哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验设计报告

开课学期：2022 年秋季

课程名称：操作系统

实验名称：mmap

实验性质：额外实验

实验时间：10.1 地点：T2

学生班级：11

学生学号：200111132

学生姓名：吴桐

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2022 年 9 月

一、 实验详细设计

Mmap：mmap 指令实现玩具版本

Mmap 的作用是什么？直接把文件的内容 map 进内存中，读写该文件就变成了对内存进行直接操作。一定意义上也满足了题目中说的多个进程共享。

我们用 vma 结构体来记录该文件一些信息：是否可用，开始位置，长度，权限，文件 file 指针等等。并且给 proc 加上 vma 数组，大小我选择的是 nofile 为进程打开的文件最大总数（总的来说判断文件最大个数还是要依据物理内存是否超额度）

Mmap 和 unmmap 都是通过系统调用的方式加入 xv6，这部分稍微略过。

这里交代一下两个函数的实现（sysfile.c）

Mmap 的参数为想映射的地址，长度，偏移，使用权限和 map 的 flag 一些标记，还有文件的 fd 号。之后先要判断一下 map 的权限和 flag 与文件的权限是否有冲突。然后从数组中找出一个未被使用的 vma，对其进行初始化。这里要注意对文件的地址对齐处理。最后别忘了对文件的 refcnt++。

Unmmap 其实是对内存中的一个范围取消映射(有可能没有把整个 vma 的取消完全？I guess，所以这里我还是保守一点，当我找到与参数 addr 对应的 vma 的 start 地址后，从 start 开始清除映射，如果 length 和 vma 不符合，那么会保留，只有刚刚好我才会选择把 vma 给再次可用以及文件引用数-1）这里清楚的时候，如果是 shared 模式，则需要把文件写回去。这里考虑到 vma 是带有文件开始读入的偏移的，但是好像咩有带偏移的写入文件函数，我就模仿着写了一个 filewrite_offset （file.c）用于带偏移的写入。

这里就要谈物理内存的分布问题。其实我们放的文件内存基本上是从 trapframe 的地址开始往下塞。所以我们会记录当前所有 vma 的下界（用于判断是否和使用的重合）。所以我们的清除的时候，就再遍历一下所有被使用的 vma，找到里面最低的下界更新一下。

然后就是实际的读入，我们选择 lazy 的方法。当触发缺页的时候，我们再把它们从文件里面写进对应地址（基本就是 la 的内容就不赘述了（I am lazy too））

二、 实验结果截图

请填写

```
== Test running mmaptest ==
$ make qemu-gdb
(4.9s)
== Test   mmaptest: mmap f ==
  mmaptest: mmap f: OK
== Test   mmaptest: mmap private ==
  mmaptest: mmap private: OK
== Test   mmaptest: mmap read-only ==
  mmaptest: mmap read-only: OK
== Test   mmaptest: mmap read/write ==
  mmaptest: mmap read/write: OK
== Test   mmaptest: mmap dirty ==
  mmaptest: mmap dirty: OK
== Test   mmaptest: not-mapped unmap ==
  mmaptest: not-mapped unmap: OK
== Test   mmaptest: two files ==
  mmaptest: two files: OK
== Test   mmaptest: fork_test ==
  mmaptest: fork_test: OK
== Test usertests ==
$ make qemu-gdb
usertests: OK (285.7s)
== Test time ==
time: OK
Score: 140/140
root@VM-8-8-ubuntu:~/myxv6/github/MIT6.S081-2020-labs#
```