Lab3

内核环境：(uname -r） 5.13.0-40-generic

第一部分：

主要结构体描述：

每个进程有一个task\_struct，里面有一个mm\_struct，里面有一个vm\_area\_struct，构成链表（循环之类的），而每个vm\_area\_struct都有一个指向mm\_struct的指针。

struct mm\_struct {

struct vm\_area\_struct \*mmap; // 虚拟区间（VMA）有序链表，按照区间起始地址递增方式组织

struct rb\_root mm\_rb; // VMA 红黑树根节点，将进程所有的 VMA 记录到红黑树中，以提高查找效率

pgd\_t \*pgd; // 页全局目录

int map\_count; // VMA 数量

struct rw\_semaphore mmap\_sem; // 读写信号量

// ...

};

struct vm\_area\_struct {

unsigned long vm\_start; // VMA 起始地址

unsigned long vm\_end; // VMA 终止地址（不包含本身）

struct vm\_area\_struct \*vm\_next, \*vm\_prev; // 链表的后一个/前一个结点

struct rb\_node vm\_rb; // 对应红黑树节点

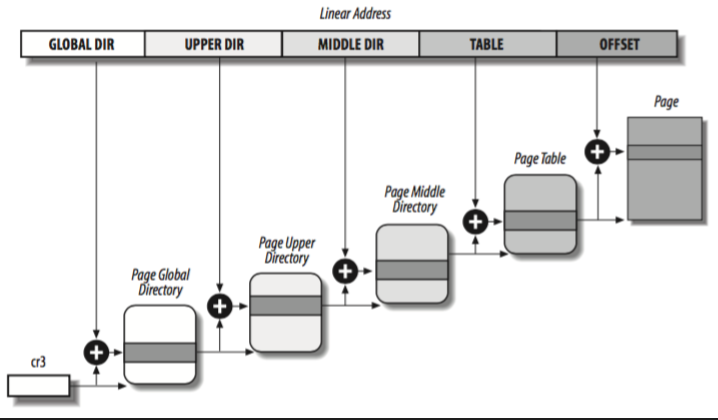
unsigned long vm\_flags; // 权限，在 mm.h 中定义

// ...  
}

伪代码流程：

pid 经过find\_get\_pid->pid结构体 经过 pid\_task->得到task\_struct结构体->得到mm\_struct结构体,根据mm和传入的虚拟地址进行多级页表映射最后得到page结构体。

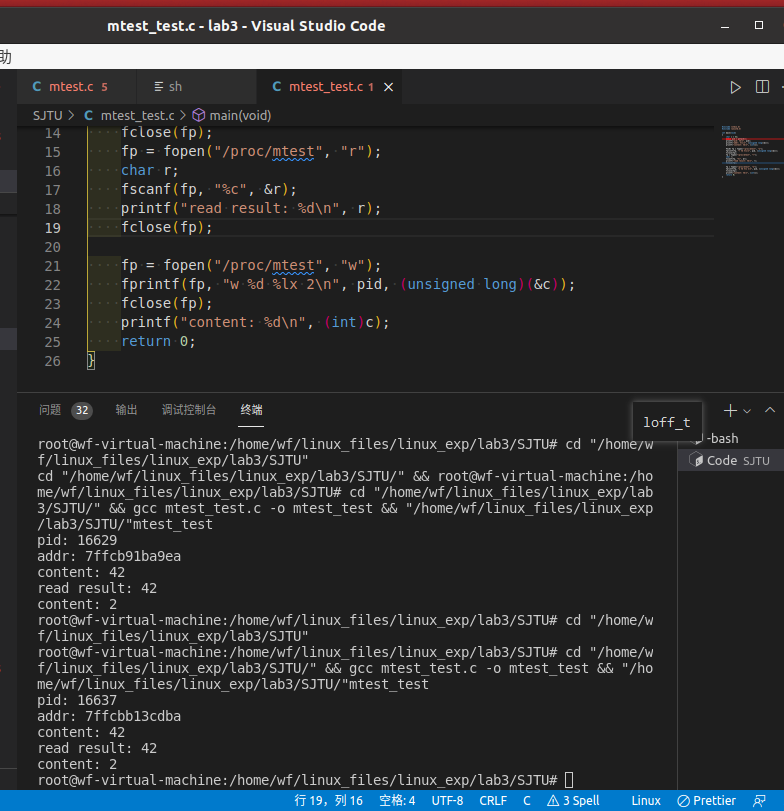
(另一种思路（根据虚拟地址 find\_vma ->得到虚拟块：vm\_area\_struct-> vm\_mm)

