本科生实验报告

学生姓名：　丁晓琪

学生学号：　22336057

专业名称：　计科

一：实验任务

实验题1：CRIU是一种在用户空间实现的进程或者容器checkp oint 和restore的方法， 从而实现进程或者容器的保存和恢复。 请利用CRIU实现进程和容器的迁移（迁移种类不限），并测试 迁移过程中的性能损耗（如进程停止时间、网络传输时间等）

二：实验过程

1. 进程迁移

- 利用`nfs`将虚拟机1的`/home`目录，挂载在虚拟机2的`/home`目录
  - 目的：在用criu生成进程迁移相关文件后，不需要从虚拟机1复制粘贴到虚拟机2，而是在虚拟机2远程访问运行虚拟机1中文件，从而复原进程
  - 注意：在网上教程中是将虚拟机1的`/home`目录挂载在了虚拟机2的`/nfs/home`目录，但是后面进程迁移时会发现找不到`/home`目录的报错，所有挂载在同一个地址

```
test2@test2-virtual-machine:/$ cd home
test2@test2-virtual-machine:/home$ cd test1
test2@test2-virtual-machine:/home/test1$ ls
checkpoint_file   nfs        test.sh          模板  文档  桌面
criu-3.14         test.log   Untitled 1.odt   视频  下载
examples.desktop  test.out   公共的           图片  音乐
```

- （虚拟机1）在后台启动一个新的会话进程运行`test.sh`脚本
  - `test.sh`脚本如下：打印时间

```
#! /bin/bash
while (true)
do
    time=$(date +%s.%N)
    echo $time
    sleep 10e-4
done
```

  - 执行并且查看输出：|
    - `setid`: 让脚本在新的会话进行，脱离当前终端会话，不会收到终端关闭影响。

- `<dev/null`:脚本无法从用户处接收输入
- `&>$PWD/test.out`：脚本运行的输出被重定向到当前目录的`test.out`下面

```
test1@test1-virtual-machine:~$ sudo setsid ./test.sh </dev/null &>$PWD/test.out
^[[A^[[B^C^C^C^C^C已杀死
test1@test1-virtual-machine:~$ tail test.out
1728484212.127702858
1728484212.135880297
1728484212.145354654
1728484212.153787007
1728484212.165464634
1728484212.178420179
1728484212.186824142
1728484212.195421070
1728484212.204080824
1728484212.219608091
test1@test1-virtual-machine:~$
```

- 查看进程相关信息：

  `ps -ef |grep test`：脚本进程pid为10539

```
test1      10376   3242  0 22:18 pts/21   00:00:00 bash
test1      10508   2427  0 22:21 ?        00:00:02 gedit /home/test1/test.out
root       10538   4226  0 22:22 pts/2    00:00:00 sudo setsid ./test.sh
root       10539  10538 14 22:22 ?        00:00:28 /bin/bash ./test.sh
test1      11536   3242  0 22:22 pts/22   00:00:00 bash
test1      52468   3344  1 22:25 ?        00:00:00 /usr/lib/firefox/firefox -contentproc -childI
test1      61173  11536  0 22:25 pts/22   00:00:00 ps -ef
```

- 虚拟机1设置检查点并且终端进程：为进程10539创建检查点并且将进程检查点文件保存在`checkpoint_file`目录

```
mkdir checkpoint_file
criu dump -D checkpoint_file -j -t 10539
```

- 在虚拟机2挂载虚拟1`/home`目录后，检查是否存在`checkpoint_file`文件，并且恢复进程

  `criu restore -D checkpoint_file -j`：恢复`checkpoint_file`中的进程

  - 问题1：有一些文件的大小和预期不符合

    解决：用`truncate -s size file`缩小文件file到预期大小size

```
test2@test2-virtual-machine:/$ cd home
test2@test2-virtual-machine:/home$ cd test1
test2@test2-virtual-machine:/home/test1$ ls
checkpoint_file   nfs        test.sh          模板  文档  桌面
criu-3.14         test.log   Untitled 1.odt   视频  下载
examples.desktop  test.out   公共的           图片  音乐
test2@test2-virtual-machine:/home/test1$ sudo criu restore -D checkpoint_file -j
 10539: Error (criu/files-reg.c:1674): File home/test1/test.out has bad size 136
6575 (expect 820680)
 10539: Error (criu/files.c:1191): Unable to open fd=1 id=0x9
Error (criu/cr-restore.c:2171): Restoring FAILED.
test2@test2-virtual-machine:/home/test1$ truncate -s 820680 test.out
test2@test2-virtual-machine:/home/test1$ sudo criu restore -D checkpoint_file -j
 10539: Error (criu/files-reg.c:1674): File usr/lib/locale/locale-archive has ba
d size 4795408 (expect 10219008)
```

