操作系统课程设计

Xv6实验报告

Lab3 页表

|  |  |
| --- | --- |
| Name | 朱从周 |
| ID | 2351893 |
| Number | 42028701 |
| major | 软件工程 |



同济大学

Tongji University

1. **环境搭建**
2. 下载VMware虚拟机，并且下载 Ubuntu 镜像并安装 Ubuntu 系统。
3. 启动Ubuntu，安装本项目所需的所有软件，运行：

$ sudo apt-get update && sudo apt-get upgrade

$ sudo apt-get install git build-essential gdb-multiarch qemu-system-misc gcc-riscv64-linux-gnu binutils-riscv64-linux-gnu

1. 下载xv6内核源码

$ git clone git://github.com/mit-pdos/xv6-riscv.git

1. 更新镜像源

$ sudo nano /etc/apt/sources.list

$ sudo apt-get update

1. 获取源码

$ git clone git://g.csail.mit.edu/xv6-labs-2021

$ cd xv6-labs-2021

$ git checkout pgtbl

1. **实验目的**

一些操作系统（如 Linux）通过在用户空间和内核之间共享只读区域中的数据来加快某些系统调用的速度。这样，在执行这些系统调用时就不需要内核交叉了。本实验旨在学习如何在页表中插入映射，首先需要在 xv6 中的 getpid() 系统调用中实现这一优化。

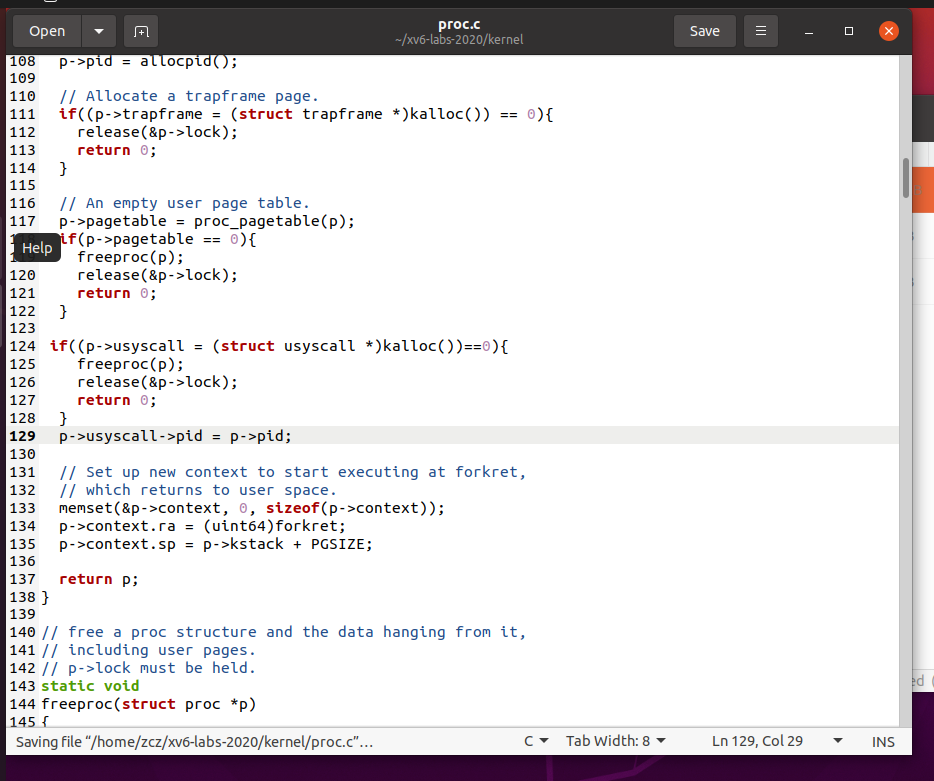
通过在用户空间和内核之间共享一个只读区域中的数据，来加速特定的系统调用。具体而言，通过在进程创建时映射一个只读页，将一个 struct usyscall 结构放置在该页的开头，该结构会存储当前进程的 PID。这将使得在执行某些系统调用时，不需要进行用户空间和内核之间的频繁切换，从而提高系统调用的性能。

深入理解 RISC-V 页表的结构和内容，并提供一个打印页表的函数 vmprint() 。通过这个实验，实现可视化页表的布局，了解页表的层次结构以及如何将虚拟地址映射到物理地址。

