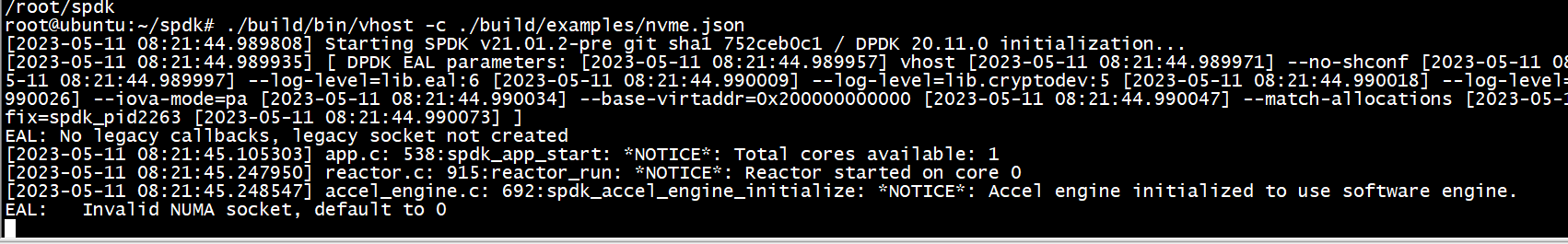
# vhost

## 准备

1. scripts/setup.sh
2. ./scripts/gen\_nvme.sh --json-with-subsystems > ./build/examples/nvme.json

## 运行vhost

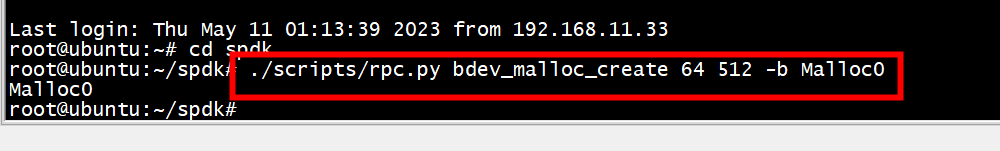
./build/bin/vhost -c ./build/examples/nvme.json

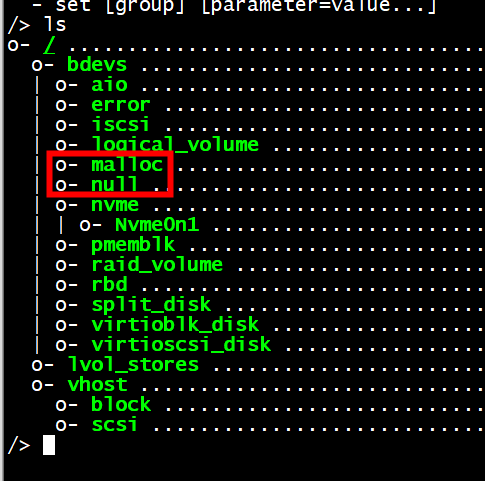


## 创建ramdisk bdv

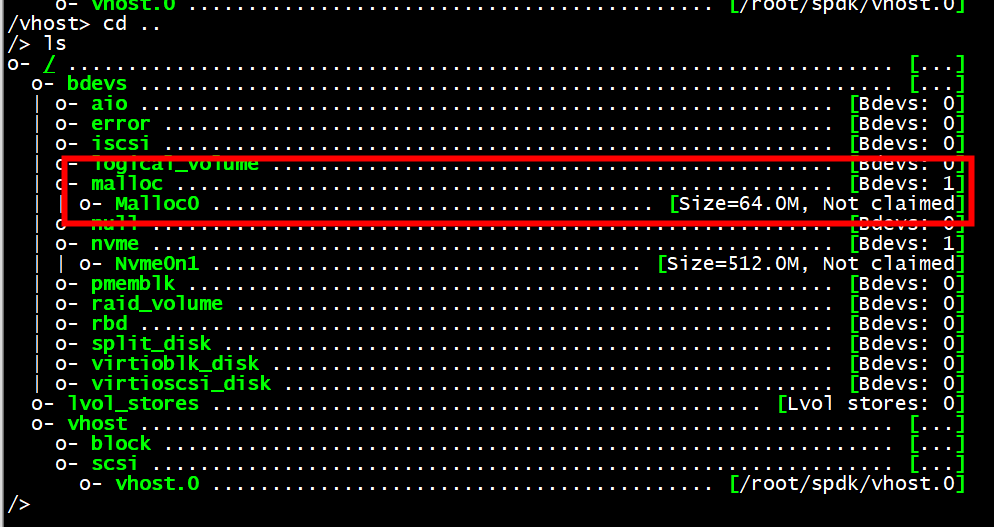
创建基于ramdisk的spdk bdev Malloc0 （vhost-scsi）:

./scripts/rpc.py bdev\_malloc\_create 64 512 -b Malloc0





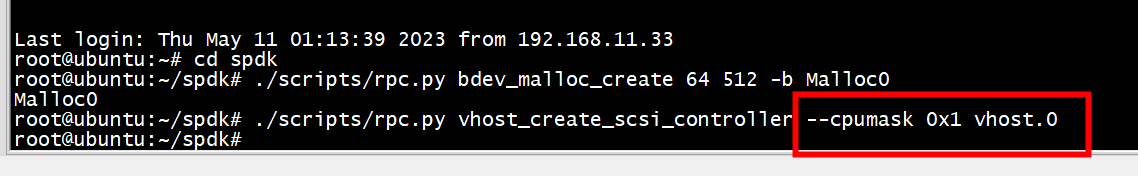
创建前



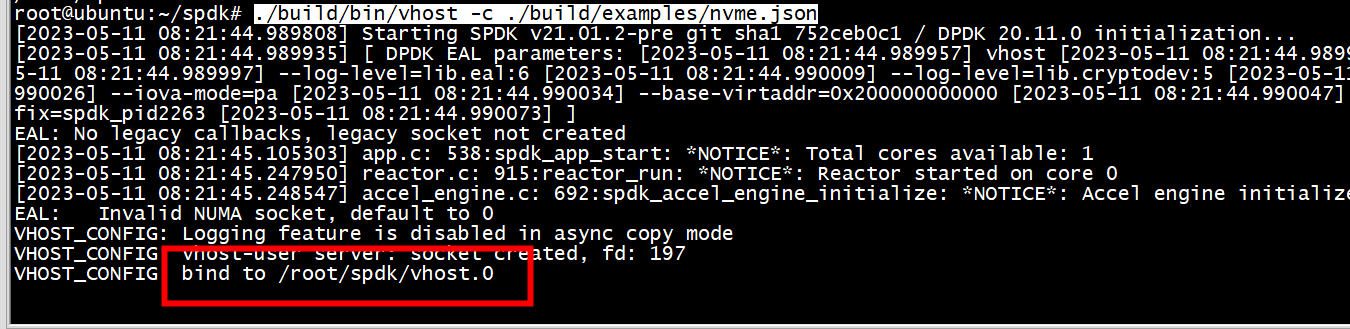
创建后

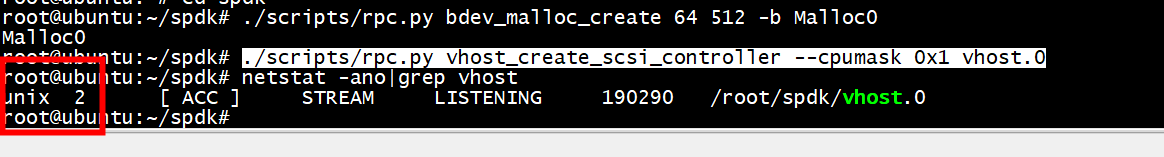
## 添加vhost controller vhost.0

./scripts/rpc.py vhost\_create\_scsi\_controller --cpumask 0x1 vhost.0

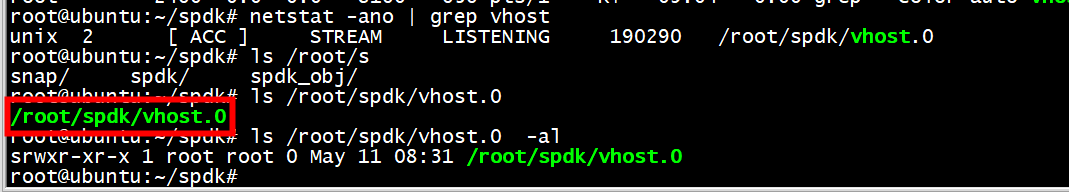


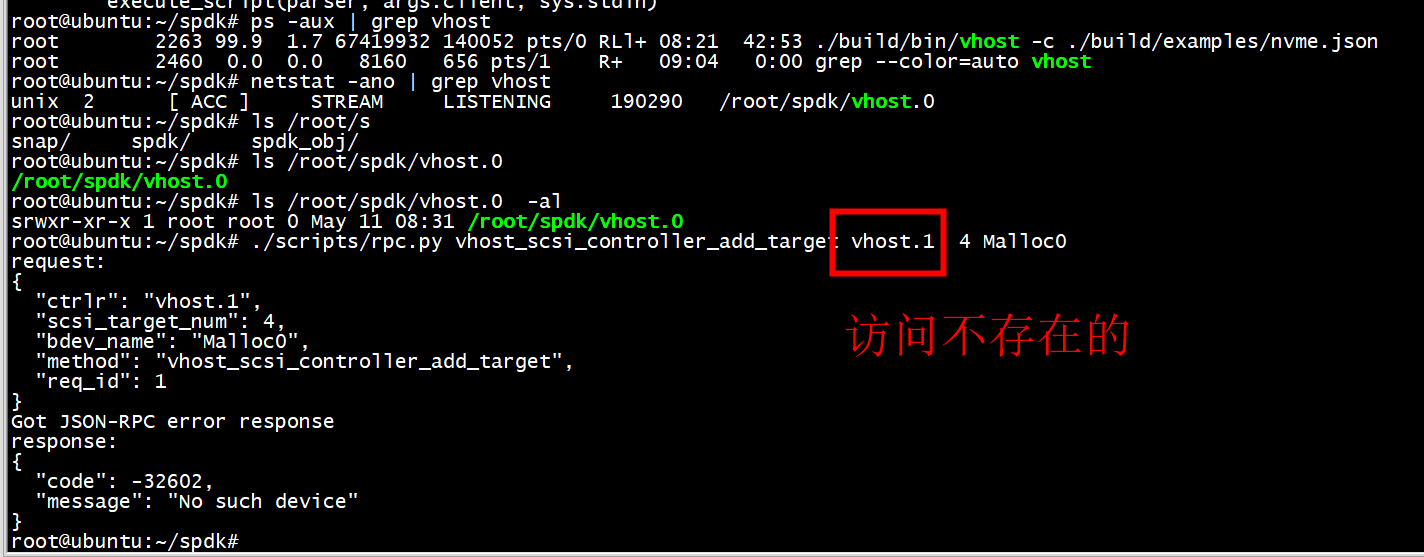
* 查看vhost进程



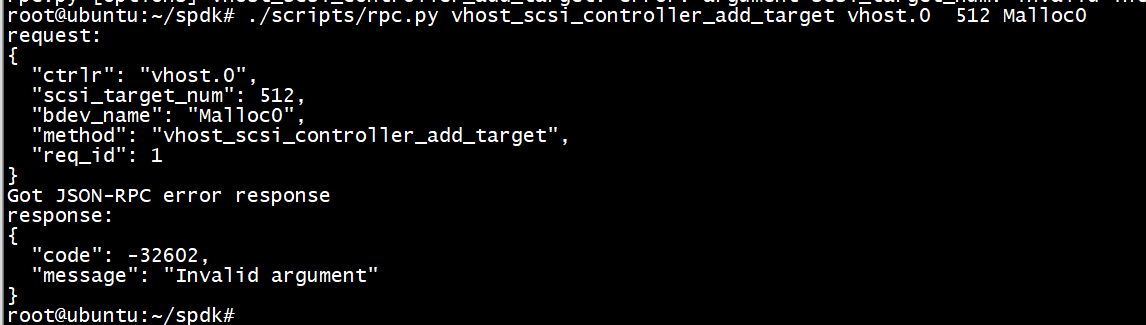


**Unix socket**





## controller vhost.0添加target





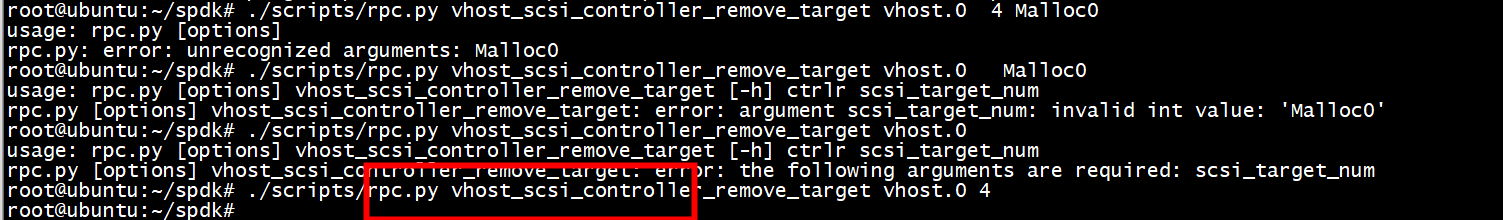
