Lab 4 — create simple container

Intro

- The idea of the work is to `clone()` process with flags enabling the separate namespaces for it, etc., to prepare rootfs image for this process to `chroot` into, to configure `cgroups`.

Features

- Rootfs is based on Ubuntu 20.04 base image. Sysbench is added to the container filesystem image through `init_container.sh`. On startup, `bash` shell is invoked.
- Container is created with its own namespaces:
  1. PID namespace (`CLONE_NEWPID`): The new process will have its own PID namespace. Processes in this namespace can only see the processes within the same namespace. The first process in this namespace is usually the init process, with a PID of 1.
  2. UTS namespace (`CLONE_NEWUTS`): The new process will have its own UTS namespace, which includes the hostname and domain name. This allows a process to have a different hostname inside and outside the namespace.
  3. Network namespace (`CLONE_NEWNET`): The new process will have its own network namespace. We create namespace within Bash script using `ip netns add` and further set it up. This means that it will have its own set of network interfaces, IP addresses, routing tables, and firewall rules, independent of the host and other processes. A pair of virtual interfaces is created: `veth_host` and `veth_container`, and assigned IP addresses 192.168.10.1 and 192.168.10.2, accordingly. The `container` binary simply joins existing namespace `container_network_ns`.
  4. Mount namespace (`CLONE_NEWNS`): The new process will have its own mount namespace. This means that it will have its own filesystem root directory and its own set of mount points, independent of the host and other processes.

```
artem@latitude:~/OneDrive/Inno/S23/TVL/container_lab (master) $ sudo ./build/container
[sudo] password for artem:
root@container:/# pwd
/
root@container:/# ps
    PID TTY          TIME CMD
      1 ?        00:00:00 container
      2 ?        00:00:00 bash
      5 ?        00:00:00 ps
root@container:/# ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
root@container:/#
```

Filesystem isolation

```
/ # ls
bin         lib         proc        sbin        usr
dev         lost+found  root        sys         var
etc         opt         run         tmp
/ # ls
bin         lib         proc        sbin        usr
dev         lost+found  root        sys         var
etc         opt         run         tmp
/ # ls
bin         lib         proc        sbin        usr
dev         lost+found  root        sys         var
etc         opt         run         tmp
```

```
root@latitude:/home/artem $ cd /
root@latitude:/ $ ls
bin   home            lib32       media  root  srv  var
boot  initrd.img      lib64       mnt    run   sys  vmlinuz
dev   initrd.img.old  libx32      opt    sbin  tmp  vmlinuz.old
etc   lib            lost+found  proc   snap  usr
root@latitude:/ $ touch nocontainer.txt
root@latitude:/ $ ls
bin   home            lib32       media             proc  snap  usr
boot  initrd.img      lib64       mnt               root  srv   var
dev   initrd.img.old  libx32      nocontainer.txt   run   sys   vmlinuz
etc   lib            lost+found  opt               sbin  tmp   vmlinuz.old
```

```
etc         opt         run         tmp
/ # touch nohost.txt
/ # ls
bin         lib         opt         run         tmp
dev         lost+found  proc        sbin        usr
etc         nohost.txt  root        sys         var
/ # ls
bin         lib         opt         run         tmp
dev         lost+found  proc        sbin        usr
etc         nohost.txt  root        sys         var
/ #
```

```
root@latitude:/ $ ls
bin   home            lib32       media             proc  snap  usr
boot  initrd.img      lib64       mnt               root  srv   var
dev   initrd.img.old  libx32      nocontainer.txt   run   sys   vmlinuz
etc   lib            lost+found  opt               sbin  tmp   vmlinuz.old
root@latitude:/ $ ls
bin   home            lib32       media             proc  snap  usr
boot  initrd.img      lib64       mnt               root  srv   var
dev   initrd.img.old  libx32      nocontainer.txt   run   sys   vmlinuz
etc   lib            lost+found  opt               sbin  tmp   vmlinuz.old
root@latitude:/ $
```

PID isolation

```
/ # ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 cmake-build-debug/container
    2 root      0:00 /bin/sh
   11 root      0:00 ps aux
/ #
```

```
root@latitude:/ $ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 166032 11356 ?        Ss   мая17   0:05 /sbin/
root           2  0.0  0.0      0     0 ?        S    мая17   0:00 [kthre
root           3  0.0  0.0      0     0 ?        I<   мая17   0:00 [rcu_g
root           4  0.0  0.0      0     0 ?        I<   мая17   0:00 [rcu_p
root           6  0.0  0.0      0     0 ?        I<   мая17   0:00 [kwork
root           8  0.0  0.0      0     0 ?        I<   мая17   0:00 [mm_pe
root           9  0.0  0.0      0     0 ?        S    мая17   0:00 [rcu_t
root          10  0.0  0.0      0     0 ?        S    мая17   0:00 [rcu_t
root          11  0.0  0.0      0     0 ?        S    мая17   0:01 [ksoft
root          12  0.0  0.0      0     0 ?        I    мая17   0:41 [rcu_s
root          13  0.0  0.0      0     0 ?        S    мая17   0:00 [migra
root          15  0.0  0.0      0     0 ?        S    мая17   0:00 [cpuhp
```

