

Docker interaction

Docker interaction

1. Bash script
 - 1.1, Basic use (Docker)
 - 1.2, Run the script
2. Shared data
 - 2.1. Shared folder
 - 2.2. Shared network
3. Shared hardware
 - 3.1. Shared graphical interface
 - 3.2, ordinary camera
 - 3.3, Depth Camera

The tutorial mainly introduces the data and hardware interaction between the host and Docker.

1. Bash script

Bash (Bourne Again SHell) is a scripting language for automated tasks. Users can combine multiple commands and execute them in sequence!

Docker commands provide optional parameters to start containers, but too many parameters will affect the reading and understanding of the command

1.1, Basic use (Docker)

Basic script

```
#!/bin/bash

docker run -it \
<image_name>:<tag> /bin/bash
```

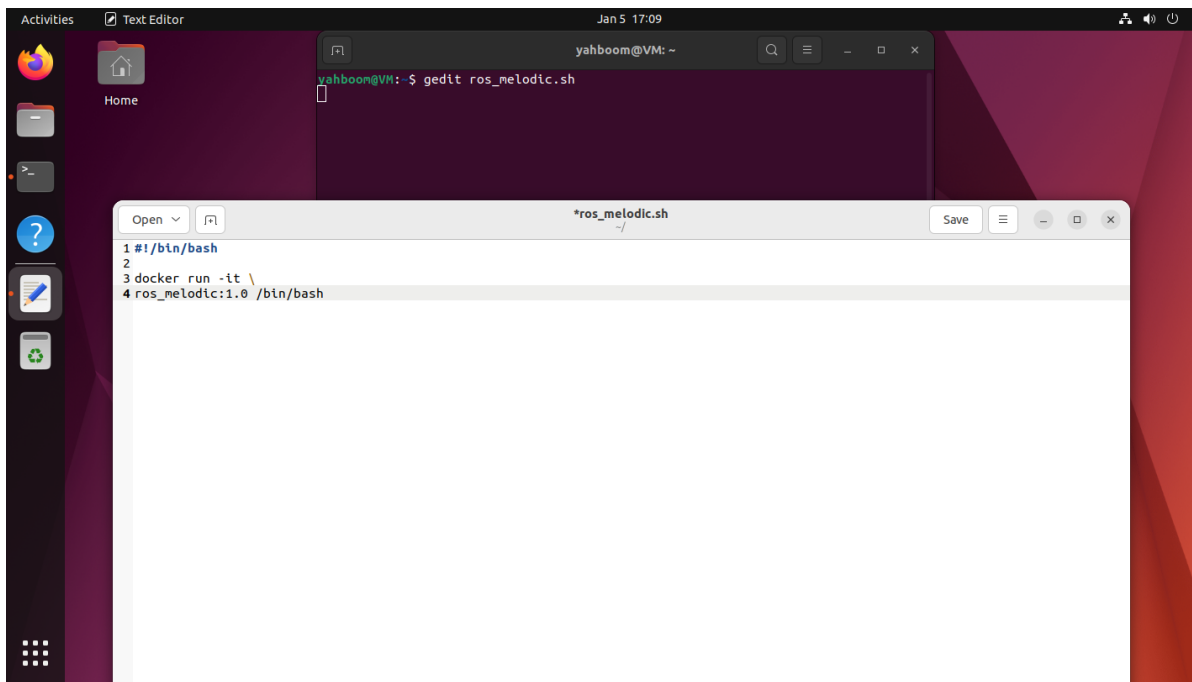
Sample script

Write the example to the `ros_melodic.sh` file:

```
gedit ros_melodic.sh
```

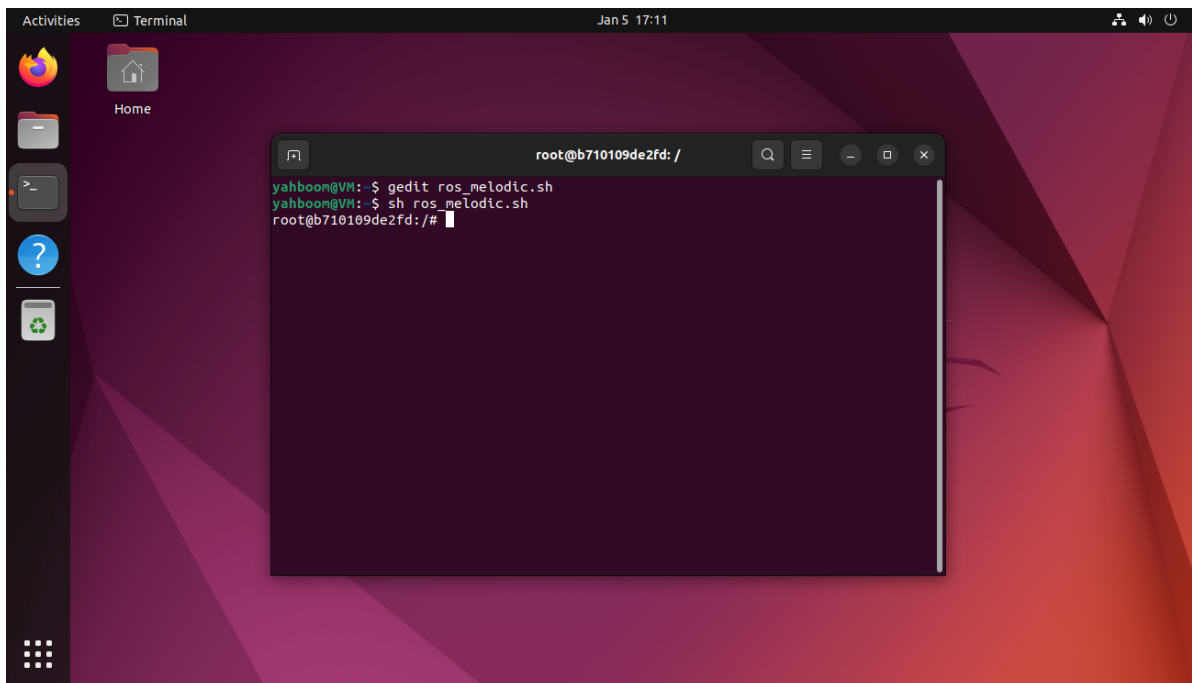
```
#!/bin/bash

docker run -it \
ros_melodic:1.0 /bin/bash
```