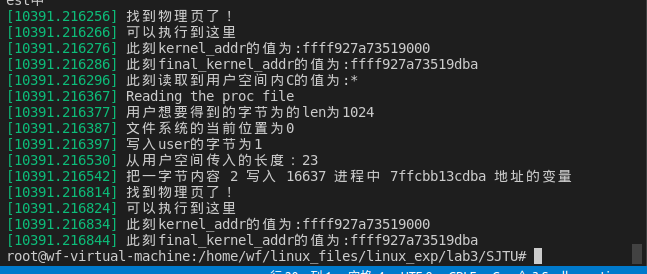


之后通过kmap\_local\_page得到内核空间的虚拟地址，低12位置零，和送进来的虚拟地址低12位（offset）做或操作得到最终要操作的地址。然后对其进行操作即可。

（记得在r情况下修改out\_len为1）。

结果：





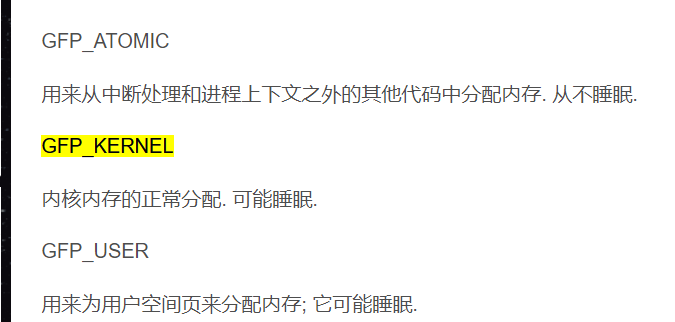
可以看到，内核中输出为\*,原因是42十进制对应的字符是\*(ASCII码)

修改了mtest\_test.c测试文件，不知道什么原因使用int c读入的char类型值不对，改成char就可以了。（21,22行）

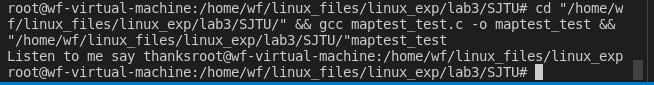
第二部分：

根据实验要求文件的提示做即可。

调用alloc\_page时用常规的GFP\_KERNEL；查看remap\_pfn\_range的参数填写。



结果：



感想：

这次实验需要自己查询很多API，而且发现有不同的方式（函数）来实现所需要的功能，很好的提高了自己根据需求查询函数和内核编程能力，同时也意识到自己能力尚且不足，需要和同学以及助教交流方且完成，独立工作的能力未来也很重要。

最后总结了做实验过程中的查找的比较好的资料供自己之后回顾看：

<https://www.jianshu.com/p/89cd35010120>

https://blog.csdn.net/prike/article/details/52722934

dmesg显示内核信息 dmesg -c（clear）

分页存储：<https://blog.csdn.net/weixin_43914604/article/details/105907291>

页，页表，页表项：<https://blog.csdn.net/HaoDaWang/article/details/78767830>

Mmap: <https://segmentfault.com/a/1190000014672234>

查看源码：https://elixir.bootlin.com/linux/latest/source

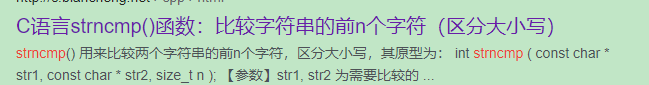
find\_vma函数功能描述：find\_vma( )函数根据一个属于某个进程的虚拟地址，找到其所属的进程虚拟区间，并返回相应的vma\_area\_struct结构体指针。

mm：是进程整个用户空间的抽象，也是总的控制结构，一个进程只有一个mm\_struct结构，一个进程整个用户空间通常有若干离散的虚拟区间，这些虚拟区间由vm\_area\_struct结构描述。

(为什么把进程的用户空间划分为一个个区间？这是因为虚拟内存的的来源不同，有的来自可执行映像，有的来自共享库，而有的来自动态分配的内存区，对不同的区间具有不同的访问权限，有可能会有不同的操作，所以需要把进程的用户空间分割管理，并用虚存区处理函数)