Network isolation (and communication)

```
/ # ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
37: veth_container@if38: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 q
disc noqueue state UP qlen 1000
    link/ether e6:dd:e0:f3:5a:5b brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.2/24 scope global veth_container
       valid_lft forever preferred_lft forever
    inet6 fe80::e4dd:e0ff:fef3:5a5b/64 scope link
       valid_lft forever preferred_lft forever
/ # ping 192.168.10.1 -c 1
PING 192.168.10.1 (192.168.10.1): 56 data bytes
64 bytes from 192.168.10.1: seq=0 ttl=64 time=0.208 ms

--- 192.168.10.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.208/0.208/0.208 ms
/ #
```

```
30: vpn: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1280 qdisc noqueue state UNKNOW
N group default qlen 1000
    link/none
    inet 10.242.104.10/24 scope global vpn
       valid_lft forever preferred_lft forever
36: vethdb728cf@if35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noque
ue master docker0 state UP group default
    link/ether de:f2:b7:a3:25:b9 brd ff:ff:ff:ff:ff:ff link-netnsid 1
38: veth_host@if37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
 state UP group default qlen 1000
    link/ether 92:74:1a:b1:95:71 brd ff:ff:ff:ff:ff:ff link-netns container_
network_ns
    inet 192.168.10.1/24 scope global veth_host
       valid_lft forever preferred_lft forever
root@latitude:/ $ ping 192.168.10.2 -c 1
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=64 time=0.104 ms

--- 192.168.10.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.104/0.104/0.104/0.000 ms
```

```
container:/# telnet 192.168.10.1 2222
Trying 192.168.10.1...
Connected to 192.168.10.1.
Escape character is '^]'.
hello from container
```

```
8: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state U
P group default
    link/ether 02:42:67:0b:e0:c8 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
30: vpn: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1280 qdisc noqueue state UNKNOW
N group default qlen 1000
    link/none
    inet 10.242.104.10/24 scope global vpn
       valid_lft forever preferred_lft forever
36: vethdb728cf@if35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noque
ue master docker0 state UP group default
    link/ether de:f2:b7:a3:25:b9 brd ff:ff:ff:ff:ff:ff link-netnsid 1
42: veth_host@if41: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
 state UP group default qlen 1000
    link/ether 92:74:1a:b1:95:71 brd ff:ff:ff:ff:ff:ff link-netns container_
network_ns
    inet 192.168.10.1/24 scope global veth_host
       valid_lft forever preferred_lft forever
root@latitude:/ $ nc -l -s 192.168.10.1 -p 2222
hello from container
```

Links

- This project on Github: https://github.com/ar7ch/lab4tv

```
root@container:/# sysbench cpu --cpu-max-prime=20000 run
sysbench 1.1.0-4228c85 (using bundled LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   270.65

Throughput:
    events/s (eps):                      270.6543
    time elapsed:                        10.0017s
    total number of events:              2707

Latency (ms):
         min:                                    3.56
         avg:                                    3.69
         max:                                   16.29
         95th percentile:                        3.89
         sum:                                 9998.66

Threads fairness:
    events (avg/stddev):           2707.0000/0.00
    execution time (avg/stddev):   9.9987/0.00

root@container:/#
```

My container

```
root@be0eecb345b1:/mnt/shared# src/sysbench cpu --cpu-max-prime=20000 run
sysbench 1.1.0-4228c85 (using bundled LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   265.57

Throughput:
    events/s (eps):                      265.5731
    time elapsed:                        10.0010s
    total number of events:              2656

Latency (ms):
         min:                                    3.55
         avg:                                    3.76
         max:                                   16.45
         95th percentile:                        4.33
         sum:                                 9996.82

Threads fairness:
    events (avg/stddev):           2656.0000/0.00
    execution time (avg/stddev):   9.9968/0.00
```

Ubuntu Docker

```
artem@latitude:~/OneDrive/Inno/S23/TVL/container_lab (master) $ sysben
ch cpu --cpu-max-prime=20000 run
sysbench 1.1.0-4228c85 (using bundled LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   270.07

Throughput:
    events/s (eps):                      270.0728
    time elapsed:                        10.0010s
    total number of events:              2701

Latency (ms):
         min:                                    3.56
         avg:                                    3.70
         max:                                   17.45
         95th percentile:                        3.89
         sum:                                 9998.16

Threads fairness:
    events (avg/stddev):           2701.0000/0.00
    execution time (avg/stddev):   9.9982/0.00
```

Host

Table with metrics

	command executed	my container	Docker (ubuntu 22.04)	host machine
CPU total time	`sysbench cpu --cpu-max-prime=20000 run`	9.9899 s	9.9968 s	9.9982 s
File IO write	`sysbench fileio --file-total-size=1G --file-num=128 --file-test-mode=seqwr run`	1073741824 bytes written in 6.60 seconds (155.11 MiB/sec).	1073741824 bytes written in 5.06 seconds (202.51 MiB/sec).	1073741824 bytes written in 7.89 seconds (129.82 MiB/sec).
File IO read	`sysbench fileio --file-total-size=1G --file-num=128 --file-test-mode=seqrd run`	IOPS=308201.48 4815.65 MiB/s (5049.57 MB/s)	IOPS=323803.01 5059.42 MiB/s (5305.19 MB/s)	IOPS=324396.52 5068.70 MiB/s (5314.91 MB/s)
Memory access	`sysbench memory --memory-block-size=1K --memory-total-size=4G run`	0.4148 s	0.4163 s	0.4164

Sources

1. https://man7.org/linux/man-pages/man7/namespaces.7.html
2. https://cesarvr.io/post/2018-05-22-create-containers/
3. https://github.com/akopytov/sysbench#general-syntax
4. https://docs.docker.com/storage/storagedriver/
5. https://man7.org/linux/man-pages/man8/ip-netns.8.html