



unshare命令详解及案例

张智溪

2021年07月22日 10:00 · 阅读 501

+ 关注

unshare命令详解

1.名字

unshare - run program with some namespaces unshared from parent(使用与父程序不共享的名称空间运行程序)

2.摘要

unshare [options] program [arguments]

3.描述

Unshares the indicated namespaces from the parent process and then executes the specified program. The namespaces to be unshared are indicated via options. Unshareable namespaces are(取消共享父进程中指定的命名空间，然后执行指定的程序。要取消共享的命名空间通过选项来指示。不可共享的命名空间有)

unshare一共可以设置6个不共享的命名空间，分别是：

mount namespace
UTS namespace
IPC namespace
network namespace
pid namespace
user namespace

4.选项

一共有如下选项：

(1)-i, --ipc
unshare the IPC namespace.
(2)-u, --mount
Unshare the mount namespace.
(3)-n, --net
Unshare the network namespace.
(4)-p, --pid
Unshare the pid namespace. See also the --fork and --mount-proc options.
(5)-u, --uts
Unshare the UTS namespace.
(6)-U, --user
Unshare the user namespace.
(7)-f, --fork
Fork the specified program as a child process of unshare rather than running it directly. This is useful if you want to unshare the namespaces of the child process.
(8)--mount-proc[=mountpoint]
Just before running the program, mount the proc filesystem at mountpoint (default is /proc). This is useful if you want to unshare the namespaces of the child process.
(9)-r, --map-root-user
Run the program only after the current effective user and group IDs have been mapped to the superuser UID (0).
(10)--propagation private|shared|slave|unchanged
Recursively sets mount propagation flag in the new mount namespace. The default is to set the propagation flag to private.
(11)--setgroups allow|deny
Allow or deny setgroups(2) syscall in user namespaces.
(12)--setgroups(2)
setgroups(2) is only callable with CAP_SETGID and CAP_SETPID in a user namespace (since Linux 3.19) does not have CAP_SETGID.
(13)-V, --version
Display version information and exit.
(14)-h, --help
Display help text and exit.

5.例子

```
# unshare --fork --pid --mount-proc readlink /proc/self
1
Establish a PID namespace, ensure we're PID 1 in it against newly mounted proc's instance.

(建立一个PID命名空间，确保我们在新安装procfs实例中是PID 1.)

$ unshare --map-root-user --user sh -c whoami
root
Establish a user namespace as an unprivileged user with a root user within it.

(将用户命名空间建立为非特权用户U，其中包含根用户R.)
```

6.各种Namespace的作用？

(1)Mount Namespace

Mount Namespace 是 Linux 内核实现的第一个 Namespace。从内核的 2.4.19 版本开始加入。它可以用来隔离不同的进程或进程组看到的挂载点。通俗地说，就是可以实现不同的进程中看到不同的挂载目录。使用 Mount Namespace 可以实现容器内只能看到自己的挂载信息，在容器内的挂载操作不会影响主机的挂载目录。

下面我们通过一个实例来演示下 Mount Namespace。我们使用一个命令行工具 unshare，unshare 是 util-linux 工具包中的一个工具。CentOS 7 系统默认已经集成了该工具。使用 unshare 命令可以实现创建并访问不同类型的 Namespace。

首先我们使用以下命令创建一个 bash 进程并且新建一个 Mount Namespace：

```
unshare --mount --fork /bin/bash
```

执行完上述命令后，这时我们已经在主机上创建了一个新的 Mount Namespace，并且当前命令行窗口加入了新创建的 Mount Namespace，下面我通过一个例子来验证下，在独立的 Mount Namespace 内创建挂载目录是不影响主机的挂载目录的。

执行完上述命令后，这时我们已经在主机上创建了一个新的 Mount Namespace，并且当前命令行窗口加入了新创建的 Mount Namespace，下面我通过一个例子来验证下，在独立的 Mount Namespace 内创建挂载目录

张智溪

博主

获得点赞17

文章被阅读13,520

春节创意
投稿大赛

掘金2022年
技术创作
大赛

2022
首次征文挑战

掘金计划创作月榜

创作新人榜

下载稀土掘金APP

一个帮助开发者成长的社区

相关文章

Share-API 更好用的 Swagger 文档视图工具

SharePreference原爆及跨进程数据共享的问题

java Web之HttpSession

nginx的Rewrite规则详解

30 道 Vue 面试题，内含详细讲解(涵盖入门到精通，善用 Vue 掌握程度)