  - 问题2：两台虚拟机的链接库版本不一样报错

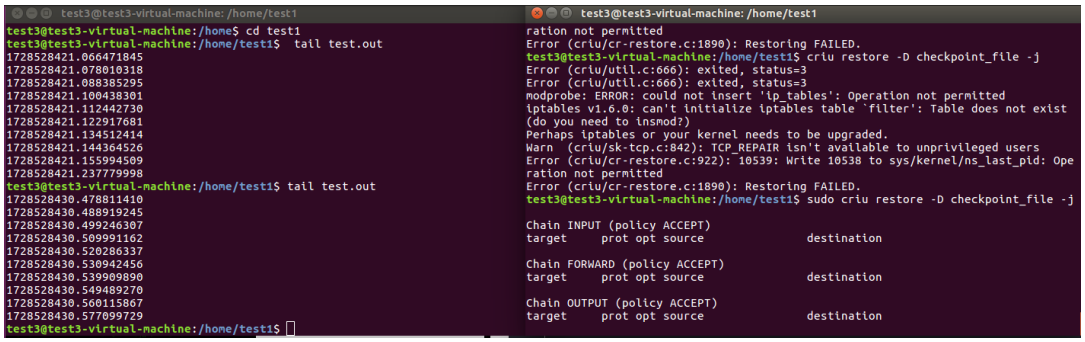
```
test1@test1-virtual-machine:~$ ls -l /lib/x86_64-linux-gnu/libc.so.6
lrwxrwxrwx 1 root root 12 10月  9 15:34 /lib/x86_64-linux-gnu/libc.so.6 -> libc-
2.23.so
```

```
test2@test2-virtual-machine:/home/test1$ ls -l /lib/x86_64-linux-gnu/libc.so.6
lrwxrwxrwx 1 root root 12 5月   3  2022 /lib/x86_64-linux-gnu/libc.so.6 -> libc-
2.27.so
```

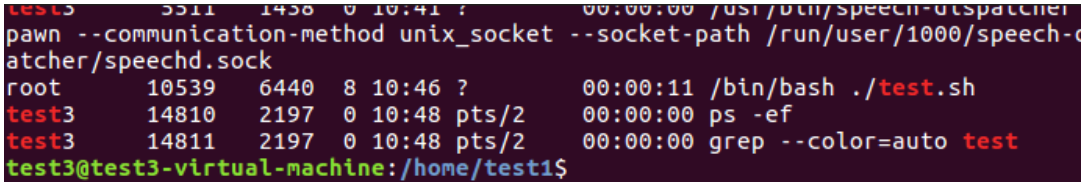
    解决：重新创建虚拟机3和虚拟机1的链接库版本一致

```
test3@test3-virtual-machine:~/桌面/vmware-tools-distrib$ sudo mount 192.168.91.1
r35:/home /home
```

  - 在虚拟机3重新恢复进程：



查看虚拟机3中恢复后的进程的pid，也是10539

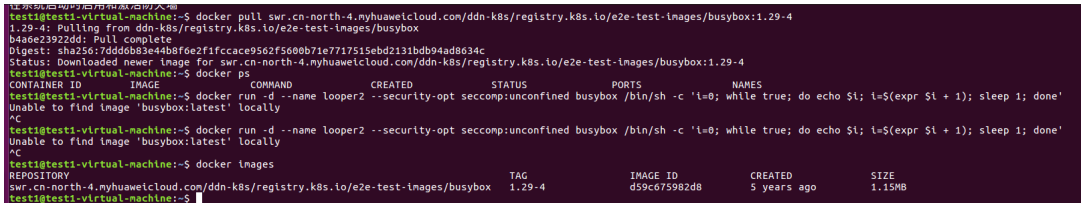


2. 容器迁移：

- 在虚拟机1创建容器：
  - 在国内的镜像网站拉取 `busybox` 的镜像：

    `docker pull swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4`
  - 查看镜像是否拉取成功：

