## Демо 1

### Повторяем пример из лекции:

|  |
| --- |
| mkdir testfolder  mkdir testfolder/bin  cp /bin/bash testfolder/bin  ldd /bin/bash  mkdir testfolder/lib  mkdir testfolder/lib64  cp /lib/x86\_64-linux-gnu/libtinfo.so.6 testfolder/lib  cp /lib/x86\_64-linux-gnu/libc.so.6 testfolder/lib  cp /lib64/ld-linux-x86-64.so.2 testfolder/lib64/  chroot testfolder |

## Задача 1

### Задание

* Продолжаем работу в командной строке. В частности - в той же папке.
* Самостоятельно необходимо добавить в измененный корень команду, которая могла бы отображать путь к папке - команда ls. Вместе с зависимостями, чтобы она запускалась внутри “измененного корня”.
* Необходимо добавить самостоятельно набор команд для вывода “корня”, чтобы убедиться, что мы действительно изолировали среду.

|  |
| --- |
| mkdir testfolder  mkdir testfolder/bin  cp /bin/bash testfolder/bin  ldd /bin/bash  mkdir testfolder/lib  mkdir testfolder/lib64  cp /lib/x86\_64-linux-gnu/libtinfo.so.6 testfolder/lib  cp /lib/x86\_64-linux-gnu/libc.so.6 testfolder/lib  cp /lib64/ld-linux-x86-64.so.2 testfolder/lib64/  #new addition  cp /bin/ls testfolder/bin/  ldd /bin/ls  cp /lib/x86\_64-linux-gnu/libselinux.so.1 testfolder/lib/  cp /lib/x86\_64-linux-gnu/libpcre2-8.so.0 testfolder/lib/  chroot testfolder/  ls / |

## Демо 2

### Задание

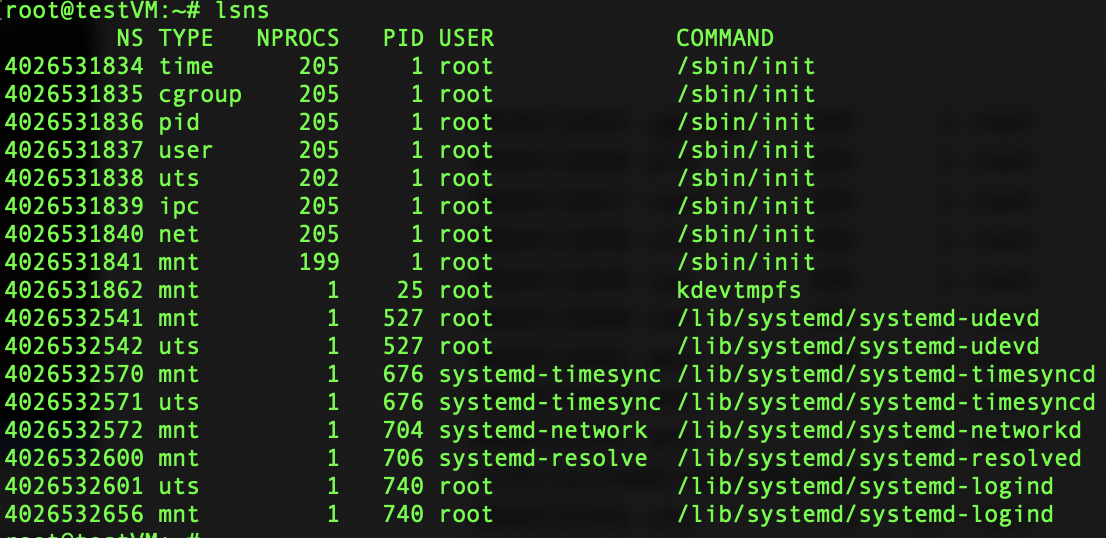
Сейчас предлагается рассмотреть более показательный пример, ради которого, по сути, и затевалась большая часть первой лекции. Для этого нам необходимо воспользоваться утилитой unshare, которая позволяет изолировать процесс от исходного пространства имен. При этом дочерний процесс изолирует сам себя, а не родительский изолирует дочерний процесс.

|  |
| --- |
| ls -l /proc/＄＄/ns  lrwxrwxrwx 1 root root 0 Nov 16 11:50 cgroup -> 'cgroup:[4026531835]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 ipc -> 'ipc:[4026531839]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 mnt -> 'mnt:[4026531841]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 net -> 'net:[4026531840]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 pid -> 'pid:[4026531836]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 pid\_for\_children -> 'pid:[4026531836]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 time -> 'time:[4026531834]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 time\_for\_children -> 'time:[4026531834]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 user -> 'user:[4026531837]'  lrwxrwxrwx 1 root root 0 Nov 16 11:50 uts -> 'uts:[4026531838]' |

Как можно видеть, наша сессия принадлежит ряду групп. Об этом и свидетельствует вывод консоли. Напомню ряд важных пространств имен:

* net - сеть
* pid - разграничение процессов
* user - маппинг юзеров
* uts - позволяет менять hostname

Все пространства имен можно просмотреть командой:



Собственно, теперь вы обладаете достаточной информацией для того, чтобы запустить приложение изолированно.

