Docker背后的内核知识

namespace资源隔离

by 李辰洋

概念

分类	系统调用参数	相关内核版本
Mount namespace	CLONE_NEWNS	Linux 2.4.19
UTS namespace	CLONE_NEWUTS	Linux 2.6.19
IPC namespace	CLONE_NEWIPC	Linux 2.6.19
PID namespace	CLONE_NEWPID	Linux 2.6.24
Network namespace	CLONE_NEWNET	Linux 2.6.29
User namespace	CLONE_NEWUSER	Linux 3.8

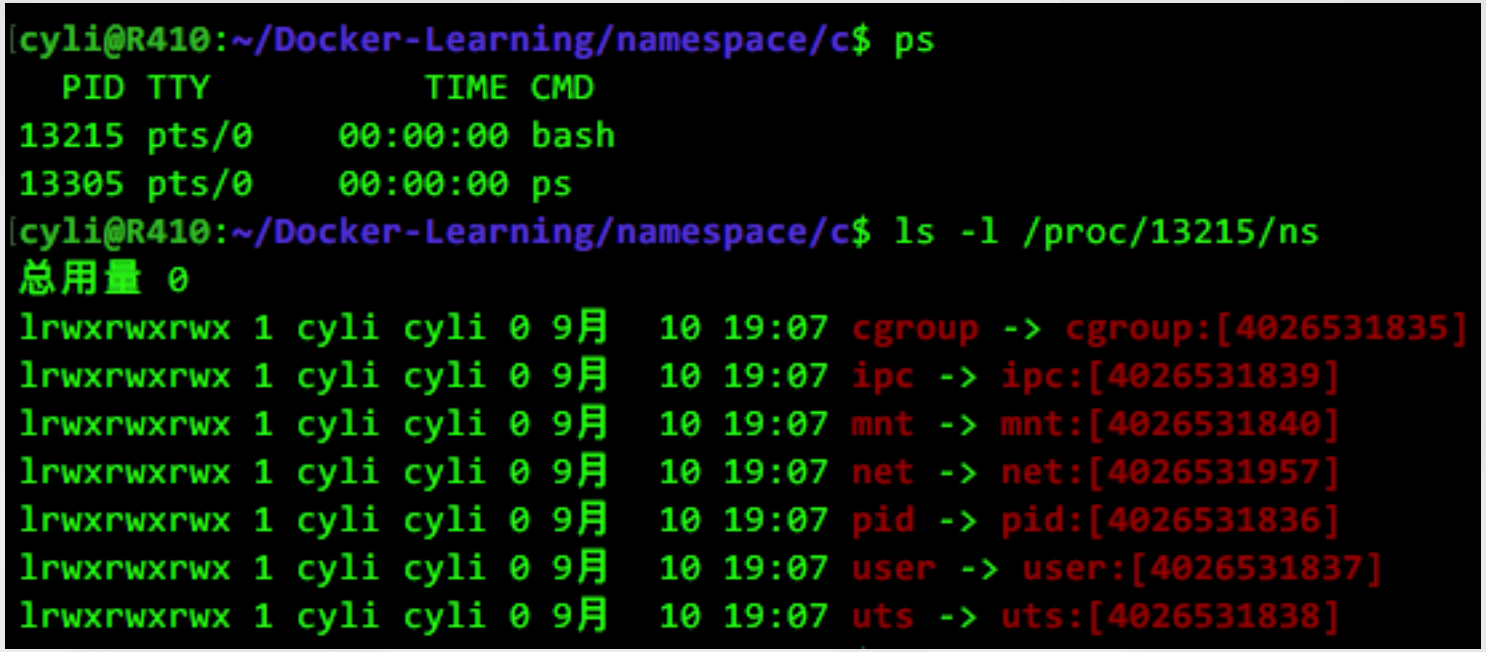
概念

- 系统调用

```
int clone(int (fn)(void ), void *child_stack, int flags, void *arg);
```

http://man7.org/linux/man-pages/man2/clone.2.html

- 查看 /proc/<pid>/ns



```
[cyli@R410:~/Docker-Learning/namespace/c$ ps
  PID TTY          TIME CMD
13215 pts/0    00:00:00 bash
13305 pts/0    00:00:00 ps
[cyli@R410:~/Docker-Learning/namespace/c$ ls -l /proc/13215/ns
总用量 0
lrwxrwxrwx 1 cyli cyli 0 9月  10 19:07 cgroup -> cgroup:[4026531835]
lrwxrwxrwx 1 cyli cyli 0 9月  10 19:07 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 cyli cyli 0 9月  10 19:07 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 cyli cyli 0 9月  10 19:07 net -> net:[4026531957]
lrwxrwxrwx 1 cyli cyli 0 9月  10 19:07 pid -> pid:[4026531836]
lrwxrwxrwx 1 cyli cyli 0 9月  10 19:07 user -> user:[4026531837]
lrwxrwxrwx 1 cyli cyli 0 9月  10 19:07 uts -> uts:[4026531838]
```