1.2, Run the script

```
sh ros_melodic.sh
```



2. Shared data

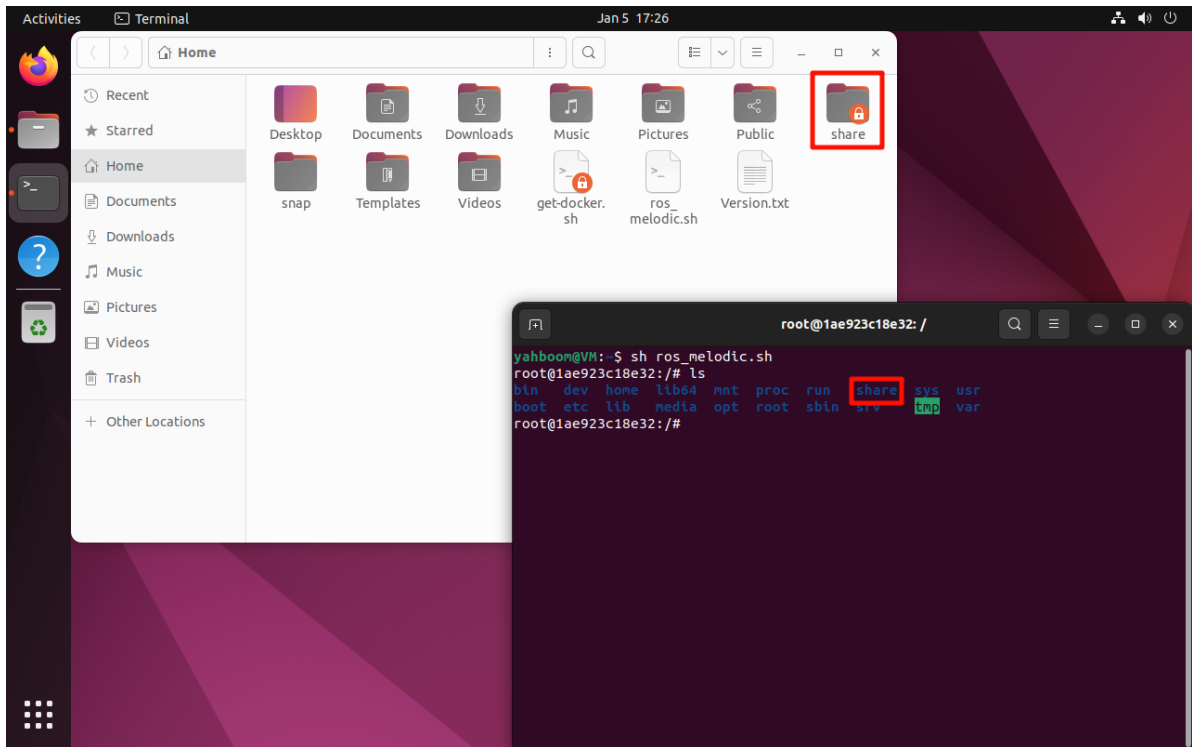
2.1. Shared folder

Share the host's `share` folder to Docker.

Sample script

```
#!/bin/bash
```

```
docker run -it \  
-v /home/yahboom/share:/share \  
ros_melodic:1.0 /bin/bash
```



2.2. Shared network

Share the host's network to Docker.

Sample script

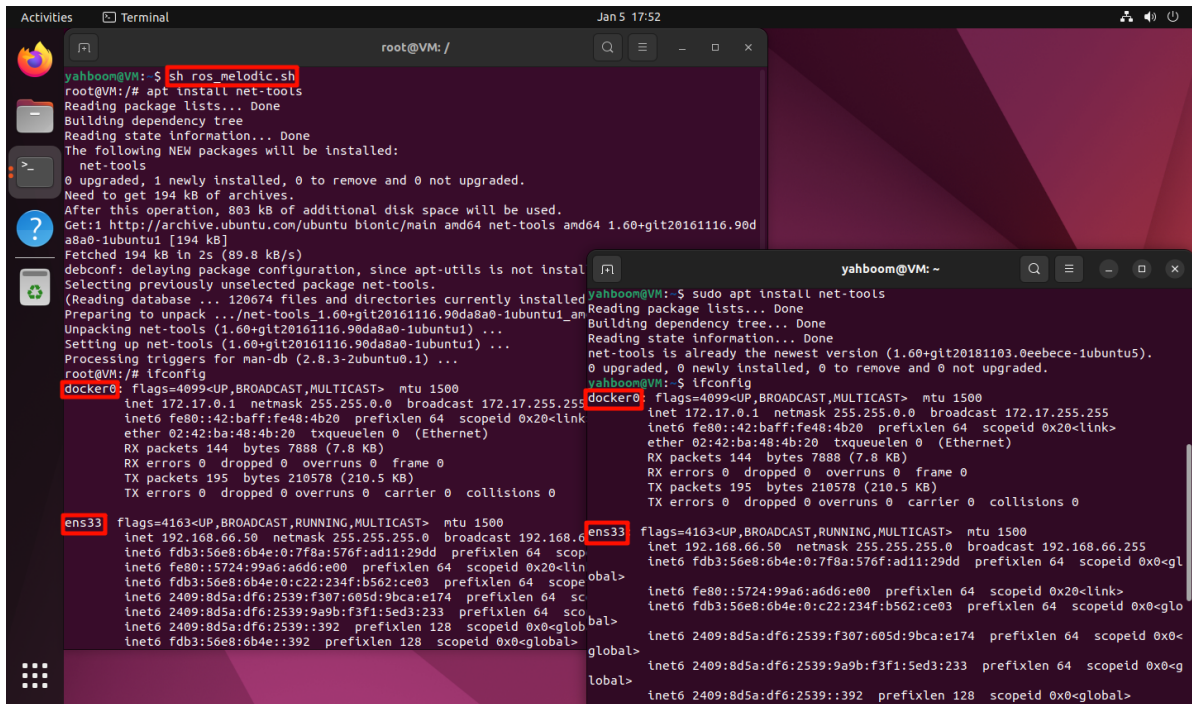
```
#!/bin/bash
```

```
docker run -it \  
--net=host \  
-v /home/yahboom/share:/share \  
ros_melodic:1.0 /bin/bash
```

