云计算技术大作业

小组成员：

纪鹏 2021E8016082025  
王耀辉 2021E8016082023  
郑玮蓁 2021E8016082024  
李雅硕 2021E8016082013  
任峥 2021E8016082032

1. **实验目的及要求**

利用Linux操作系统的Namespace和Cgroup机制，实现一个简单的容器引擎，容器引擎应该具有以下功能：

• 实现进程、用户、文件系统、网络等方面的隔离

• 能够在Ubuntu系统上运行CentOS环境

• 能够实现同一操作系统下两个容器之间的网络通信

• 能够为容器分配定量的CPU和内存资源

1. **实验原理及思路**

前期通过查阅资料，小组内讨论，实现容器引擎的要求，主要需要以下技术，namespace，cgroup，AUFS、Vethpair或Linux Bridge。

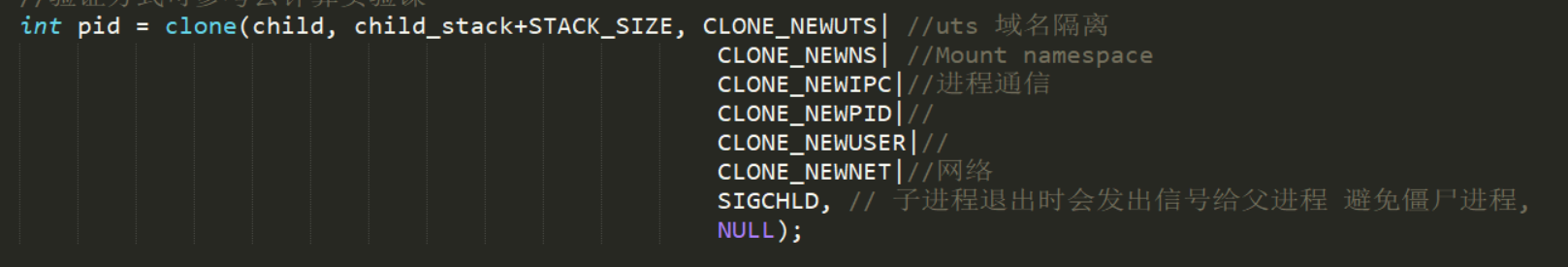
编程语言采用C语言及Go语言同时进行。

1、Linux Namespace是Linux提供的一种内核级别环境隔离的方法。



在调用clone函数时，设定不同系统调用参数，可分别实现不同类型的隔离，如果想实现多种类型namespace同时隔离，可使用“|“连接系统调用参数。

如：“CLONE\_NEWUTS | CLONE\_NEWNS | CLONE\_NEWIPC | CLONE\_NEWPID“



2、Linux CGroup全称Linux Control Group， 是Linux内核的一个功能，用来限制，控制与分离一个进程组群的资源（如CPU、内存、磁盘输入输出等）。

有三个部分：

①cgroup是对进程分组管理的一种机制，一个cgroup包含一组进程，并可以在cgroup上增加Linux subsystem的各种参数配置，将一组进程和一组subsystem的系统参数关联起来

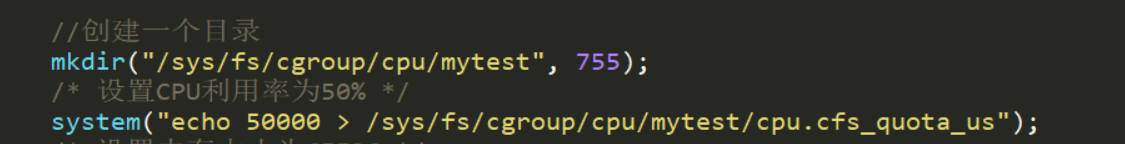
②subsystem是一组资源控制模块

③hierarchy是把一组cgroup串成一个树状的结构，一个这样的树便是一个hierarchy，通过这种树状结构，Cgroup可以做到继承。

在linux下将Cgroup实现成为了一个file system。

在/sys/fs下有一个cgroup的目录，这个目录下还有很多子目录，比如： cpu，cpuset，memory，blkio……这些，这些都是cgroup的子系统。分别用于干不同的事的。

实现思路：到/sys/fs/cgroup的各个子目录下去创建个自己的目录，如mytest，然后将mytest下的相关资源进行限制，再将容器的PID添加至mytest/tasks下，即可实现对容器资源的限制。





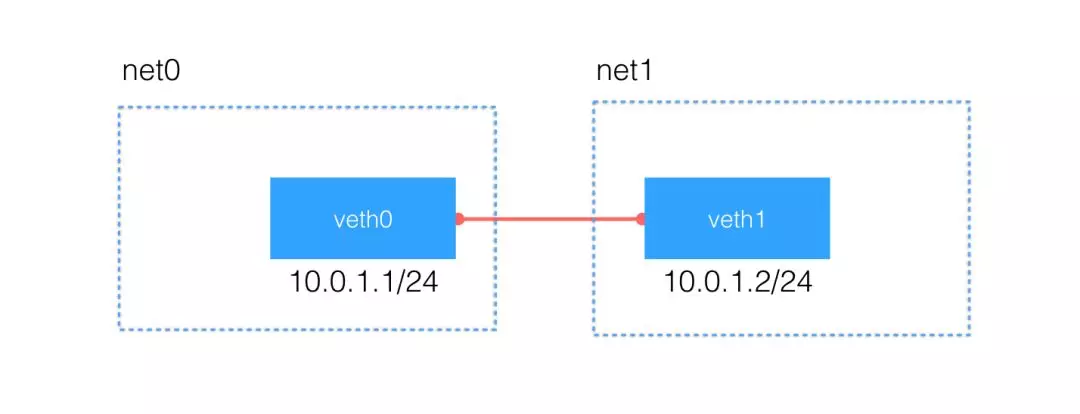
3、Vethpair和Linux Bridge

容器在创建时，进行了Network namespace隔离，每个Network namespace就有了自己独立的网络栈信息，包括独立的网卡、路由表、ARP表、iptables等网络资源，因此可以给主机上的虚拟机或者容器提供独立的、相互隔离的网络资源。