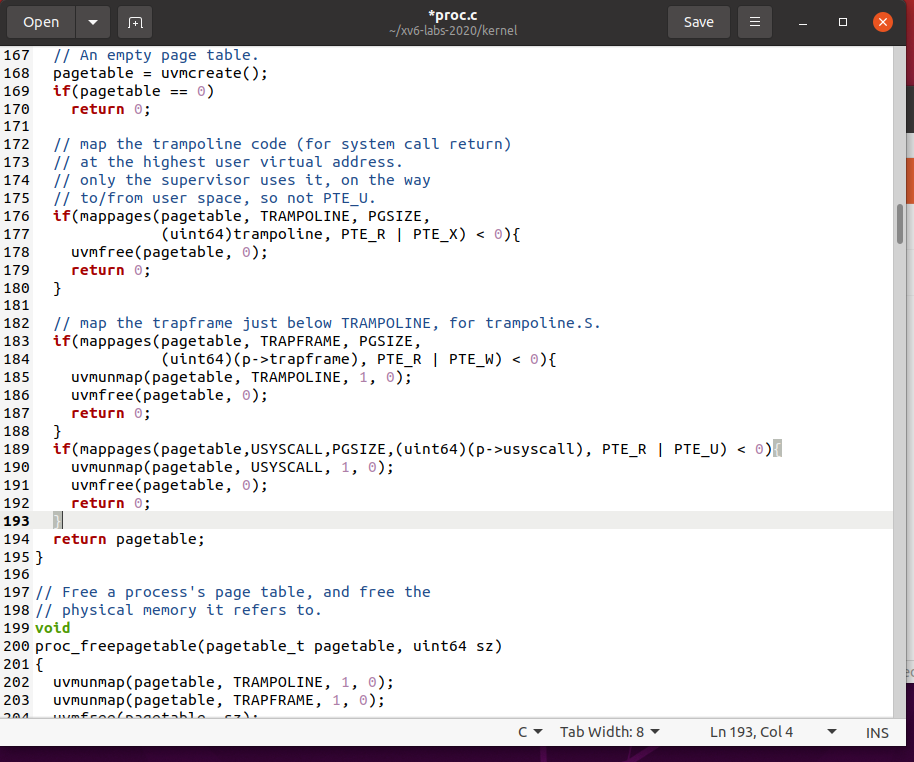
并且向 xv6 内核添加一个新特性，即通过检查 RISC-V 页表中的访问位，实现一个系统调用 pgaccess()，该系统调用可以报告哪些页面已被访问（读取或写入）。这对于垃圾回收器等应用有一定的益处。

**2. 实验内容**

首先在 kernel/proc.h proc 结构体中添加一项指针来保存这个共享页面的地址。之后需要在 kernel/proc.c 的 allocproc() 中为其分配空间(kalloc)。并初始化其保存当前进程的PID。