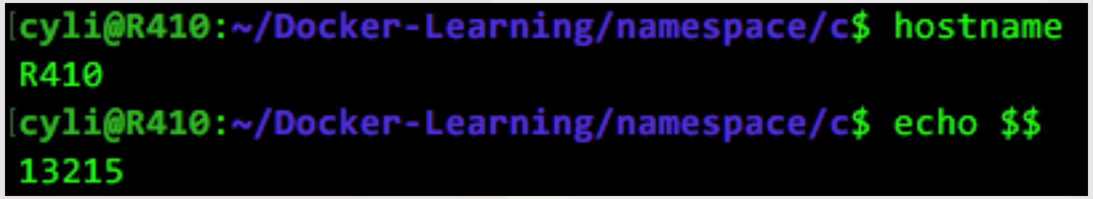
UTS namespace

- 核心代码

```
#define STACK_SIZE (1024*1024)

static char container_stack[STACK_SIZE];

char* const container_args[] = { "/bin/bash", NULL };

int container_main(void *args) {
    sethostname("tinylcy", 7);
    execv(container_args[0], container_args);
    return -1;
}

int main(void) {
    int container_pid = clone(container_main, container_stack + STACK_SIZE
                                          , SIGCHLD | CLONE_NEWUTS, NULL);
    waitpid(container_pid, NULL, 0);
    return 0;
}
```

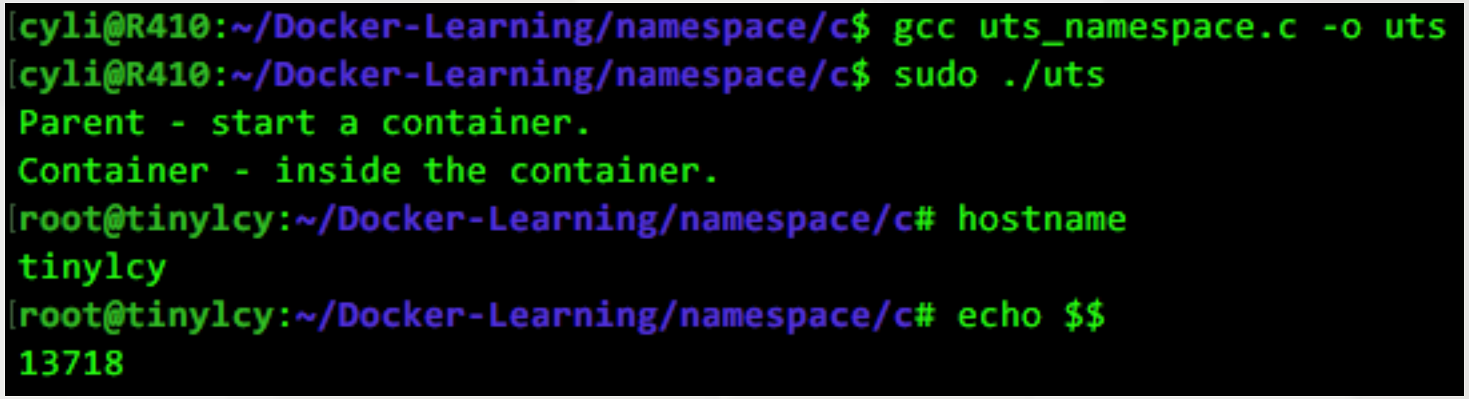
UTS namespace

- 宿主机



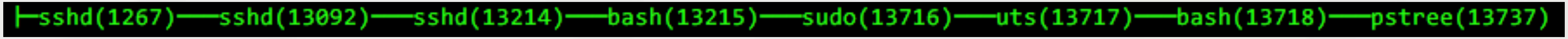
```
[cyli@R410:~/Docker-Learning/namespace/c$ hostname
R410
[cyli@R410:~/Docker-Learning/namespace/c$ echo $$
13215
```

- 容器



```
[cyli@R410:~/Docker-Learning/namespace/c$ gcc uts_namespace.c -o uts
[cyli@R410:~/Docker-Learning/namespace/c$ sudo ./uts
Parent - start a container.
Container - inside the container.
[root@tinylcy:~/Docker-Learning/namespace/c# hostname
tinylcy
[root@tinylcy:~/Docker-Learning/namespace/c# echo $$
13718
```

- pstree –pl



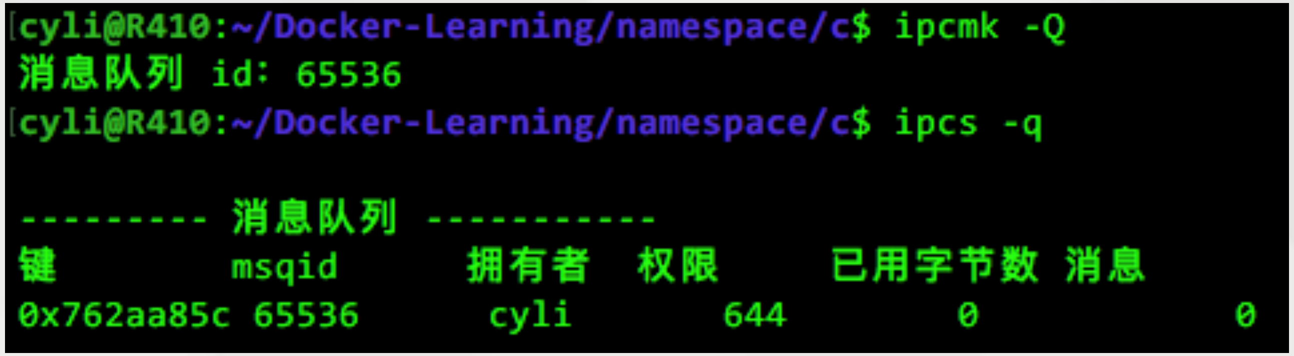
```
├─sshd(1267)───sshd(13092)───sshd(13214)───bash(13215)───sudo(13716)───uts(13717)───bash(13718)───pstree(13737)
```

IPC namespace

- 核心代码

```
int container_pid = clone(container_main, container_stack + STACK_SIZE
                                                 , SIGCHLD | CLONE_NEWIPC, NULL);
```