Linux 实际上是通过网络设备去操作和使用网卡的，系统安装了一个网卡后会为其生成一个网络设备实例，比如eth0。常见的虚拟化设备有Veth、Bridge、802.1.q VLAN device、TAP，容器网络通信主要用到Veth和Bridge。

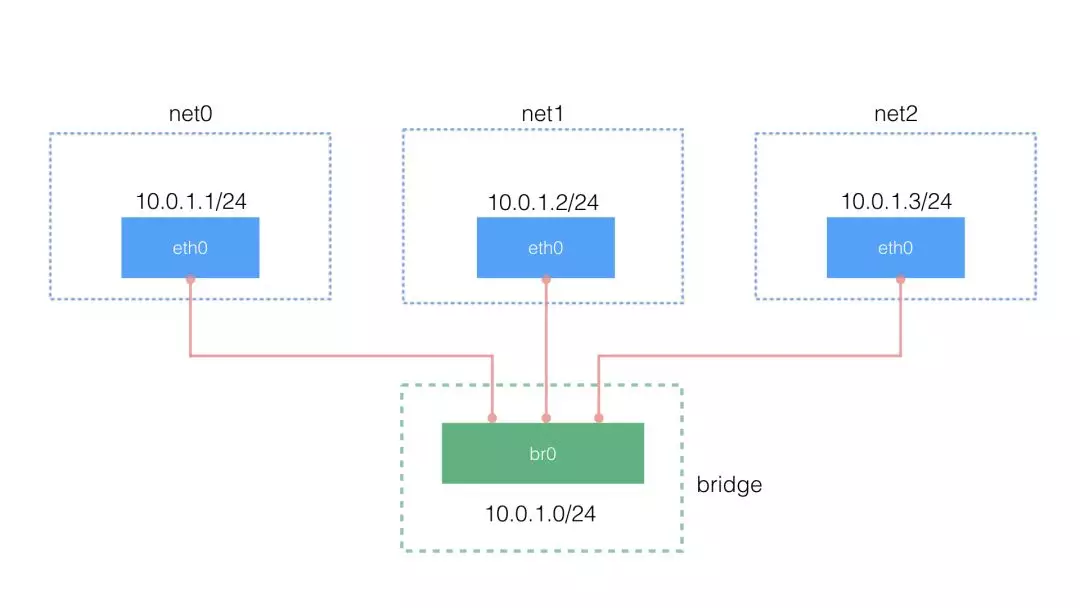
①Vethpair

创建一对veth pair，将veth pair的一端放入容器1中，另一端放在容器2中，给veth pair配上IP地址，然后将其上线，两个容器就可以通信了。这个方法适用于两个容器间通信。



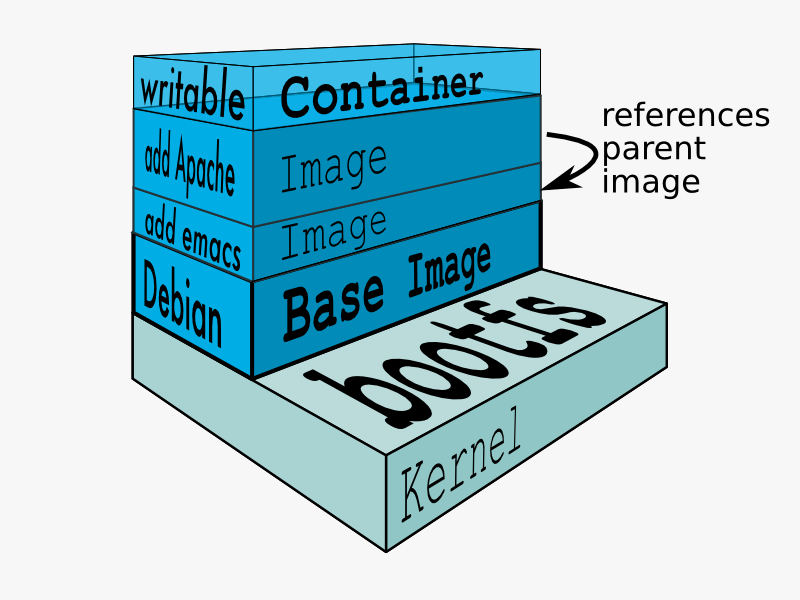
②Bridge

与上面原理类似，在宿主机上创建Bridge，然后创建veth pair后，将一端放在容器中，但另一端放在Bridge上，每对veth pair都是如此，这样每个容器都可以通过Bridge进行通信。



4、AUFS

Docker在运行镜像时使用AUFS，AUFS是一种Union File System，所谓UnionFS就是把不同物理位置的目录合并mount到同一个目录中。每一个Docker image都是由一系列read-only layer组成的。在操作镜像时，可以只对最上层的读写层进行操作，而不影响其他层，这样可以做各种修改而不用担心把系统搞坏。