把大小改小点

**./scripts/rpc.py vhost\_scsi\_controller\_add\_target vhost.0 4 Malloc0**

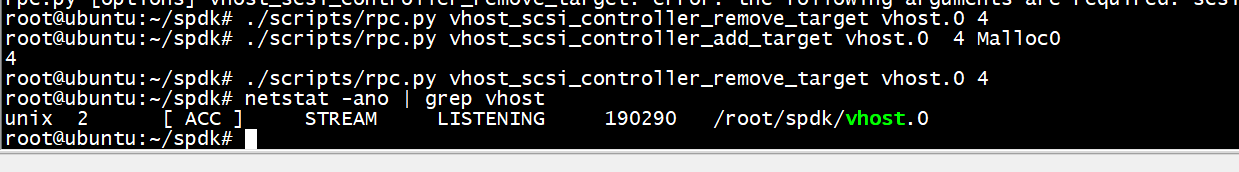


* 删除

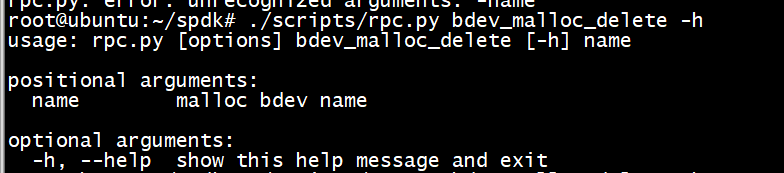
./scripts/rpc.py vhost\_scsi\_controller\_remove\_target vhost.0 4



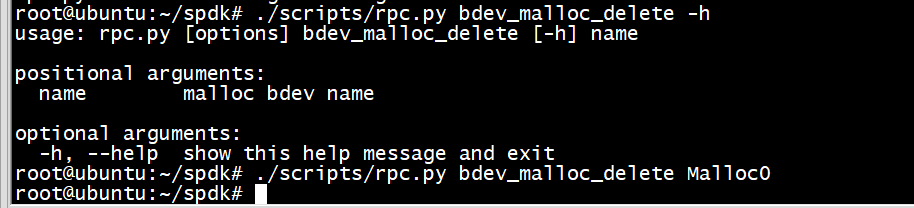
溢出后，unix socket server还存在



## 删除 ramdisck bdv

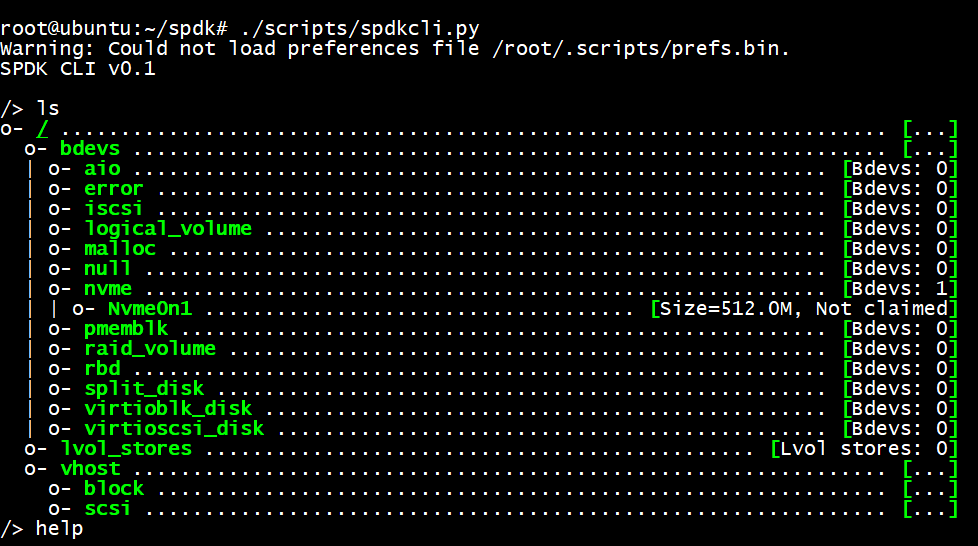


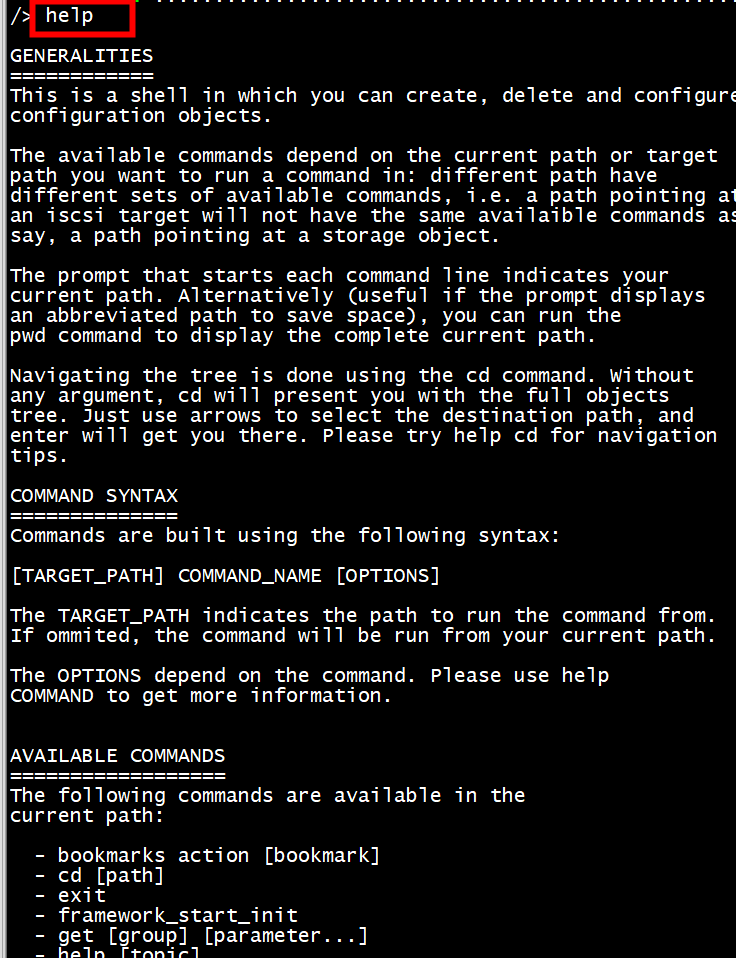
./scripts/rpc.py bdev\_malloc\_delete Malloc0





