

分布式系统作业二

一、实验要求：

利用 CRIU 实现进程和容器的热迁移。

二、实验过程：

环境：Ubuntu 18.04 LTS, CRIU 3.6, Docker 17.06.0-ce

I.进程的迁移：

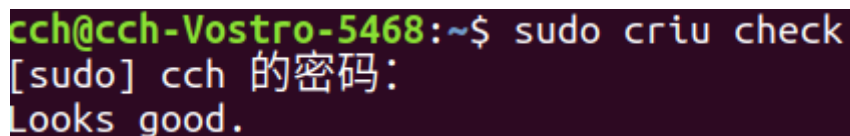
1.安装 CRIU：

根据老师给的 github 链接安装，时间会很长：

```
sudo apt-get update && sudo apt-get install -y protobuf-c-compiler libprotobuf-c0-dev protobuf-compiler libprotobuf-dev:amd64 gcc build-essential bsdmainutils python git-core asciidoc make httpd git curl supervisor cgroup-lite libapparmor-dev libseccomp-dev libprotobuf-dev libprotobuf-c0-dev protobuf-c-compiler protobuf-compiler python-protobuf libnl-3-dev libcap-dev libaio-dev apparmor libnet-dev
$ git clone https://github.com/xemu1/criu criu
$ cd criu
$ sudo make clean
$ sudo make
$ sudo make install
# Then check if your criu works well
$ sudo criu check
$ sudo criu check --all
```

需要注意的就是依赖包的安装，缺少某个依赖或者某个依赖不太对会很麻烦！

如果出现如下内容则证明没有问题：



```
cch@cch-Vostro-5468:~$ sudo criu check
[sudo] cch 的密码:
Looks good.
```

2.准备工作：

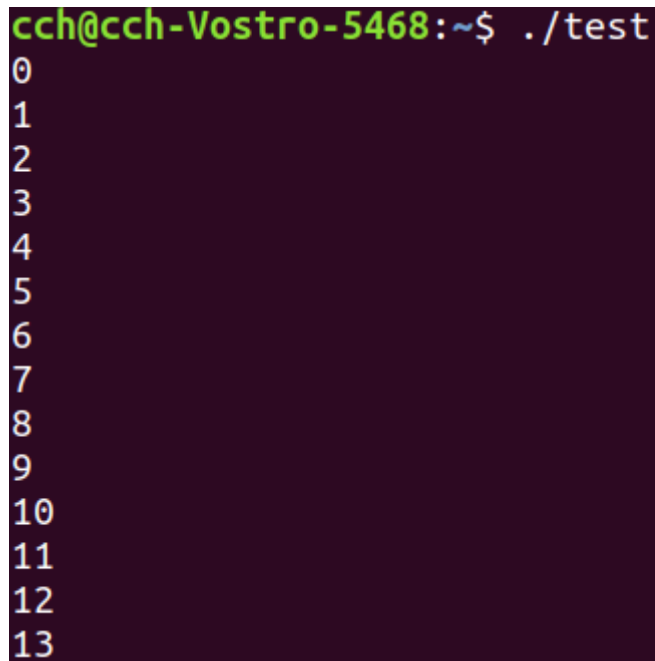
2.1 编写一个测试程序 test.c：

```
#include <stdio.h>
#include <unistd.h>
int main()
{
    int i=0;
    for(;i<100;i++){
        printf("%d\n",i);
        sleep(1);
    }
    return 0;
}
```

2.2 编译运行:

```
gcc test.c -o test
./test
```

可以看到程序正在运行，滚屏输出:



```
cch@cch-Vostro-5468:~$ ./test
0
1
2
3
4
5
6
7
8
9
10
11
12
13
```

3.迁移:

压缩后用U盘拷贝至另外一个设备上，在新设备上:

3.1 dump:

