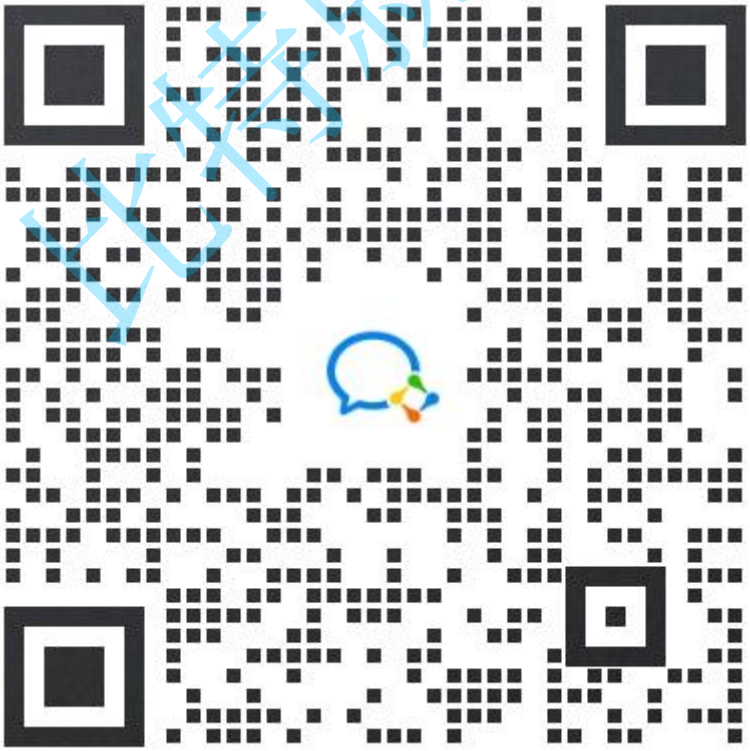
空间隔离实战

版权说明

对比特项目感兴趣，可以联系这个微信。



NameSpace 隔离实战

实战目的

了解隔离能力并不是 docker 提供的，而是操作系统内核提供基本能力。

基础知识

dd 命令详解

Linux dd 命令用于读取、转换并输出数据。

dd 可从标准输入或文件中读取数据，根据指定的格式来转换数据，再输出到文件、设备或标准输出。

- 语法

```Shell
dd OPTION
```

- 参数
  - if=文件名：输入文件名，默认为标准输入。即指定源文件。
  - of=文件名：输出文件名，默认为标准输出。即指定目的文件。
  - ibs=bytes：一次读入 bytes 个字节，即指定一个块大小为 bytes 个字节。
    obs=bytes：一次输出 bytes 个字节，即指定一个块大小为 bytes 个字节。
    bs=bytes：同时设置读入/输出的块大小为 bytes 个字节。
  - cbs=bytes：一次转换 bytes 个字节，即指定转换缓冲区大小。
  - skip=blocks：从输入文件开头跳过 blocks 个块后再开始复制。
  - seek=blocks：从输出文件开头跳过 blocks 个块后再开始复制。
  - count=blocks：仅拷贝 blocks 个块，块大小等于 ibs 指定的字节数。
  - conv=<关键字>，关键字可以有以下 11 种：
    - conversion：用指定的参数转换文件。
    - ascii：转换 ebcdic 为 ascii
    - ebcdic：转换 ascii 为 ebcdic
    - ibm：转换 ascii 为 alternate ebcdic
    - block：把每一行转换为长度为 cbs，不足部分用空格填充
    - unblock：使每一行的长度都为 cbs，不足部分用空格填充