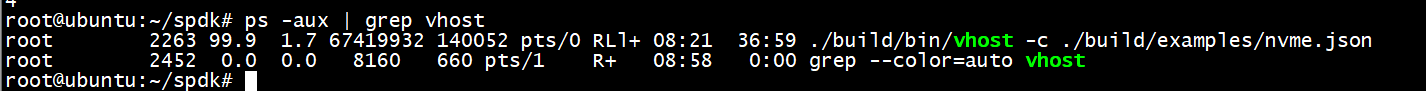
## ./scripts/spdkcli.py



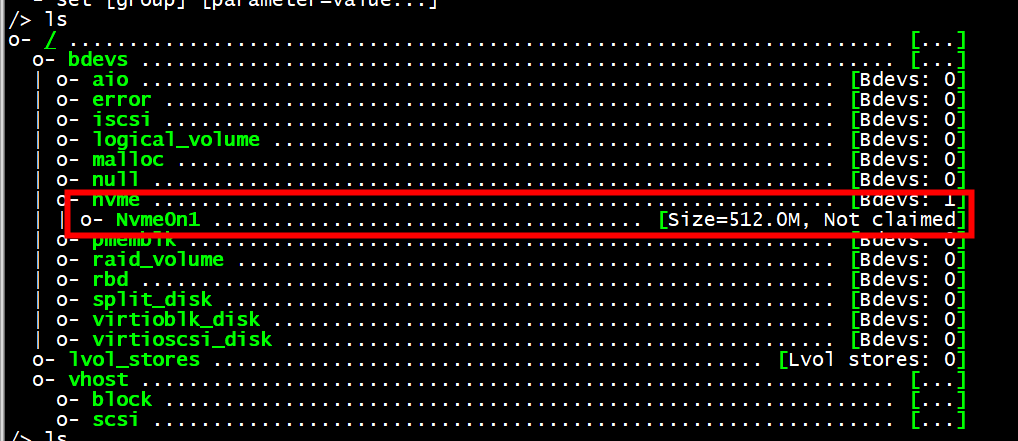


## 问题排查

* 判断vhost 进程是否存在



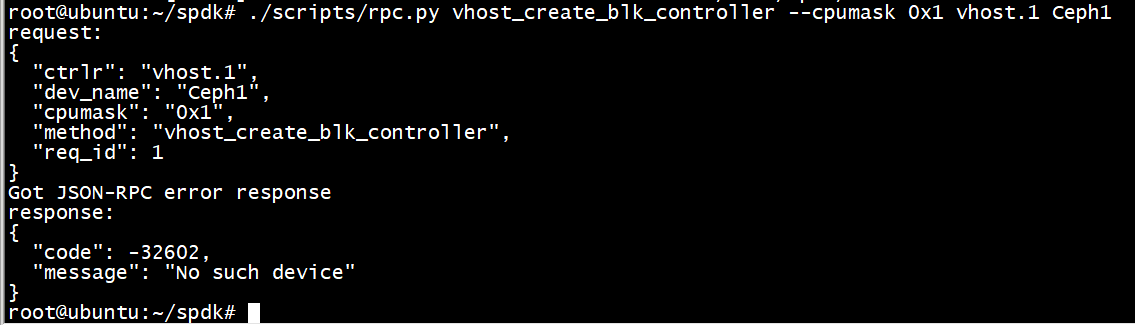
* bdev 是否创建

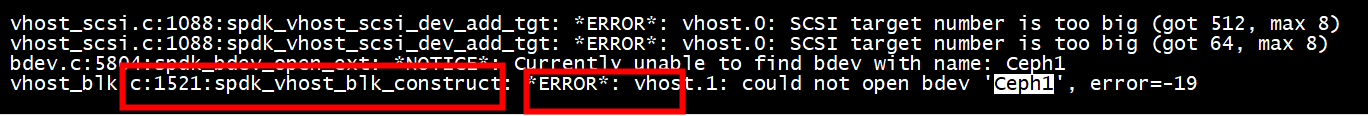


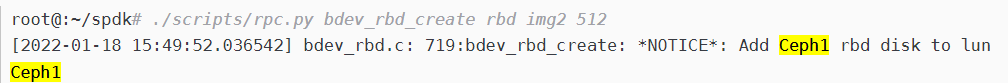
### spdk\_vhost\_blk\_construct: \*ERROR

没有创建Ceph1设备

./scripts/rpc.py vhost\_create\_blk\_controller --cpumask 0x1 vhost.1 Ceph1







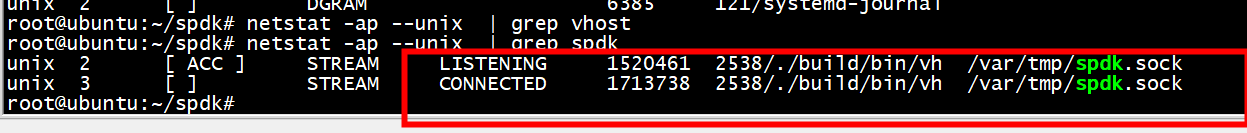
# Nvme Vhost test2

## Unix soscket

使用SPDK的RPC需要首先在SPDK的应用启动时使用“-r”参数指定RPC server的监听地址，默认地址为“/var/tmp/spdk.sock”

