# Linux namespace机制

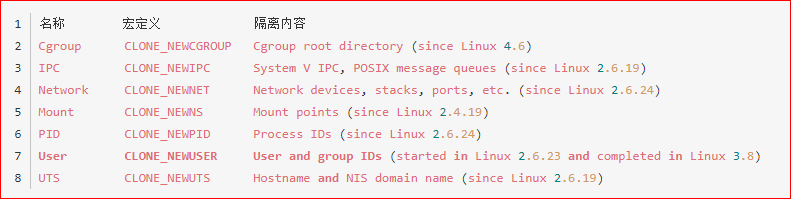
## First question：what is namespace?

Namespace是linux内核提供的一种机制，用于对系统的全局资源进行隔离。

## Second question：which resource was be isolated by namespace in linux system?

在linux系统中，进程间通信机制IPC，主机名UTS，进程ID，网络资源，文件系统结构mount，用户和组都属于全局性资源，启用namespace机制后，namespace对以上资源进行了隔离，在不同的namespace内部，用户感知到的主机名可以是独立的。网络资源（接口，协议栈，路由表等）也可以是独立的。

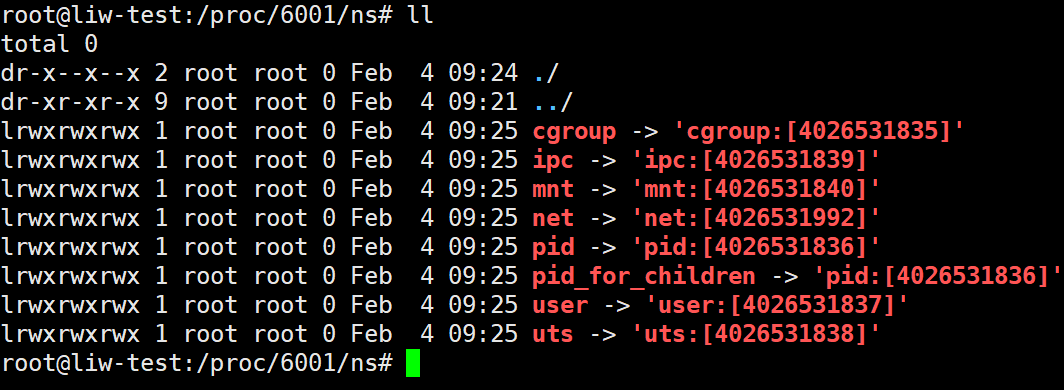
Linux中的namespace主要有



## How to find the namespace for a particular process

每一个Linux进程都可以在/proc下发现一个对应进程id的文件目录

该目录保存了进程运行时的信息，其中ns目录记录了该进程所属的namespace空间。



对不同的进程，其可能属于同一个namespace，因此可以通过检测进程下所属的ns id得到当前系统中总计有多少个namespace空间。

## Third question：how to manipulate namespace

### 1：network namespace

一般情况下，我们可以通过ifconfig/route等基础命令操作linux网络设备，协议栈等。当系统支持namespace时，则可以使用ip命令进行网络接口，协议栈的操作。

1. 创建一个netns

Ip netns add <network namespace name>

例如：ip netns add nsttest

1. 查询netns

Ip netns list

1. 删除netns

Ip netns delete <network namespace name>

1. 在netns中执行网络命令

Ip netns exec <network namespace name> <command>

例如：ip netns exec nsttest ip addr

1. 创建network namespace后的一些工作

首先会使用ip命令将lo口状态设置为up状态。

Ip netns exec nstest ip link set dev lo up

创建veth pair虚拟设备，并将该设备的另外一端移动到namespace空间中

Ip link add veth-a type veth peer name veth-b //创建veth设备

Ip link set veth-b netns nstest //将veth设备的一端veth-b移动到namespace nstest内

为veth接口两端分配IP地址

Ip addr add 10.0.0.1/24 dev veth-a

Ip link set dev veth-a up

Ip netns exec nstest ip addr add 10.0.0.2/24 dev veth-b

Ip netns exec nstest ip link set dev veth-b up

### 2：uts namespace

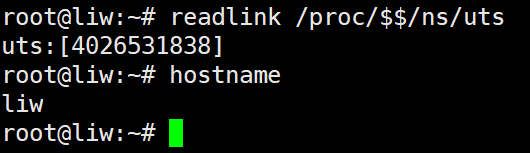
一般情况下，进程获取主机名称都是当前配置的主机名称，当启用uts namespace机制后，不同的进程将可以获取不同的主机名称，从而uts namespace将主机名进行了隔离，可以让进程感知到自己是运行在不同的主机上。