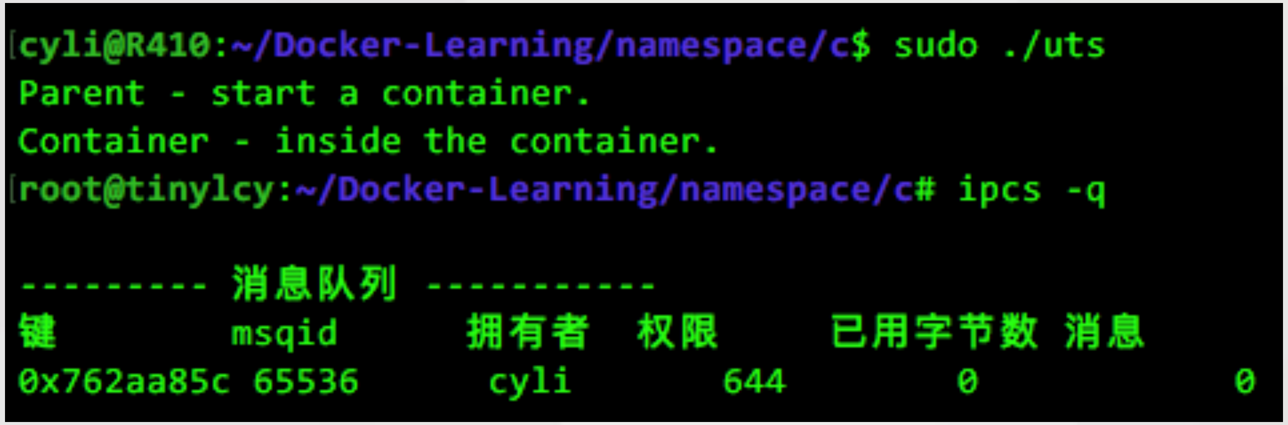
- 创建消息队列



```
cyli@R410:~/Docker-Learning/namespace/c$ ipcmk -Q
消息队列 id: 65536
cyli@R410:~/Docker-Learning/namespace/c$ ipcs -q

--------- 消息队列 -----------
键        msqid      拥有者  权限     已用字节数 消息
0x762aa85c 65536      cyli       644         0            0
```

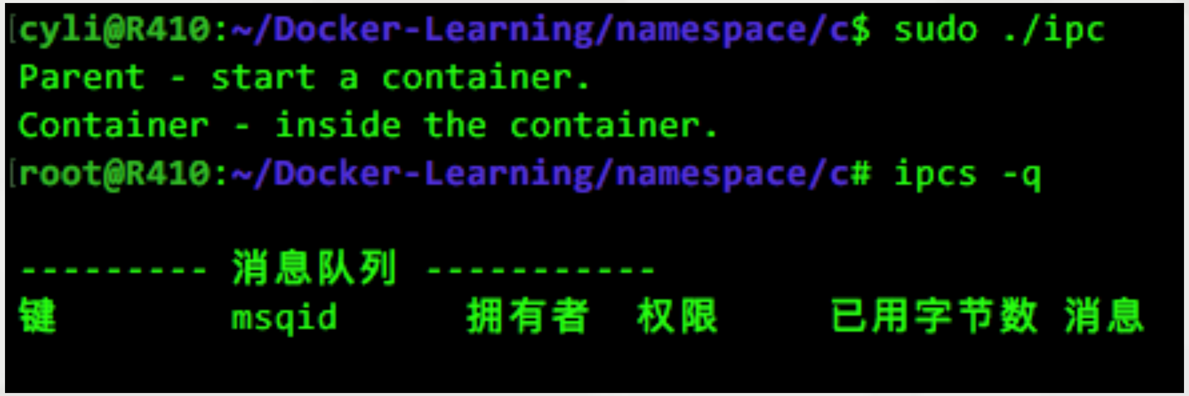
IPC namespace

- 执行 UTS



```
[cyli@R410:~/Docker-Learning/namespace/c$ sudo ./uts
Parent - start a container.
Container - inside the container.
[root@tinylcy:~/Docker-Learning/namespace/c# ipcs -q

--------- 消息队列 -----------
键        msqid      拥有者  权限     已用字节数 消息
0x762aa85c 65536      cyli       644        0            0
```

- 执行 IPC



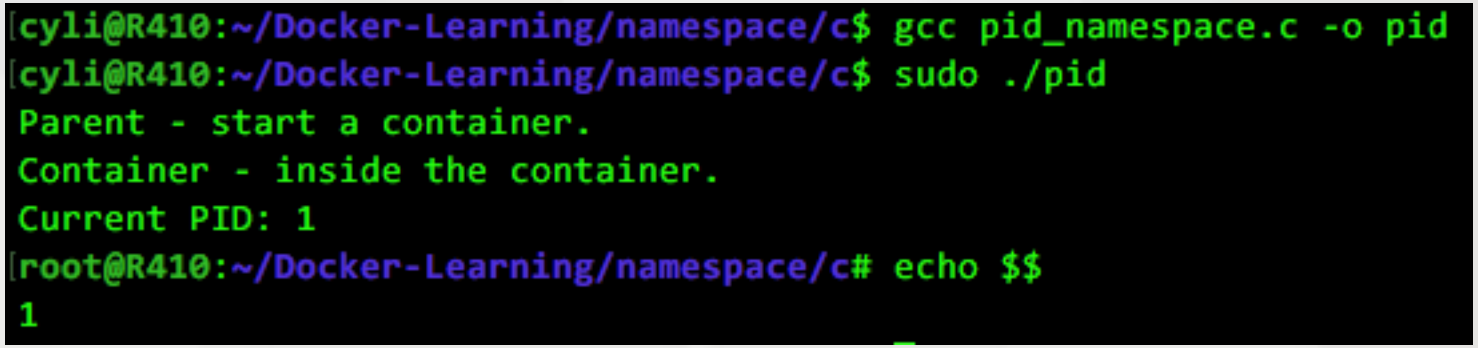
```
[cyli@R410:~/Docker-Learning/namespace/c$ sudo ./ipc
Parent - start a container.
Container - inside the container.
[root@R410:~/Docker-Learning/namespace/c# ipcs -q

--------- 消息队列 -----------
键        msqid      拥有者  权限     已用字节数 消息
```

