Lab: mmap

Git链接：[Jeery1/xv6-6.S081 at mmap](https://github.com/Jeery1/xv6-6.S081/tree/mmap)

**实验目的：**本实验旨在通过实现 mmap 系统调用，深入理解操作系统中的 内存映射文件（Memory-Mapped Files） 机制，掌握以下核心概念和技术：

1. 理解 mmap 的核心机制

内存映射原理：将文件直接映射到进程的虚拟地址空间，实现文件内容与内存的自动同步。

与传统 I/O 的对比：相比 read/write 系统调用，mmap 避免了用户态与内核态的数据拷贝，提升性能。

应用场景：数据库、动态库加载、大文件处理等。

2. 实现 mmap 的核心功能

基本映射：支持将文件映射到进程的虚拟地址空间，并指定权限（PROT\_READ/PROT\_WRITE）。

延迟加载（Lazy Loading）：仅在访问内存时触发缺页异常，按需加载文件内容到物理内存。

写回机制：对映射内存的修改自动同步到文件（MAP\_SHARED）或仅保留在内存（MAP\_PRIVATE）。

3. 处理缺页异常（Page Fault）

按需加载：在 usertrap() 中处理 PGFLT（缺页异常），动态加载文件内容到物理页。

权限检查：确保进程对映射区域的访问符合 mmap 指定的权限（如禁止写入只读区域）。

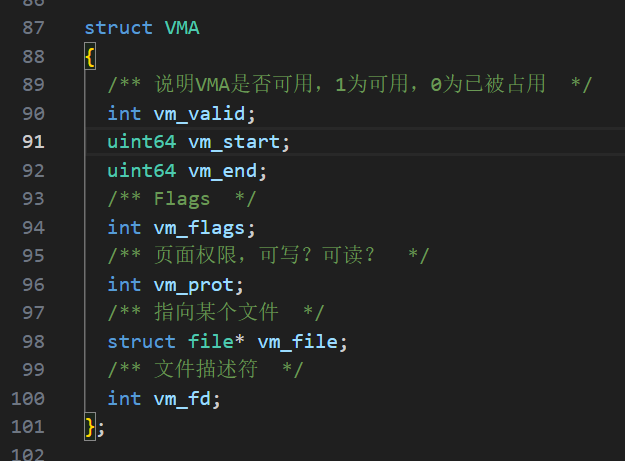
4. 资源管理与释放

munmap 实现：支持解除映射，释放虚拟内存和物理页，并处理脏页写回（MAP\_SHARED 模式）。

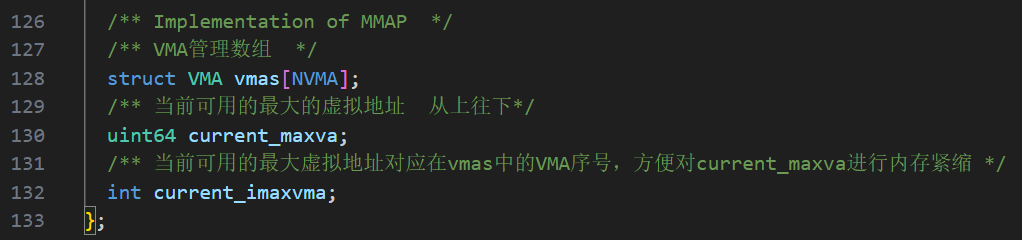
进程退出时的清理：在 exit() 中释放所有映射区域，避免资源泄漏。

