自动化生成的代码很工整。

# 15.16 无外接显示器时设置显示桌面分辨率

如果没有显示器连接，默认 VNC 连接后的分辨率为 640x480 ，通过修改添加如下内容，将其默认 VNC 分辨率设置

```
sudo vi /etc/X11/xorg.conf
Section "Screen"
Identifier "Default Screen"
Monitor "Configured Monitor"
Device "Tegra0"
SubSection "Display"
Depth 24
Virtual 1920 1080 # Modify the resolution by editing these values
EndSubSection
EndSection
```

# 15.17 编写 linux 桌面图标启动

直接在桌面上建立一个后缀为 desktop 的文件，可以按照 ubuntu 官方提示修改。
https://help.ubuntu.com/community/UnityLaunchersAndDesktopFiles

```
#-- 全局安装（所有用户可用),将 xxx.desktop 复制到/usr/share/applications
#-- 当前用户可用, 将 xxx.desktop 复制到 ~/.local/share/applications 目录即可
#--appName.desktop
[Desktop Entry]
Version=1.0 #app 的版本
Name=ADCIV#app 的名字
Comment= this app use for xxx #说明信息
Exec=/path/to/your/QtApp/ADCIV#app 的执行路径，绝对路径
Icon=/path/to/your/app_icon/ADCIV.ico #icon 路径，绝对路径
Terminal=false #是否在终端启动，效果自己试一下就知道了
Type=Application
Categories=Utility;Application;
```

# 15.18 Ubuntu 上实现 Win 下 windeployqt 功能脚本

```
#! /bin/bash
EXE='embeddedSerialPort'
PWD=`pwd`
files=`ldd $EXE | awk '{ if(match($3,"^/")) printf("%s "),$3 }'`
cp $files $PWD
```

# 15.19 Qt 程序发布

**以下内容来自网络，如有侵权请联系删除！**

0：在 ubuntu 下将 Qt 程序打包，发布成 Debian 包的过程

下面的步骤对于所有的打包程序都是通用的，所以完全可以写一个脚本，把这个包过程自动化，但是在自动化之前，我们需要弄清楚，一步一步打包是如何进行的，下面详细讲解。

1：建立好如下文件（夹）结构

```
└──── mydeb  #目录 名字自取
  |
  ├──── application #目录 名字确定
  |   ├──── catchGG #目录 自己应用程序的名字
  |   |   ├──── catchGG   #程序或文件 Qt 生成的 或其它 可运行程序
  |   |   └──── pycatchgg #程序或文件 Qt 生成的 或其它 可运行程序
  |   └──── lib
  |       ├──── catchGG.desktop  #文件 需要复制到/usr/share/applications/目录，dash 中可搜索
  |       ├──── catchGG.png     #文件  应用程序在 unity 中显示的图标 需要复制到/usr/share/pixmaps/目录
  |
  |
  └──── DEBIAN      #目录 名字确定
     ├──── control   #文件 名字不可改
     ├──── postinst  #脚本文件 名字不可改 ，运行 sudo dpkg -i xx.deb 命令时，会运行这个脚本
     ├──── postrm    #脚本文件 名字不可改 ，运行 sudo dpkg -r app 命令时，会运行这个脚本
```

上面结构中： mydeb 目录下面的两个目录 application 和 DEBIAN 名字不可改变的

mydeb/application/lib 目录下面的 应用程序图标文件就不用说了，主要是.desktop 文件 里面应该写些什么内容

mydeb/DEBIAN/目录下面的三个文件的内容，是我们需要自己动手写的

2：.desktop 文件的基本内容

#下面是一个基本的例子，按下面格式，改成自己需要的内容即可

```
[Desktop Entry]
Version=0.1   #应用程序版本
Name=catchGG  #应用程序名
Comment=Back up your data with one click  #应用程序描述
Exec=/usr/bin/catchGG                #可运行应用程序最终的绝对路径
keywords=google,catchgg,catchGG        #在 dasn 串搜索时，可用的关键字
Terminal=false                    #运行时不需要打开终端
Type=Application                  #应用程序类型，在 dash 和分类中会有显示，还有其它的类型
Categories=Utility;Application;        #应用程序的分类，工具/应用程序
```