首先创建一个文件夹:

```
mkdir imgdir
```

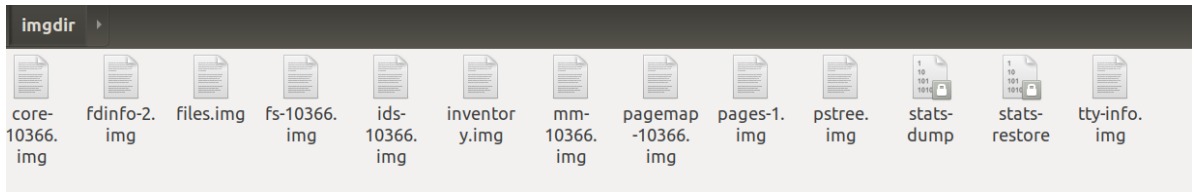
获取进程号:

```
ps -ef | grep test
```

得到进程号为10366，然后创建 checkpoint:

```
criu dump -D imgdir -j -t 10366
```

打开 `imgdir` 文件夹，可以看到许多 `img` 文件，这些文件记录了程序的状态以便恢复时使用；与此同时，可以看到滚屏停止，也就是说程序停止了执行。



3.2 restore:

```
criu restore -D imgdir -j
```

可以看到此时程序又重新执行。

II.容器的迁移:

1. Docker 安装:

很简单，几行命令即可:

```
apt update
apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
xenial stable"
apt install docker-ce=17.06.0~ce-0~ubuntu
```

2.修改配置文件:

如果使用默认的配置文​​件，则会发现根本进行不下去，会弹出各种各样的bug；我也是在网上尝试了好多解决方案，最终这一解决方案解决了出现的bug:

2.1 自己在 `etc/docker` 下创建个 `daemon.json`，即输入如下指令:

```
sudo vim etc/docker/daemon.json
```

加入如下内容:

```
{
  "insecure-registries" : ["storage-shshukun:5050"],
  "registry-mirrors": ["https://wwuqa7no.mirror.aliyuncs.com"],
  "runtimes": {
    "nvidia": {
      "path": "nvidia-container-runtime",
      "runtimeArgs": []
    }
  },
  "experimental"=true,
  "live-restore"=true
}
```

2.2 进入系统配置，执行如下命令:

```
sudo vim /etc/sysconfig/docker
```

修改如下内容：

```
--selinux-enabled=false
```

2.3 更新修改的信息，执行如下命令：

```
sudo systemctl daemon-reload
sudo systemctl start docker.service
sudo docker info
```

可以看到已经是修改后的结果：

```
Server Version: 17.06.0-ce
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 11
 Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: nvidia runc
Default Runtime: runc
Init Binary: docker-init
containerd version: cfb82a876ecc11b5ca0977d1733adbe58599088a
runc version: 2d41c047c83e09a6d61d464906feb2a2f3c52aa4
init version: 949e6fa
Security Options:
 apparmor
 seccomp
  Profile: default
Kernel Version: 5.4.0-52-generic
Operating System: Ubuntu 18.04.5 LTS
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 3.669GiB
Name: cch-Vostro-5468
ID: Q5RV:YU7F:RY46:5YYT:NN4H:AQ7F:TYII:7OXG:6G7G:ZBVC:5XHA:PHRX
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Experimental: true
Insecure Registries:
 storage-shshukun:5050
 127.0.0.0/8
Registry Mirrors:
 https://wwuqa7no.mirror.aliyuncs.com/
Live Restore Enabled: true

WARNING: No swap limit support
```

准备就绪后，正式开始使用 Docker。

3. 迁移：

3.1 运行一个容器：

```
docker run -d --name loop2 --security-opt seccomp:unconfined busybox \
/bin/sh -c 'i=0; while true; do echo $i; i=$((i+1)); sleep 1; done'
```

运行后可以看到获得了容器的 ID：

```
cch@cch-Vostro-5468:~/docker$ docker run -d --name loop2 --security-opt seccomp:unconfined busybox \
> /bin/sh -c 'i=0; while true; do echo $i; i=$((i + 1)); sleep 1; done'
2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b
```