1. **实验结果验证**

• C语言v1版本和Go语言的程序均实现了进程、用户、文件系统、网络等方面的隔离的要求；

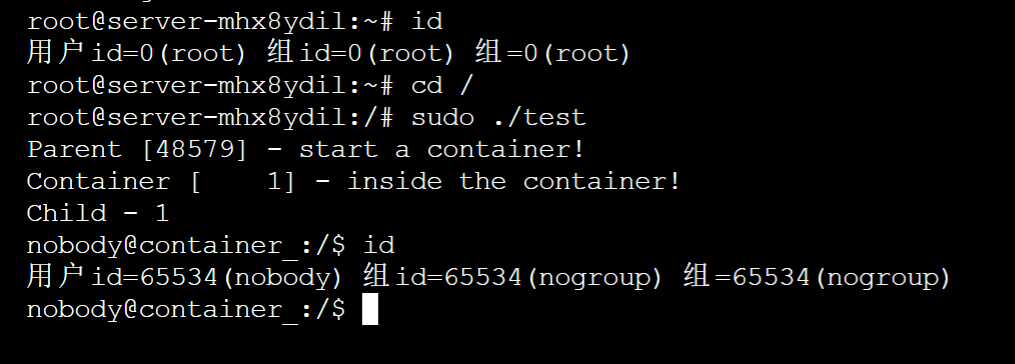
• C语言v1版本和Go语言的程序均实现了能够为容器分配定量的CPU和内存资源的要求；

• C语言v2运行后的容器是Centos系统。实现能够在Ubuntu系统上运行CentOS环境的要求；需要先在宿主机上将Centos镜像打包导出后，解压至/home/namespace-experiment下：tar -xf Centos.tar -C Centos；