If you cannot use the `ifconfig` command, install `net-tools`:

```
sudo apt install net-tools -y
```

```
ifconfig
```



```
root@VM: /  
yahboom@VM:~$ sh ros_melodic.sh  
root@VM:~# apt install net-tools  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
net-tools  
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 194 kB of archives.  
After this operation, 803 kB of additional disk space will be used.  
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 net-tools amd64 1.60+git20161116.90da8a0-1ubuntu1 [194 kB]  
Fetched 194 kB in 2s (89.8 kB/s)  
debconf: delaying package configuration, since apt-utils is not installed  
Selecting previously unselected package net-tools.  
(Reading database ... 120674 files and directories currently installed.)  
Preparing to unpack .../net-tools_1.60+git20161116.90da8a0-1ubuntu1_amd64.deb ...  
Unpacking net-tools (1.60+git20161116.90da8a0-1ubuntu1) ...  
Setting up net-tools (1.60+git20161116.90da8a0-1ubuntu1) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
root@VM:~# ifconfig  
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255  
    inet6 fe80::42:baff:fe48:4b20 prefixlen 64 scopeid 0x20<link>  
    ether 02:42:ba:48:4b:20 txqueuelen 0 (Ethernet)  
    RX packets 144 bytes 7888 (7.8 KB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 195 bytes 210578 (210.5 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.66.50 netmask 255.255.255.0 broadcast 192.168.66.255  
    inet6 fdb3:56e8:6b4e:0:7f8a:576f:ad11:29dd prefixlen 64 scopeid 0x20<link>  
    inet6 fe80::5724:99a6:a6d6:e00 prefixlen 64 scopeid 0x20<link>  
    inet6 fdb3:56e8:6b4e:0:c22:234f:b562:ce03 prefixlen 64 scopeid 0x20<link>  
    inet6 2409:8d5a:df6:2539:f307:605d:9bca:e174 prefixlen 64 scopeid 0x20<link>  
    inet6 2409:8d5a:df6:2539:9a9b:f3f1:5ed3:233 prefixlen 64 scopeid 0x20<link>  
    inet6 2409:8d5a:df6:2539:392 prefixlen 128 scopeid 0x0<global>  
    inet6 fdb3:56e8:6b4e::392 prefixlen 128 scopeid 0x0<global>  
  
yahboom@VM:~$ sudo apt install net-tools  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
net-tools is already the newest version (1.60+git20161116.90da8a0-1ubuntu1).  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
yahboom@VM:~$ ifconfig  
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255  
    inet6 fe80::42:baff:fe48:4b20 prefixlen 64 scopeid 0x20<link>  
    ether 02:42:ba:48:4b:20 txqueuelen 0 (Ethernet)  
    RX packets 144 bytes 7888 (7.8 KB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 195 bytes 210578 (210.5 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.66.50 netmask 255.255.255.0 broadcast 192.168.66.255  
    inet6 fdb3:56e8:6b4e:0:7f8a:576f:ad11:29dd prefixlen 64 scopeid 0x20<link>  
    inet6 fe80::5724:99a6:a6d6:e00 prefixlen 64 scopeid 0x20<link>  
    inet6 fdb3:56e8:6b4e:0:c22:234f:b562:ce03 prefixlen 64 scopeid 0x20<link>  
    inet6 2409:8d5a:df6:2539:f307:605d:9bca:e174 prefixlen 64 scopeid 0x20<link>  
    inet6 2409:8d5a:df6:2539:9a9b:f3f1:5ed3:233 prefixlen 64 scopeid 0x20<link>  
    inet6 2409:8d5a:df6:2539:392 prefixlen 128 scopeid 0x0<global>  
    inet6 fdb3:56e8:6b4e::392 prefixlen 128 scopeid 0x0<global>
```

