Linux namespaces

СПбАУ
Егор Горбунов
10 ноября 2016 г.

Содержание

1. Напоминание о namespaces
2. Использование UTS NS
3. Использование PID NS
4. Namespaces внутри
5. Устройство UTS NS
6. Устройство PID NS

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

...

Что позволяет UTS NS?

1. Изменять *hostname* для процесса (контейнера)
   - Идентификация контейнера
   - Используется для администрирования (логи, ...)
2. Изменять *NIS domain name* для процесса (контейнера)
   - NIS — Network Information Service — устаревшая технология, используется для централизованного хранения файлов конфигураций (на сервере) и распространения их по клиентам
   - Разные NIS domain name $\Rightarrow$ разные конфигурации

Как работать с UTS NS?

```
$ sudo unshare --uts
$ hostname leonid
$ ypdomainname some-nis-domain
$ ping leonid
```

Что позволяет PID NS?

- Создавать свежее дерево процессов (набор PIDs)
- Важно для контейнеров

```
$ unshare --fork --pid --user
$ echo $$
1
```

Многие приложения рассчитывают на то, что PID константный, `getpid()` кэширует PID $\Longrightarrow$ `unshare()`, `setns()` вступает в силу лишь для первого ребёнка (после `fork()`)

Namespaces внутри

sched.h

```
struct task_struct {
    ...
/* namespaces */
    struct nsproxy *nsproxy;
    ...
}
```

nsproxy.h

```
struct nsproxy {
    atomic_t count;
    struct uts_namespace *uts_ns;
    struct ipc_namespace *ipc_ns;
    struct mnt_namespace *mnt_ns;
    struct pid_namespace *pid_ns_for_children;
    struct net *net_ns;
    struct cgroup_namespace *cgroup_ns;
};
```

nsproxy

- copy on write
- изменять **`task->nsproxy`** можно только если **`task`** — текущая задача

nsproxy methods

```
int copy_namespaces(unsigned long flags, struct task_struct *tsk);
void exit_task_namespaces(struct task_struct *tsk);
void switch_task_namespaces(struct task_struct *tsk, struct nsproxy *new);
void free_nsproxy(struct nsproxy *ns);
int unshare_nsproxy_namespaces(unsigned long, struct nsproxy **, struct cred *, struct fs_struct *);
int __init nsproxy_cache_init(void);

static inline void put_nsproxy(struct nsproxy *ns){
      if (atomic_dec_and_test(&ns->count)) {
            free_nsproxy(ns);
      }
}

static inline void get_nsproxy(struct nsproxy *ns){
      atomic_inc(&ns->count);
}
```

Структура одного namespace

ns_common

```
struct ns_common {
   atomic_long_t stashed;
   const struct proc_ns_operations *ops;
   unsigned int inum;
};
```

proc_ns_operations

```
struct proc_ns_operations {
   const char *name;
   int type;
   struct ns_common *(*get)(struct task_struct *task);
   void (*put)(struct ns_common *ns);
   int (*install)(struct nsproxy *nsproxy, struct ns_common *ns);
};
```

namespaces в fork(), clone(), unshare()

fork(), clone()

copy_namespaces

unshare()

unshare_nsproxy_namespaces ⇒ switch_task_namespaces

exit()

exit_task_namespaces

setns()

create_new_namespaces ⇒ ns->ops->install

Устройство UTS NS

uts_namespace

```
struct uts_namespace {
   struct kref kref;
   struct new_utsname name;
   struct user_namespace *user_ns;
   struct ns_common ns;
};
```

utsname.h

```
struct new_utsname {
   char sysname[__NEW_UTS_LEN + 1];
   char nodename[__NEW_UTS_LEN + 1];
   char release[__NEW_UTS_LEN + 1];
   char version[__NEW_UTS_LEN + 1];
   char machine[__NEW_UTS_LEN + 1];
   char domainname[__NEW_UTS_LEN + 1];
};
```

Устройство PID NS

pid_namespace.h

```
struct pid_namespace {
      ...
      struct pidmap pidmap[PIDMAP_ENTRIES];
      int last_pid;
      unsigned int nr_hashed;
      struct task_struct *child_reaper;
      struct kmem_cache *pid_cachep;
      unsigned int level;
      struct pid_namespace *parent;
      struct user_namespace *user_ns;
      struct work_struct proc_work;
      kgid_t pid_gid;
      int hide_pid;
      struct ns_common ns;
      ...
};
```