PID & Mount namespace

- 核心代码

```
int container_pid = clone(container_main, container_stack + STACK_SIZE
                                      , SIGCHLD | CLONE_NEWPID, NULL);
```

- 执行 PID



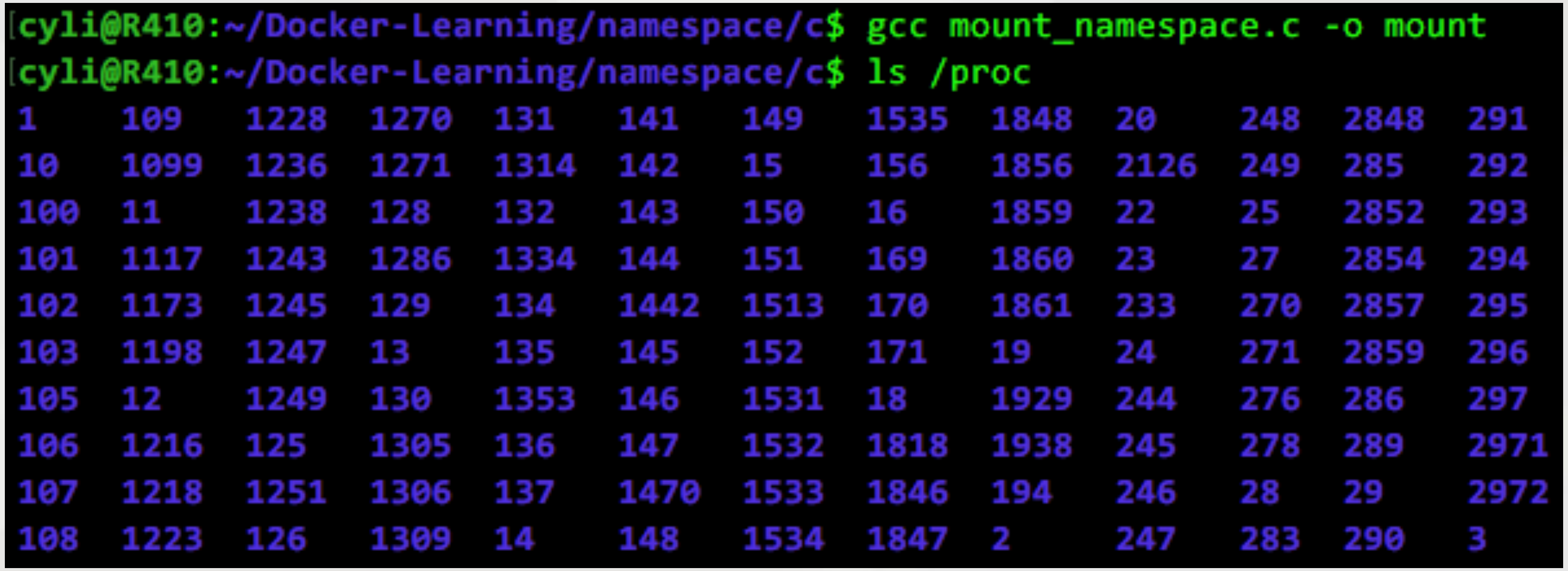
```
[cyli@R410:~/Docker-Learning/namespace/c$ gcc pid_namespace.c -o pid
[cyli@R410:~/Docker-Learning/namespace/c$ sudo ./pid
Parent - start a container.
Container - inside the container.
Current PID: 1
[root@R410:~/Docker-Learning/namespace/c# echo $$
1
```

PID & Mount namespace

- 核心代码

```
int container_pid = clone(container_main, container_stack + STACK_SIZE
                        , SIGCHLD | CLONE_NEWNS | CLONE_NEWPID, NULL);

system("mount -t proc proc /proc");  // http://tldp.org/LDP/lfs/5.0/html/chapter06/proc.html
```

