4. docker hardware interaction and data processing

The operating environment and hardware and software reference configuration are as follows:

- Reference model: ROSMASTER X3
- Robot hardware configuration: Arm series main control, Silan A1 LiDAR, AstraPro Plus depth camera.
- Robot system: Ubuntu (version not required) + docker (version 20.10.21 and above)
- PC virtual machine: Ubuntu (20.04) + ROS2 (Foxy)
- Usage scenario: use on a relatively clean 2D plane

4.1. Hardware mounting (port binding)

1. Establish udev rules in the host (/etc/udev/rules.d/), see [VI. Linux operating system ---- 6. Binding device ID 】 chapter
2. Then when you open the container, mount the device with the rule set to the docker container with --device=/dev/myserial --device=/dev/rplidar and so on.

```
docker run -it --device=/dev/myserial --device=/dev/rplidar ubuntu:latest
/bin/bash
```

3, docker container will be able to discover the device

```
jetson@ubuntu:~$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED              SIZE
ubuntu                        1.0       78ca7be949b6   About an hour ago    69.2MB
pengan88/ubuntu               1.0       78ca7be949b6   About an hour ago    69.2MB
yahboomtechnology/ros-foxy    3.4.0     49581aa78b6b   6 hours ago          24.3GB
yahboomtechnology/ros-foxy    3.3.9     cefb5ac2ca02   4 days ago           20.5GB
yahboomtechnology/ros-foxy    3.3.8     49996806c64a   4 days ago           20.5GB
yahboomtechnology/ros-foxy    3.3.7     8989b8860d17   5 days ago           17.1GB
yahboomtechnology/ros-foxy    3.3.6     326531363d6e   5 days ago           16.1GB
mysql                         latest    5371f8c3b63e   6 days ago           592MB
ubuntu                        latest    bab8ce5c00ca   6 weeks ago          69.2MB
hello-world                   latest    46331d942d63   13 months ago        9.14kB
```

```
jetson@ubuntu:~$ ll /dev | grep ttyUSB*
lrwxrwxrwx   1 root   root           7 Apr 23 18:07 myserial -> ttyUSB0
lrwxrwxrwx   1 root   root           7 Apr 23 18:07 rplidar -> ttyUSB1
crwxrwxrwx   1 root   dialout 188,   0 Apr 23 18:07 ttyUSB0
crwxrwxrwx   1 root   dialout 188,   1 Apr 23 18:07 ttyUSB1
jetson@ubuntu:~$ docker run -it   --device=/dev/myserial --device=/dev/rplidar
ubuntu:latest  /bin/bash
root@03522257ba30:/# ls /dev   # docker中已经有myserial和rplidar
# myserial and rplidar already in docker
console  fd  full  mqueue  myserial  null  ptmx  pts  random  rplidar  shm
stderr  stdin  stdout  tty  urandom  zero
```

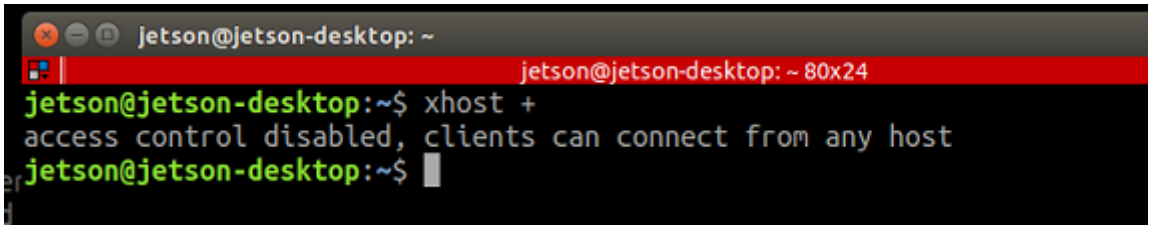
4.2. GUI display in docker

1. Install it in the host:

```
sudo apt-get install tigervnc-standalone-server tigervnc-viewer
sudo apt-get install x11-xserver-utils
```

2. Execute: xhost + in the host computer.

After displaying the following figure normally, execute 3 steps:



3. Execute the command to enter the container in the host:

```
docker run -it \                                    # 交互式运行docker镜像
--env="DISPLAY" \                                   # 开启显示GUI界面
--env="QT_X11_NO_MITSHM=1" \                        # 采用X11的端口1进行显示
-v /tmp/.X11-unix:/tmp/.X11-unix \                  # 映射显示服务节点目录
yahboomtechnology/ros-foxy:3.3.9                    # 要启动的镜像名称
/bin/bash                                           # 在容器内执行/bin/bash命令

docker run -it \ # Interactively run docker image
--env="DISPLAY" \ # Enable to display GUI interface.
--env="QT_X11_NO_MITSHM=1" \ # Use port 1 of X11 for the display
-v /tmp/.X11-unix:/tmp/.X11-unix \ # Mapping display service node directory
yahboomtechnology/ros-foxy:3.3.9 # Name of the image to be launched
/bin/bash # Execute the /bin/bash command inside the container
```