        - lcase：把大写字符转换为小写字符
        - ucase：把小写字符转换为大写字符
        - swap：交换输入的每对字节
        - noerror：出错时不停止
        - notrunc：不截短输出文件
        - sync：将每个输入块填充到 ibs 个字节，不足部分用空（NUL）字符补齐。
    - --help：显示帮助信息
    - --version：显示版本信息
- 案例

```
Shell
# 生成 1 个镜像文件
dd if=/dev/zero of=fdimage.img bs=8k count=10240

#将 testfile 文件中的所有英文字母转换为大写，然后转成为 testfile_1 文件
dd if=testfile_2 of=testfile_1 conv=ucase
```

mkfs 命令详解

用于在设备上创建 Linux 文件系统,俗称格式化，比如我们使用 U 盘的时候可以格式化。

- 语法

```
Shell
mkfs [-V] [-t fstype] [fs-options] filesys [blocks]
```

- 参数

```
Shell
-t fstype：指定要建立何种文件系统；如 ext3，ext4
filesys ：指定要创建的文件系统对应的设备文件名；
blocks：指定文件系统的磁盘块数。
-V ：详细显示模式
fs-options:传递给具体的文件系统的参数
```

- 实例

```
Shell
#将 sda6 分区格式化为 ext4 格式
```

```
mkfs -t ext4 /dev/sda6
#格式化镜像文件为 ext4
mkfs -t ext4 ./fdimage.img
```

df 命令详解

Linux df（英文全拼：disk free） 命令用于显示目前在 Linux 系统上的文件系统磁盘使用情况统计。

- 语法

```
Shell
df [OPTION]... [FILE]...
```

- 常见参数
  - -a, --all 包含所有的具有 0 Blocks 的文件系统
  - -h, --human-readable 使用人类可读的格式(预设值是不加这个选项的...)
  - -H, --si 很像 -h, 但是用 1000 为单位而不是用 1024
  - -t, --type=TYPE 限制列出文件系统的 TYPE
  - -T, --print-type 显示文件系统的形式
- 案例

```
Shell
#查看磁盘使用情况
df -h
#查看磁盘的系统类型
df -Th
```

mount 命令详解

mount 命令用于加载文件系统到指定的加载点。此命令的也常用于挂载光盘，使我们可以访问光盘中的数据，因为你将光盘插入光驱中，Linux 并不会自动挂载，必须使用 Linux mount 命令来手动完成挂载。

Linux 系统下不同目录可以挂载不同分区和磁盘设备，它的目录和磁盘分区是分离的，可以自由组合(通过挂载)

不同的目录数据可以跨越不同的磁盘分区或者不同的磁盘设备。

挂载的实质是为磁盘添加入口（挂载点）。

- mount 常见用法

```
Shell
mount [-l]
mount [-t vfstype] [-o options] device dir
```

- 常见参数

-l：显示已加载的文件系统列表;

-t: 加载文件系统类型支持常见系统类型的 ext3,ext4,iso9660,tmpfs,xfs 等,大部分情况可以不指定，mount 可以自己识别

-o options 主要用来描述设备或档案的挂接方式。

loop：用来把一个文件当成硬盘分区挂接上系统

ro：采用只读方式挂接设备

rw：采用读写方式挂接设备

device: 要挂接(mount)的设备。

dir: 挂载点的目录

- 案例

```
Shell
#将 /dev/hda1 挂在 /mnt 之下。
mount /dev/hda1 /mnt

#将镜像挂载到/mnt/testext4 下面,需要确保挂载点也就是目录存在
mkdir -p /mnt/testext4
mount ./fdimage.img /mnt/testext4
```