目录

unshare命令详解

1.名字

2.摘要

3.描述

4.选项

5.例子

6.各种Namespace的作用？

(1)Mount Namespace

(2)PID Namespace

(3)UTS Namespace

(4)IPC Namespace

(5)User Namespace

下一篇

如何给虚拟机配置Yum源

是不影响主机的挂载目录的。

首先在 /tmp 目录下创建一个目录。

```
[root@centos7 centos]# mkdir /tmp/tmpfs
```

创建好目录后使用 mount 命令挂载一个 tmpfs 类型的目录。命令如下：

```
[root@centos7 centos]# mount -t tmpfs -o size=20m tmpfs /tmp/tmpfs
```

然后使用 df 命令查看一下已经挂载的目录信息：

```
[root@centos7 centos]# df -h

Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        500G  1.4G  499G   1% /
devtmpfs         16G   0  16G   0% /dev
tmpfs            16G   0  16G   0% /dev/shm
tmpfs            16G   0  16G   0% /sys/fs/cgroup
tmpfs            16G  57M   16G   1% /run
tmpfs            3.2G   0  3.2G   0% /run/user/1000
tmpfs            20M   0  20M   0% /tmp/tmpfs
```

可以看到 /tmp/tmpfs 目录已经被正确挂载。为了验证主机上并没有挂载此目录，我们新打开一个命令行窗口，同样执行 df 命令查看主机的挂载信息：

```
[centos@centos7 ~]$ df -h

Filesystem      Size  Used Avail Use% Mounted on
devtmpfs         16G   0  16G   0% /dev
tmpfs            16G   0  16G   0% /dev/shm
tmpfs            16G  57M   16G   1% /run
tmpfs            16G   0  16G   0% /sys/fs/cgroup
/dev/vda1        500G  1.4G  499G   1% /
tmpfs            3.2G   0  3.2G   0% /run/user/1000
```

通过上面输出可以看到主机上并没有挂载 /tmp/tmpfs，可见我们独立的 Mount Namespace 中执行 mount 操作并不会影响主机。

为了进一步验证我们的想法，我们继续在当前命令行窗口查看一下当前进程的 Namespace 信息。命令如下：

```
[root@centos7 centos]# ls -l /proc/self/ns/

total 0

lrwxrwxrwx. 1 root root 0 Sep 4 08:20 ipc -> ipc:[4026531839]
lrwxrwxrwx. 1 root root 0 Sep 4 08:20 mnt -> mnt:[4026532239]
lrwxrwxrwx. 1 root root 0 Sep 4 08:20 net -> net:[4026531956]
lrwxrwxrwx. 1 root root 0 Sep 4 08:20 pid -> pid:[4026531836]
lrwxrwxrwx. 1 root root 0 Sep 4 08:20 user -> user:[4026531837]
lrwxrwxrwx. 1 root root 0 Sep 4 08:20 uts -> uts:[4026531838]
```

然后新打开一个命令行窗口，使用相同的命令查看一下主机上的 Namespace 信息：

```
[centos@centos7 ~]$ ls -l /proc/self/ns/

total 0

lrwxrwxrwx. 1 centos centos 0 Sep 4 08:20 ipc -> ipc:[4026531839]
lrwxrwxrwx. 1 centos centos 0 Sep 4 08:20 mnt -> mnt:[4026531840]
lrwxrwxrwx. 1 centos centos 0 Sep 4 08:20 net -> net:[4026531956]
lrwxrwxrwx. 1 centos centos 0 Sep 4 08:20 pid -> pid:[4026531836]
lrwxrwxrwx. 1 centos centos 0 Sep 4 08:20 user -> user:[4026531837]
lrwxrwxrwx. 1 centos centos 0 Sep 4 08:20 uts -> uts:[4026531838]
```

通过对比两次命令的输出结果，我们可以看到，除了 Mount Namespace 的 ID 位不一样外，其他 Namespace 的 ID 位均一致。

通过以上结果我们可以得出结论，使用 unshare 命令可以新建 Mount Namespace，并且在新建的 Mount Namespace 内 mount 是和外部完全隔离的。

(2)PID Namespace

PID Namespace 的作用是用来隔离进程。在不同的 PID Namespace 中，进程可以有相同的 PID 号，利用 PID Namespace 可以实现每个容器的主进程为 1 号进程，而容器内的进程在主机上却有不同的PID，例如一个进程在主机上 PID 为 122，使用 PID Namespace 可以实现该进程在容器内看到的 PID 为 1。

下面我们通过一个实例来演示下 PID Namespace 的作用。首先我们使用以下命令创建一个 bash 进程，并且新建一个 PID Namespace：

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

执行完上述命令后，我们在主机上创建了一个新的 PID Namespace，并且当前命令行窗口加入了新创建的 PID Namespace，在当前的命令行窗口使用 ps aux 命令查看一下进程信息：

```
[root@centos7 centos]# ps aux

USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1  0.0  0.0 115544 3084 pts/0    S   10:57   0:00 bash
root     10  0.0  0.0 115444 1764 pts/0    R+  10:59   0:00 ps aux
```

