Linux network namespaces

СПбАУ
Власов Святослав
17 ноября 2016 г.

Содержание

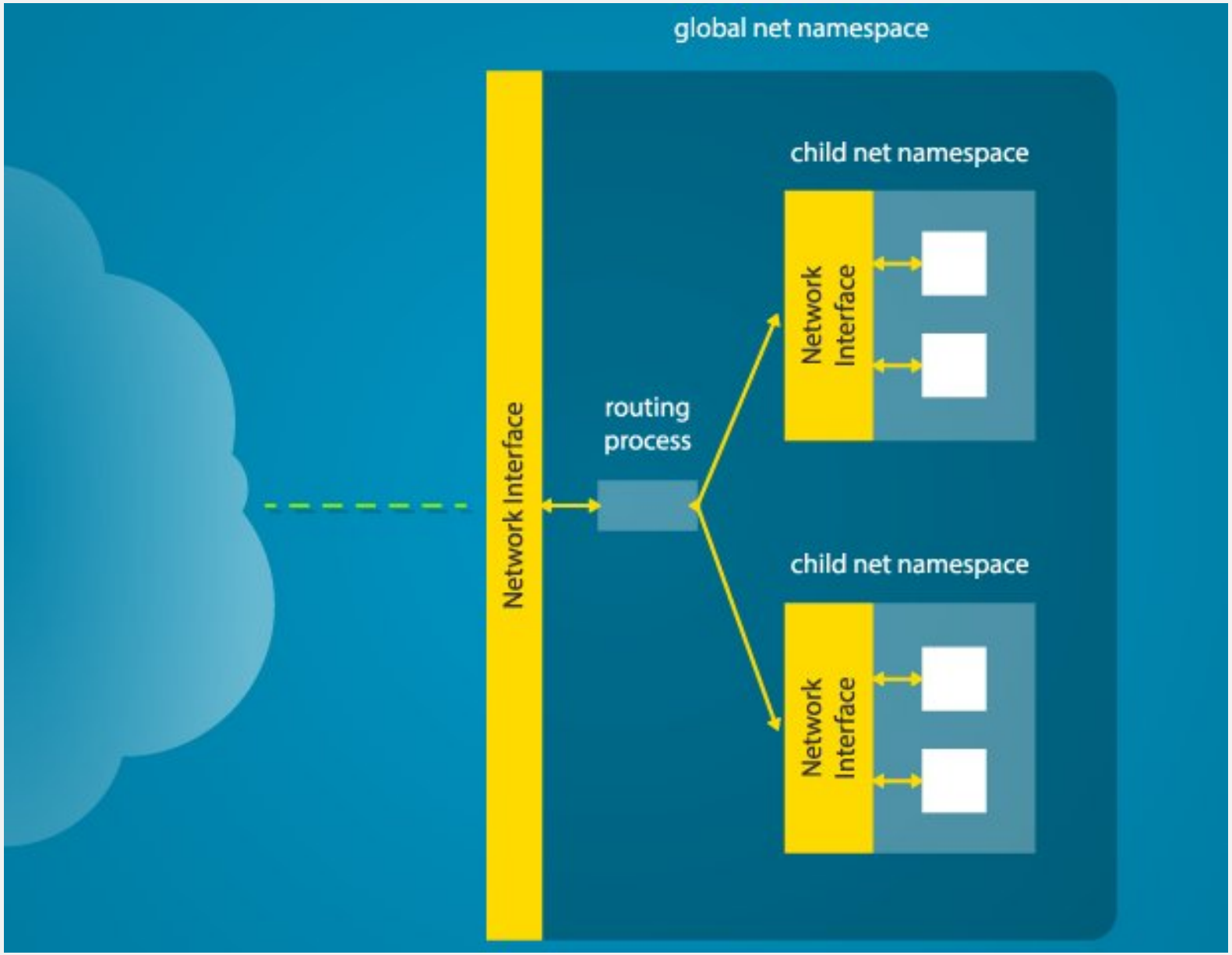
Namespaces

User	Изоляция ID пользователя, ID группы
PID	Изоляция ID процесса
UTS	Изоляция Hostname и доменного имени NIS
Network	**Изоляция сетевого стека: таблица маршрутизации, firewall rules, сетевые устройства, порты**
IPC	Изоляция объектов межпроцессного взаимодействия: семафоры (sem), очереди сообщений (msg), общая память (shm)
Mount	Изоляция точек монтирования
Cgroup	Изоляция иерархии cgroup

...

Чем является NET NS?

Network Namespace — логически это копия сетевого стека со своими правилами маршрутизации, firewall-rules и сетевыми устройствами. Позволяет внутри каждого NET NS видеть отдельный изолированный набор сетевых устройств, даже loopback.


global net namespace
child net namespace
Network Interface
routing process
Network Interface
child net namespace
Network Interface

Особенности NET NS

- Каждое сетевое устройство может принадлежать только одному сетевому неймспейсу,
- Каждый сокет может принадлежать только одному сетевому неймспейсу,
- Физические устройства принадлежат только корневому неймспейсу
- Для маршрутизации между неймспейсами используется механизм виртуальных сетевых устройств (veth)
- В отличие от других неймспейсов, сетевые неймспейсы требуют явного освобождения и могут "переживать"использующие их процессы.