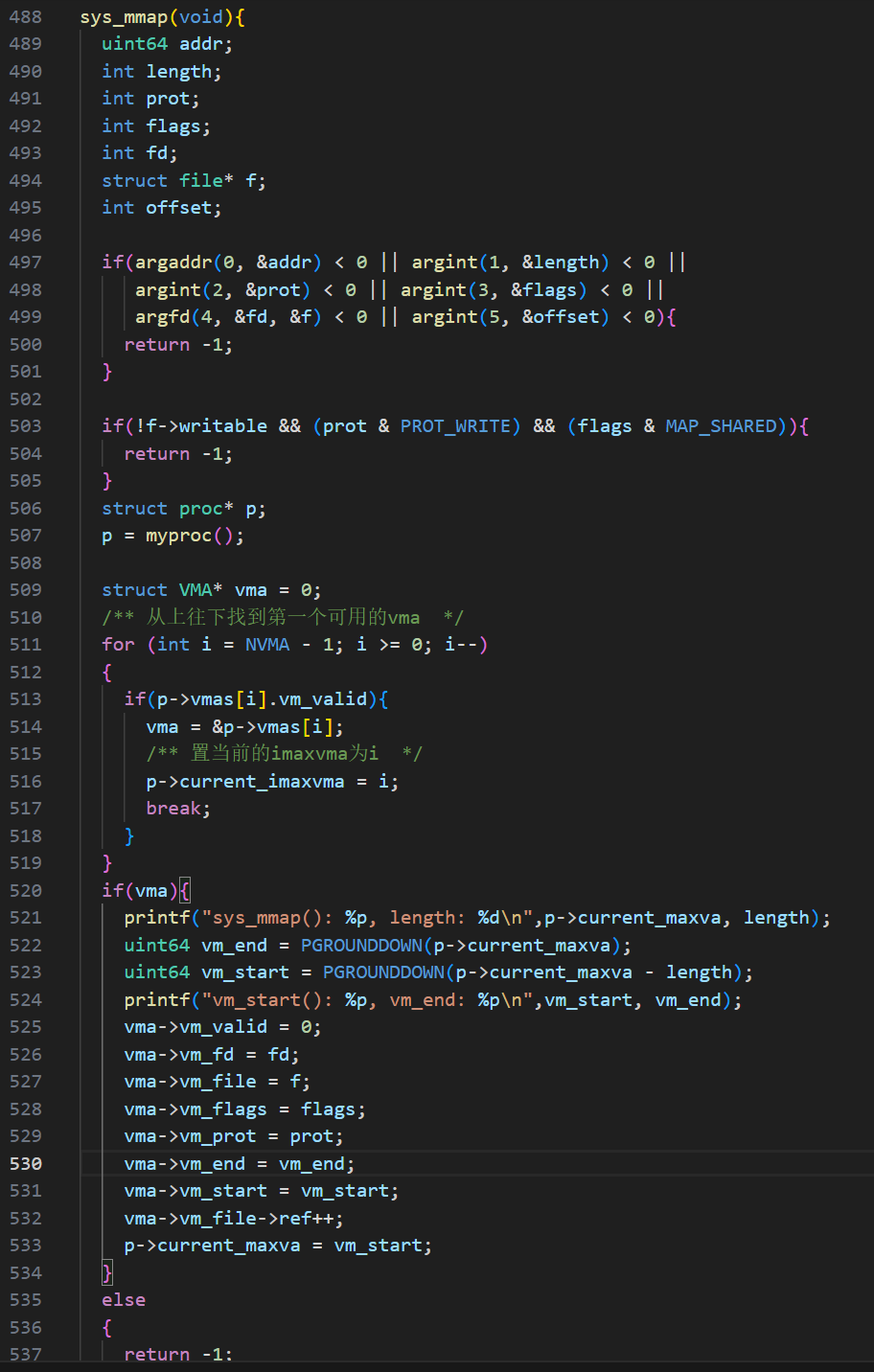
**实验步骤：**

首先定义一个struct VMA 这是虚拟内存区域的基本数据结构，用于管理进程的地址空间映射。

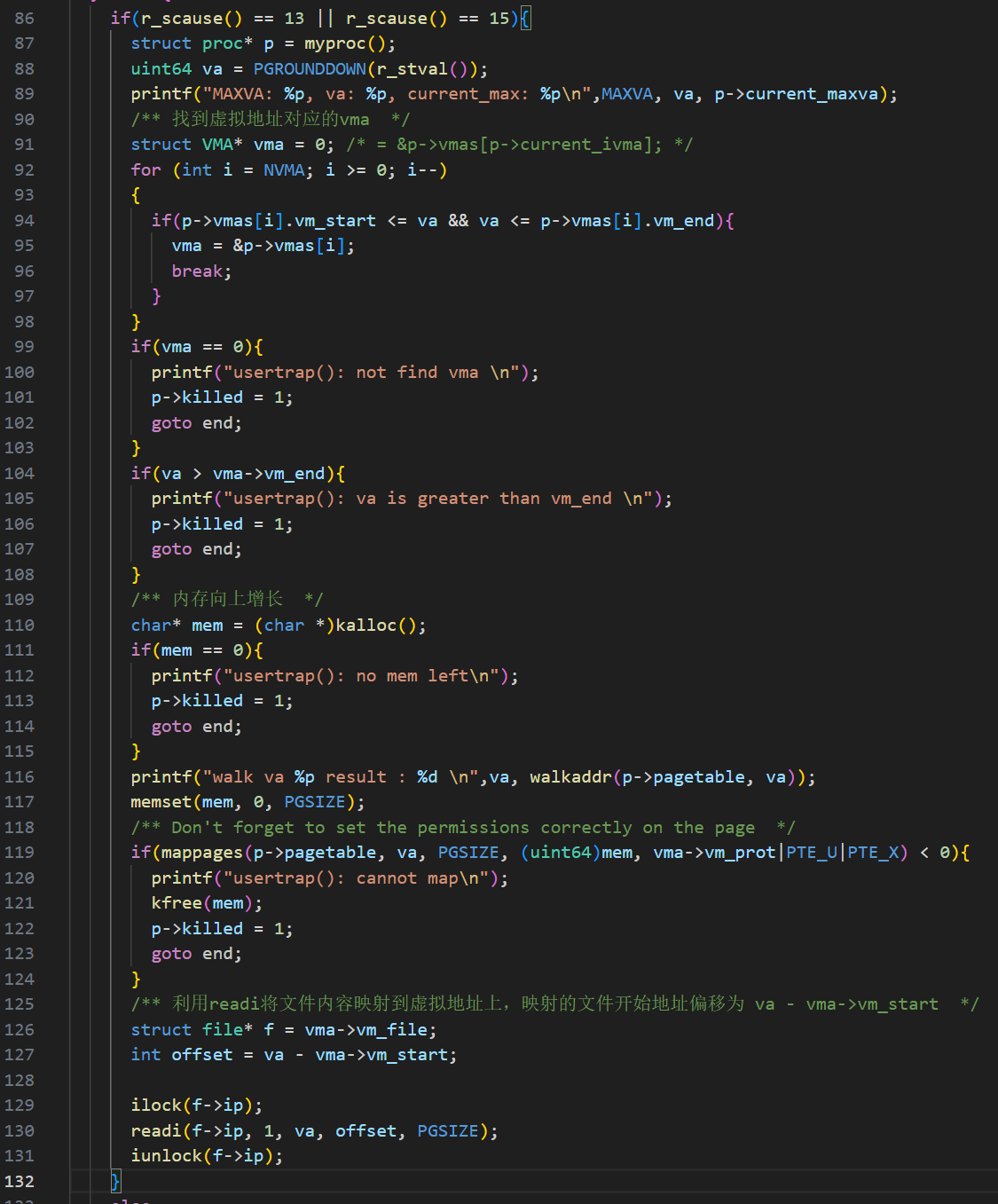


接下来，我们修改proc中的内容，使之能够管理VMA，这里我们定义最多管理NVMA=100个VAM。我们需要一个当前最大VMA地址指针来记录heap段增长到了那个位置





下面开始正式实现mmap。基本思路是：首先找到第一个可用的VMA，然后再根据current\_maxva和传入的参数为其设置各种字段。需要注意的是，vm\_start应该小于vm\_end，因为在之后的trap.c中我们会利用kalloc分配内存，kalloc分配的内存是向上增长的，而我们想要的正是让内存从vm\_start向上增长。