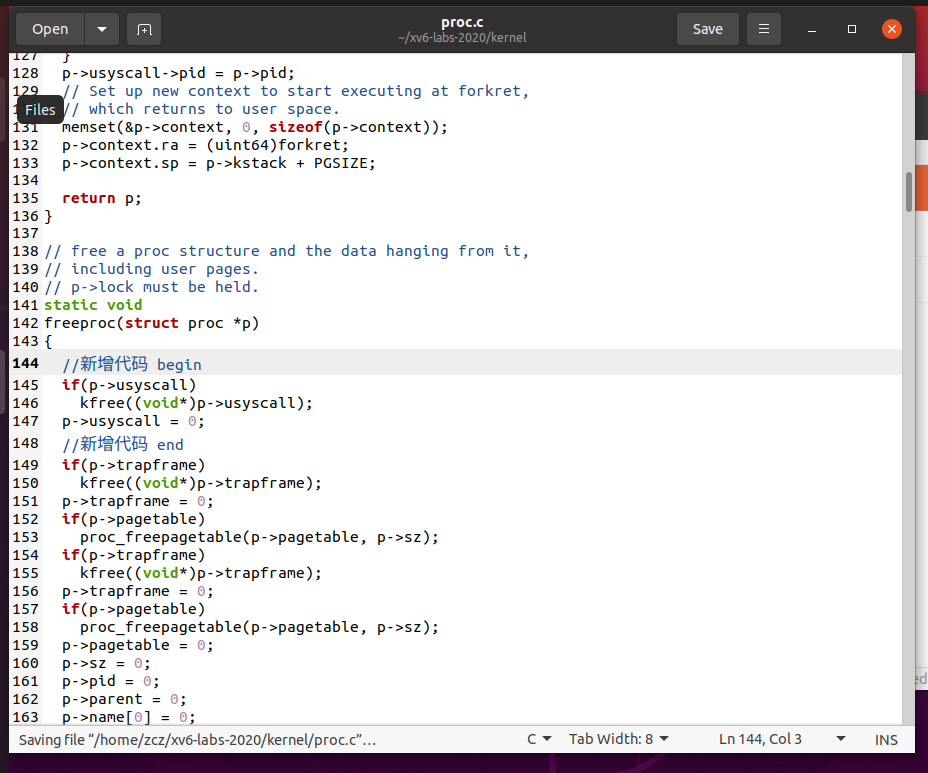


在(kernel/proc.c)中修改proc\_pagetable方法，仿照给trapframe新增映射关系，这里实验有要求许用户空间只读取页面的权限位，所以使用权限PTE\_R与PTE\_U