1）启动方式 ./build/bin/vhost -S /var/tmp -s 1024 -m 0x3

2） netstat -ap --unix | grep spdk



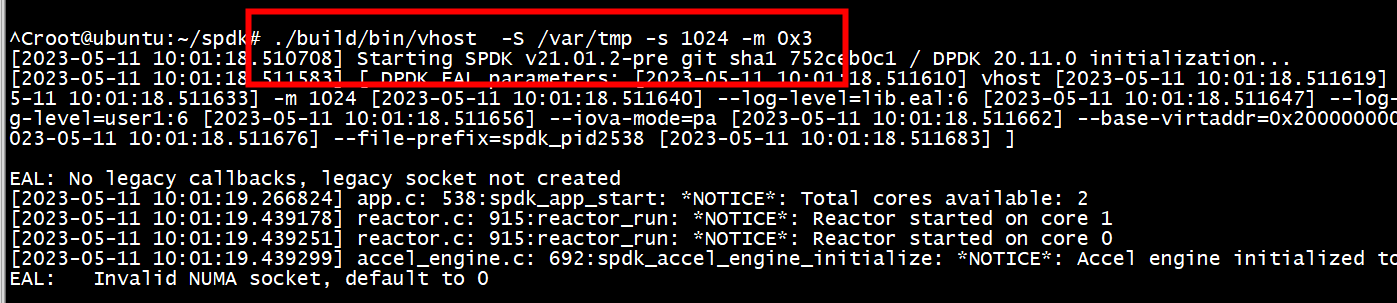
使用SPDK提供的client端命令行工具“scripts/rpc.py”，可以方便的向SPDK server端发起RPC调用。它的使用方法如下：

rpc.py [-h] [-s SERVER\_ADDR] [-p PORT] [-v]<command> [parameters list]

其中，“-s”“-p”参数指定SPDK RPC server端的监听地址和端口；“command”“parameters list” 指定具体的RPC命令和对应的参数。

## 运行vhost

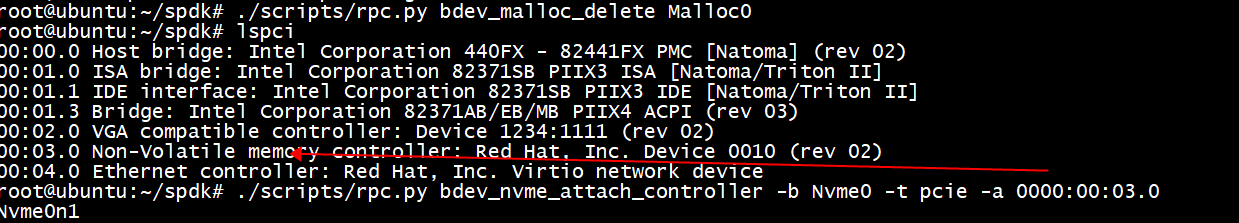
./build/bin/vhost -S /var/tmp -s 1024 -m 0x3



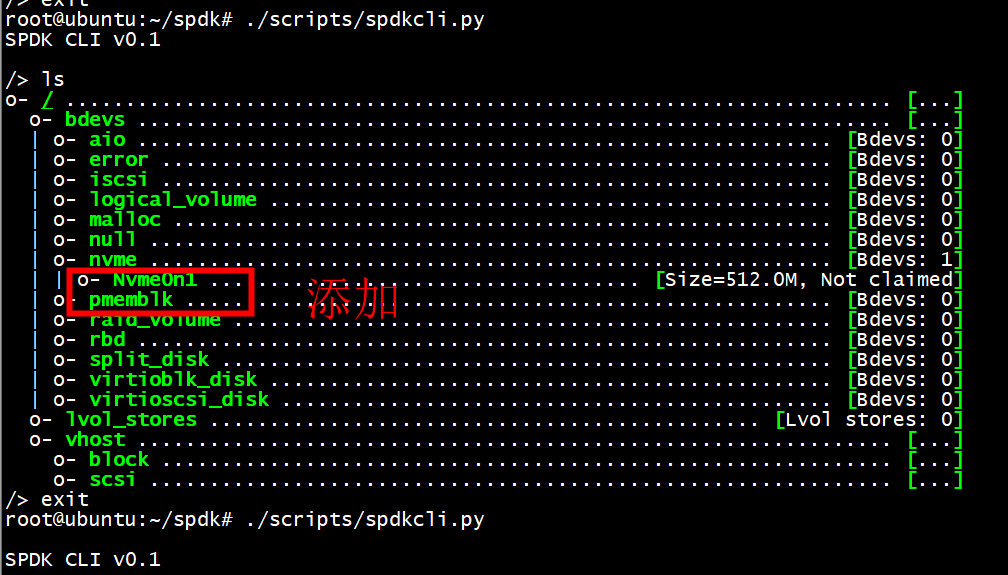
## 添加nvme

### Attch controller

1. 查看pci nvme设备lscpi
2. ./scripts/rpc.py bdev\_nvme\_attach\_controller -b Nvme0 -t pcie -a 0000:00:03.0



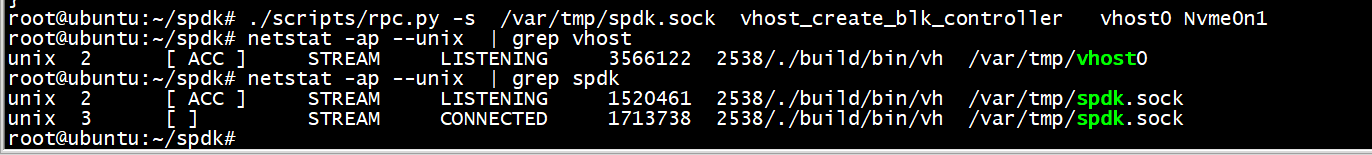
查看 Nvme0n1

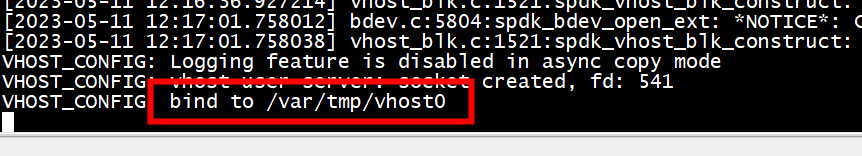


### 添加 controller

./scripts/rpc.py -s /var/tmp/spdk.sock vhost\_create\_blk\_controller vhost0 Nvme0n1