• 能够实现同一操作系统下两个容器之间的网络通信的要求。两个容器网络间通信通过配置Veth pair可实现；

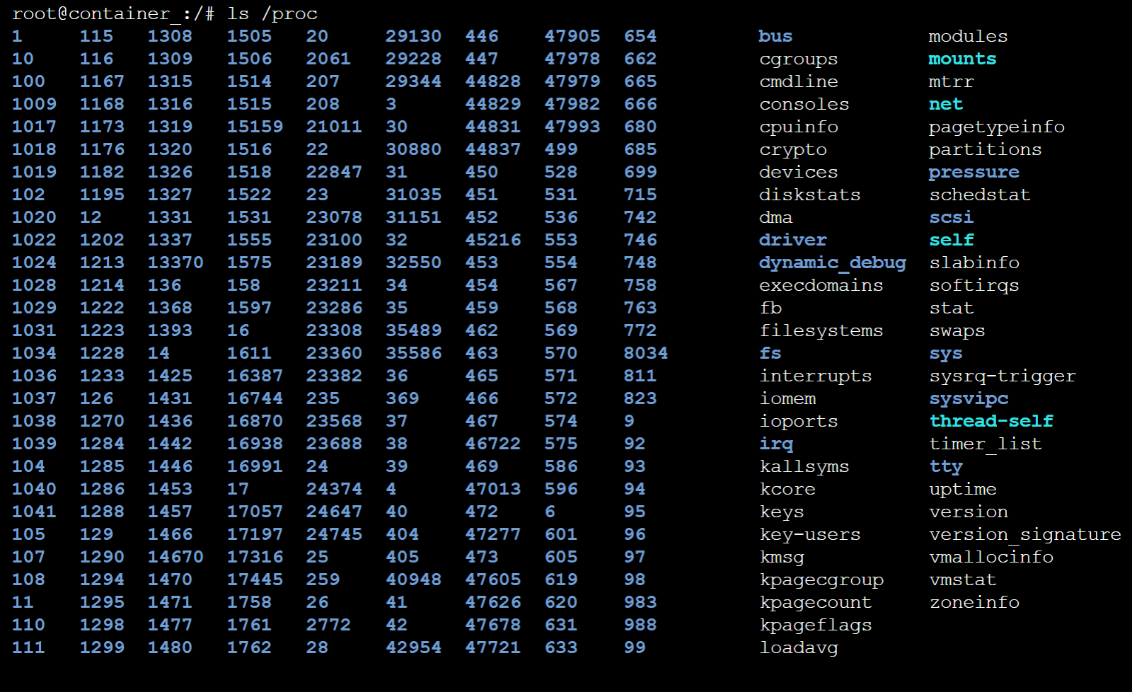
1. namespace隔离
   1. user namespace

在运行容器前后通过id命令可以看到用户是不同的，实现了user隔离；

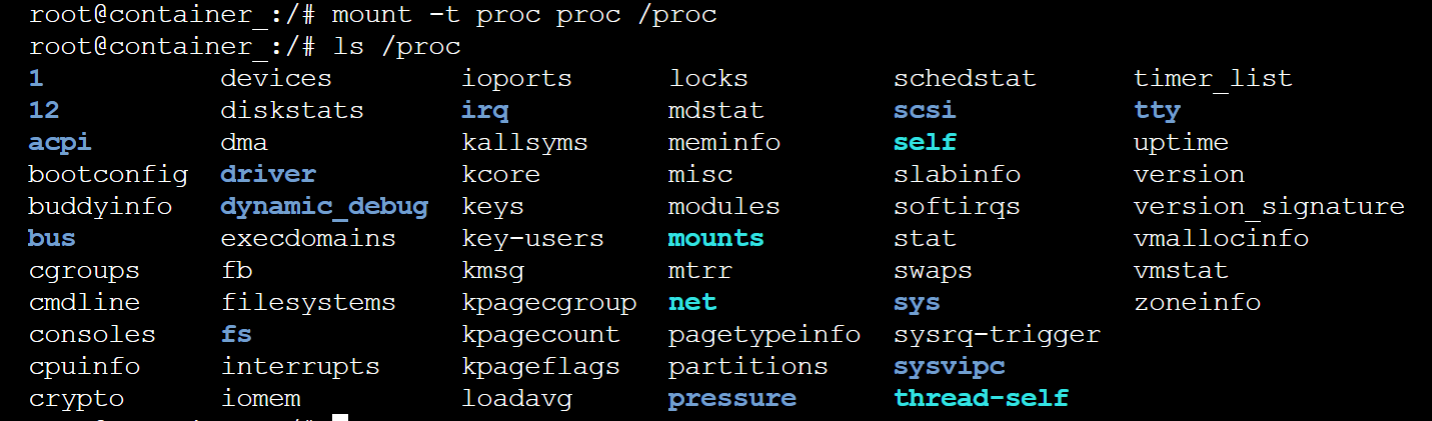


* 1. mount namespace

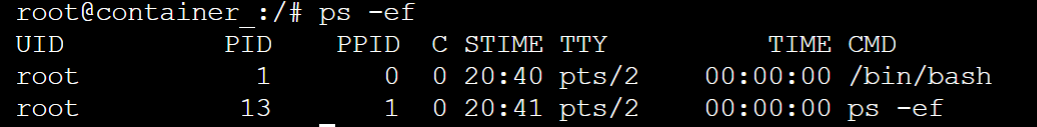
启动容器后在容器内查看/proc，内容如下，



在容器内执行mount命令，将/proc转移到容器内，此时再查看/proc内容已经改变,



再利用ps -ef查看进程，可以看到，在当前的Namespace中，bash进程是PID为1的进程。说明当前的Mount Namespace中的mount和外部空间已经隔离了，mount操作没有影响到外部。