通过上述命令输出结果可以看到当前 Namespace 下 bash 为 1 号进程，而且我们也看不到主机上的其他进程信息。

(3)UTS Namespace

UTS Namespace 主要是用来隔离主机名的，它允许每个 UTS Namespace 拥有一个独立的主机名。例如我们的主机名称为 docker，使用 UTS Namespace 可以实现容器内的主机名称为 lagoudocker 或者其他任意自定义主机名。

同样我们通过一个实例来验证下 UTS Namespace 的作用，首先我们使用 unshare 命令来创建一个 UTS Namespace：

```
unshare --fork --uts /bin/bash
```

创建好 UTS Namespace 后，当前命令行窗口已经处于一个独立的 UTS Namespace 中，下面我们使用 hostname 命令(hostname 可以用来查看主机名称)设置一下主机名：

```
hostname -b lagoudocker
```

然后再查看一下主机名：

```
[root@centos7 centos]# hostname

lagoudocker
```

通过上面命令的输出，我们可以看到当前UTS Namespace 内的主机名已经被修改为 lagoudocker，然后我们新打开一个命令行窗口，使用相同的命令查看一下主机的 hostname：

```
[centos@centos7 ~]$ hostname
centos7
```

可以看到主机的名称仍然为 centos7, 并没有被修改。由此, 可以验证 UTS Namespace 可以用来隔离主机名。

(4)IPC Namespace

IPC Namespace 主要是用来隔离进程间通信的。例如 PID Namespace 和 IPC Namespace 一起使用可以实现同一 IPC Namespace 内的进程彼此可以通信, 不同 IPC Namespace 的进程却不能通信。

同样我们通过一个实例来验证下IPC Namespace的作用, 首先我们使用 unshare 命令来创建一个 IPC Namespace:

```
unshare --fork --ipc /bin/bash
```

下面我们需要借助两个命令来实现对 IPC Namespace 的验证。

ipcs -q 命令:用来查看系统间通信队列列表。

ipcmk -Q 命令:用来创建系统间通信队列。

我们首先使用 ipcs -q 命令查看一下当前 IPC Namespace 下的系统通信队列列表:

```
[centos@centos7 ~]$ ipcs -q
----- Message Queues -----
key      msgid   owner   perms  used-bytes  messages
```

由上可以看到当前无任何系统通信队列, 然后我们使用 ipcmk -Q 命令创建一个系统通信队列:

```
[root@centos7 centos]# ipcmk -Q
Message queue id: 0
```

再次使用 ipcs -q 命令查看当前 IPC Namespace 下的系统通信队列列表

```
[root@centos7 centos]# ipcs -q
----- Message Queues -----
key      msgid   owner   perms  used-bytes  messages
0x73682a32 0      root    644    0           0
```

可以看到我们已经成功创建了一个系统通信队列。然后我们新打开一个命令行窗口, 使用ipcs -q 命令查看一下主机的系统通信队列:

```
[centos@centos7 ~]$ ipcs -q
----- Message Queues -----
key      msgid   owner   perms  used-bytes  messages
```

通过上面的实验, 可以发现, 在单独的 IPC Namespace 内创建的系统通信队列在主机上无法看到, 即 IPC Namespace 实现了系统通信队列的隔离。

(5)User Namespace

User Namespace 主要是用来隔离用户和用户组的。一个比较典型的应用场景就是在主机上以非 root 用户运行的进程可以在一个单独的 User Namespace 中映射成 root 用户。使用 User Namespace 可以实现进程在容器内拥有 root 权限, 而在主机上却只是普通用户。

User Namespce 的创建是可以不使用 root 权限的。下面我们以普通用户的身份创建一个 User Namespace, 命令如下:

```
[centos@centos7 ~]$ unshare --user -r /bin/bash
[root@centos7 ~]#
```

CentOS7 默认允许创建的 User Namespace 为 0, 如果执行上述命令失败(unshare 命令返回的错误为 unshare: unshare failed: Invalid argument), 需要使用以下命令修改系统允许创建的 User Namespace 数量, 命令为:echo 65535 > /proc/sys/user/max_user_namespaces, 然后再次尝试创建 User Namespace。

然后执行 id 命令查看一下当前的用户信息:

```
[root@centos7 ~]# id
uid=0(root) gid=0(root) groups=0(root),65534(nfsnobody) context=unconfined_u:unconfined_r:unconfined_t:s0-
```

通过上面的输出可以看到我们在新的 User Namespace 内已经是 root 用户了。下面我们使用只有主机 root 用户才可以执行的 reboot 命令来验证一下, 在当前命令行窗口执行 reboot 命令:

```
[root@centos7 ~]# reboot
Failed to open /dev/initctl: Permission denied
Failed to talk to init daemon.
```