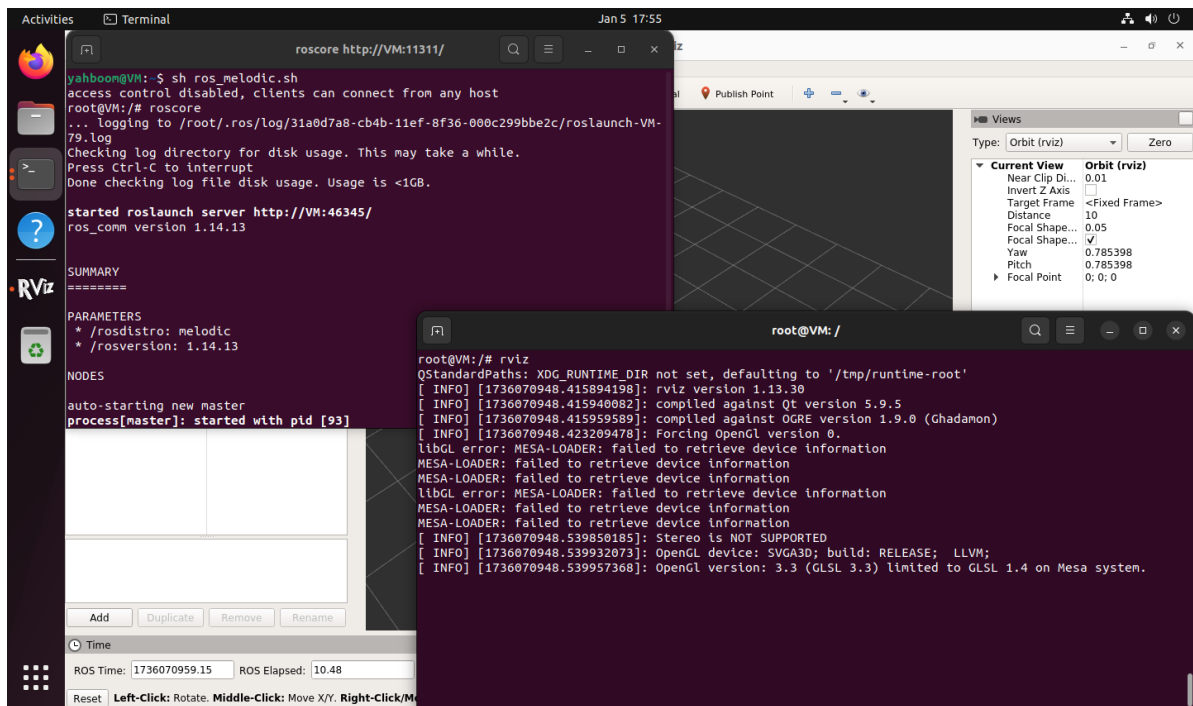
3. Shared hardware

3.1. Shared graphical interface

Docker runs graphical interface programs and needs to output to the local display graphical interface.

Sample script

```
#!/bin/bash  
  
xhost +  
  
docker run -it \  
--net=host \  
-e DISPLAY=$DISPLAY \  
-e "QT_X11_NO_MITSHM=1" \  
-v /tmp/.X11-unix:/tmp/.X11-unix \  
-v /home/yahboom/share:/share \  
ros_melodic:1.0 /bin/bash
```



3.2, ordinary camera

Ordinary camera mapping to Docker: just need to add the `/dev/video*` device number corresponding to the camera

