

Введение в Linux namespaces

Namespace vs CGroups

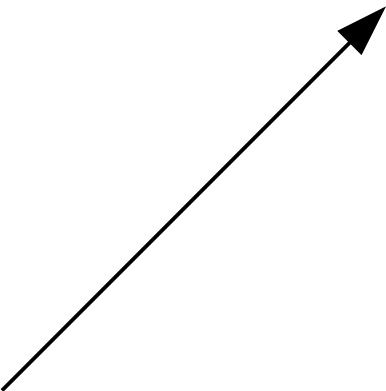
- Namespace – механизм изоляции и группировки структур данных ядра.
- Control groups – механизм изоляции ресурсов ядра

Представление процесса в linux

include/linux/sched.h, include/linux/nsproxy.h

```
1274 struct task_struct {
1275     volatile long state;    /* -1 unrunnable, 0 runnable */
1276     void *stack;
1277     atomic_t usage;
1278     unsigned int flags;    /* per process flags, */
1279     unsigned int ptrace;
1280
1372     pid_t pid;
1373     pid_t tgid;
1374
1460 /* namespaces */
1461     struct nsproxy *nsproxy;

```



```
29 struct nsproxy {
30     atomic_t count;
31     struct uts_namespace *uts_ns;
32     struct ipc_namespace *ipc_ns;
33     struct mnt_namespace *mnt_ns;
34     struct pid_namespace *pid_ns_for_children;
35     struct net *net_ns;
36 };

```

<http://lxr.free-electrons.com/source/include/linux/sched.h#L127>

3

<http://lxr.free-electrons.com/source/include/linux/nsproxy.h#L29>

API

- `clone()`
- `unshare()`
- `setns()`

clone / unshare

```
clone(..., CLONE_NEWXXX, ....);
```

VS.

```
if (fork() == 0)
    unshare(CLONE_NEWXXX);
```

Namespaces

- Mount (_NEWNS)
- UTS (_NEWUTS)
- IPC (_NEWIPC)
- PID (_NEWPID)
- user (_NEWUSER)

Mount namespace

- mount namespace – **копия** дерева файловой системы, ассоциированная с процессом
- Создание:

```
clone (..., ..., CLONE_NEWNS,...)
```

- Опции:
 - распространение событий монтирования
 - запрет перемонтирования

Флаги mount

– bind – смонтировать существующее дерево в другую точку (поддерево будет доступно в обоих местах)

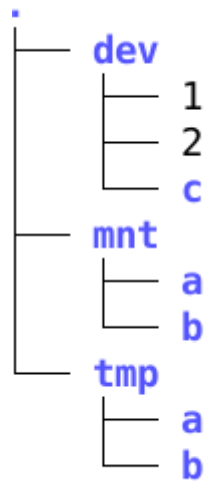
Распространение изменений:

- make-shared
- make-slave
- make-private
- make-unbindable

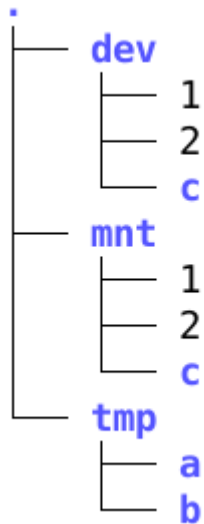
Подробнее:

[Documentation/filesystems/sharedsubtree.txt](#)

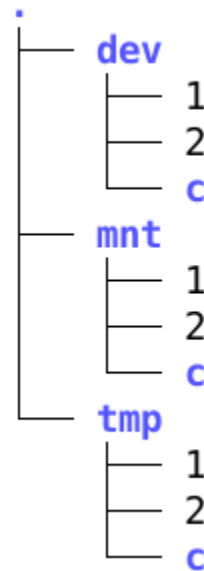
Пример: приватное монтирование



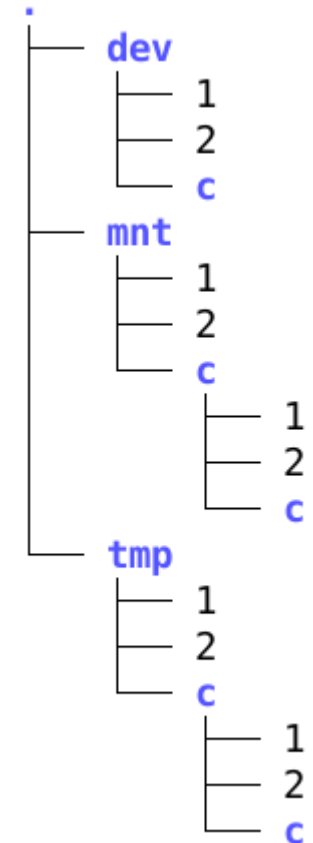
`sudo mount --bind dev mnt`



`sudo mount --make-shared mnt`
`sudo mount --bind mnt tmp`



`sudo mount --bind dev mnt/c`



UTS namespace

- Изоляция имени хоста и доменного имени

- utsname

```
struct utsname {  
    char sysname[];  
    char nodename[];  
  
    char release[];  
    char version[];  
    char machine[];  
#ifdef _GNU_SOURCE  
    char domainname[];  
#endif  
};
```