名字必须是Nvme0n1，否则会报错

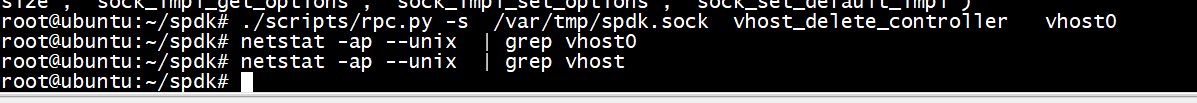




## 删除nvme

### 删除controller

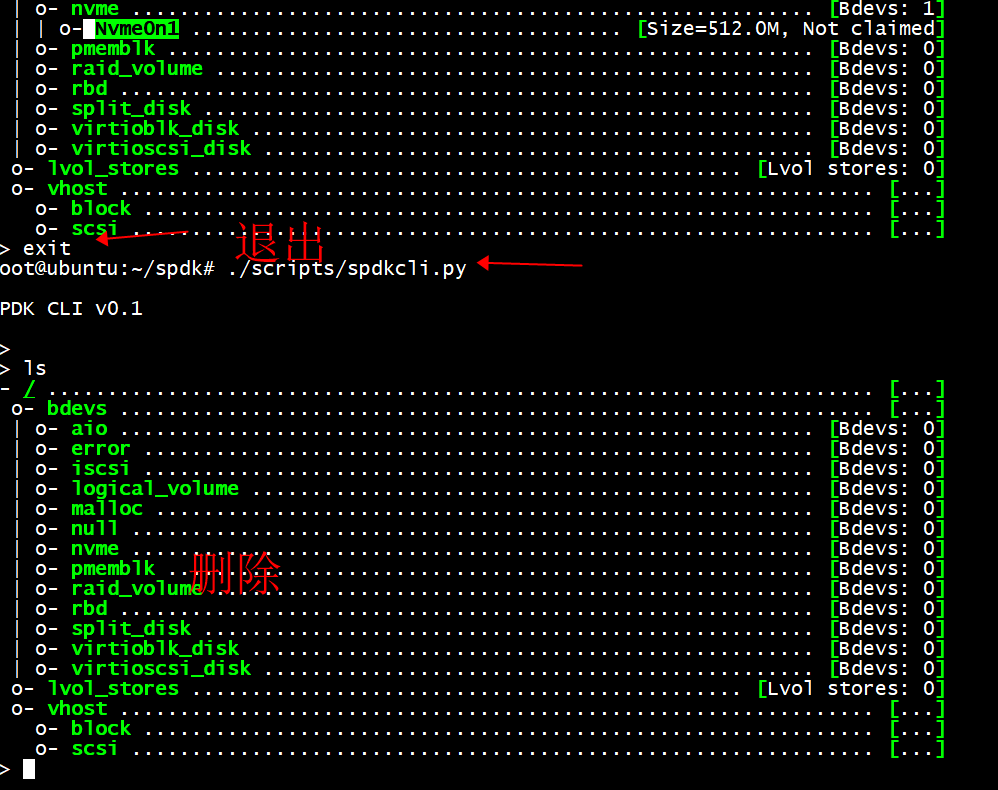
./scripts/rpc.py -s /var/tmp/spdk.sock vhost\_delete\_controller vhost0



### detach controller

./scripts/rpc.py bdev\_nvme\_detach\_controller Nvme0



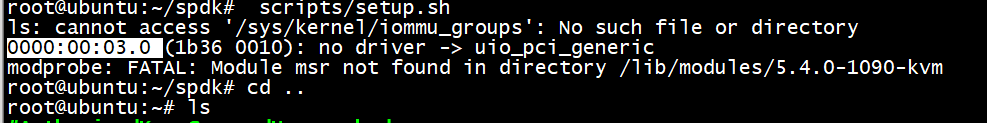


# Nvme tcp

## Server

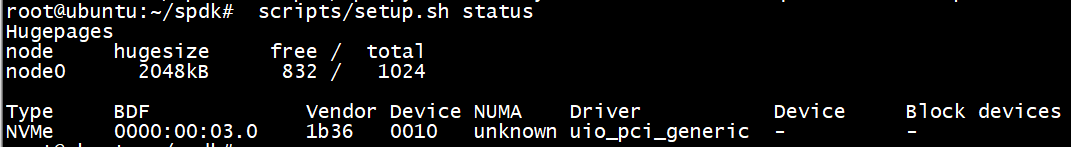
1. # 运行脚本转让设备控制权给SPDK

$ sudo scripts/setup.sh



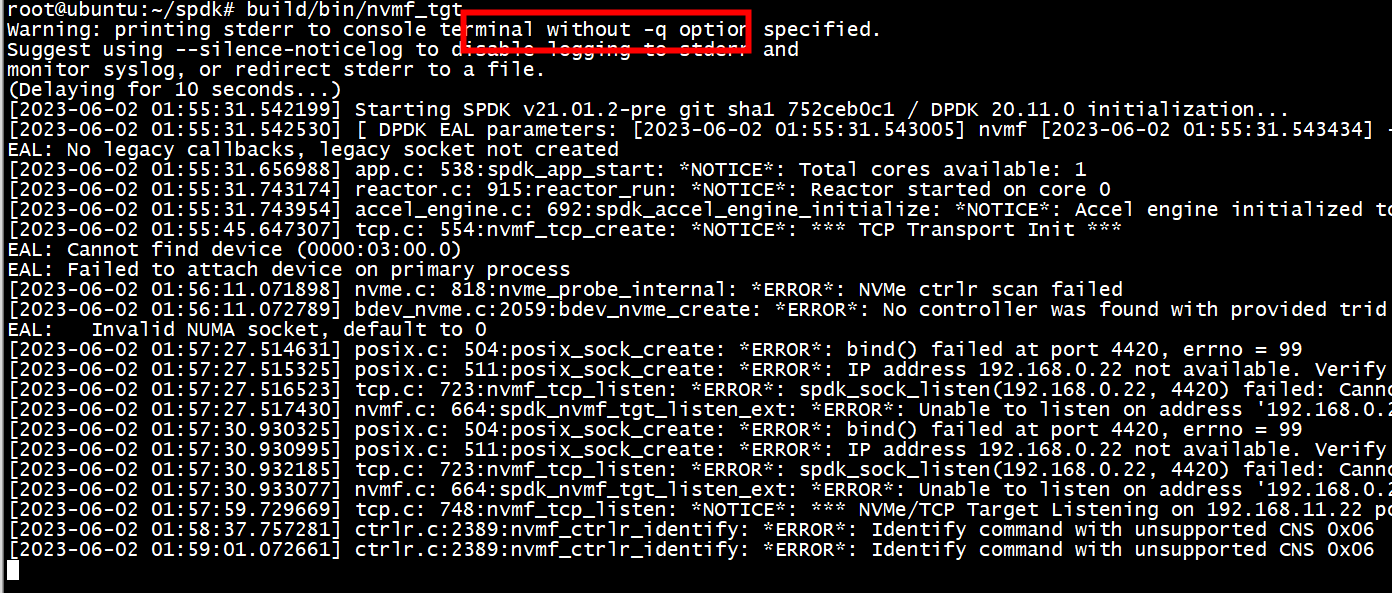
1. # 查看设备状态

$ sudo scripts/setup.sh status



1. 启动 nvmf-tgf

spdk/build/bin/nvmf\_tgt &



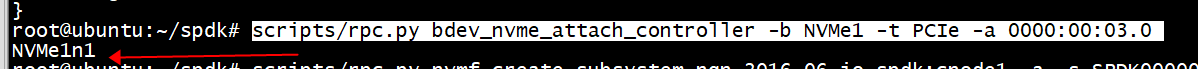
1. 创建TCP传输

root@ubuntu:~/spdk# scripts/rpc.py nvmf\_create\_transport -t TCP -u 16384 -m 8 -c 8192