    `docker images`



  - 在docker容器中启动一个进程：

    `docker run -D`：从给定的镜像启动新的容器，并且容器在后台运行

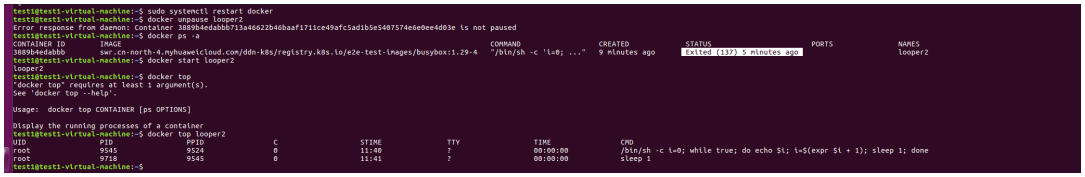
    `--name looper2`：容器的名称为looper2

    `swr.cn-north-4.myhuaweicloud.com/ddn-k8s/registry.k8s.io/e2e-test-images/busybox:1.29-4`：创建容器的镜像

    `/bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'`：容器启动后要执行的命令。用`/bin/sh`（一个 shell）执行一个无限循环的脚本。脚本的内容是初始化一个变量 `i` 为 0，进入一个无限循环，每次循环都会打印当前的 `i` 值，将 `i` 的值加 1，然后休眠 1 秒。

