虚拟内存实验指南

ICS回课

rpm工具介绍

- 实验需要用到rpm工具，一个可以格式化查看pagemap的命令行工具
- 仓库地址：https://github.com/EchoStone1101/rpm
- 基本用法（注意由于权限问题，要使用sudo或者在管理员权限下运行）：

命令	效果
sudo ./rpm <pid>	查看<pid>的所有内存区域
sudo ./rpm <pid> addr	查看<pid>的虚拟地址addr的信息
sudo ./rpm <pid> addr –a	查看<pid>的虚拟地址addr对应的内存区域的所有虚拟地址的信息

实验代码

实现代码存放在github中，可以通过如下命令拉取并编译

- git clone https://github.com/wzf03/vm_demo
- cd vm_demo
- make

并通过以下命令将rpm工具下载到实验代码目录以便使用

- wget https://github.com/EchoStone1101/rpm/releases/download/v1.1/rpm
- chmod +x rpm

实验环境

- OS： Ubuntu 23.04
- Arch: x86_64
- Kernel: 6.2.0-39-generic

NOTE: 由于环境的复杂性，不能保证在其他环境下能复现实验结果。例如，ICS CLASSMACHINE由于是容器，没有读取/proc/<pid>/pagemap的权限，所以无法使用rpm工具；WSL由于内核版本为5.15，处理COW的策略与6.2不同，无法观察到COW实验的结果。

实验一：写时复制(Copy on Write)

- 实验代码在copy-on-write-demo目录下
- 实验程序cow_fork定义了一个全局变量a_number，由于初始化为0，故存放在程序的.bss节中。当程序运行时，会fork出一个子进程并打印父子进程的信息，然后暂停。但程序接收到SIGUSR1信号时，程序会将a_number的值改为1。通过观察fork后和发送信号后程序的data区域的虚拟地址空间和对应的物理地址空间信息，可以观察到写时复制的效果。

实验步骤

1. 在**后台**运行cow_fork实验程序，观察到程序输出了父子进程的pid

```
❯ ./copy_on_write_demo/cow_fork &
[1] 601038
parent pid: 601038
child pid: 601040
```

2. 在使用rpm工具查看父子进程的虚拟地址空间结构。可能可以观察到进程的栈地址随机化的效果；以及虚拟地址空间中有多个路径结尾为cow_fork的区域，通过他们权限以及offset，结合readelf –l的输出，我们可以判断出各个区域对应的节。同时，父子进程对应虚拟地址区域的地址范围相同。

NOTE：为什么说栈地址随机化（ASLR）的效果是可能观察到的？
因为gcc是默认关闭ASLR的，有些发行版提供的gcc会在编译时把ASLR更改为默认打开（如Archlinux，Ubuntu），但有些发行版不会（如Rocky Linux）。由于笔者的实验环境是Ubuntu，所以可以观察到ASLR的效果