1. 创建 NVMe 块设备

scripts/rpc.py bdev\_nvme\_attach\_controller -b NVMe1 -t PCIe -a 0000:00:03.0



1. 将NVMe块设备分配给子系统
2. # 创建子系统

$ root@ubuntu:~/spdk# scripts/rpc.py nvmf\_create\_subsystem nqn.2016-06.io.spdk:cnode1 -a -s SPDK00000000000001 -d SPDK\_Controller1



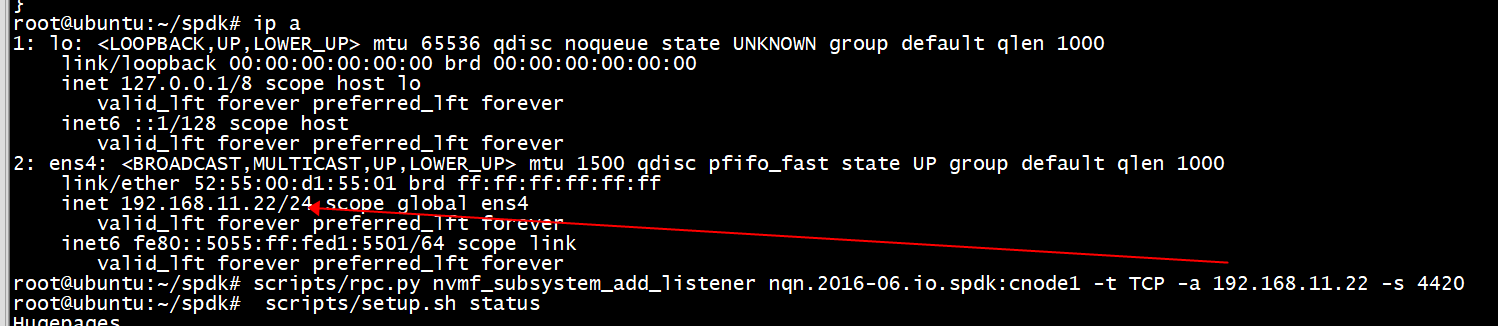
1. # 将NVMe块设备分配给刚刚创建的子系统

root@ubuntu:~/spdk# scripts/rpc.py nvmf\_subsystem\_add\_ns nqn.2016-06.io.spdk:cnode1 NVMe1n1



1. # 为子系统添加TCP监听器

$ scripts/rpc.py nvmf\_subsystem\_add\_listener nqn.2016-06.io.spdk:cnode1 -t TCP -a 192.168.11.22 -s 4420



## Client

1. 安装 nvme-cli

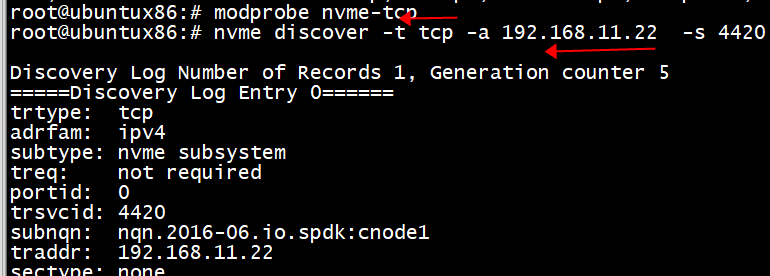
apt install nvme-cli

1. 加载驱动程序

modprobe nvme-tcp

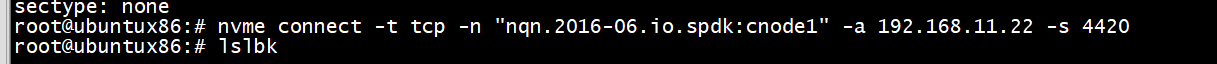
1. 发现 Target

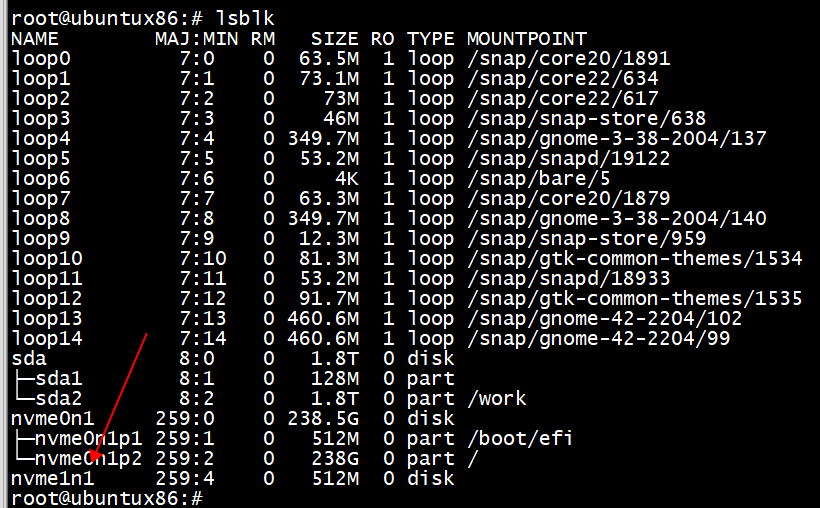
nvme discover -t tcp -a 192.168.11.22 -s 4420