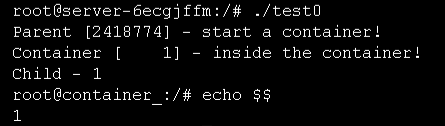


* 1. pid namespace

运行容器后，在宿主机上查看进程信息，可以看到运行的进程的真实PID如下，

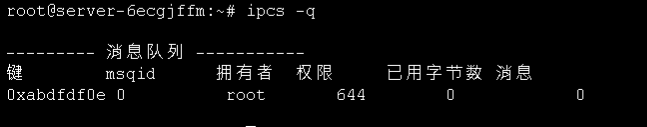


然后在容器空间查看PID如下，与真实值不同，证明已隔离，

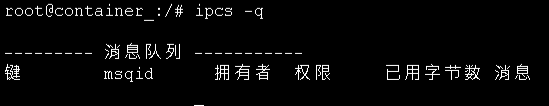


* 1. ipc namespace

首先在宿主机上查看message queue如下，



然后在容器空间中查看message queue如下，即看不到宿主机上的message queue，证明已经隔离；

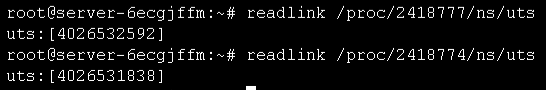


* 1. uts namespace

在容器空间中我们运行pstree -pl，可以看到系统与进程关系如下，



然后查看父进程2418774与子进程2418777所在uts空间的情况



可以看到两者不在同一个uts namespace中，验证了隔离性；

uts namespace对hostname做了隔离，所以我们还可以通过修改hostname并查看结果来验证隔离性；

首先在宿主机上查看hostname如下，



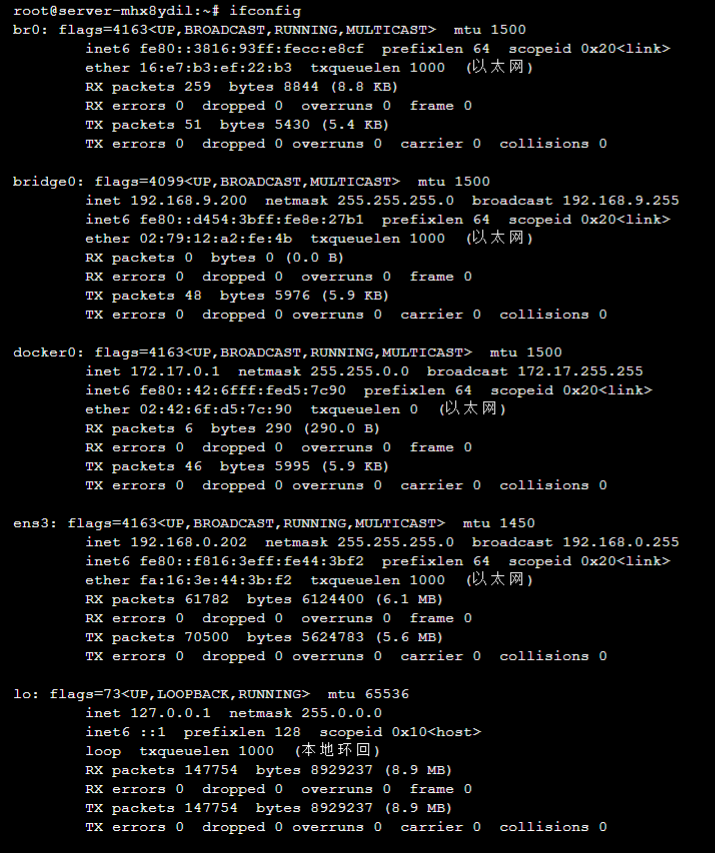
进入容器查看容器hostname如下，



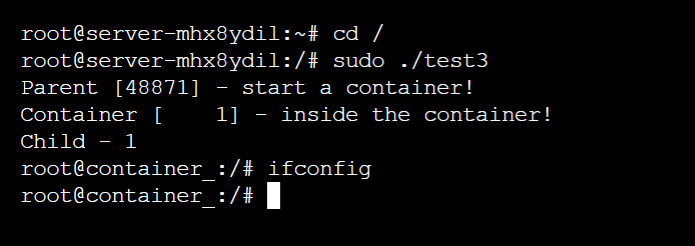
证明已经隔离；

* 1. network namepace

首先在宿主机上查看机器上的网络设备如下，



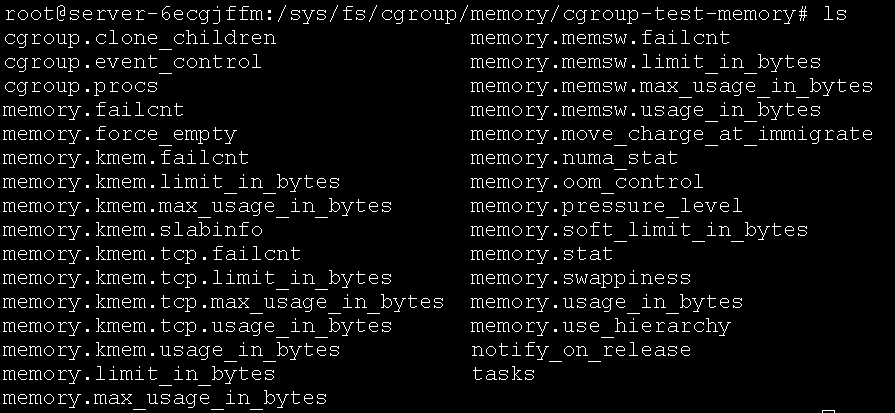
然后进入容器中查看网络设备如下，



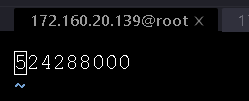
证明已经隔离；

1. cgroup资源分配
   1. 内存分配

首先，为目的文件夹cgroup-test创建hierarchy，系统默认subsystem在sys/fs/cgroup/中；所以要对其进行内存限制，我们建立文件cgroup-test-memory文件夹，系统自动添加subsystem的配置文件，其中memory.limit\_in\_bytes则是负责cgroup中进程内存的分配；



可以看到当前memory.limit\_in\_bytes文件中数值为524288000即52428800B/1024/1024 = 500M，



我们利用stress命令进行内存占用的验证，首先在对应cgroup中执行stress命令，通过ps -ef指令查看到对应指令PID2268345，再通过top指令查看到对应内存占用率约为5%；



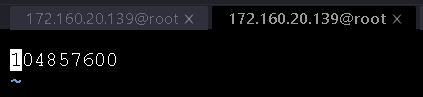


然后我们执行代码cgroup.go，修改其cgoup策略限制内存为100M，再通过指令查看结果，发现此时占用率变成了2.5%；





再打开对应subsystem中的memory.limit\_in\_bytes发现内容被修改为104857600即100M，

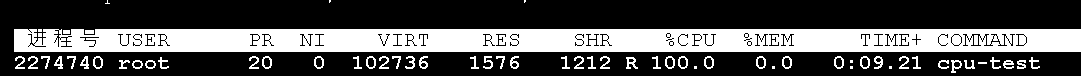


验证完毕；

* 1. Cpu分配

同理于内存分配，cpu分配通过修改/sys/fs/cgroup/cpu/cgroup-test-cpu中的cpu.cfs\_quota\_us文件，默认值为-1，即无限制；