- 执行 Mount 前



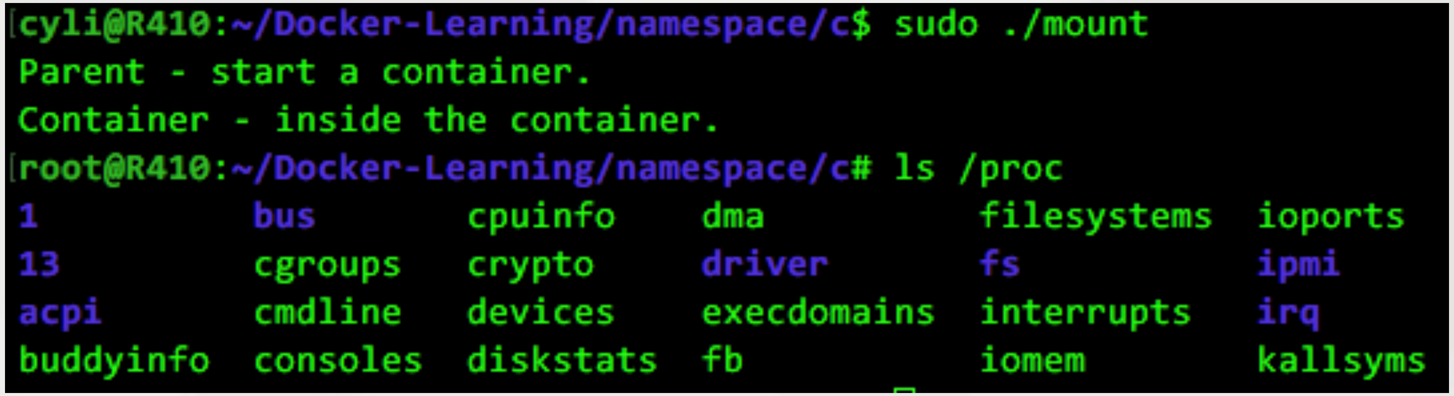
```
[cyli@R410:~/Docker-Learning/namespace/c$ gcc mount_namespace.c -o mount
[cyli@R410:~/Docker-Learning/namespace/c$ ls /proc
1    109   1228  1270  131   141   149   1535  1848  20    248  2848  291
10   1099  1236  1271  1314  142   15    156   1856  2126  249  285   292
100  11    1238  128   132   143   150   16    1859  22    25   2852  293
101  1117  1243  1286  1334  144   151   169   1860  23    27   2854  294
102  1173  1245  129   134   1442  1513  170   1861  233   270  2857  295
103  1198  1247  13    135   145   152   171   19    24    271  2859  296
105  12    1249  130   1353  146   1531  18    1929  244   276  286   297
106  1216  125   1305  136   147   1532  1818  1938  245   278  289   2971
107  1218  1251  1306  137   1470  1533  1846  194   246   28   29    2972
108  1223  126   1309  14    148   1534  1847  2     247   283  290   3
```

PID & Mount namespace

- 核心代码

```
int container_pid = clone(container_main, container_stack + STACK_SIZE
                        , SIGCHLD | CLONE_NEWNS | CLONE_NEWPID, NULL);

system("mount -t proc proc /proc");  // http://tldp.org/LDP/lfs/5.0/html/chapter06/proc.html
```

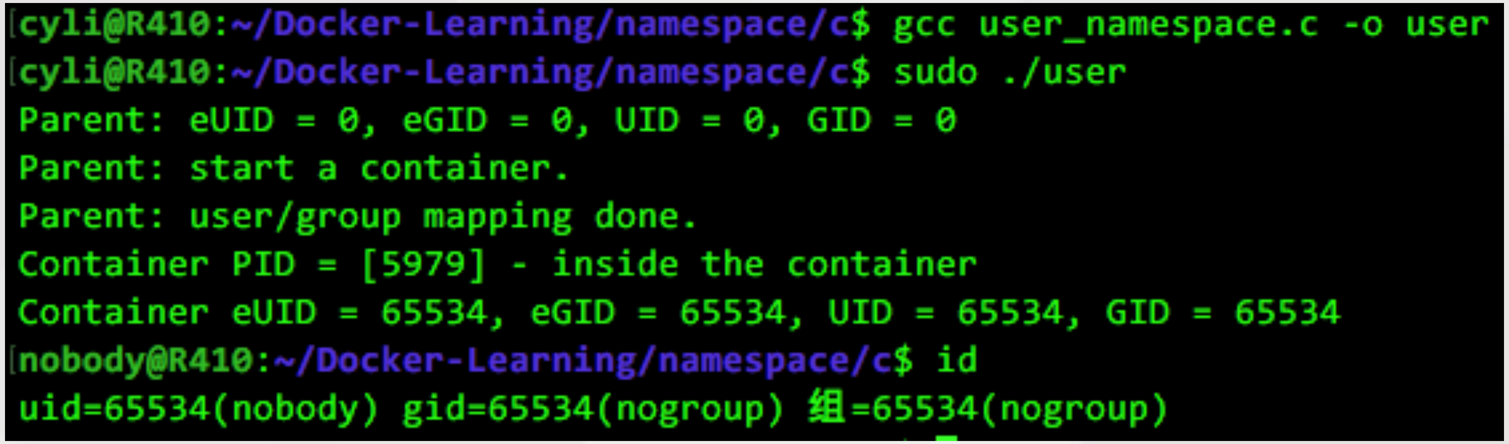
- 执行 Mount 后



```
[cyli@R410:~/Docker-Learning/namespace/c$ sudo ./mount
Parent - start a container.
Container - inside the container.
[root@R410:~/Docker-Learning/namespace/c# ls /proc
1          bus        cpuinfo    dma          filesystems  ioports
13         cgroups    crypto     driver       fs           ipmi
acpi       cmdline    devices    execdomains  interrupts   irq
buddyinfo  consoles   diskstats  fb           iomem        kallsyms
```

User namespace

- 设置CLONE_NEWUSER



```
[cyli@R410:~/Docker-Learning/namespace/c$ gcc user_namespace.c -o user
[cyli@R410:~/Docker-Learning/namespace/c$ sudo ./user
 Parent: eUID = 0, eGID = 0, UID = 0, GID = 0
 Parent: start a container.
 Parent: user/group mapping done.
 Container PID = [5979] - inside the container
 Container eUID = 65534, eGID = 65534, UID = 65534, GID = 65534
[nobody@R410:~/Docker-Learning/namespace/c$ id
 uid=65534(nobody) gid=65534(nogroup) 组=65534(nogroup)
```