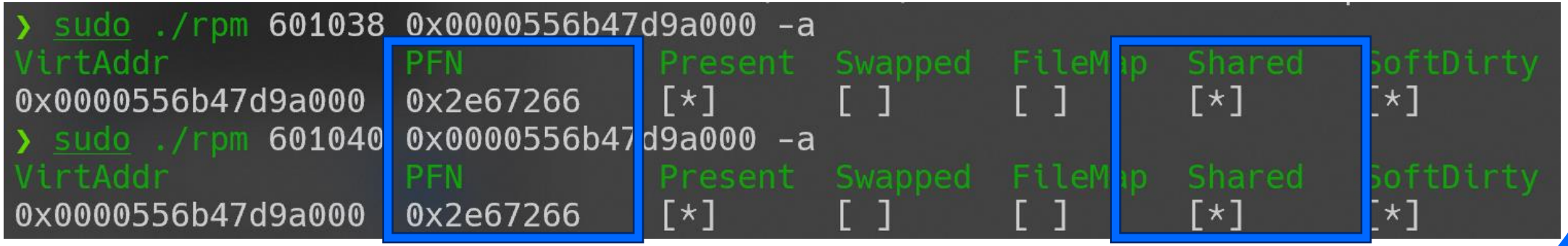
```
❯ sudo ./rpm 601038
Memory Region (size)                                Perm    Offset    Device    Inode      Path
0x0000556b47d96000-0x0000556b47d97000 (    4K)      r--p    0         253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d97000-0x0000556b47d98000 (    4K)      r-xp    4096      253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d98000-0x0000556b47d99000 (    4K)      r--p    8192      253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d99000-0x0000556b47d9a000 (    4K)      r--p    8192      253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d9a000-0x0000556b47d9b000 (    4K)      rw-p    12288     253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b48bcc000-0x0000556b48bed000 (  132K)      rw-p    0         0:0       0          [heap]
0x00007fe7c6c00000-0x00007fe7c6c22000 (  136K)      r--p    0         252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6c22000-0x00007fe7c6d9a000 ( 1504K)      r-xp    139264    252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6d9a000-0x00007fe7c6df2000 (  352K)      r--p    1679360   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6df2000-0x00007fe7c6df6000 (   16K)      r--p    2035712   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6df6000-0x00007fe7c6df8000 (    8K)      rw-p    2052096   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6f91000-0x00007fe7c6f92000 (    4K)      r--p    0         252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6f92000-0x00007fe7c6fba000 (  160K)      r-xp    4096      252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6fba000-0x00007fe7c6fc4000 (   40K)      r--p    167936    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6fc4000-0x00007fe7c6fc6000 (    8K)      r--p    208896    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6fc6000-0x00007fe7c6fc8000 (    8K)      rw-p    217088    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007ffc8f291000-0x00007ffc8f2b2000 (  132K)      rw-p    0         0:0       0          [stack]
0x00007ffc8f37e000-0x00007ffc8f382000 (   16K)      r--p    0         0:0       0          [vvar]
0x00007ffc8f382000-0x00007ffc8f384000 (    8K)      r-xp    0         0:0       0          [vdso]
❯ sudo ./rpm 601040
Memory Region (size)                                Perm    Offset    Device    Inode      Path
0x0000556b47d96000-0x0000556b47d97000 (    4K)      r--p    0         253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d97000-0x0000556b47d98000 (    4K)      r-xp    4096      253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d98000-0x0000556b47d99000 (    4K)      r--p    8192      253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d99000-0x0000556b47d9a000 (    4K)      r--p    8192      253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b47d9a000-0x0000556b47d9b000 (    4K)      rw-p    12288     253:0     329652329  /home/zhuofeng/vm_demo/copy_on_write_demo/cow_fork
0x0000556b48bcc000-0x0000556b48bed000 (  132K)      rw-p    0         0:0       0          [heap]
0x00007fe7c6c00000-0x00007fe7c6c22000 (  136K)      r--p    0         252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6c22000-0x00007fe7c6d9a000 ( 1504K)      r-xp    139264    252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6d9a000-0x00007fe7c6df2000 (  352K)      r--p    1679360   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6df2000-0x00007fe7c6df6000 (   16K)      r--p    2035712   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6df6000-0x00007fe7c6df8000 (    8K)      rw-p    2052096   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007fe7c6f91000-0x00007fe7c6f92000 (    4K)      r--p    0         252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6f92000-0x00007fe7c6fba000 (  160K)      r-xp    4096      252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6fba000-0x00007fe7c6fc4000 (   40K)      r--p    167936    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6fc4000-0x00007fe7c6fc6000 (    8K)      r--p    208896    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007fe7c6fc6000-0x00007fe7c6fc8000 (    8K)      rw-p    217088    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007ffc8f291000-0x00007ffc8f2b2000 (  132K)      rw-p    0         0:0       0          [stack]
0x00007ffc8f37e000-0x00007ffc8f382000 (   16K)      r--p    0         0:0       0          [vvar]
0x00007ffc8f382000-0x00007ffc8f384000 (    8K)      r-xp    0         0:0       0          [vdso]
```

rpm输出

- Perm 栏表示区域的读写权限，r为读，w为写，x为执行，p为私有，s为共享，-表示没有对应权限
- Offset 栏表示区域对应文件中的偏移量。可以与readelf的结果相对照。比如offset为4096的就对应了.text等，offset为12288的对应了.data和.bss
- Device 栏表示区域对应的物理设备。由于实验环境的/usr目录和/home分别在两个不同的磁盘分区中，所以不同条目的Device栏对应的值可能不同
- Path 栏表示区域对应的文件的路径

实验步骤

3. 我们想要观察的a_number位于.bss节，对应的区域就是0x0000556b47d9a000。用rpm工具查看，可以发现两个进程相应的虚拟地址对应的PFN(page frame number)相同，即映射到了相同的物理地址，并且被标记为了Shared。