  - 确认容器状态：`docker ps -a`

    检查容器中是否有计数进程：`docker top looper2`



- （虚拟机1）设置`checkpoint`：

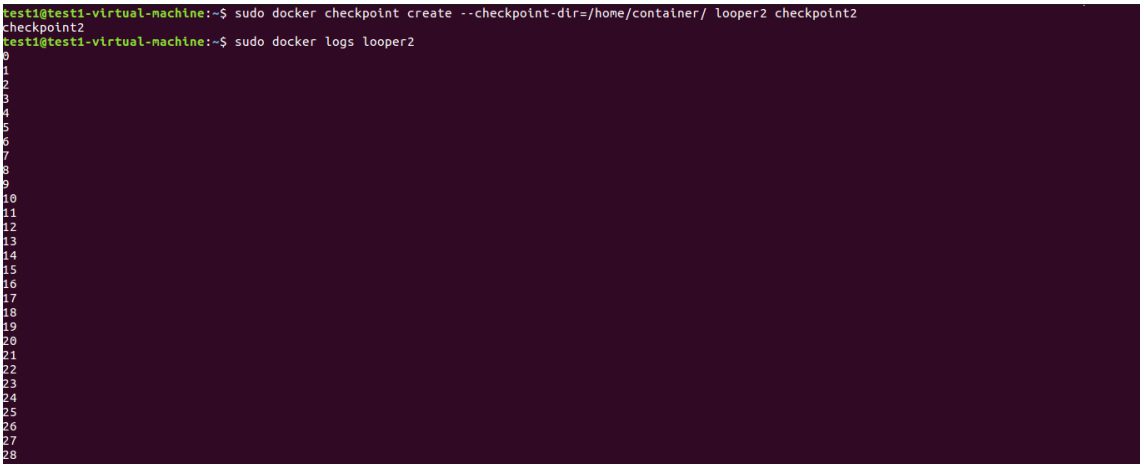
  `docker checkpoint create`：创建新的检查点

`--checkpoint-dir=/home/container/`：检查点保存目录

创建检查点的容器是`looper2`

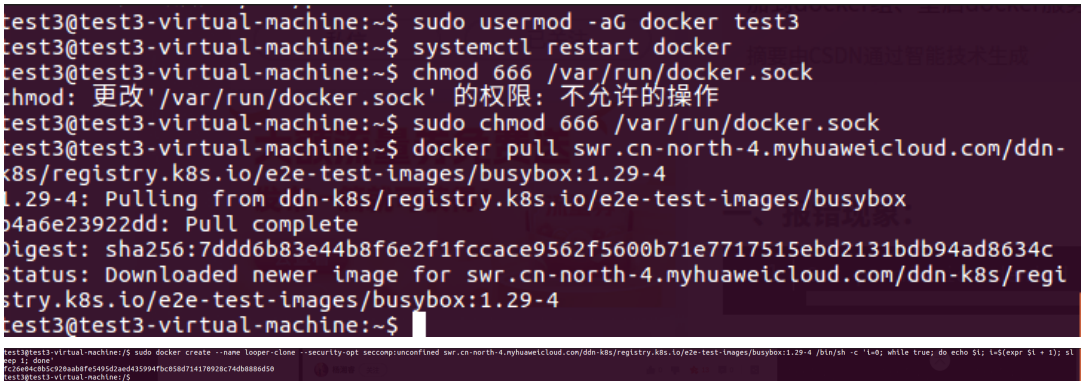
检查点名称为`checkpoint2`

查看容器的输出：`docker logs looper2`



注意设置检查点后容器中断,最后输出为91

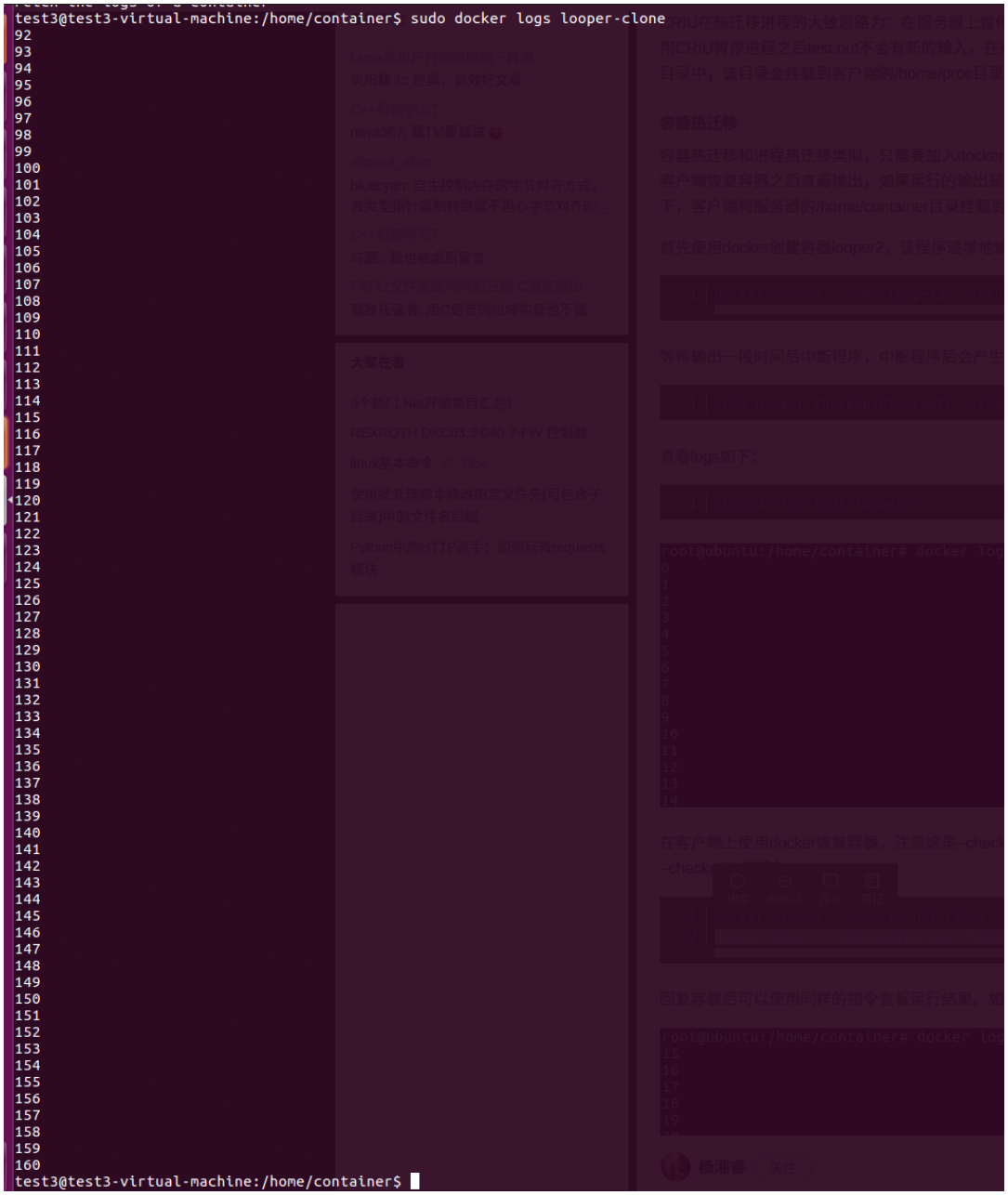
- （虚拟机3）恢复容器：
  - 恢复前准备：拉取busybox镜像和创建克隆容器`looper-clone`（在克隆容器中恢复容器）



  - 在克隆容器中恢复容器：

    `docker start`：

- 查看迁移恢复后的容器状态和输出：从92开始打印（接上前面的91）



三：实验总结

由于不熟悉进程迁移和容器迁移的过程，创建检查点后将时间大部分花在解决恢复进程和容器的报错以及一些环境问题上。在网络传输上，时间主要花费在挂载检查文件所在的目录上。成功执行恢复命令时也需要耗费一定时间等待进程/容器启动。