- UID / GID 映射
  - /proc/<pid>/uid_map
  - /proc/<pid>/gid_map
  - ID-inside-ns ID-outside-ns length

User namespace

- 核心代码

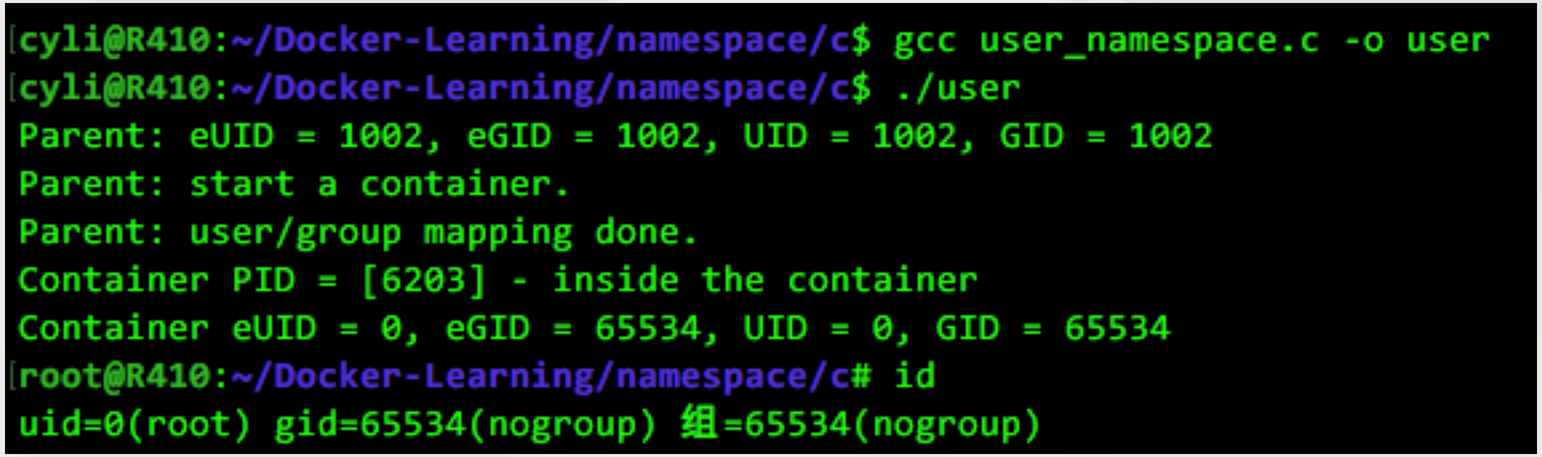
```
int pipefd[2];

int main(void) {
    const int gid = getgid(), uid = getuid();

    pipe(pipefd);

    int container_pid = clone(container_main, container_stack + STACK_SIZE
                                         , CLONE_NEWUSER | SIGCHLD, NULL);

    set_uid_map(container_pid, 0, uid, 1);
    set_gid_map(container_pid, 0, gid, 1);

    close(pipefd[1]);

    waitpid(container_pid, NULL, 0);

    return 0;
}
```

User namespace

- 核心代码

```
int container_main(void *args) {
    char ch;
    close(pipefd[1]);
    read(pipefd[0], &ch, 1);
    execv(container_args[0], container_args);
    return -1;
}
```

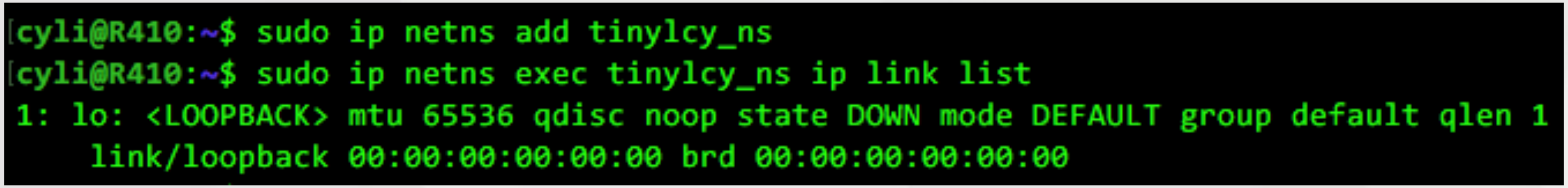
- 执行User



```
cyli@R410:~/Docker-Learning/namespace/c$ gcc user_namespace.c -o user
cyli@R410:~/Docker-Learning/namespace/c$ ./user
Parent: eUID = 1002, eGID = 1002, UID = 1002, GID = 1002
Parent: start a container.
Parent: user/group mapping done.
Container PID = [6203] - inside the container
Container eUID = 0, eGID = 65534, UID = 0, GID = 65534
root@R410:~/Docker-Learning/namespace/c# id
uid=0(root) gid=65534(nogroup) 组=65534(nogroup)
```