## Демо 3

ПРОДОЛЖАЕМ!

Прежде чем дать следующее задание, необходимо дать подсказку со словами:

Да, чуть не забыл, прежде чем вы дальше будете выполнять задачи, вам стоит знать о том, что утилита unshare имеет ряд флагов, доступных для выполнения. Полную инструкцию по использованию вы можете запросить, вызвав **man unshare**, но основные хотелось бы также озвучить:

--pid

--net

--fork

--mount-proc

Каждый из флагов соответствует пространству имен, которое было изучено ранее.

Рассмотрим простой пример вместе, затем каждый выполнит аналогичные действия самостоятельно.

Давайте поработает, например, с сетевым пространством имен. Изначально в системе нет никаких ns:

|  |
| --- |
| ip netns list |

Давайте создадим:

|  |
| --- |
| ip netns add testns  ip netns list |

Вы могли заметить, что list возвращал только наш вновь созданный namespace. Разве он не должен возвращать по крайней мере два, одним из которых был бы исходным namespace, о котором мы упоминали ранее? Причина этого в том, что ip создаёт то, что называется named network namespace, который является просто network namespace, идентифицируемый уникальным именем (в нашем случае testns). Только именованные пространства имён network отображаются подкомандой list, а изначальный network namespace неименованный.

Проще всего получить именованные пространства имён network. Например, в каждом именованном network namespace создаётся файл в каталоге /var/run/netns и им сможет воспользоваться процесс, который хочет переключиться на свой namespace. Другое свойство именованных пространств имён network заключается в том, что они могут существовать без наличия какого-либо процесса — в отличие от неименованных, которые будут удалены как только завершатся все принадлежащие им процессы.

Теперь же давайте запустим командную оболочку с изоляцией сетевых интерфейсов:

|  |
| --- |
| ip netns exec testns123 bash |

При вводе ifconfig мы можем видеть, что у нас есть только 1 интерфейс, находящийся в состоянии DOWN. В этом случае, разумеется, ничего не сделать (показать, что пинг не работает).

Мы должны теперь свыкнуться с тем, что настройки по умолчанию для пространств имён обычно очень строгие. Для пространств имён network, как мы видим, никаких устройств, помимо loopback, не будет доступно. Мы можем поднять интерфейс loopback без всяких формальностей:

|  |
| --- |
| root@testVM:~# ip link set dev lo up  root@testVM:~# ip a  1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  inet 127.0.0.1/8 scope host lo  valid\_lft forever preferred\_lft forever  inet6 ::1/128 scope host  valid\_lft forever preferred\_lft forever |

Теперь, когда разобрались с lo-интерфейсом, добавим виртуальный интерфейс для дальнейшего взаимодействия. Мы будем использовать сетевое устройство virtual ethernet (или сокращённо veth). Устройства veth всегда создаются как пара устройств, связанных по принципу туннеля, так что сообщения, отправленные на одном конце, выходят из устройства на другом. Вы могли бы предположить, что мы могли бы легко иметь один конец в исходном network namespace, а другой — в нашем дочернем network namespace, а всё общение между пространствами имён network проходило бы через соответствующее оконечное устройство veth (и вы были бы правы).

Выполняем команды в исходном пространстве:

|  |
| --- |
| ip link add veth0 type veth peer name veth1  ip link set veth1 netns testns123  ip a  ip addr add 10.0.0.1/24 dev veth0  ip link set dev veth0 up  ip a |

В новом пространстве:

|  |
| --- |
| ip addr add 10.0.0.2/24 dev veth1  ip link set dev veth1 up |

## Задача 2

### Задание

Нужно создать новое сетевое пространство имен. И убедиться, что это получилось, проверив командой if a

### Пример решения

Это примерный вариант решения, главное – чтобы логика была верной.

|  |
| --- |
| root@testVM:~# sudo unshare --net /bin/bash  root@testVM:/home/lvl001# ip a  1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 |

## Задача 3

### Задание

Нужно найти правильный вариант решения изоляции и дать аргументацию.

### Пример решения

Это примерный вариант решения, главное – чтобы логика была верной.

|  |
| --- |
| root@testVM:~# sudo unshare --pid --net --fork --mount-proc /bin/bash  root@testVM:~# ps -aux  USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND  root 1 0.0 0.2 5044 4124 pts/2 S 13:03 0:00 /bin/bash  root 8 0.0 0.1 7744 3664 pts/2 R+ 13:03 0:00 ps -aux |