unshare 命令详解

unshare 主要能力是使用与父程序不共享的名称空间运行程序。

- 语法

```
Shell
unshare [options] program [arguments]
```

- 常用参数

参数	含义

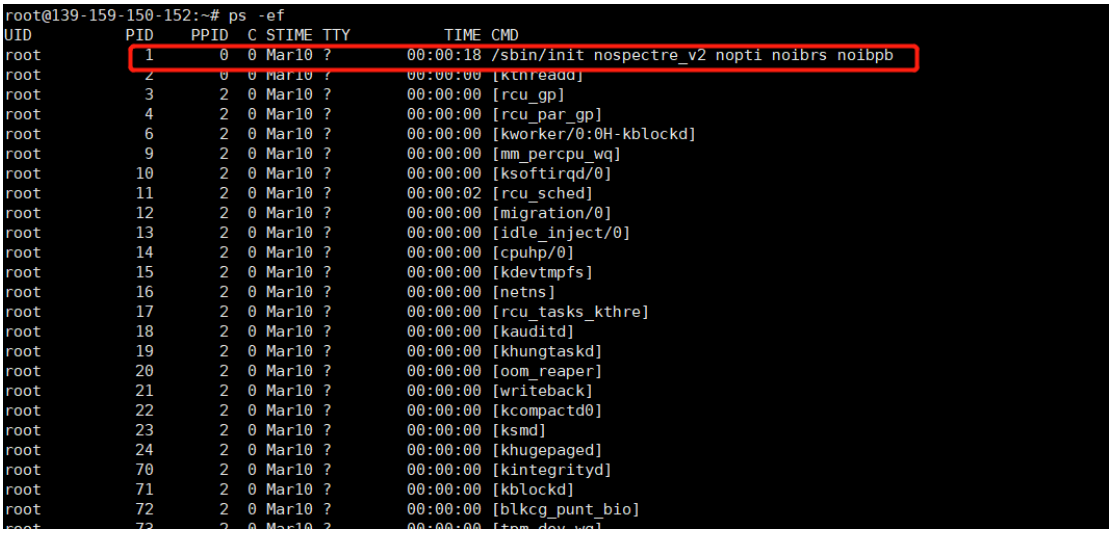
-i, --ipc	不共享 IPC 空间
-m, --mount	不共享 Mount 空间
-n, --net	不共享 Net 空间
-p, --pid	不共享 PID 空间
-u, --uts	不共享 UTS 空间
-U, --user	不共享用户
-V, --version	版本查看
--fork	执行 unshare 的进程 fork 一个新的子进程，在子进程里执行 unshare 传入的参数。
--mount-proc	执行子进程前，将 proc 优先挂载过去

- 案例

```
Shell
#hostname 隔离
root@139-159-150-152:~# unshare -u /bin/bash
root@139-159-150-152:~# hostname test1
root@139-159-150-152:~# hostname
test1
root@139-159-150-152:~# exit
exit
root@139-159-150-152:~# hostname
139-159-150-152
root@139-159-150-152:
```

实战操作一（PID 隔离）

1. 在主机上执行 ps -ef,可以看到进程列表如下，其中启动进程 PID 1 为 init 进程



2. 我们打开另外一个 shell，执行下面命令创建一个 bash 进程，并且新建一个 PID Namespace：

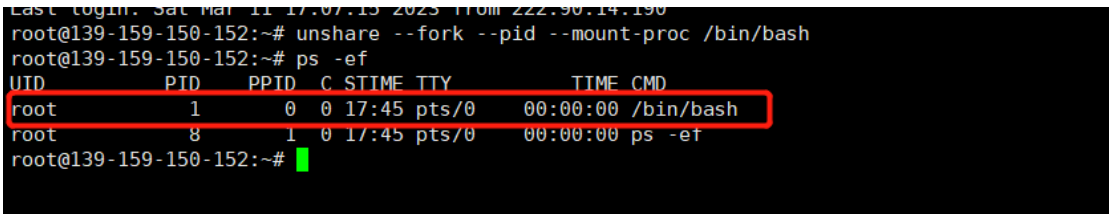
--fork 新建了一个 bash 进程，是因为如果不建新进程，新的 namespace 会用 unshare 的 PID 作为新的空间的父进程，而这个 unshare 进程并不在新的 namespace 中，所以会报个错 `Cannot allocate memory`

--pid 表示我们的进程隔离的是 pid，而其他命名空间没有隔离

mount-proc 是因为 Linux 下的每个进程都有一个对应的 /proc/PID 目录，该目录包含了大量的有关当前进程的信息。 对一个 PID namespace 而言，/proc 目录只包含当前 namespace 和它所有子孙后代 namespace 里的进程的信息。创建一个新的 PID namespace 后，如果想让子进程中的 top、ps 等依赖 /proc 文件系统的命令工作，还需要挂载 /proc 文件系统。而文件系统隔离是 mount namespace 管理的，所以 linux 特意提供了一个选项--mount-proc 来解决这个问题。如果不带这个我们看到的进程还是系统的进程信息。

```
Shell
unshare --fork --pid --mount-proc /bin/bash
```

3. 执行 ps -ef 查看进程信息，我们可以看到此时进程空间内的内容已经变了，而且启动进程也变成了我们的 bash 进程。说明我们已经看不到主机上的进程空间了，我们的进程空间发生了隔离。