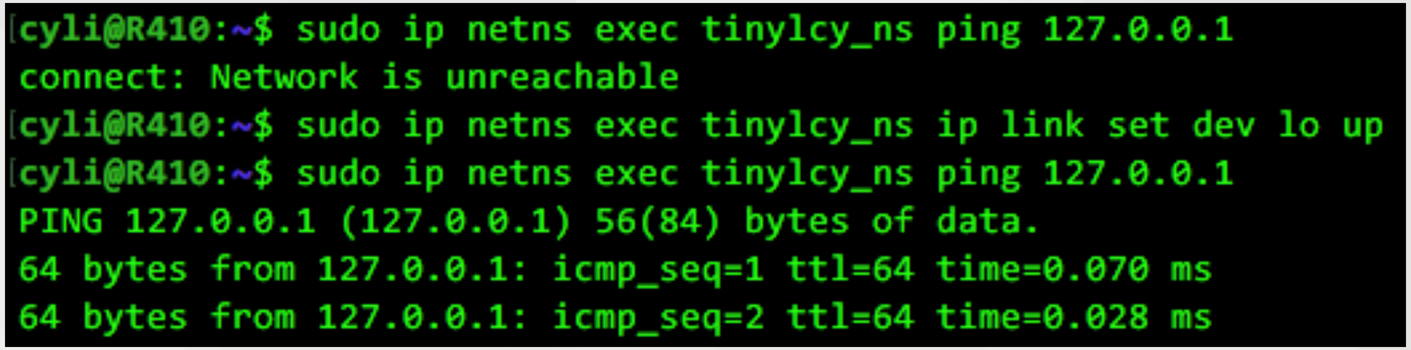
Network namespace

- 创建 namespace



```
[cyli@R410:~$ sudo ip netns add tinylcy_ns
[cyli@R410:~$ sudo ip netns exec tinylcy_ns ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

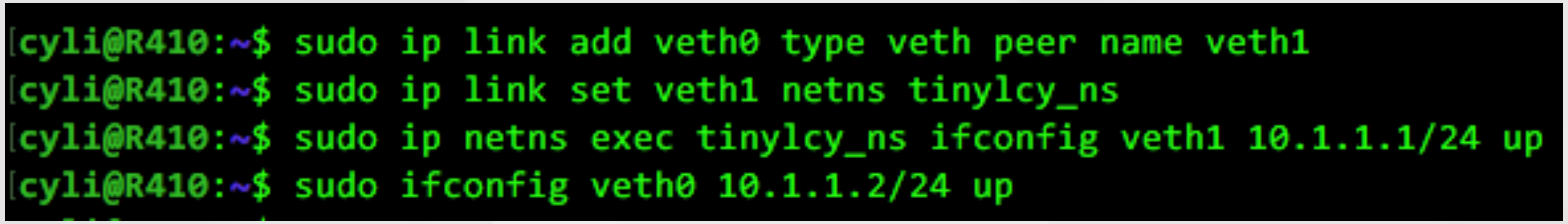
- 启动 loopback 设备



```
[cyli@R410:~$ sudo ip netns exec tinylcy_ns ping 127.0.0.1
connect: Network is unreachable
[cyli@R410:~$ sudo ip netns exec tinylcy_ns ip link set dev lo up
[cyli@R410:~$ sudo ip netns exec tinylcy_ns ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.070 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.028 ms
```

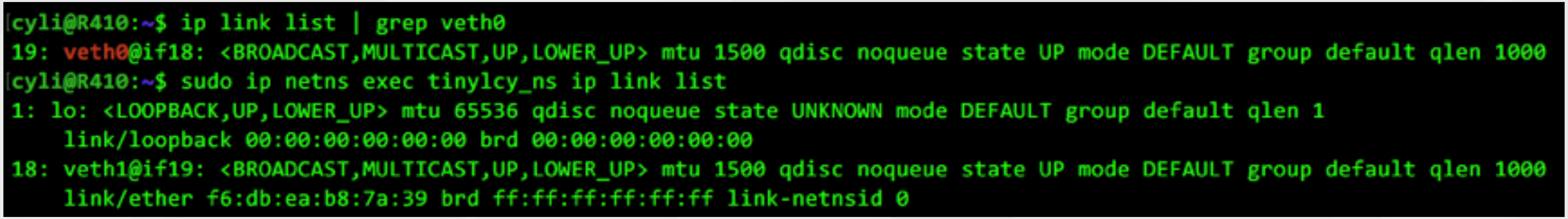
Network namespace

- 创建 VETH(Virtual ETHernet) Pair