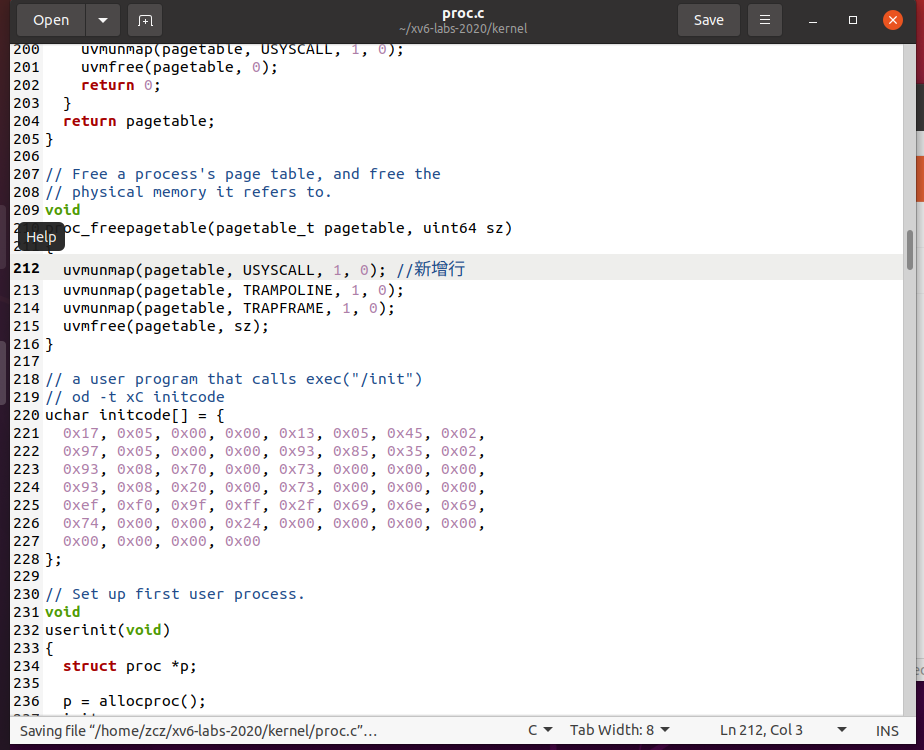


在(kernel/proc.c)中修改freeproc与proc\_freepagetable，在进程释放的时候将对应内存释放掉。

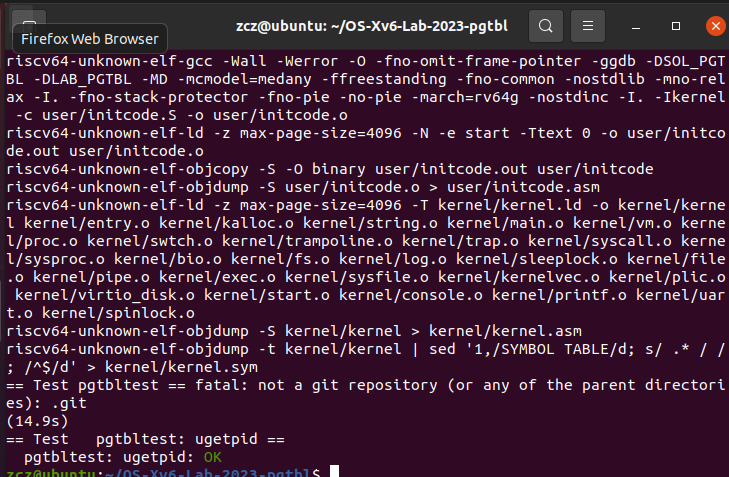
freeproc中增加



proc\_freepagetable中增加



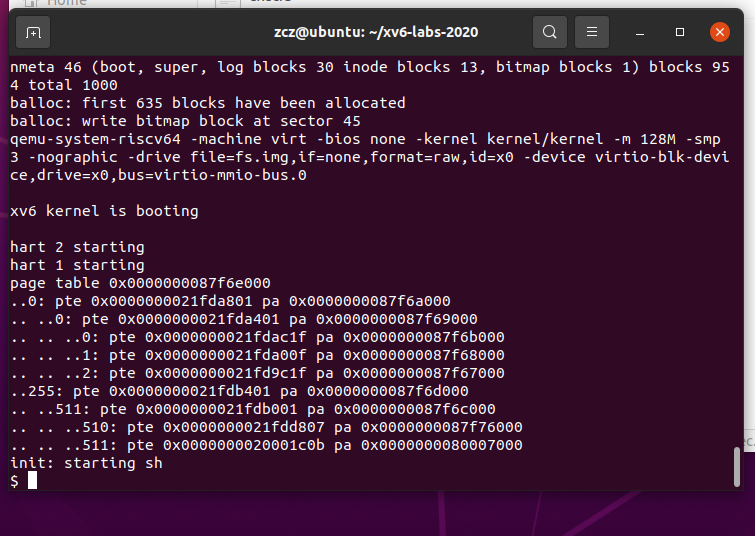
测试结果



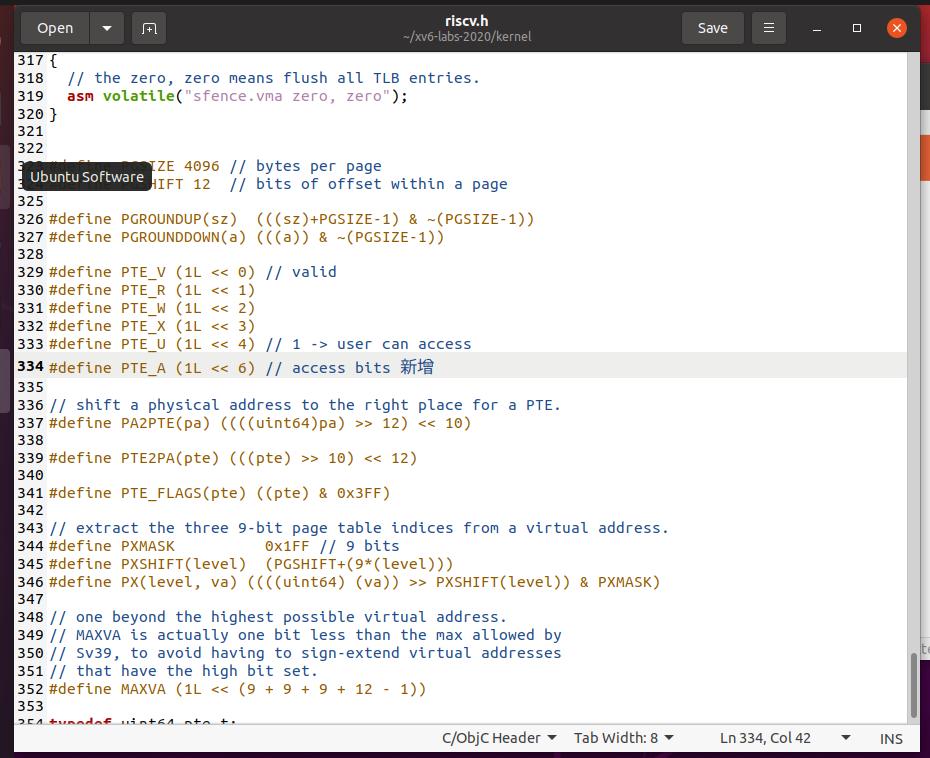
**实现 vmprint 函数**：参考 freewalk 函数，在vm.c中编写 vmprint 函数，该函数将以指定格式递归打印页表的内容。