4.执行 exit 退出进程

```
Shell
exit
```

实战操作二（Mount 隔离）

1. 打开第一个 shell 窗口 A，执行命令， df -h ,查看主机默认命名空间的磁盘挂载情况

```
Shell
root@139-159-150-152:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            948M     0  948M   0% /dev
tmpfs           199M  1.1M  198M   1% /run
/dev/vda1        40G  8.0G   30G  22% /
tmpfs           992M     0  992M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           992M     0  992M   0% /sys/fs/cgroup
tmpfs           199M     0  199M   0% /run/user/0
/dev/loop0       50M   50M     0 100% /snap/snapd/18357
/dev/loop1       56M   56M     0 100% /snap/core18/2697
/dev/loop2       55M   55M     0 100% /snap/erlang/101
/dev/loop3       56M   56M     0 100% /snap/core18/2708
```

2. 打开一个新的 shell 窗口 B，执行 Mount 隔离命令

```
Shell
# --mount 表示我们要隔离 Mount 命名空间了
# --fork 表示新建进程
unshare --mount --fork /bin/bash
mkdir -p /data/tmpmount
```

3. 在窗口 B 中添加新的磁盘挂载

```
Shell
dd if=/dev/zero of=fdimage.img bs=8k count=10240
mkfs -t ext4 ./fdimage.img
mount ./fdimage.img   /data/tmpmount
```

4. 在窗口 B 挂载的磁盘中添加文件

```
Shell
```

```
echo "Hello world!" > /data/tmpmount/hello.txt
```

5. 查看窗口 B 中的磁盘挂载信息

```
Shell
root@139-159-150-152:/data/maxhou/mounttest# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        40G   23G   15G  61% /
udev            948M     0  948M   0% /dev
tmpfs           992M     0  992M   0% /dev/shm
tmpfs           199M  2.4M  196M   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           199M  8.0K  199M   1% /run/user/1000
tmpfs           199M     0  199M   0% /run/user/1002
tmpfs           199M     0  199M   0% /run/user/1001
tmpfs           199M     0  199M   0% /run/user/0
tmpfs           992M     0  992M   0% /sys/fs/cgroup
/dev/loop2       55M   55M     0 100% /snap/erlang/101
/dev/loop4       56M   56M     0 100% /snap/core18/2714
overlay          40G   23G   15G  61% /data/myworkdir/fs/merged
/dev/loop1       50M   50M     0 100% /snap/snapd/18596
/dev/loop5       56M   56M     0 100% /snap/core18/2721
/dev/loop3       54M   54M     0 100% /snap/snapd/18933
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/922af407c456f95d898fea95ca148b30607b
20ff6c3e7c3ff1b61cff3fae4cfd/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/e0783839b4cf83c86574efa690a8d7f0ee3a
b56cfc58932f61b7259a29169f2d/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/9b230a4ec2179c894e7ef7e02ad8110a2fc8
2f73947b9b1d6ac52d8f530888a5/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/136daee7deb5833e762c4f15f669446a0953
cb7cbc4402db8ede275b360bd860/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/56b1378d1428401284a83392403f82816c5a
e6dde3c446d0e9f5bc59a1ce7d22/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/2673569d857e88b099d669886ef555934bb6
156d24e96fec47ad6edfc2ac861b/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/b30c60e4b02fa47646ab76bc0fc8f44ca411
8d03d1b7c976e4e9398396ae9c85/merged
```

```
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/0c1c00f53f152528ccf805682e71335901a3
5aef0a51656a4fd148b4bf63de73/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/59e9cbe3e4e352cfafcbd14fed1b9ac346ab
2aa41121130d6d6005c46192fd1f/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/9efca1404a5c76c5f02d8955c364837c9e65
5292c4a43335645aa6dfbd3c2e82/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/14f66040cb71227dbcbb6234a08355a45e54
f241eccedfce3ffb349eb48041ab/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/1a1750c0319256058b1d6ca2cb88f000a751
38126bf8db4ddd95c1abfb1bcb84/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/970c9cfe0e5dfbacee73b19ac28a15c81607
fd329113d392db3d085c3fcba334/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/71d23ff71232c534c92c2f900737360c3a63
92a0858f6e4becfe36379e17c9be/merged
/dev/loop0       74M   60K   68M   1% /data/tmpmount
```