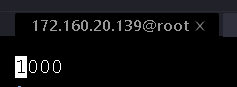
启动计cpu测试程序cpu-test.go，通过top观察其cpu占用



然后我们执行从group-test.go代码修改cpu.cfs\_quota\_us的值为1000，重新执行cpu-test，结果如下：



文件内容被修改为1000，



验证完毕；

1. 容器中运行CentOS镜像

宿主机为Ubuntu环境，所以验证思路为通过centos特有的yum指令以及通过与docker启动centos的结果进行对比验证系统是否启动；

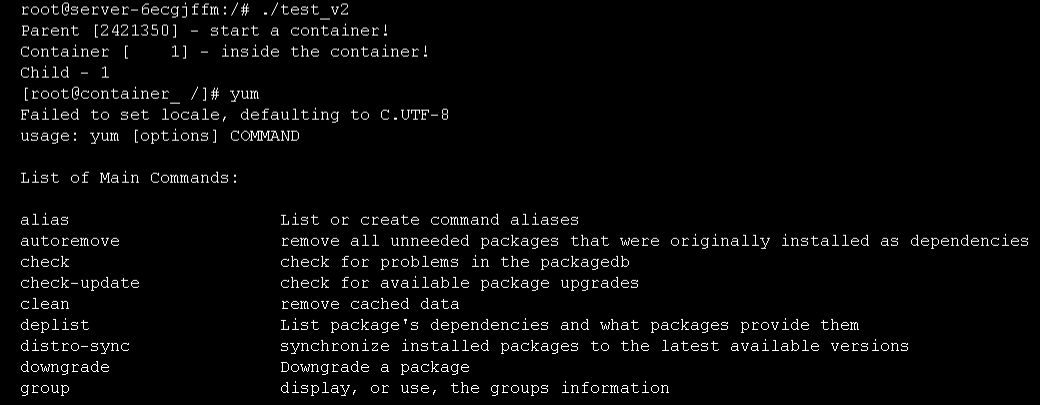
首先建立目录，

mkdir -p /home/namespace-experiment/centos/

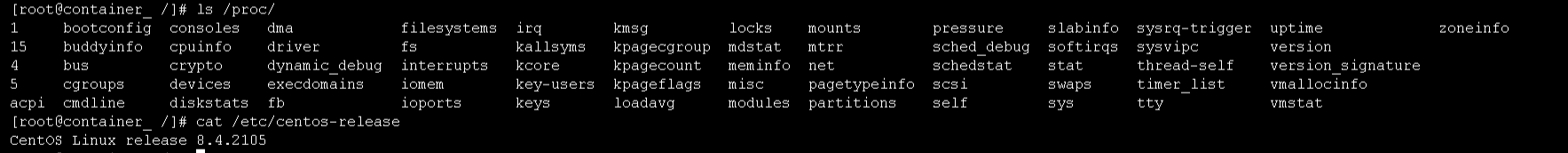
通过docker将centos镜像拉下来并打包为centos.tar，然后将镜像导入宿主机的centos文件夹，此时文件夹内容如下；



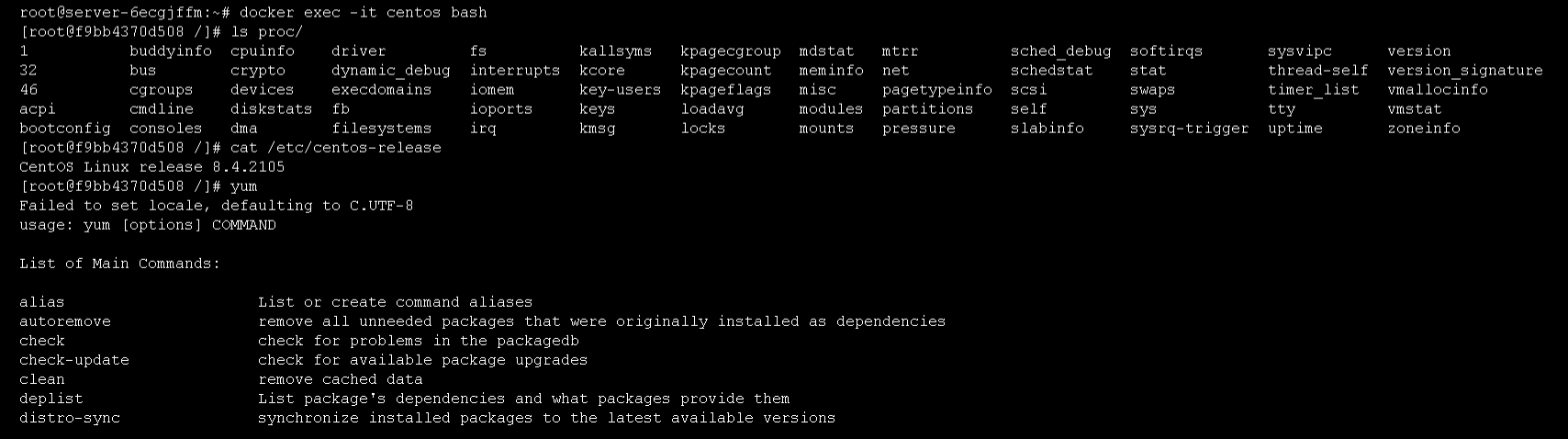
然后我们运行test\_v2程序，启动容器并启动centos系统，然后在容器中执行yum指令，结果如下，