Icon=/usr/share/pixmaps/catchGG.png　　#应用程序图标名的绝对路径

3：control 文件的基本内容
#下面是一个基本的例子，就不一个一个说了，都很容易
#有一点需要说一下，很多人都不知道 Depends 后面的依赖库名字是如何得到的，实际上非常容易
使用 ldd 命令，后面加你开发的 Qt 程序的名字，就可以得到你的 Qt 程序需要哪些动态链接库（配合 grep qt 命令使用）

Package:catchGG
Version:0.1
Section:utils
Priority:extra
Maintainer:Me You(QQ:397916230)
Depends:libqt5widgets5(>=5.0),libqt5gui5(>=5.0),libqt5core5a(>=5.0),libqt5x11extras5(>=5.0),libqt5x11extras5-dev(>=5.0)
Architecture:amd64
Description: The software is an opensource package from 397916230@qq.com

4:postinst 脚本的基本内容
#在说脚本里面内容时，先告诉大家如何创建这个脚本，并使这个脚本具有个运行的权限
touch postinst &&  chmod 755 postinst

#下面是这个脚本的基本内容----一个例子
#可以看到就是一个基本的 shell 脚本，做了如下事情
#1：告诉用户，安装过程中做了什么 echo 命令显示
#2：把应用程序图标文件和.desktop 文件 mv 到对应的系统目录中 （注意使用的是/applicaton/lib 这种路径格式）
#3：把可运行的 Qt 开发的程序，或其它二进行可运行程序（或其它库文件）复制到系统的 path 路径下
#4：设置一些基本的用户环境变量

echo "Start to install"
mv -f /application/lib/catchGG.desktop /usr/share/applications/
mv -f /application/lib/catchGG.png /usr/share/pixmaps/
mv -f /application/catchGG/* /usr/bin/

echo "set QT_PLUGIN_PATH in file .profile"
echo "QT_PLUGIN_PATH=/usr/lib/x86_64-linux-gnu/qt5/plugins" >> ~/.profile
echo "Install ok"

5:postrm 脚本的基本内容
#在说脚本里面内容时，先告诉大家如何创建这个脚本，并使这个脚本具有个运行的权限
touch postrm &&  chmod 755 postrm

#下面是这个脚本的基本内容----一个例子

#把之间安装在系统中的文件都给删除，就搞定了，

```
echo "Start to remove"
rm -rf  /usr/share/applications/catchGG.desktop
rm -rf  /usr/share/pixmaps/catchGG.png
rm -rf  /usr/bin/catchGG
rm -rf  /usr/bin/pycatchgg
echo "Remove finished!
```

6：总结
打包的过程照着上面的步骤，一步一步来，还是非常方便的

打包命令也非常容易,如下：
sudo dpkg -b mydeb/ Qtapp_0.1_amd64.deb
打包完成后，进行安装，删除测试

```
# 安装
sudo dpkg -i  Qtapp_0.1_amd64.deb
sudo apt-get install -f #自动 安装程序需要的依赖库
# 删除
sudo dpkg -r app_name
```

# 附录 1 PC 专用的环境配置

## 1 Ubuntu 下(SVN&Git)客户端 rabbitcs 安装步骤

#更新源操作
$ sudo apt-get update

#ubuntu18 直接跳过此步骤，ubuntu16 先下载安装依赖库
$ sudo apt-get install python-nautilus python-configobj python-gtk2 python-glade2 python-svn python-dbus python-dulwich subversion meld

#rabbitcs 安装
$ sudo apt install rabbitvcs-cli rabbitvcs-core rabbitvcs-gedit rabbitvcs-nautilus