3.2 用 checkpoint 保存下来当前的状态：

```
docker checkpoint create --checkpoint-dir=/home loop2 checkpoint2
```

可以看到保存为 checkpoint2：

```
cch@cch-Vostro-5468:~/docker$ docker checkpoint create --checkpoint-dir=/home loop2 checkpoint2
checkpoint2
```

3.3 通过日志查看中断前运行情况（后面会用到）：

```
cch@cch-Vostro-5468:~/docker$ docker logs loop2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

3.4 可以在 /home 下的 checkpoint2 内看到镜像文件：

```
cch@cch-Vostro-5468:~/docker$ cd /home
cch@cch-Vostro-5468:/home$ ls
2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b  lost+found
cch
cch@cch-Vostro-5468:/home$ cd 2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b
cch@cch-Vostro-5468:/home/2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b$ ls
checkpoints
cch@cch-Vostro-5468:/home/2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b$ cd checkpoints
cch@cch-Vostro-5468:/home/2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b/checkpoints$ ls
checkpoint2
cch@cch-Vostro-5468:/home/2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b/checkpoints$ cd checkpoint2
cch@cch-Vostro-5468:/home/2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b/checkpoints/checkpoint2$ ls
cgroup.img      fdinfo-3.img    ipcns-var-10.img  pages-2.img
config.json     files.img       mm-1.img          pipes-data.img
core-1.img      fs-1.img        mm-48.img         pstree.img
core-48.img     fs-48.img       mountpoints-12.img tmpfs-dev-67.tar.gz.img
criu.work       ids-1.img       pagemap-1.img     tmpfs-dev-70.tar.gz.img
descriptors.json ids-48.img      pagemap-48.img    tmpfs-dev-71.tar.gz.img
```

压缩后用U盘拷贝至另外一个设备上，在新设备上：

3.5 创建一个容器：

```
docker create --name loop22 --security-opt seccomp:unconfined busybox \
/bin/sh -c 'i=0; while true; do echo $i; i=$((i+1)); sleep 1; done'
```

结果如下：

```
see docker create help :
cch@cch-Vostro-5468:~/docker$ docker create --name looper22 --security-opt seccomp:unconfined busybox /bin/sh -c 'i=0; while true; do echo $i; i=$((expr $i + 1)); sleep 1; done'
e325ecadd26f1f68828cd8867832b31eb8072ad7b536ea5d2753414baa2b6852
```

3.6 恢复：

让新创建的容器 looper22 执行 checkpoint2，然后获取 looper22 的日志输出：

```
cch@cch-Vostro-5468:~/docker$ docker start --checkpoint-dir=/home/2c860274253080546d7655e0966cb0524d86870ad917e546864c1e61257e1b7b/checkpoints/ --checkpoint=checkpoint2 looper22
cch@cch-Vostro-5468:~/docker$ docker logs looper22
21
22
23
24
25
26
```

可以看到输出是从21开始的（前面有说终止于20），这说明我们迁移是没有问题的。

三、实验体会：

这个实验搞得我很自闭。。在之前提交的实验报告里并未完成全部任务，Docker 的迁移还差一步没有实现。后来重新装了 Docker 和 CRIU（换了另外一个版本），发现按照官网所给示例就可以跑通了！看来是之前在安装时版本的匹配以及依赖包出现了一些问题。

四、参考资料：

<https://blog.csdn.net/NelsonCheung/article/details/109164437>

<https://www.cnblogs.com/shmily3929/p/12085163.html>

<https://criu.org/Docker>

<https://phpor.net/blog/post/4956>

https://blog.csdn.net/victory_lei/article/details/97099769

https://blog.csdn.net/weixin_43142797/article/details/105895167

<https://www.cnblogs.com/xiangzhuo/p/9484449.html>