在linux中，pfn全称“page frame number”，是物理内存区域编号

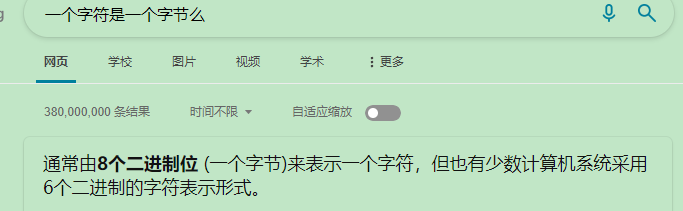




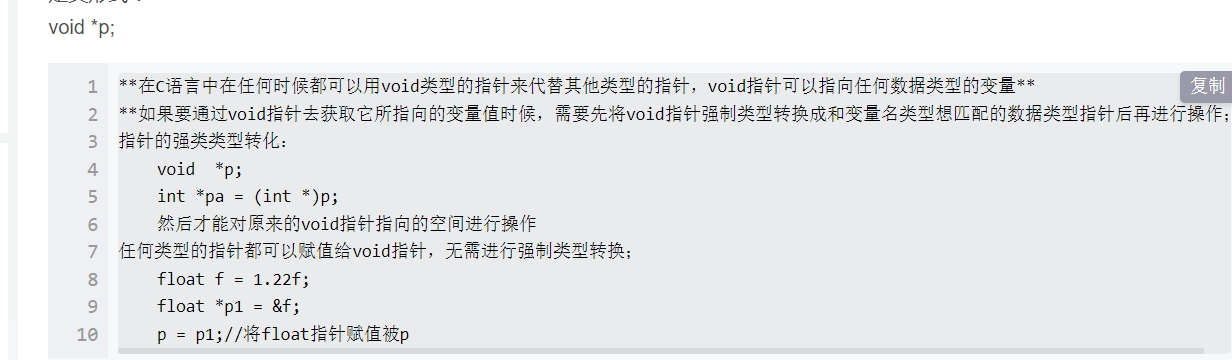


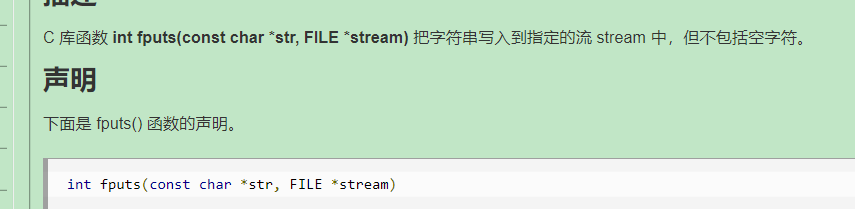
输出的虚拟地址是48位的：

https://cn.etsoutdoors.com/858614-why-do-x86-64-systems-MQBRCB



宏：https://zhuanlan.zhihu.com/p/98470867





内核空间与物理地址之间有直接的映射关系：

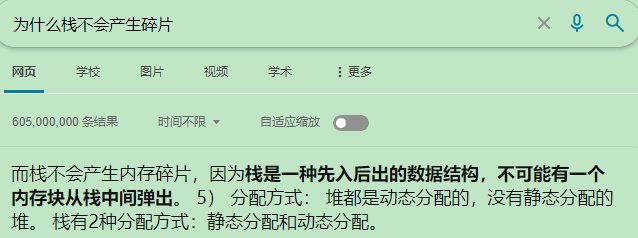
https://www.cnblogs.com/bizhu/archive/2012/10/09/2717303.html











分段 分页：

<https://blog.csdn.net/OOFFrankDura/article/details/84403741>

直接赋值 和 间接赋值：

<https://blog.csdn.net/shulianghan/article/details/121458154>

指针：

https://blog.csdn.net/qq\_21583681/article/details/78572009#commentBox

问题：

char\* buf = (char\*) mmap(NULL, SIZE, PROT\_READ, MAP\_PRIVATE, fd, 0);需要SIZE

ret = remap\_pfn\_range(vma, vma->vm\_start, page\_to\_pfn(page), (unsigned long)index, vma->vm\_page\_prot);需要一个size((unsigned long)index)，二者有什么关系？

如何找到指针变量的地址？