4. Testing

```
在容器中执行：rviz2
Execute in container: rviz2
```

4.3 Transferring files between docker containers and the host computer

4.3.1, using cp naming

4.3.1.1, copying files from containers to hosts

```
# 命令
docker cp 容器id:容器内路径 目的主机路径

# 测试
# 容器内执行，创建一个文件测试
# The command
docker cp container id:in-container path destination host path

# test
# Execute inside the container, create a file for testing

jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE           COMMAND       CREATED       STATUS
  PORTS     NAMES
c54bf9efae47   ubuntu:latest   "/bin/bash"   2 hours ago   Up 9 minutes
            funny_hugle
3b9c01839579   hello-world     "/hello"      3 hours ago   Exited (0) 3 hours ago
            jovial_brown
jetson@ubuntu:~$ docker attach c5
root@c54bf9efae47:/# ls
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  root  run  sbin  srv  sys
tmp  usr  var
root@c54bf9efae47:/# cd
root@c54bf9efae47:~# ls
root@c54bf9efae47:~# touch test.txt
root@c54bf9efae47:~# ls
test.txt
root@c54bf9efae47:~# pwd
/root
root@c54bf9efae47:/# read escape sequence    #按ctrl+P+Q 容器不停止退出
#Press ctrl+P+Q the container does not stop to exit
jetson@ubuntu:~$ docker cp c54bf9efae47:/root/test.txt ~/
jetson@ubuntu:~$ ls      # test.txt文件已经拷贝进来了
  # The test.txt file has been copied in.
Desktop  Documents  Downloads  fishros  Music  openvino  Pictures  Public
rootOnNVMe  run_docker.sh  sensors  snap  temp  Templates  test.txt  Videos
```

4.3.1.2 Copying files from the host to the container

```
# 命令
docker cp 宿主机文件路径 容器id:容器内路径

# Command
docker cp host file path container id:path within container
```

```
#测试#Testing
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE           COMMAND       CREATED       STATUS
  PORTS     NAMES
c54bf9efae47   ubuntu:latest   "/bin/bash"   2 hours ago   Up 5 minutes
            funny_hugle
3b9c01839579   hello-world     "/hello"      3 hours ago   Exited (0) 3 hours ago
            jovial_brown
jetson@ubuntu:~$ ls
Desktop  Documents  Downloads  fishros  Music  openvino  Pictures  Public
rootOnNVMe  run_docker.sh  sensors  snap  temp  Templates  test.txt  Videos
jetson@ubuntu:~$ touch 11.txt
jetson@ubuntu:~$ ls
11.txt  Desktop  Documents  Downloads  fishros  Music  openvino  Pictures  Public
rootOnNVMe  run_docker.sh  sensors  snap  temp  Templates  test.txt  Videos
jetson@ubuntu:~$ docker cp 11.txt c54bf9efae47:/root/
jetson@ubuntu:~$ docker attach c5
root@c54bf9efae47:/# ls
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  root  run  sbin  srv  sys
tmp  usr  var
root@c54bf9efae47:/# cd /root/
root@c54bf9efae47:~# ls     # 11.txt文件已经拷贝进来了
# 11.txt file has been copied in
11.txt  test.txt
```

4.3.2 Using Data Volumes

4.3.2.1, Data Volumes Overview

The application and run the environment packaged to form a container to run, the run can be accompanied by the container, but our requirements for the data, is that we want to be able to persist! As if, you install a mysql, as a result, you delete the container, it is equivalent to deleting the library run, which is certainly not right! So we hope that it is possible to share data between containers, docker container generated data, if not through docker commit to generate a new image, so that the data as part of the image saved, then when the container is deleted, the data is naturally gone! This won't work!

To be able to save the data in docker we can use volumes! so that the data is mounted locally! This way the data is not lost when the container is deleted!

features:

1, data volume can be shared between containers or reuse data

2, changes in the volume can take effect directly

3, changes in the data volume will not be included in the update of the image

4, the life cycle of the data volume continues until no container uses it

4.3.2.2. Data volume utilization

```
# 命令
docker run -it -v 宿主机绝对路径目录:容器内目录 镜像名

# Command
docker run -it -v host absolute path directory:in-container directory image name

# 测试  # Testing
docker run -it -v /home/jetson/temp:/root/temp yahboomtechnology/ros-foxy:3.4.0
/bin/bash

宿主机中的/home/jetson/temp目录和容器内的/root/temp目录就可以共享数据了
The /home/jetson/temp directory in the host and the /root/temp directory in the
container can then share data
```