实验步骤

4. 使用kill给父进程发送USR1信号，用rpm工具再次观察，发现父进程对应的PFN发生变化，并且子进程对应的PFN不变。同时，两个虚拟地址的Shared都被取消标记。证明发生了写时复制。

```
❯ kill -USR1 601038
❯ sudo ./rpm 601038 0x0000556b47d9a000 -a
VirtAddr           PFN              Present  Swapped  FileMap  Shared   SoftDirty
0x0000556b47d9a000 0x2767c8f        [*]      [ ]      [ ]      [ ]      [*]
❯ sudo ./rpm 601040 0x0000556b47d9a000 -a
VirtAddr           PFN              Present  Swapped  FileMap  Shared   SoftDirty
0x0000556b47d9a000 0x2e67266        [*]      [ ]      [ ]      [ ]      [*]
```

实验步骤

5. 给子进程发送SIGUSR1信号，使用rpm工具观察。发现子进程的PFN未变，即没有再发生复制。

```
❯ kill -USR1 601040
❯ sudo ./rpm 601038 0x0000556b47d9a000 -a
VirtAddr           PFN              Present  Swapped  FileMap  Shared   SoftDirty
0x0000556b47d9a000  0x2767c8f       [*]      [ ]      [ ]      [ ]      [*]
❯ sudo ./rpm 601040 0x0000556b47d9a000 -a
VirtAddr           PFN              Present  Swapped  FileMap  Shared   SoftDirty
0x0000556b47d9a000  0x2e67266       [*]      [ ]      [ ]      [ ]      [*]
```

实验二：共享库

- 实验代码在share_lib_demo目录下
- add.c和add.h中定义了一个int add(int,int)函数，执行两个int类型的加操作并范围。这里将add.c编译为了libadd.so。
- main1.c调用libadd.c中的add函数将两个数相加并打印。编译为prog1。
- main2.c调用libadd.c中的add函数求1到10之和并打印。编译为prog2。
- prog1和prog2结尾均使用getchar()暂停程序以方便观察。

实验步骤

1. 后台运行两个程序，并复制pid。主要要在share_lib_demo目录下运行，否则程序将无法找到libadd.so

```
❯ cd share_lib_demo
❯ ./prog1&
[4] 610859
1 + 2 = 3
[4]  + suspended (tty input)  ./prog1
❯ ./prog2&
[5] 610874
sum = 55
[5]  + suspended (tty input)  ./prog2
```