Создание неймспейсов

```
# ip netns add netns1
```

Эта команда создает новый сетевой неймспейс netns1. Помимо этого также создается маунт соответствующий неймспейсу в директории `/var/run/netns`.

Команда `ip netns exec` позволяет запускать команды внутри сетевых неймспейсов:

```
# ip netns exec netns1 ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
# ip netns exec netns1 ping 127.0.0.1
connect: Network is unreachable
# ip netns exec netns1 ip link set dev lo up
# ip netns exec netns1 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=0.051 ms
```

Освобождение неймспейсов

```
# ip netns del netns1
```

- Отмонтирует `/var/run/netns/netns1`
- Неймспейс не будет освобожден, если он используется хотя бы одним процессом
- "Мигрируемые" устройства будут перемещены в сетевой неймспейс по умолчанию
- "Неперемещаемые" устройства (имеющие флаг `NETIF_F_NETNS_LOCAL` в свойствах) будут уничтожены
- Примеры неперемещаемых устройств: loopback, ppp, bridge.

Пример

```
# modprobe dummy
# ip addr add 192.168.100.199/24 brd + dev dummy0
# ip link set dev dummy0 up
# ifconfig | grep dummy
dummy0 Link encap:Ethernet HWaddr fa:62:00:cd:d0:c2
# ip link set dummy0 netns netns1
# ifconfig | grep dummy
# ip netns exec netns1 bash
# ifconfig
dummy0 Link encap:Ethernet HWaddr fa:62:00:cd:d0:c2
       inet addr:192.168.100.199 Bcast:192.168.100.255 Mask:255.255.255.0
       ...
# mkdir -p /etc/netns/netns1
# echo '192.168.100.199 dummy.com' > /etc/netns/netns1/hosts
# ping dummy.com
PING dummy.com (192.168.100.199) 56(84) bytes of data.
64 bytes from dummy.com (192.168.100.199): icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from dummy.com (192.168.100.199): icmp_seq=2 ttl=64 time=0.042 ms
```

Виртуальный сетевой интерфейс

Взаимодействовать между сетевыми неймспейсами позволяет механизм виртуальных сетевых интерфейсов (veth). Виртуальный сетевой интерфейс напоминает pipe: это пара сетевых интерфейсов связанных друг с другом и находящихся в разных неймспейсах.

Создать veth можно следующей командой:

```
# ip link add name veth0 type veth peer name veth1 netns <pid>
```

где `veth0` и `veth1` — имена сетевых интерфейсов, а `<pid>` — индентификатор процесса в дочернем сетевом немспейсе.

Под капотом

В ядре сетевой неймспейс представлен структурой `struct net` объявленной в `include/net/net_namespace.h`. В структуре `struct net_device` имеется указатель на неймспейс к которому данное сетевое устройство принадлежит.

netdevice.h

```
struct net_device {
    ...
    net* nd_net;
    ...
}
```

С помощью функций `struct net* dev_net(const struct net_device *dev)` и `void dev_net_set(struct net_device *dev, struct net *net)` можно получить доступ к немспейсу данного сетевого устройства.

Под капотом

В структуре `struct sock` также имеется поле `struct net* sk_net` а также функции `sock_net` и `sock_net_set` для доступа к неймспейсу сокета соответственно.

Все имеющиеся неймспейсы связаны в список `extern struct list_head net_namespace_list;`, а макрос `for_each_net()` позволяет проходить в цикле по ним.

Сетевой неймспейс по умолчанию представлен структурой `extern struct net init_net;` включающий в себя loopback и все физические сетевые устройства.

Под капотом

ipnetns.c

```
   /* Create the base netns directory if it doesn't exist */
mkdir(NETNS_RUN_DIR, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH|S_IXOTH);
/* Create the filesystem state */
fd = open(netns_path, O_RDONLY|O_CREAT|O_EXCL, 0);
if (fd < 0) {
...
   }
close(fd);
if (unshare(CLONE_NEWNET) < 0) {
...
   }

/* Bind the netns last so I can watch for it */
if (mount("/proc/self/ns/net", netns_path, "none", MS_BIND, NULL) < 0) {
...
   }
return 0;
```

Под капотом

```
static void __net_exit default_device_exit(struct net *net)
{
	struct net_device *dev, *aux;
	/*
	 * Push all migratable network devices back to the
	 * initial network namespace
	 */
	rtnl_lock();
	for_each_netdev_safe(net, dev, aux) {
		int err;
		char fb_name[IFNAMSIZ];

		/* Ignore unmoveable devices (i.e. loopback) */
		if (dev->features & NETIF_F_NETNS_LOCAL)
			continue;

		/* Leave virtual devices for the generic cleanup */
		if (dev->rtnl_link_ops)
			continue;

		/* Push remaining network devices to init_net */
		snprintf(fb_name, IFNAMSIZ, "dev%d", dev->ifindex);
		err = dev_change_net_namespace(dev, &init_net, fb_name);
		if (err) {
			...
		}
	}
	rtnl_unlock();
}
```