如下通过unshare命令明确如何使用uts namespace

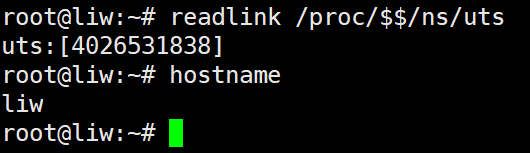
1：打开两个shell窗口，shell1和shell2

2：在两个窗口中执行hostname检查当前系统的主机名称

Shell1输出内容：

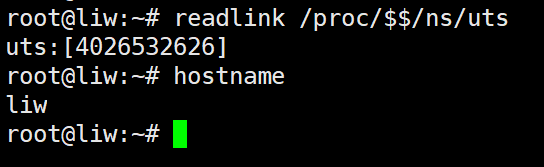


Shell2输出内容



从输出可以看到，两个shell位于同一个namespace内，因此查询到的hostname取值也是相同的。

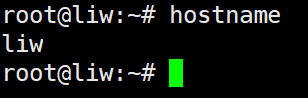
3：我们在shell2中执行unshare -u命令切换到一个新的namespace中



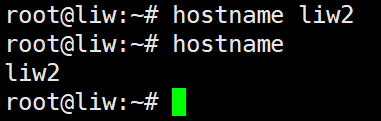
从这个输出可以看到，此时namespace已经发生变化，但hostname没有发生变化，

4：在shell2中我们用hostname liw2修改主机名称后继续查询

Shell1中的输出



Shell2中的输出：



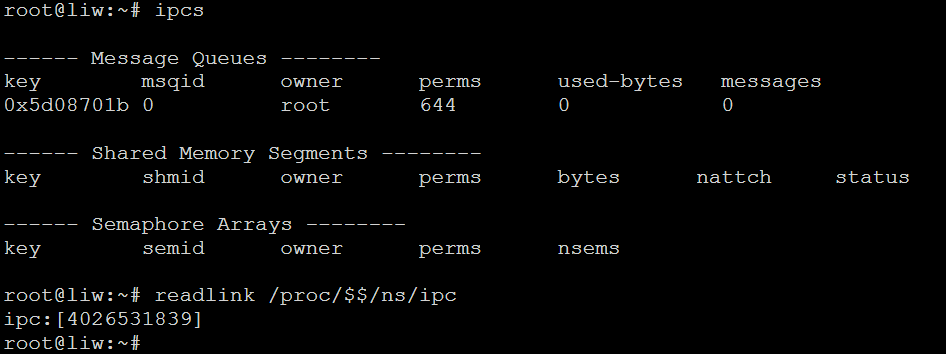
从以上输出可以看到此时shell2中的hostname已经发生了变化，但该变化并没有影响到shell1中hostname的变化。

### 3：ipc namespace

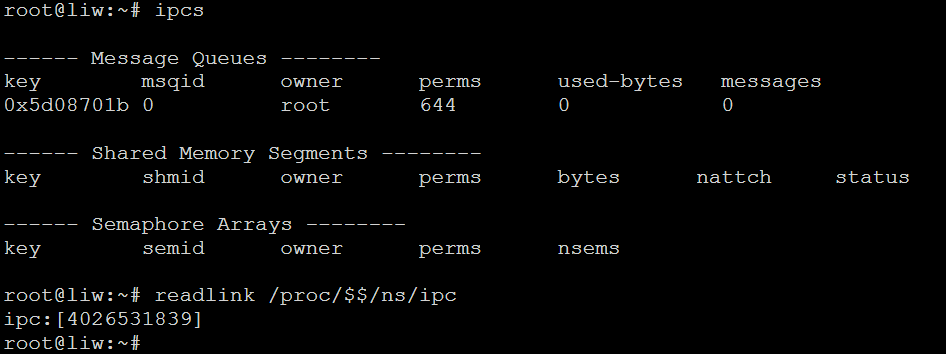
在Linux系统中，ipc用于进程之间的通信，在linux系统中，提供了管道，信号，消息队列，共享内存，套接口几类ipc通信机制。当使用了ipc namespace功能后，划分到不同namespace域下的多个进程，将无法通过以上几种ipc机制进行互相通信，主要是消息队列和共享内存将无法被使用进行不同namespace下的互通（socket的隔离会被network namespace隔离，信号会被pid namespace隔离）