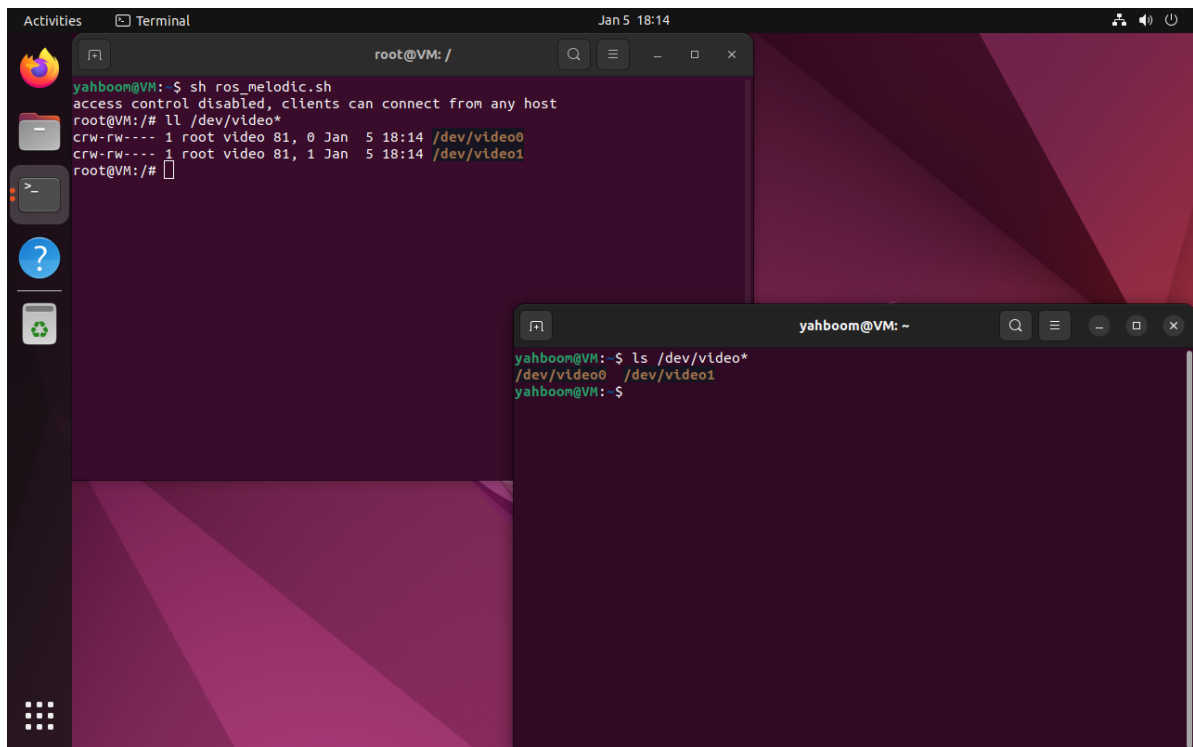
If only one camera is connected, it generally corresponds to `/dev/video0`

Sample script

```
#!/bin/bash

xhost +

docker run -it \
--net=host \
-e DISPLAY=$DISPLAY \
-e "QT_X11_NO_MITSHM=1" \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /home/yahboom/share:/share \
--device=/dev/video0 \
ros_melodic:1.0 /bin/bash
```