测试结果：

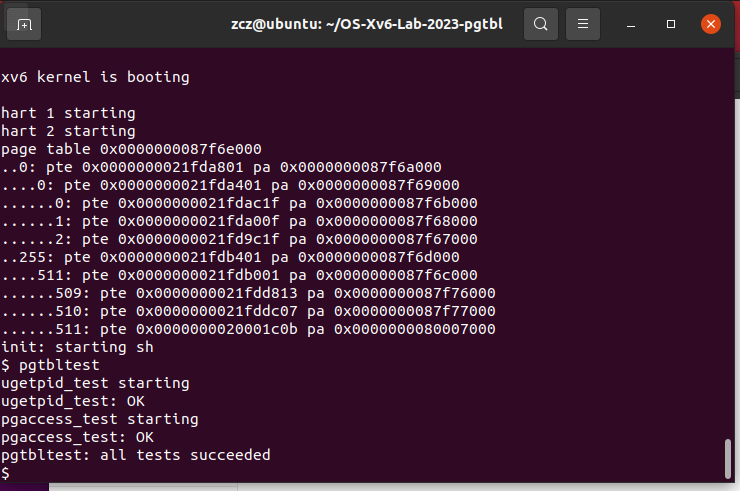


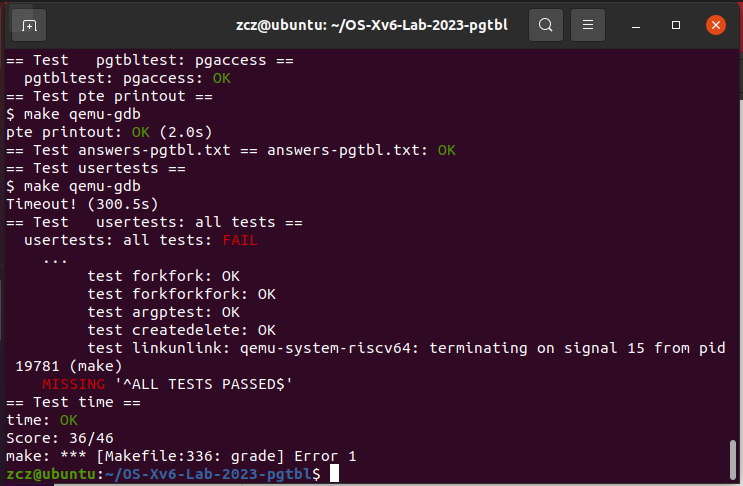
3、确定PTE\_A，访问位在页表项中的位置。查阅手册可以知道是在第六位。因此在kernel/riscv.h中进行如下修改。





测试结果：





**3.实验中遇到的问题和解决方法**

1. **递归错误：** 在递归地遍历页表时，起初在递归的设计上，思路还不太清晰，后来参考了freewalk函数的形式，让我对递归遍历的逻辑有了更好的理解。
2. **格式化输出问题：** 我在格式化输出的语法上出现了一些错误。后来我结合控制台调试，使用 %p 格式化符正确地打印了 64 位的十六进制 PTE 和物理地址。
3. **页表层次理解：** 在理解页表的层次结构时，可能显得过于抽象，但通过观察源码、结合课程教材中的图 3-4和解释 vmprint() 输出的页表内容，我可以更清晰地了解每个级别的页表是如何映射虚拟地址到物理地址的。

**3.实验心得**

通过本次实验，首先我看到了性能优化的重要性，这给了我很大的启示：通过将系统调用的相关数据放在只读页中，以减少内核和用户空间之间的数据传输次数，从而加速系统调用的执行。

此外这个实验使我更深入地理解了系统调用的工作原理以及它们是如何在用户空间和内核空间之间进行通信的。这为您提供了更清晰的操作系统工作流程的认识。通过在每个进程的页表中插入只读页，掌握操作页表的方法，从而实现用户空间与内核空间之间的数据共享。尽管在实验过程中遇到了不少问题，但是这让我对映射与回收、权限授予等有了更深刻的理解。

通过这个实验，我学习了操作系统的内存管理机制，包括页表的结构和作用；理解了如何为进程分配页表，映射虚拟地址到物理地址，以及如何使用页表权限来实现不同的访问控制。

在实现系统调用过程中，我深入了解了内核代码的组织结构和运行方式，以及如何将用户态的请求转换为内核态的操作，并了解了如何从用户空间传递参数到内核空间。

此外我还能够结合教材中的页表结构来编写程序，这更让我理解了上层算法实现和底层硬件逻辑之间的紧密联系，这为我今后的学习打下基础、积累了一定的经验。