6. 查看窗口 A 中的磁盘挂载信息

```
Shell
root@139-159-150-152:/data/tmpmount# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            948M     0  948M   0% /dev
tmpfs           199M  2.4M  196M   2% /run
/dev/vda1        40G   23G   15G  61% /
tmpfs           992M     0  992M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           992M     0  992M   0% /sys/fs/cgroup
/dev/loop2       55M   55M     0 100% /snap/erlang/101
/dev/loop4       56M   56M     0 100% /snap/core18/2714
tmpfs           199M  8.0K  199M   1% /run/user/1000
tmpfs           199M     0  199M   0% /run/user/1002
overlay          40G   23G   15G  61% /data/myworkdir/fs/merged
tmpfs           199M     0  199M   0% /run/user/1001
/dev/loop1       50M   50M     0 100% /snap/snapd/18596
/dev/loop5       56M   56M     0 100% /snap/core18/2721
tmpfs           199M     0  199M   0% /run/user/0
```

```
/dev/loop3       54M   54M     0 100% /snap/snapd/18933
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/922af407c456f95d898fea95ca148b30607b
20ff6c3e7c3ff1b61cff3fae4cfd/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/e0783839b4cf83c86574efa690a8d7f0ee3a
b56cfc58932f61b7259a29169f2d/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/9b230a4ec2179c894e7ef7e02ad8110a2fc8
2f73947b9b1d6ac52d8f530888a5/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/136daee7deb5833e762c4f15f669446a0953
cb7cbc4402db8ede275b360bd860/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/56b1378d1428401284a83392403f82816c5a
e6dde3c446d0e9f5bc59a1ce7d22/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/2673569d857e88b099d669886ef555934bb6
156d24e96fec47ad6edfc2ac861b/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/b30c60e4b02fa47646ab76bc0fc8f44ca411
8d03d1b7c976e4e9398396ae9c85/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/0c1c00f53f152528ccf805682e71335901a3
5aef0a51656a4fd148b4bf63de73/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/59e9cbe3e4e352cfafcbd14fed1b9ac346ab
2aa41121130d6d6005c46192fd1f/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/9efca1404a5c76c5f02d8955c364837c9e65
5292c4a43335645aa6dfbd3c2e82/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/14f66040cb71227dbcbb6234a08355a45e54
f241eccedfce3ffb349eb48041ab/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/1a1750c0319256058b1d6ca2cb88f000a751
38126bf8db4ddd95c1abfb1bcb84/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/970c9cfe0e5dfbacee73b19ac28a15c81607
fd329113d392db3d085c3fcba334/merged
overlay          40G   23G   15G  61%
/data/var/lib/docker/overlay2/71d23ff71232c534c92c2f900737360c3a63
92a0858f6e4becfe36379e17c9be/merged
```

7. 查看窗口 B 中的文件信息

```
Shell
root@139-159-150-152:~# ll /data/tmpmount/
total 8
drwxrwxrwt 2 root root   60 Mar 11 18:13 ./
drwx--x--x 4 root root 4096 Mar 11 17:59 ../
-rw-r--r-- 1 root root   13 Mar 11 18:13 hello.txt
root@139-159-150-152:~# cat /data/tmpfs/hello.txt
Hello world!
```

8. 查看窗口 A 中的文件信息，可以看到窗口 B 中新建的文件和磁盘挂载在主机的窗口中并没有，说明我们实现了文件系统隔离。

```
Shell
root@139-159-150-152:/data/tmpfs# ll /data/tmpmount
total 8
drwxr-xr-x 2 root root 4096 Mar 11 17:59 ./
drwx--x--x 4 root root 4096 Mar 11 17:59 ../
root@139-159-150-152:/data/tmpfs# cat /data/tmpmount/hello.txt
cat: /data/tmpfs/hello.txt: No such file or directory
```

9. 窗口 B 执行 exit，退出

```
Shell
exit
```