2. 使用rpm工具观察两个程序的虚拟内存区域，可以看到对应共享库的Inode相同，证明对应同样的磁盘文件。

```
❯ sudo ../rpm 610859
Memory Region (size)                                Perm    Offset    Device    Inode      Path
0x0000559b0f4ad000-0x0000559b0f4ae000 (    4K)      r--p    0         253:0     329658681  /home/zhuofeng/vm_demo/share_lib_demo/prog1
0x0000559b0f4ae000-0x0000559b0f4af000 (    4K)      r-xp    4096      253:0     329658681  /home/zhuofeng/vm_demo/share_lib_demo/prog1
0x0000559b0f4af000-0x0000559b0f4b0000 (    4K)      r--p    8192      253:0     329658681  /home/zhuofeng/vm_demo/share_lib_demo/prog1
0x0000559b0f4b0000-0x0000559b0f4b1000 (    4K)      r--p    8192      253:0     329658681  /home/zhuofeng/vm_demo/share_lib_demo/prog1
0x0000559b0f4b1000-0x0000559b0f4b2000 (    4K)      rw-p    12288     253:0     329658681  /home/zhuofeng/vm_demo/share_lib_demo/prog1
0x0000559b0fe8d000-0x0000559b0feae000 (  132K)      rw-p    0         0:0       0          [heap]
0x00007f08f4e00000-0x00007f08f4e22000 (  136K)      r--p    0         252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f08f4e22000-0x00007f08f4f9a000 ( 1504K)      r-xp    139264    252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f08f4f9a000-0x00007f08f4ff2000 (  352K)      r--p    1679360   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f08f4ff2000-0x00007f08f4ff6000 (   16K)      r--p    2035712   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f08f4ff6000-0x00007f08f4ff8000 (    8K)      rw-p    2052096   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f08f50e8000-0x00007f08f50e9000 (    4K)      r--p    0         253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f08f50e9000-0x00007f08f50ea000 (    4K)      r-xp    4096      253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f08f50ea000-0x00007f08f50eb000 (    4K)      r--p    8192      253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f08f50eb000-0x00007f08f50ec000 (    4K)      r--p    8192      253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f08f50ec000-0x00007f08f50ed000 (    4K)      rw-p    12288     253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f08f50ef000-0x00007f08f50f0000 (    4K)      r--p    0         252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f08f50f0000-0x00007f08f5118000 (  160K)      r-xp    4096      252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f08f5118000-0x00007f08f5122000 (   40K)      r--p    167936    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f08f5122000-0x00007f08f5124000 (    8K)      r--p    208896    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f08f5124000-0x00007f08f5126000 (    8K)      rw-p    217088    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007ffe43632000-0x00007ffe43653000 (  132K)      rw-p    0         0:0       0          [stack]
0x00007ffe43799000-0x00007ffe4379d000 (   16K)      r--p    0         0:0       0          [vvar]
0x00007ffe4379d000-0x00007ffe4379f000 (    8K)      r-xp    0         0:0       0          [vdso]
❯ sudo ../rpm 610874
Memory Region (size)                                Perm    Offset    Device    Inode      Path
0x000055b605e4a000-0x000055b605e4b000 (    4K)      r--p    0         253:0     329658682  /home/zhuofeng/vm_demo/share_lib_demo/prog2
0x000055b605e4b000-0x000055b605e4c000 (    4K)      r-xp    4096      253:0     329658682  /home/zhuofeng/vm_demo/share_lib_demo/prog2
0x000055b605e4c000-0x000055b605e4d000 (    4K)      r--p    8192      253:0     329658682  /home/zhuofeng/vm_demo/share_lib_demo/prog2
0x000055b605e4d000-0x000055b605e4e000 (    4K)      r--p    8192      253:0     329658682  /home/zhuofeng/vm_demo/share_lib_demo/prog2
0x000055b605e4e000-0x000055b605e4f000 (    4K)      rw-p    12288     253:0     329658682  /home/zhuofeng/vm_demo/share_lib_demo/prog2
0x000055b6073a7000-0x000055b6073c8000 (  132K)      rw-p    0         0:0       0          [heap]
0x00007f2dd0000000-0x00007f2dd0022000 (  136K)      r--p    0         252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f2dd0022000-0x00007f2dd019a000 ( 1504K)      r-xp    139264    252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f2dd019a000-0x00007f2dd01f2000 (  352K)      r--p    1679360   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f2dd01f2000-0x00007f2dd01f6000 (   16K)      r--p    2035712   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f2dd01f6000-0x00007f2dd01f8000 (    8K)      rw-p    2052096   252:2     1705442    /usr/lib/x86_64-linux-gnu/libc.so.6
0x00007f2dd03ec000-0x00007f2dd03ed000 (    4K)      r--p    0         253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f2dd03ed000-0x00007f2dd03ee000 (    4K)      r-xp    4096      253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f2dd03ee000-0x00007f2dd03ef000 (    4K)      r--p    8192      253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f2dd03ef000-0x00007f2dd03f0000 (    4K)      r--p    8192      253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f2dd03f0000-0x00007f2dd03f1000 (    4K)      rw-p    12288     253:0     329658680  /home/zhuofeng/vm_demo/share_lib_demo/libadd.so
0x00007f2dd03f3000-0x00007f2dd03f4000 (    4K)      r--p    0         252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f2dd03f4000-0x00007f2dd041c000 (  160K)      r-xp    4096      252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f2dd041c000-0x00007f2dd0426000 (   40K)      r--p    167936    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f2dd0426000-0x00007f2dd0428000 (    8K)      r--p    208896    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007f2dd0428000-0x00007f2dd042a000 (    8K)      rw-p    217088    252:2     1705439    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x00007ffc3baf6000-0x00007ffc3bb17000 (  132K)      rw-p    0         0:0       0          [stack]
0x00007ffc3bb7e000-0x00007ffc3bb82000 (   16K)      r--p    0         0:0       0          [vvar]
0x00007ffc3bb82000-0x00007ffc3bb84000 (    8K)      r-xp    0         0:0       0          [vdso]
```