可以看到, 我们在新创建的 User Namespace 内虽然是 root 用户, 但是并没有权限执行 reboot 命令。这说明在隔离的 User Namespace 中, 并不能获取到主机的 root 权限, 也就是说 User Namespace 实现了用户和用户组的隔离。

(6)Net Namespace

Net Namespace 是用来隔离网络设备、IP 地址和端口等信息的, Net Namespace 可以让每个进程拥有自己独立的 IP 地址、端口和网卡信息。例如主机 IP 地址为 172.16.4.1, 容器内可以设置独立的 IP 地址为 192.168.1.1。

同样用实例验证, 我们首先使用 ip a 命令查看一下主机上的网络信息:

```
$ 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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 02:11:b0:14:01:0c brd ff:ff:ff:ff:ff:ff
    inet 172.20.1.11/24 brd 172.20.1.255 scope global dynamic eth0
        valid_lft 86063337sec preferred_lft 86063337sec
    inet6 fe80::11:b0ff:fe14:10c/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:82:b0:a0:d9 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:82ff:fe8d:a0ff/64 scope link
        valid_lft forever preferred_lft forever
```

然后我们使用以下命令创建一个 Net Namespace:

```
unshare --net --fork /bin/bash
```

同样的我们使用 ip a 命令查看一下网络信息:


```
[root@centos7 centos]# 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
```

可以看到, 宿主机上有 lo, eth0, docker0 等网络设备, 而我们新建的 Net Namespace 内则与主机上的网络设备不同。

分类: 后端 标签: Linux 后端

文章被收录于专栏:

 linux
linux


关注

 安装掘金浏览器插件

多内容聚合浏览, 多引擎快捷搜索, 多工具便捷提效, 多模式随心切换, 你想要的, 这里都有!

前往安装

评论



输入评论 (Enter 执行, Ctrl + Enter 发送)

相关推荐

- 爱吃薯条的Ray | 6月前 | 微小程序 前端
- 小程序全局分享onShareAppMessage (亲测有效)**
- 662 3 评论

xiaofu0825 | 1年前 | Vue.js

uni-app+uniCloud 初体验

921 5 评论

小肥羊冲冲冲Android | 6月前 | Android

share SDK 导入实现第三方登录和分享

206 1 评论

杰哥的IT之旅 | 5月前 | 后端 运维 Linux

Linux 终端下记不住命令的使用方法? 这个开屏项目帮你解决。

3336 38 2

程序员ouan | 1月前 | Linux 后端 运维

看完这篇 Linux 权限后, 通透了!

6461 33 2

Cyandev | 5年前 | Linux 后端 操作系统

5 分钟让你明白“软链接”和“硬链接”的区别

4214 110 10

Lion | 10月前 | Linux 后端

2万字系统总结, 带你实现 Linux 命令自由!

1.8w 562 30

刘志军 | 3年前 | 后端 爬虫 GitHub

中间人攻击(爬虫工具) mitmproxy 使用指南

6021 68 5

DT黑白 | 6月前 | Linux Docker 后端

全网最细Docker安装Minio, 搞定最新版本坑(强烈推荐收藏)

5298 31 12

HonestTly | 5月前 | 后端 Linux

TCP 为什么是三次握手, 而不是两次或四次?

5387 47 2

南橘nyc | 1年前 | 分布式

【进阶之路】可靠消息最终一致性解决方案

3180 22 评论

Authing | 8月前 | API 安全 前端

Authing Share | 理解 SAML2 协议

413 4 评论

MacroZheng | 1年前 | Java Linux

还在百度Linux命令? 推荐一套我用起来特顺手的命令!

1.5w 244 19

十年上路 | 1年前 | Linux

虚拟机中linux系统实现路由转发功能

65 2 评论

hsfuebao | 9月前 | Linux

每天一个linux命令(26): 用SecureCRT来上传和下载文件

98 点赞 评论

Authing | 8月前 | 安全 前端

Authing Share | 系统安全与防范

253 2 评论

渡比小金刚 | 3年前 | React.js 程序员 运维

StackShare 发布了 2017 年排名前 50 的开发工具

2005 27 评论

杰哥的IT之旅 | 5月前 | 后端 Linux 运维

17 个有趣却无用的 Linux 彩蛋, 真是好玩到爆啦!

3694 60 10

晨逸 | 1年前 | axios

【已解决】使用Axios发送请求, 有时错误的时候没有返回任何东... 偶, 发送请求会返回东西, 不会报错。

473 2 9

红烧不是清蒸 | 3年前 | Linux 后端 安全

防止linux(被)入侵, 服务器受攻击

稀土掘金浏览器插件——你的一站式工作台

多内容聚合浏览, 多引擎快捷搜索, 多工具便捷提效, 多模式随心切换, 你想要的, 这里都有。

安装浏览器插件