#重启你的文件管理系统
$ nautilus -q
$ nautilus

#安装完成后效果图


# 2 ubuntu18.04 命令行安装微信步骤

## 2.1 安装步骤

cd ~
mkdir wechat
cd wechat

git clone https://gitee.com/wszqkzqk/deepin-wine-for-ubuntu.git
cd deepin-wine-for-ubuntu
chmod +x *.sh
geidt install_2.8.22.sh

```
#!/bin/bash
mkdir ./deepintemp
cd ./deepintemp
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine/deepin-wine_2.18-22~rc0_all.deb  #之前阿里云的源不能用了
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine/deepin-wine32_2.18-22~rc0_i386.deb
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine/deepin-wine32-preloader_2.18-22~rc0_i386.deb
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine-helper/deepin-wine-helper_1.2deepin8_i386.deb
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine-plugin/deepin-wine-plugin_1.0deepin2_amd64.deb
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine-plugin-virtual/deepin-wine-plugin-virtual_1.0deepin3_all.deb
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine-uninstaller/deepin-wine-uninstaller_0.1deepin2_i386.deb
wget http://packages.deepin.com/deepin/pool/non-free/u/udis86/udis86_1.72-2_i386.deb
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine/deepin-fonts-wine_2.18-22~rc0_all.deb
wget http://packages.deepin.com/deepin/pool/non-free/d/deepin-wine/deepin-libwine_2.18-22~rc0_i386.deb
wget http://packages.deepin.com/deepin/pool/main/libj/libjpeg-turbo/libjpeg62-turbo_1.5.1-2_amd64.deb
wget http://packages.deepin.com/deepin/pool/main/libj/libjpeg-turbo/libjpeg62-turbo_1.5.1-2_i386.deb

echo '准备添加 32 位支持'
sudo dpkg --add-architecture i386
```

```
echo '添加成功，准备刷新 apt 缓存信息...'
sudo apt update
echo '即将开始安装...'
sudo dpkg -i *.deb
echo '安装完成，正在自动安装依赖...'
sudo apt install -fy

rm -vfr ./deepintemp
```

./install_2.8.22.sh

cd ..

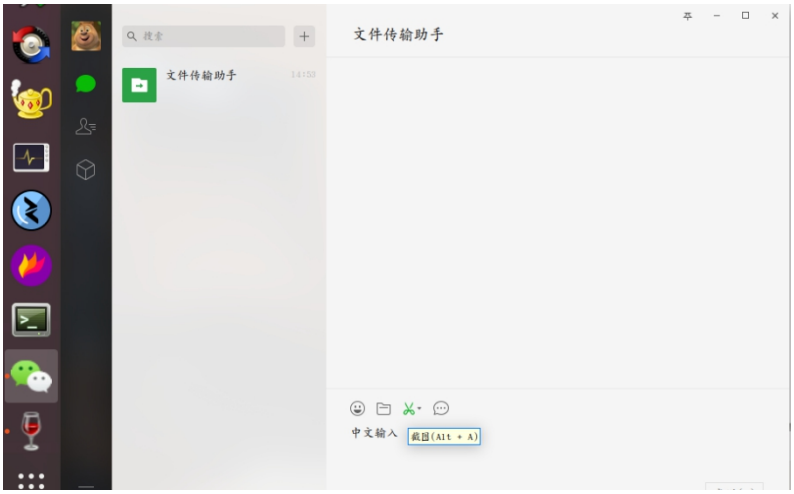
wget https://mirrors.aliyun.com/deepin/pool/non-free/d/deepin.com.wechat/deepin.com.wechat_2.6.8.65deepin0_i386.deb

sudo dpkg -i deepin.com.wechat_2.6.8.65deepin0_i386.deb

## 2.2 输入中文显示乱码问题解决办法

sudo locale-gen en_US.UTF-8
echo "export LC_ALL=en_US.UTF-8" >> ~/.bashrc
source ~/.bashrc

## 2.3 运行效果



# 附录 2 硬件故障排查

1、如果出现设备异常重启，请检查电源供电是否良好，电源的波纹系数是否小。如果电压被突然拉低造成重启，请加装稳压器。另外尽量使用标配电源适配器或者直流稳压开关电源。

2、如果出现路由器 wifi 不稳定，请检查是否存在金属屏蔽，供电电压是否稳定，是否存在相同信道的无线设备干扰，是否存在伺服电机、线圈等电磁干扰设备。

3、如果出现激光雷达数据不能被捕获到，请通过 ping 验证网络的连通性，请检查是否关闭防火墙，端口是否配置正确。

4、如果 CAN 总线没有数据，请首先检查设备是否上电，接着检查 120 欧姆的终端电阻是否接入，波特率是否一致，检查与 CAN 卡连接的 CAN 设备是否正常。如果是 socket can，请终端输入 ifconfig can0/1 查看设备状态；如果是设备 can，查看设备是否已经被其他程序打开。