实验步骤

3. 以libadd.so的.text节为例观察虚拟内存地址对应的物理内存地址，发现PFN相同，即对应相同的物理内存地址，并且被标记为Shared。

```
❯ sudo ../rpm 610859 0x00007f08f50e9000
VirtAddr           PFN           Present  Swapped  FileMap  Shared   SoftDirty
0x00007f08f50e9000  0x32796a8    [*]      [ ]      [*]      [*]      [*]
❯ sudo ../rpm 610874 0x00007f2dd03ed000
VirtAddr           PFN           Present  Swapped  FileMap  Shared   SoftDirty
0x00007f2dd03ed000  0x32796a8    [*]      [ ]      [*]      [*]      [*]
```

实验三：mmap

- 实验代码在mmap_demo目录下
- gen.c随机生成1,000,000个double类型的随机数并将结果使用mmap以二进制的格式写入到data.bin文件中。
  - 其中ftruncate的效果是拓展文件大小以便mmap写入。
- read.c使用mmap读取data.bin的内容并统计随机数的分布情况。
- 建议阅读代码以体会mmap在读写二进制数据上的便利。

实验步骤

1. 依次运行gen程序和read程序，可以观察到data.bin被正确生成；read程序成功统计随机数的分布情况。

```
❯ cd mmap_demo
❯ ls
Makefile  gen  gen.c  read  read.c
❯ ./gen
❯ ls
Makefile  data.bin  gen  gen.c  read  read.c
❯ ./read
Out of range: 0
0.0 - 0.1: 100197
0.1 - 0.2: 100094
0.2 - 0.3: 99916
0.3 - 0.4: 100045
0.4 - 0.5: 99779
0.5 - 0.6: 100073
0.6 - 0.7: 99857
0.7 - 0.8: 99570
0.8 - 0.9: 100517
0.9 - 1.0: 99952
```

实验步骤

2. 将gen.c的第23行中MAP_SHARED改为MAP_PRIVATE，即将内存映射类型从共享对象改为私有对象。删除之前生成的data.bin后重新编译运行gen程序，发现此时data.bin中全为0，证明对私有对象的修改不会反应到磁盘文件中。

```
❯ vim gen.c
❯ make
cc -O2 -Wall -o gen gen.c
❯ rm data.bin
❯ ./gen
❯ ls
Makefile  data.bin  gen  gen.c  read  read.c
❯ ./read
Out of range: 0
0.0 - 0.1: 1000000
0.1 - 0.2: 0
0.2 - 0.3: 0
0.3 - 0.4: 0
0.4 - 0.5: 0
0.5 - 0.6: 0
0.6 - 0.7: 0
0.7 - 0.8: 0
0.8 - 0.9: 0
0.9 - 1.0: 0
```

实验四：代码段共享

- 不只是父子进程，只要是相同的可执行文件，就会共享代码段。
- 可以使用sleep程序来实验该特性。

实验步骤

1. 后台启动两个sleep程序，记录pid。

```
❯ sleep 1000 &
[1] 641125
❯ sleep 1000 &
[2] 641133
```

2. 使用rpm工具具体查看代码段对应的物理内存地址，发现完全一致。注：省略了查询代码段对应虚拟内存地址的步骤。

```
❯ sudo ./vm_demo/rpm 641125 0x0000555caeb11000 -a
VirtAddr           PFN          Present  Swapped  FileMap  Shared   SoftDirty
0x0000555caeb11000  0x134310     [*]      [ ]      [*]      [*]      [*]
0x0000555caeb12000  0x1612fb     [*]      [ ]      [*]      [*]      [*]
0x0000555caeb13000  0x1324c9     [*]      [ ]      [*]      [*]      [*]
0x0000555caeb14000  0x13045c     [*]      [ ]      [*]      [*]      [*]
```

```
❯ sudo ./vm_demo/rpm 641133 0x000055a650aba000 -a
VirtAddr           PFN          Present  Swapped  FileMap  Shared   SoftDirty
0x000055a650aba000  0x134310     [*]      [ ]      [*]      [*]      [*]
0x000055a650abb000  0x1612fb     [*]      [ ]      [*]      [*]      [*]
0x000055a650abc000  0x1324c9     [*]      [ ]      [*]      [*]      [*]
0x000055a650abd000  0x13045c     [*]      [ ]      [*]      [*]      [*]
```

END