接下来在trap.c中加入缺页异常的处理。当用户态访问未映射的虚拟地址时（r\_scause() == 13 或 15），进入此逻辑。

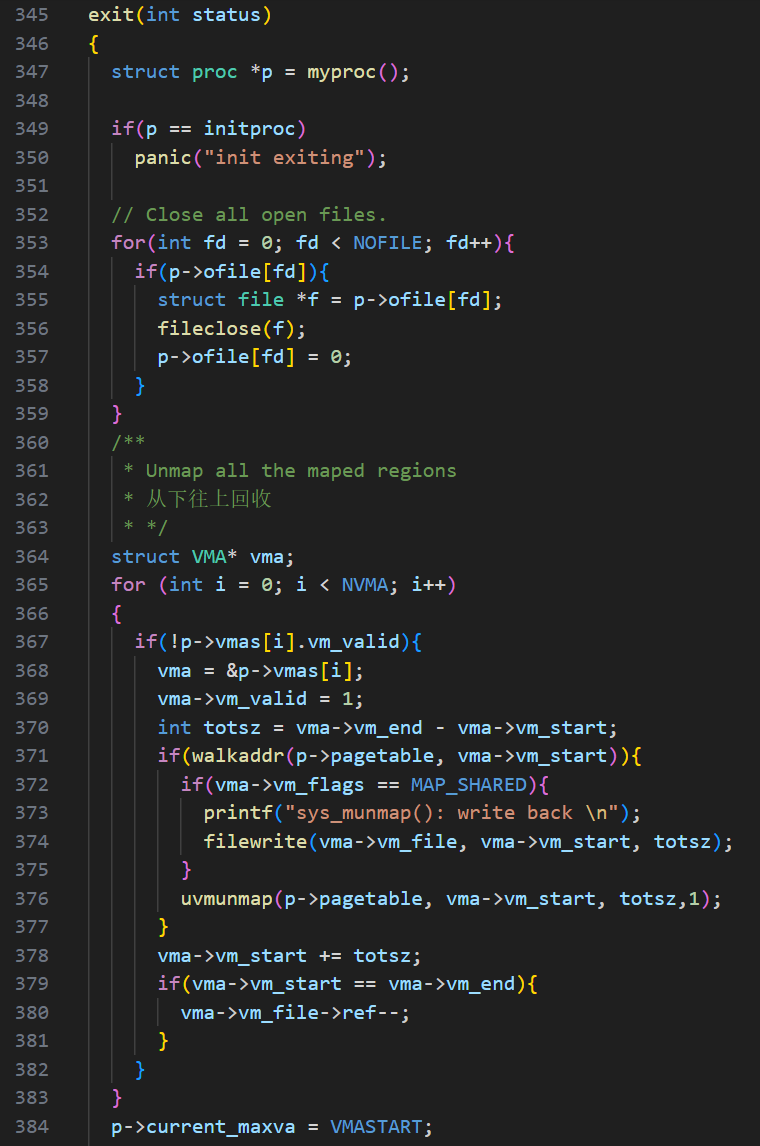
找到触发缺页的虚拟地址对应的 VMA（虚拟内存区域）。分配物理页并映射到用户页表。如果是文件映射，从磁盘读取文件内容到该页。

接下来该实现munmap了。munmap就是释放mmap的空间，当发现mmap的flages是MAP\_SHARED时我们要回写文件，否则不用（本次实验中是这样的）。另外我们我们还需要考虑一个问题，那就是内部碎片的问题，设想下面的情况。VMA[1]被释放了，但是我们的current\_maxva并不能移动。那么在此后再进行mmap时，按照我们的算法这块被释放的内存将永远不会被用到了。