仍然使用两个shell窗口举例说明

在shell 1执行如下命令创建消息队列

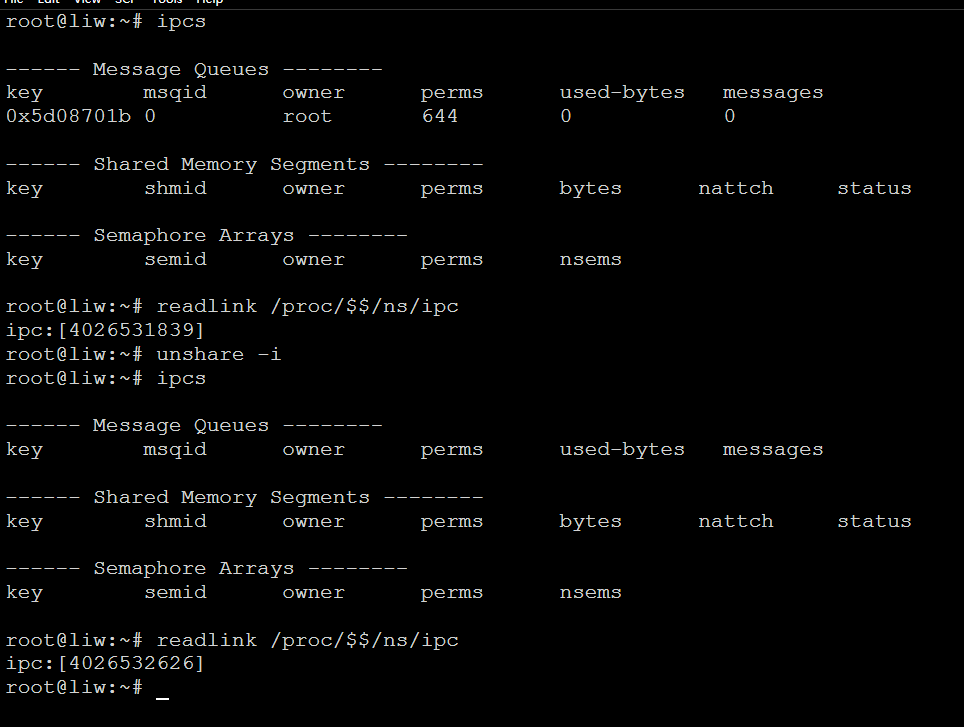


在shell2中执行ipc查询命令



可以看到默认的两个shell都可以查询到生成的消息队列，同时查询到的namespace也属于同一个namespace。

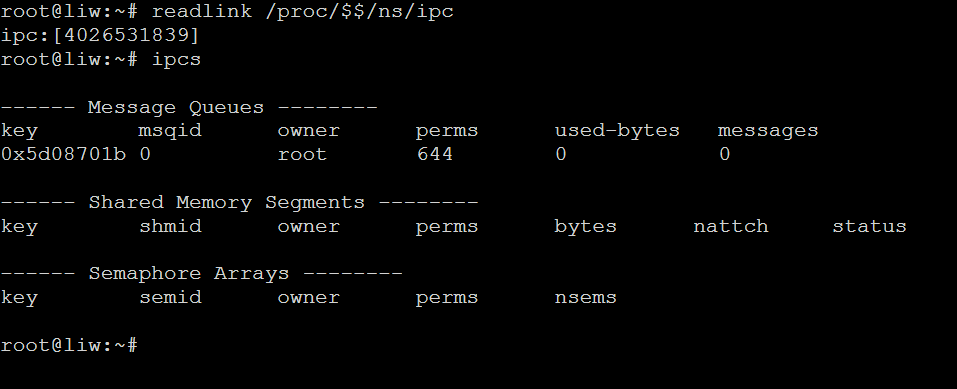
此时在shell2中执行如下命令，



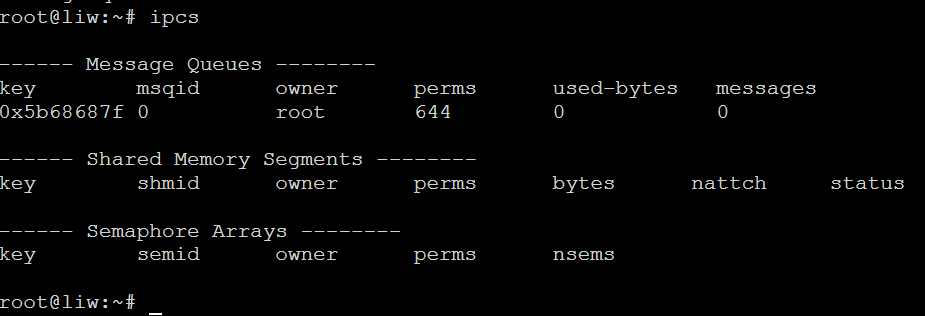
可以看到shell2中已经查询不到之前在shell1中创建的消息队列。

此时如果在shell2中创建新的消息队列，在shell1中也无法观察到该消息队列

Shell1中的内容：



Shell2中的内容：



可以看到此时存在两个完全不一样的消息队列，且互相在对方shell中无法观察到。

### 4：pid namespace

在Linux系统中，namespace机制是关联到具体进程pid的，那么pid namespace又是什么原理了。

其实在Linux系统中，会存在一个全局的pid编号，当一个进程加入到某一个pid namespace空间后，则会存在一个局部的pid编号，在该namespace空间内，看到的则是该区部pid编号。

详细原理可以参考互联网上如下总结：

<https://www.cnblogs.com/ck1020/p/5954135.html>