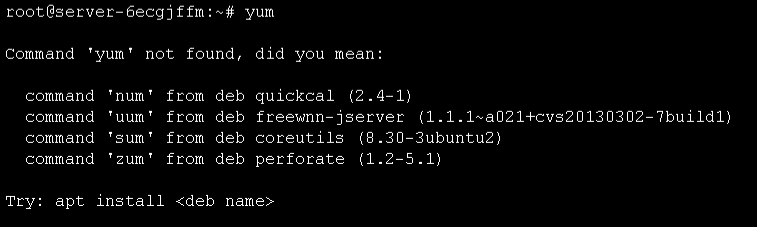
查看/proc以及/etc/centos-release，结果如下，



通过docker启动相同版本镜像，查看相同内容，结果如下，



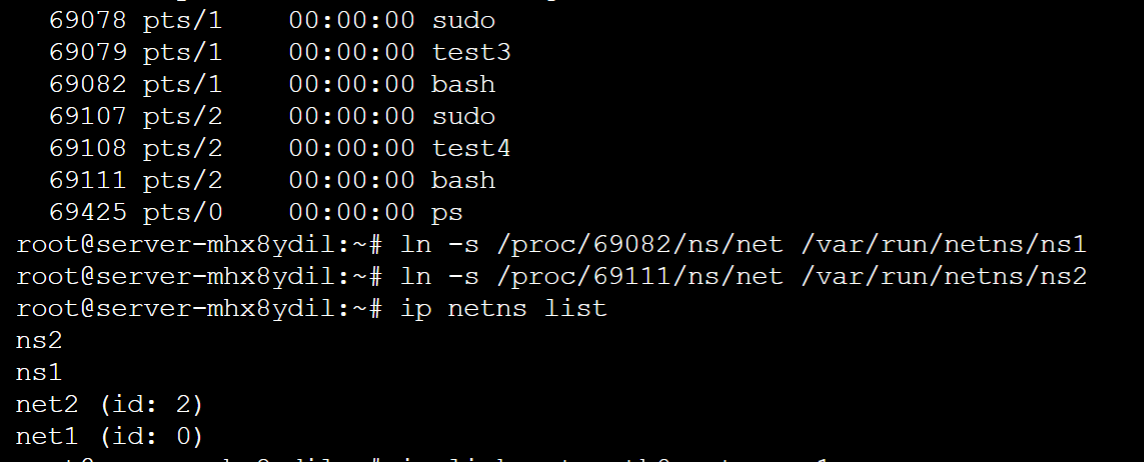
然后再在宿主机上执行yum，结果如下，



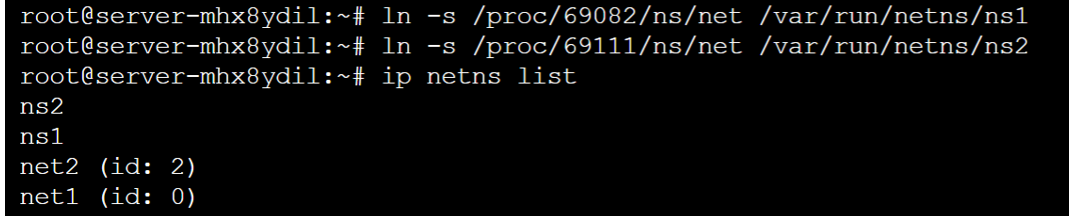
证明centos已经在Ubuntu上运行起来了；

1. 容器之间网络通信
   1. Veth pair

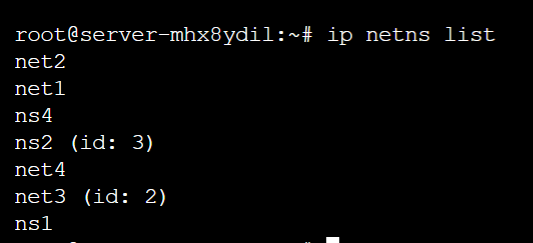
首先运行test3、test4创建两个容器，并查看其对应PID如下，



再根据其对应PID创建对应的namespace，分别命名为ns1和ns2，



可以看到通过test3和test4创建的ns1和ns2



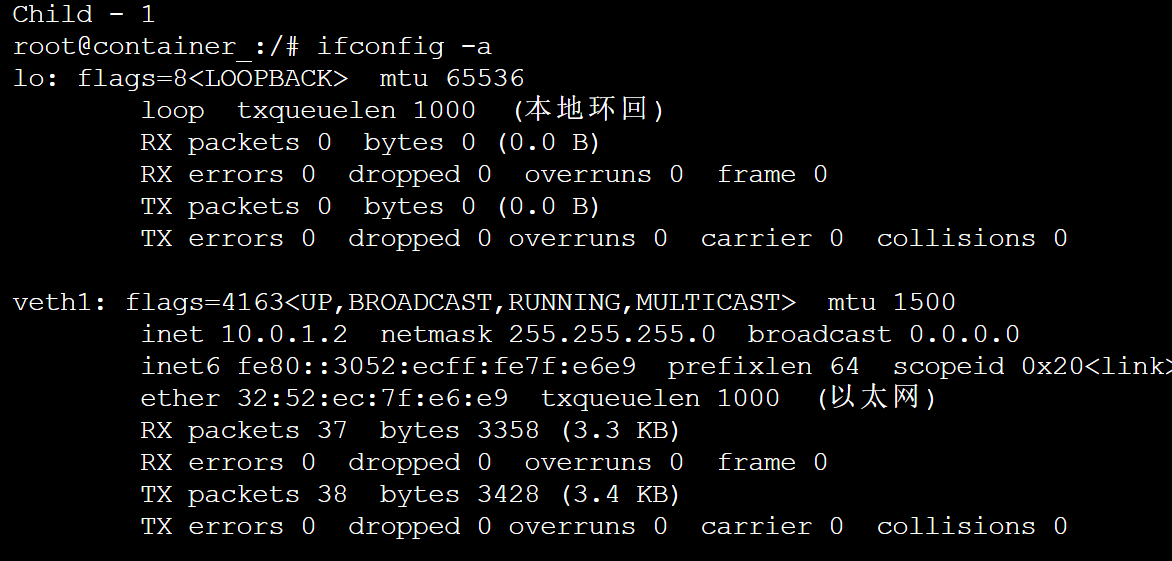
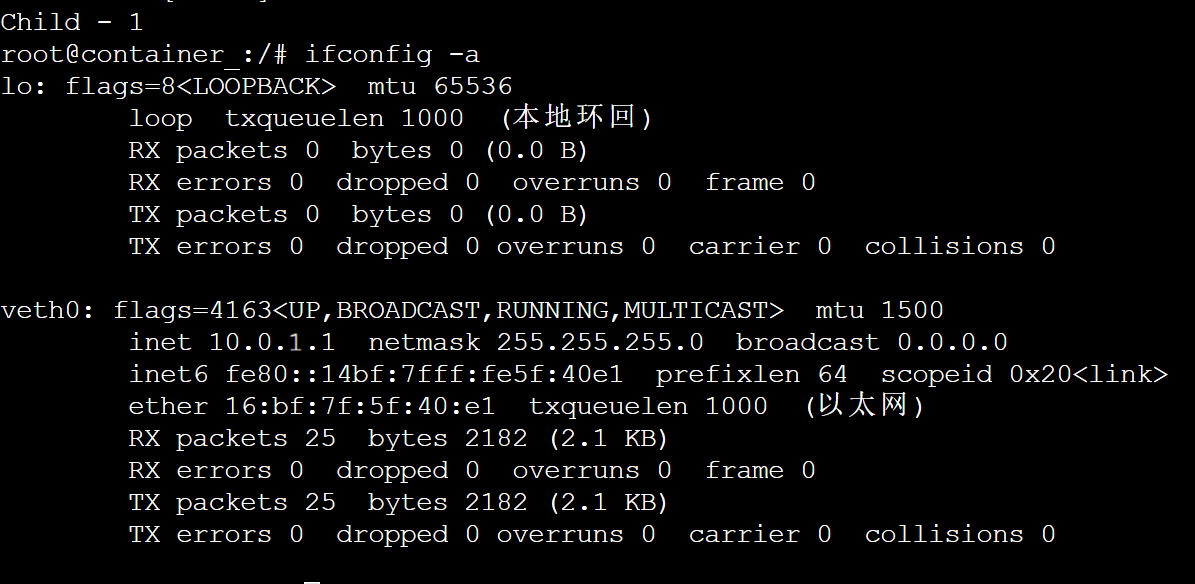
通过veth pair连接两个namespace并分配ip