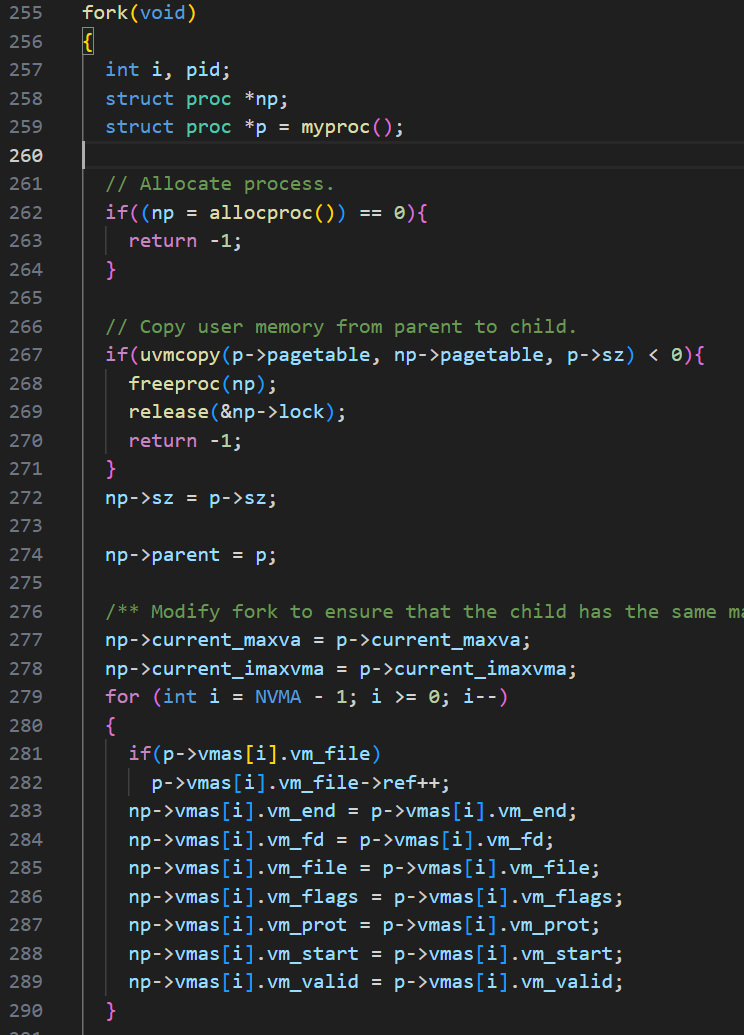


首先从系统调用参数中获取要解除映射的地址和长度。然后从后向前遍历进程的VMA数组，找到包含目标地址的VMA。找到匹配的VMA后：如果页表中存在映射：对于共享映射(MAP\_SHARED)，将修改写回文件，调用 uvmunmap 解除页表映射。调整VMA的起始地址，如果VMA范围变为空，释放相关文件引用并标记VMA为可用。查找第一个可用的VMA，更新进程的最大虚拟地址和索引。如果没有找到，重置为初始值(VMASTART)。