```
cyli@R410:~$ sudo ip link add veth0 type veth peer name veth1
cyli@R410:~$ sudo ip link set veth1 netns tinylcy_ns
cyli@R410:~$ sudo ip netns exec tinylcy_ns ifconfig veth1 10.1.1.1/24 up
cyli@R410:~$ sudo ifconfig veth0 10.1.1.2/24 up
```

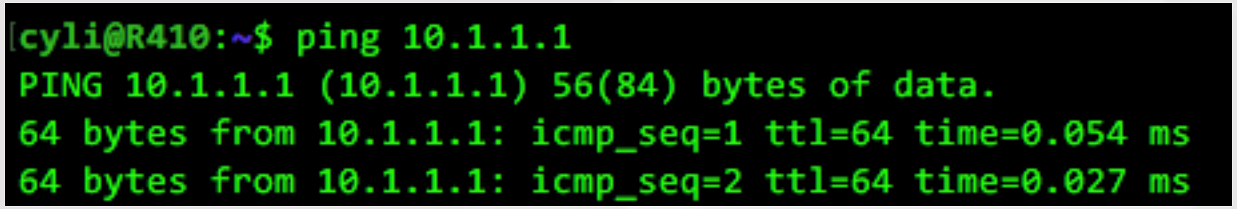
- 测试 (VETH)



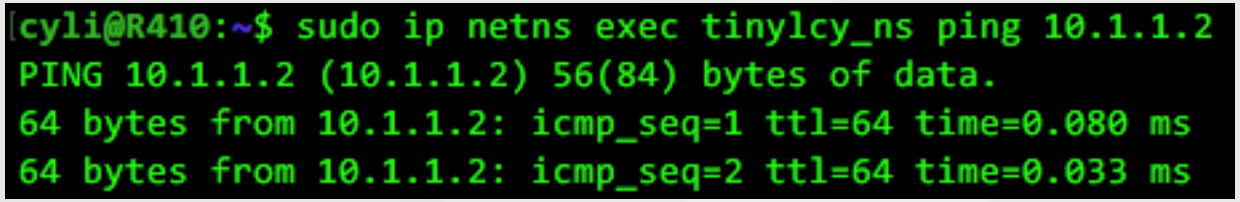
```
cyli@R410:~$ ip link list | grep veth0
19: veth0@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
cyli@R410:~$ sudo ip netns exec tinylcy_ns ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
18: veth1@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether f6:db:ea:b8:7a:39 brd ff:ff:ff:ff:ff:ff link-netnsid 0
```

Network namespace

- 测试 (ping)

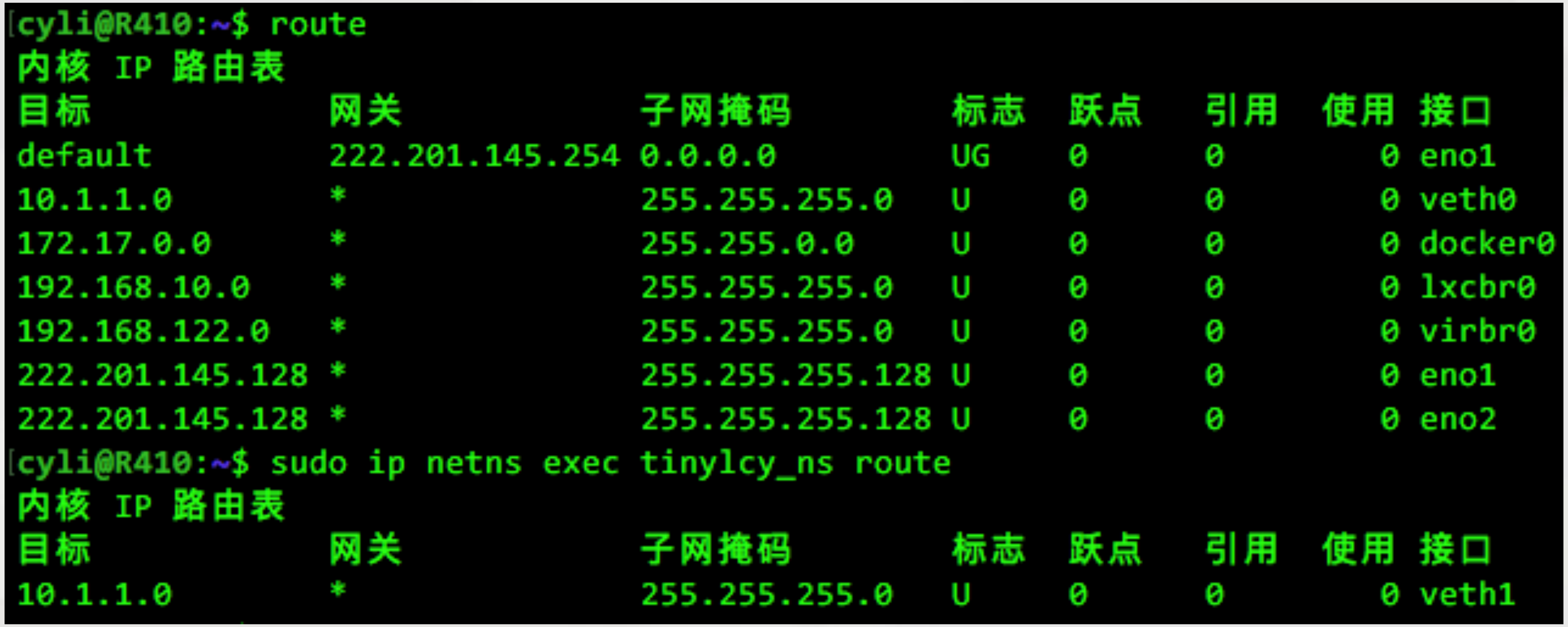


```
cyli@R410:~$ ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.054 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=0.027 ms
```



```
cyli@R410:~$ sudo ip netns exec tinylcy_ns ping 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=64 time=0.033 ms
```

- 测试 (路由表)



```
cyli@R410:~$ route
内核 IP 路由表
目标            网关            子网掩码        标志  跃点   引用  使用 接口
default         222.201.145.254 0.0.0.0         UG    0      0        0 eno1
10.1.1.0        *               255.255.255.0   U     0      0        0 veth0
172.17.0.0      *               255.255.0.0     U     0      0        0 docker0
192.168.10.0    *               255.255.255.0   U     0      0        0 lxcbr0
192.168.122.0   *               255.255.255.0   U     0      0        0 virbr0
222.201.145.128 *               255.255.255.128 U     0      0        0 eno1
222.201.145.128 *               255.255.255.128 U     0      0        0 eno2
cyli@R410:~$ sudo ip netns exec tinylcy_ns route
内核 IP 路由表
目标            网关            子网掩码        标志  跃点   引用  使用 接口
10.1.1.0        *               255.255.255.0   U     0      0        0 veth1
```

THANKS

https://github.com/scut-ccmdp-lab/Docker-Learning/tree/master/namespace