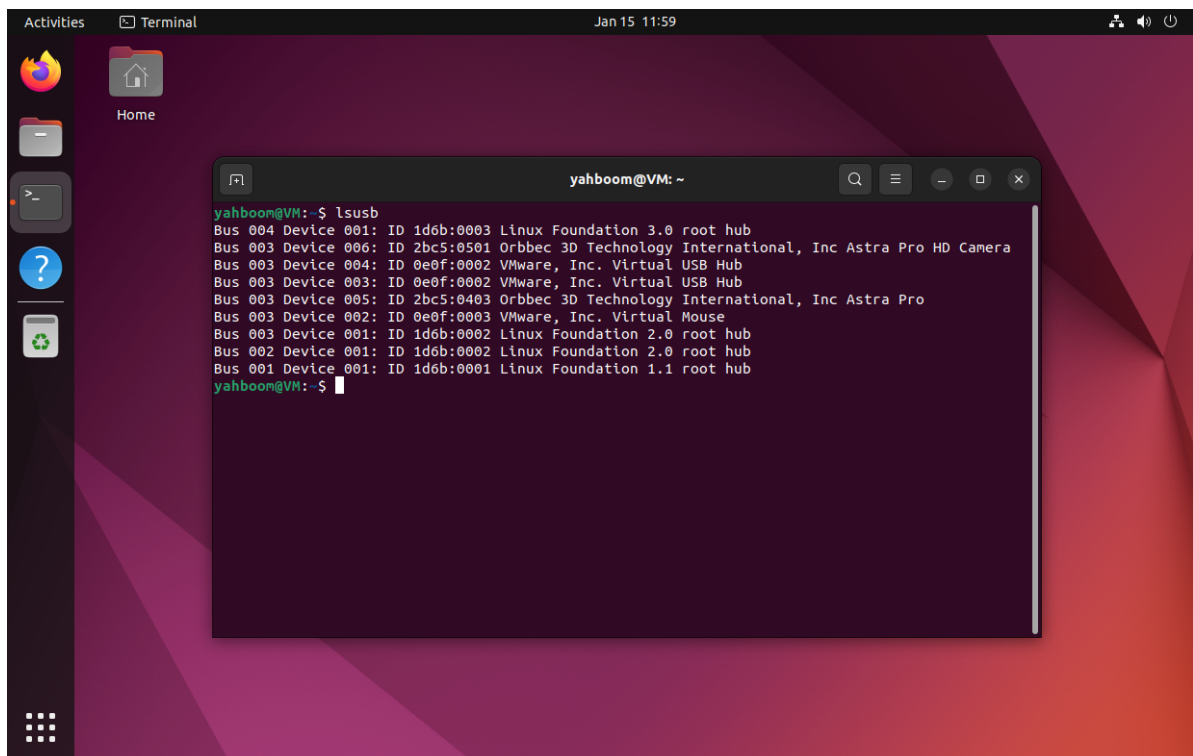


3.3, Depth Camera

Compared with ordinary cameras, depth cameras require additional information to call the camera: Take **Astra Pro** as an example

View USB device information

```
lsusb
```



Information we need to pay attention to: USB bus number, device number on the USB bus, device vendor ID, product ID

Bus 003 Device 005: ID 2bc5:0403 Orbbec 3D Technology International, Inc Astra Pro

Bus 003 Device 006: ID 2bc5:0501 Orbbec 3D Technology International, Inc Astra Pro HD Camera