- Создание:

```
clone (..., ..., CLONE_NEWUTS,...)
```

Пример: создание UTS

см: namespaces/demo_uts_namespaces.c

```
kkv@thinkpad:/ws/bz/oslinux-seminars-2015/namespaces$ sudo ./demo_uts_namespaces newhostname
PID of child created by clone() is 6221
uts.nodename in child: newhostname
uts.nodename in parent: thinkpad
==== parent namespaces ls -l /proc/6220/ns
total 0
lrwxrwxrwx 1 root root 0 mapta 5 09:34 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 net -> net:[4026531968]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 pid -> pid:[4026531836]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 uts -> uts:[4026531838]
==== child namespaces ls -l /proc/6221/ns
total 0
lrwxrwxrwx 1 root root 0 mapta 5 09:34 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 net -> net:[4026531968]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 pid -> pid:[4026531836]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 mapta 5 09:34 uts -> uts:[4026532562]
```

■

Удержание пространства имен

```
#touch ./uts
```

```
#mount —bind /proc/6221/ns/uts ./uts
```

```
cm: namespaces/ns_exec.c
```

```
fd = open("./uts", O_RDONLY);  
if (fd == -1)  
    errExit("open");
```

```
# hostname
```

- thinkpad

```
if (setns(fd, 0) == -1)  
    errExit("setns");
```

```
#./ns_exec ./uts hostname
```