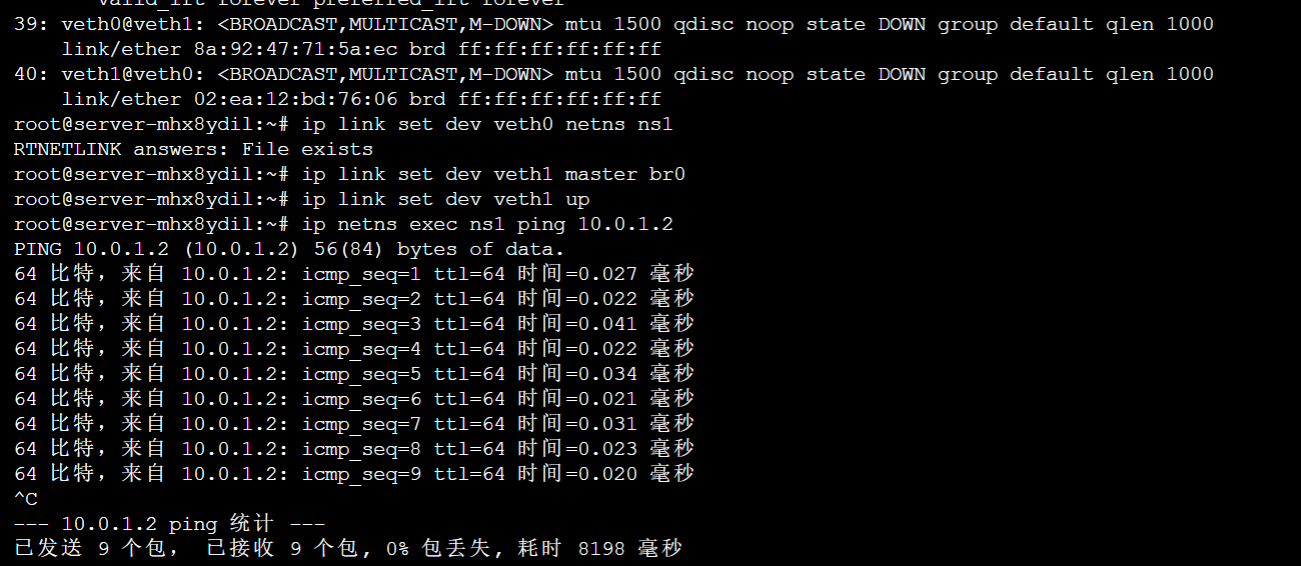




进入容器查看各自ip，可以看到分别是配置的10.0.1.1和10.0.1.2



使用ping进行测试



证明已经实现了网络通信；

1. **参考资料**

1《自己动手写Dokcer》 电子工业出版社 陈显鹭著

2《Linux开源存储全栈详解》电子工业出版社 英特尔亚太研发有限公司著

3 [DOCKER基础技术 | 酷 壳 - CoolShell](https://coolshell.cn/?s=DOCKER%E5%9F%BA%E7%A1%80%E6%8A%80%E6%9C%AF)

4 <https://blog.csdn.net/flynetcn/article/details/119567350>

5 [一文搞懂 Linux network namespace - bakari - 博客园 (cnblogs.com)](https://www.cnblogs.com/bakari/p/10443484.html)