USB bus number: Bus 003

Device number on the USB bus: Device 005, Device 006

Vendor ID of the device: 2bc5

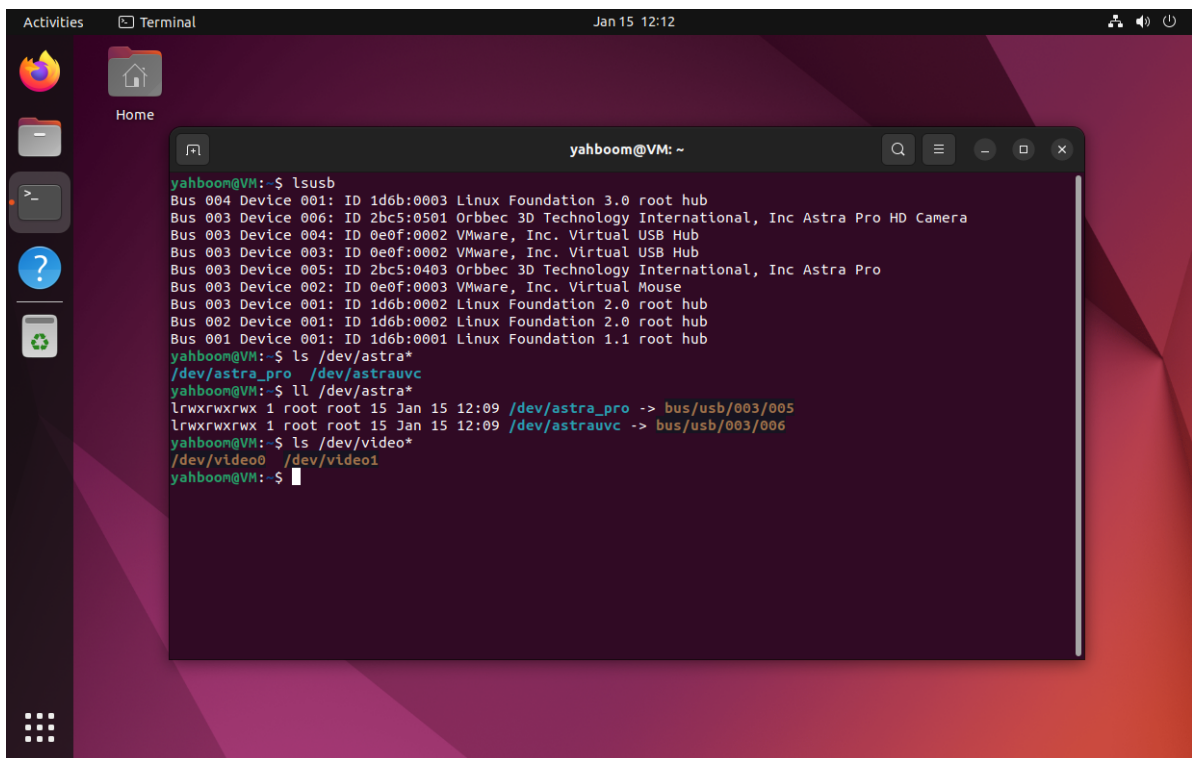
Product ID: 0403, 0501

Device can be located and device ID remapped based on vendor ID and product ID, where USB bus number and device number on USB bus affect the device connected in Docker.

Device ID remap

The binding process and method are not introduced here, and the results are directly displayed. Device ID remap is not part of this tutorial.

Every time you start docker, you need to pay attention to the USB bus number and device number on USB bus corresponding to the depth camera. If the two are inconsistent, the camera will always prompt that the device cannot be connected when it is started in Docker.

A screenshot of a Linux desktop environment with a terminal window open. The terminal shows the output of several commands: 'lsusb' lists USB devices including the Astra Pro HD Camera on bus 003; 'ls /dev/astra*' shows the device paths; 'll /dev/astra*' shows permissions and timestamps; and 'ls /dev/video*' shows video device paths. The terminal window title is 'yahboom@VM: ~'.

```
yahboom@VM:~$ lsusb
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 006: ID 2bc5:0501 Orbbec 3D Technology International, Inc Astra Pro HD Camera
Bus 003 Device 004: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 005: ID 2bc5:0403 Orbbec 3D Technology International, Inc Astra Pro
Bus 003 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
yahboom@VM:~$ ls /dev/astra*
/dev/astra_pro  /dev/astrauvc
yahboom@VM:~$ ll /dev/astra*
lrwxrwxrwx 1 root root 15 Jan 15 12:09 /dev/astra_pro -> bus/usb/003/005
lrwxrwxrwx 1 root root 15 Jan 15 12:09 /dev/astrauvc -> bus/usb/003/006
yahboom@VM:~$ ls /dev/video*
/dev/video0  /dev/video1
yahboom@VM:~$
```

Sample Script

```
#!/bin/bash

xhost +

docker run -it \
--net=host \
-e DISPLAY=$DISPLAY \
-e "QT_X11_NO_MITSHM=1" \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /home/yahboom/share:/share \
```

```
-v /dev/bus/usb/003/005:/dev/bus/usb/003/005 \  
-v /dev/bus/usb/003/006:/dev/bus/usb/003/006 \  
--device=/dev/astra_pro \  
--device=/dev/astrauvc \  
--device=/dev/video0 \  
--device=/dev/video1 \  
ros_melodic:1.0 /bin/bash
```