- newhostname

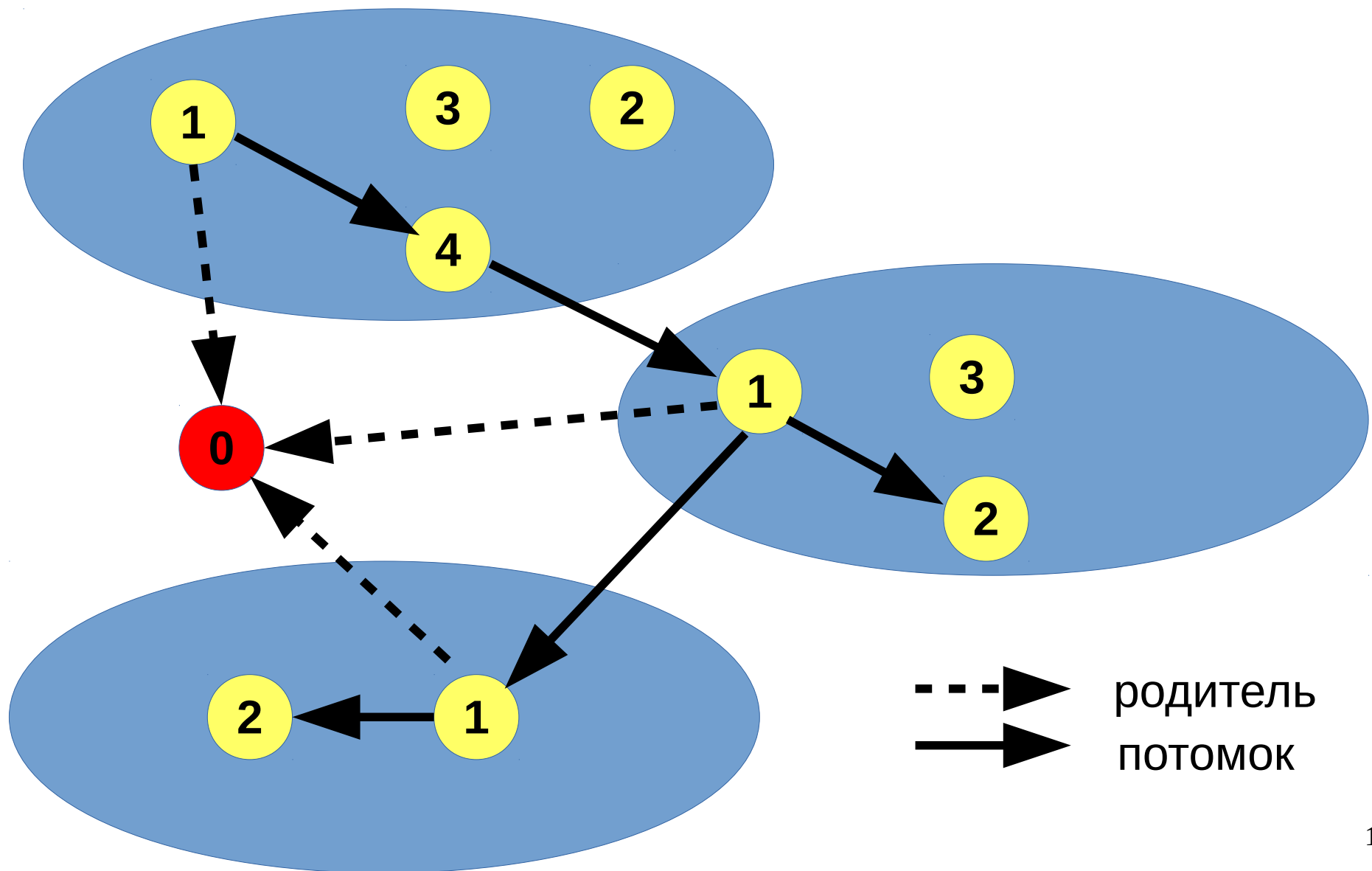
PID namespace

- Назначение: изоляция идентификаторов процессов
- Создание

```
clone (..., ..., CLONE_NEWPID, ...)
```

- Возможности:
 - миграция контейнеров с сохранением PIDs
 - имитация init-процесса
 - могут быть вложенными

Иерархия PIDNS



PIDs & TGIDs

PPID	TGID	PID	Command
0	1	1	/sbin/init
1	4300	4300	— init --user --startup-event indicator-services-sta
1	3308	3308	— /usr/sbin/cupsd -f
1	2999	2999	— /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 121:132
1	2827	2827	— /usr/lib/udisks2/udisksd --no-debug
1	2827	2844	└─ /usr/lib/udisks2/udisksd --no-debug
1	2827	2839	└─ /usr/lib/udisks2/udisksd --no-debug
1	2827	2836	└─ /usr/lib/udisks2/udisksd --no-debug
1	2827	2833	└─ /usr/lib/udisks2/udisksd --no-debug
1	2819	2819	— /usr/lib/rtkit/rtkit-daemon
1	2819	2823	└─ /usr/lib/rtkit/rtkit-daemon

Пример: PID namespace

- см: pidns_init_sleep.c

```
kkv@thinkpad:/ws/bz/oslinux-seminars-2015/namespaces$ sudo ./pidns_init_sleep newproc
```

```
PID returned by clone(): 7410
```

```
childFunc(): PID = 1
```

```
childFunc(): PPID = 0
```

```
Mounting procfs at newproc
```

```
es$ ls -la newproc/  
h file or directory
```

```
10:38 .  
10:38 ..  
10:39 1  
10:39 acpi  
10:39 asound  
10:39 buddyinfo  
10:39 bus  
10:39 cgroups
```

```
kkv@thinkpad:/ws/bz/oslinux-seminars-2015/namespaces$
```

```
[sudo] password for kkv:
```

```
Caller PID = 7597, levels = 5
```

```
Mounting procfs at ./multi_proc4
```

```
Mounting procfs at ./multi_proc3
```

```
Mounting procfs at ./multi_proc2
```

```
Mounting procfs at ./multi_proc1
```

```
Mounting procfs at ./multi_proc0
```

```
Final child sleeping
```

```
└
```


IPC namespace

- Изоляция ресурсов IPC (System V IPC) – очередей сообщений, разделяемая память...
- Создание:

```
clone (..., ..., CLONE_NEWIPC, ...)
```

USER namespace

- Изоляция идентификаторов пользователей и групп
- Создание:

```
clone (..., ..., CLONE_NEWUSER,...)
```

- Возможности:
 - Предоставление привилегированных операций непривилегированному пользователю внутри пространства имен
 - Отображение пользователей и групп
- `/proc/sys/kernel/overflowuid`

Пример: user namespace

- см: demo_users.c

```
kkv@thinkpad:/ws/bz/oslinux-seminars-2015/namespaces$ ./demo_users
eUID = 1000; eGID = 1000; PPID=7941; PID=7941 capabilities: =
eUID = 65534; eGID = 65534; PPID=7941; PID=7942 capabilities: = cap_chown,
verridge,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_se
setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_a
net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chrc
s_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resol
ys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_cor
setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_s
```

```
./_patch_user_maps 7942
```

```
echo '0 1000 1' >/proc/$1/uid_map
echo '0 1000 1' >/proc/$1/gid_map
```

```
eUID = 0; eGID = 0; PPID=7941; PID=7942 capabilities: = cap_chown
cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_s
cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_
cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chr
,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resc
cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_cc
```

Network namespace

- Изоляция сетевой конфигурации, интерфейсов, правил маршрутизации
- Создание: `clone (..., ..., CLONE_NEWNET, ...)`

Примеры (через ip):

- `ip netns add netns1`
- `ip netns exec netns1 ip link list`
- `ip netns delete netns1`
- `ip netns exec netns1 ip link set dev lo up`
- `ip netns exec netns1 ping 127.0.0.1`

Для чтения

- <http://www.ibm.com/developerworks/library/l-month-namespaces/>
- <http://lwn.net/Articles/531114/>