1. 连接Target

nvme connect -t tcp -n "nqn.2016-06.io.spdk:cnode1" -a 192.168.11.22 -s 4420

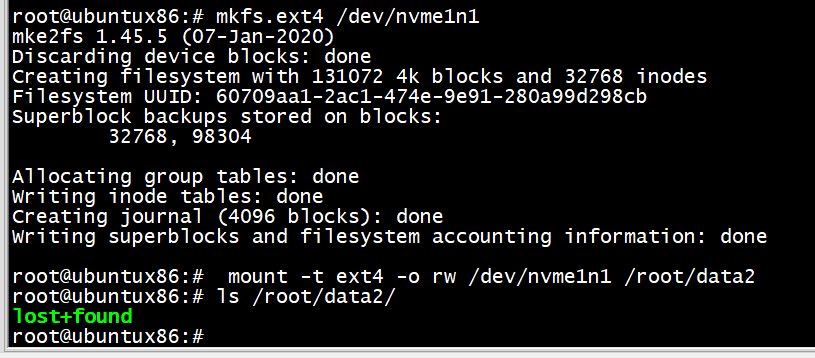




此时可将块设备nvme0n1p1进行挂载访问

1. # 将/dev/nvme0n1挂载到/root/data2/下

$ mount -t ext4 -o rw /dev/nvme1n1 /root/data2



1. 取消连接Target

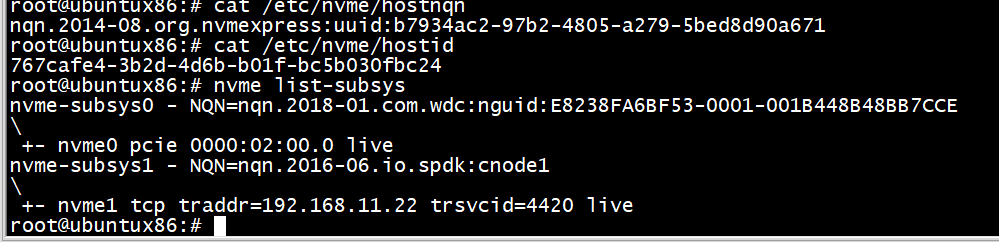
$ nvme disconnect -n "nqn.2016-06.io.spdk:cnode1"

# 或者

$ nvme disconnect-all

### 命令

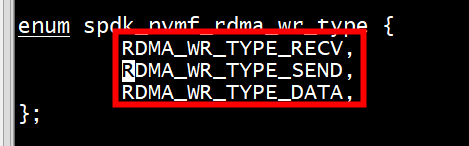
|  |  |  |
| --- | --- | --- |
|  | cat /etc/nvme/hostnqn |  |
|  | cat /etc/nvme/hostid |  |
|  | nvme list-subsys |  |



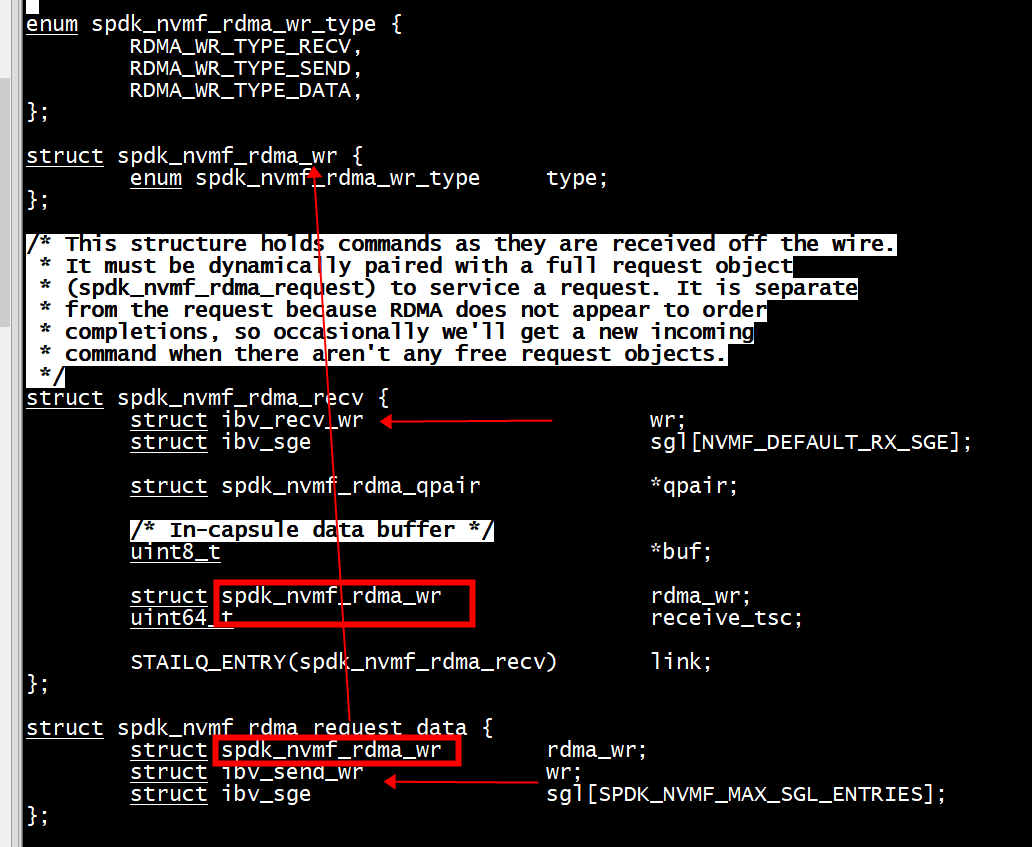
# Spdk rdma

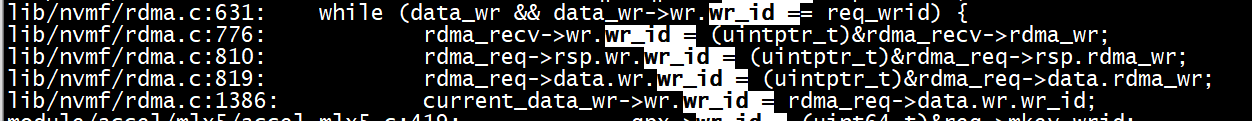
# Spdk nvmf

## struct spdk\_nvmf\_rdma\_wr



struct spdk\_nvmf\_rdma\_wr和struct ibv\_recv\_wr、struct ibv\_send\_wr 的关系





# 参考

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | [spdk(三)----qemu使用spdk](https://www.cnblogs.com/powerrailgun/p/12394086.html) |  |
|  | SPDK NVMe-oF多路径结合Delay Bdev使用场景 |  |
|  | NVMe over TCP Write/Read命令下发流程梳理 |  |