然后，修改exit，与munmap的实现基本一样，不过因为它是释放所有VMA及其映射的内存，因此不需要考虑内存紧缩问题，代码如下：



最后，完成fork时proc中添加字段的复制：



**实验现象：**

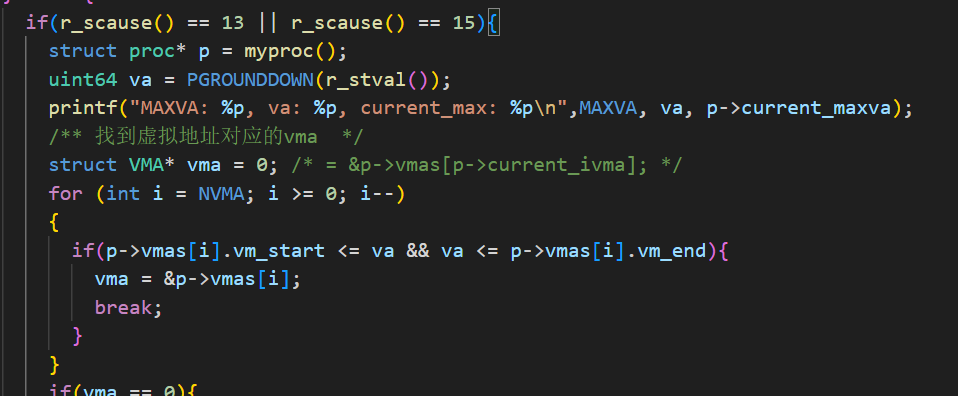


Mmaptest通过

**实验中遇到的问题和解决方法：**

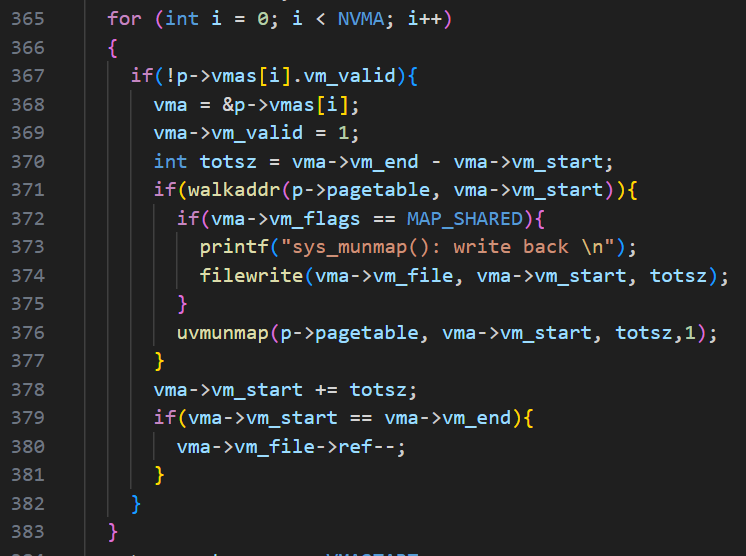
mmap 采用懒加载策略，但缺页处理程序未正确处理 MAP\_PRIVATE 和 MAP\_SHARED 情况，导致访问未映射内存时 panic。

解决办法：修改 usertrap() 中的缺页处理逻辑：



进程调用 exit() 时，未释放 mmap 分配的 VMA 和文件引用，导致内存泄漏。

解决办法：在 exit() 中遍历 VMA 并释放资源：

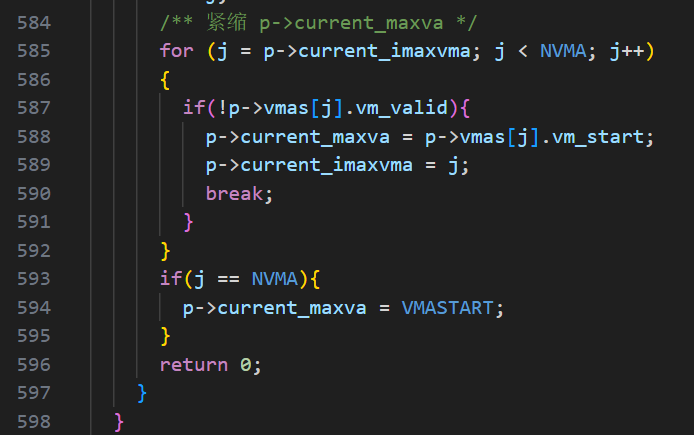


问题：munmap 后未调整 current\_maxva

问题描述

munmap 解除映射后，current\_maxva 未正确更新，导致后续 mmap 可能错误覆盖有效内存。

解决办法：在 munmap 后重新计算 current\_maxva：



**实验心得：**

通过本次mmap实验，我对操作系统的内存管理机制有了更深刻的理解。在实现过程中，我遇到了VMA管理混乱、懒加载缺页处理不当、文件回写错误等多个挑战，这些困难反而成为最好的学习机会。通过反复调试，我不仅掌握了mmap/munmap的核心实现原理，更体会到操作系统设计中性能与安全性的精妙平衡。实验让我认识到，一个看似简单的系统调用背后，需要页表管理、缺页异常处理、文件系统等多个模块的协同工作。这次实践极大地提升了我的系统编程能力和调试技巧，也让我对操作系统的兴趣更加浓厚。这段经历让我明白，只有亲自动手实现，才能真正理解系统设计的精妙之处。