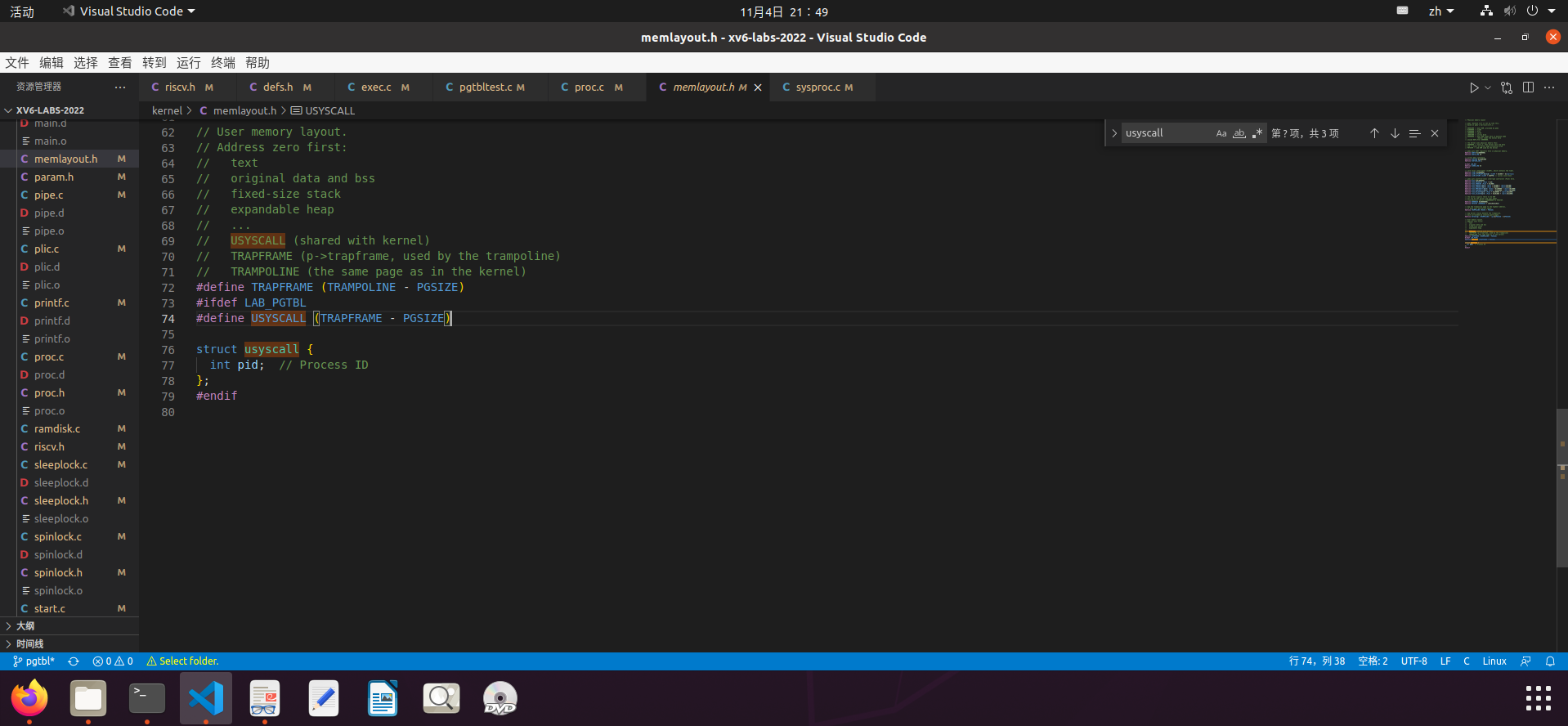
**Lab3实验报告**

一、实验思路、相关代码截图、结果截图

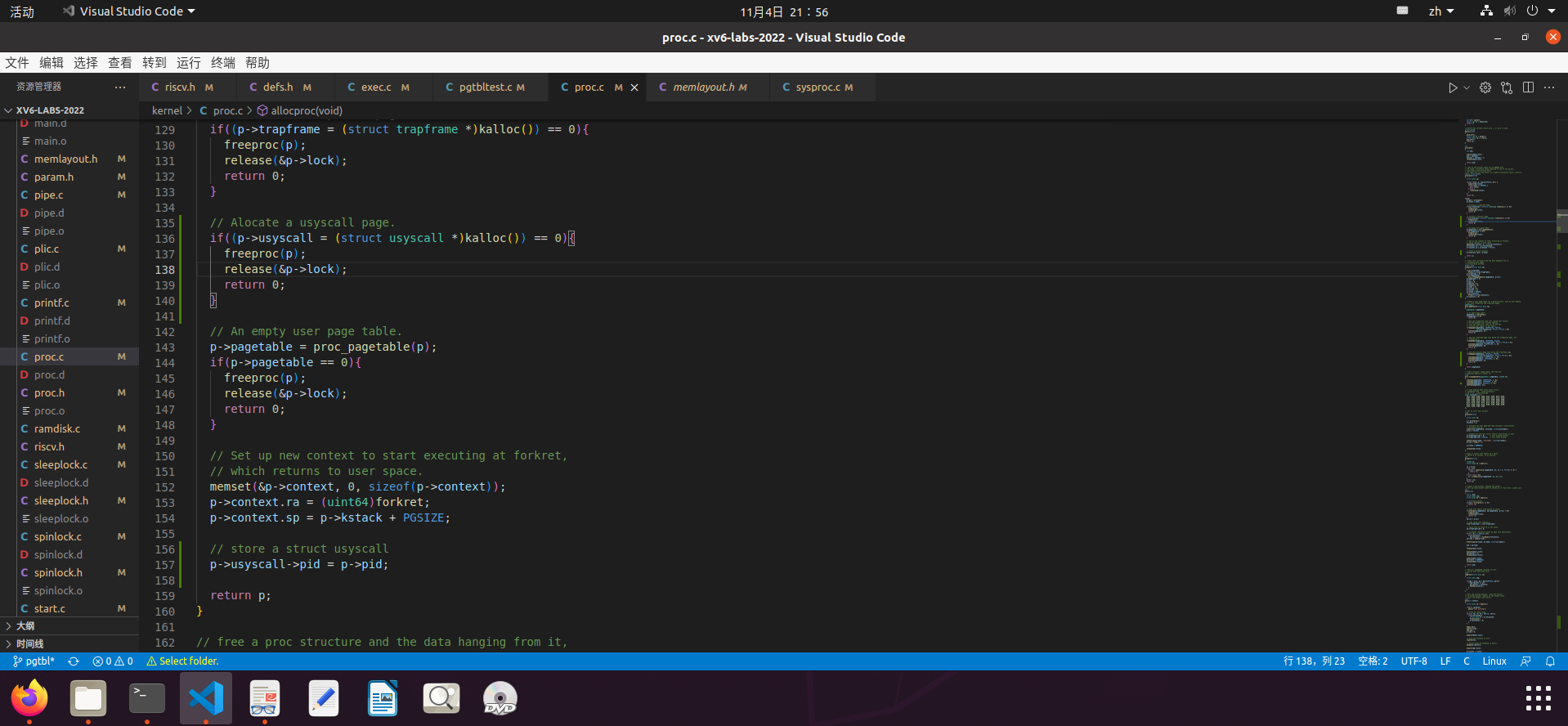
PartA: Speed up system calls:

在xv6中，如果用户态调用系统调用，就会切换到内核态，这中间一定是有开销的，至少CPU要保存用户态进程的上下文，然后CPU被内核占有，系统调用完成后再切换回来。

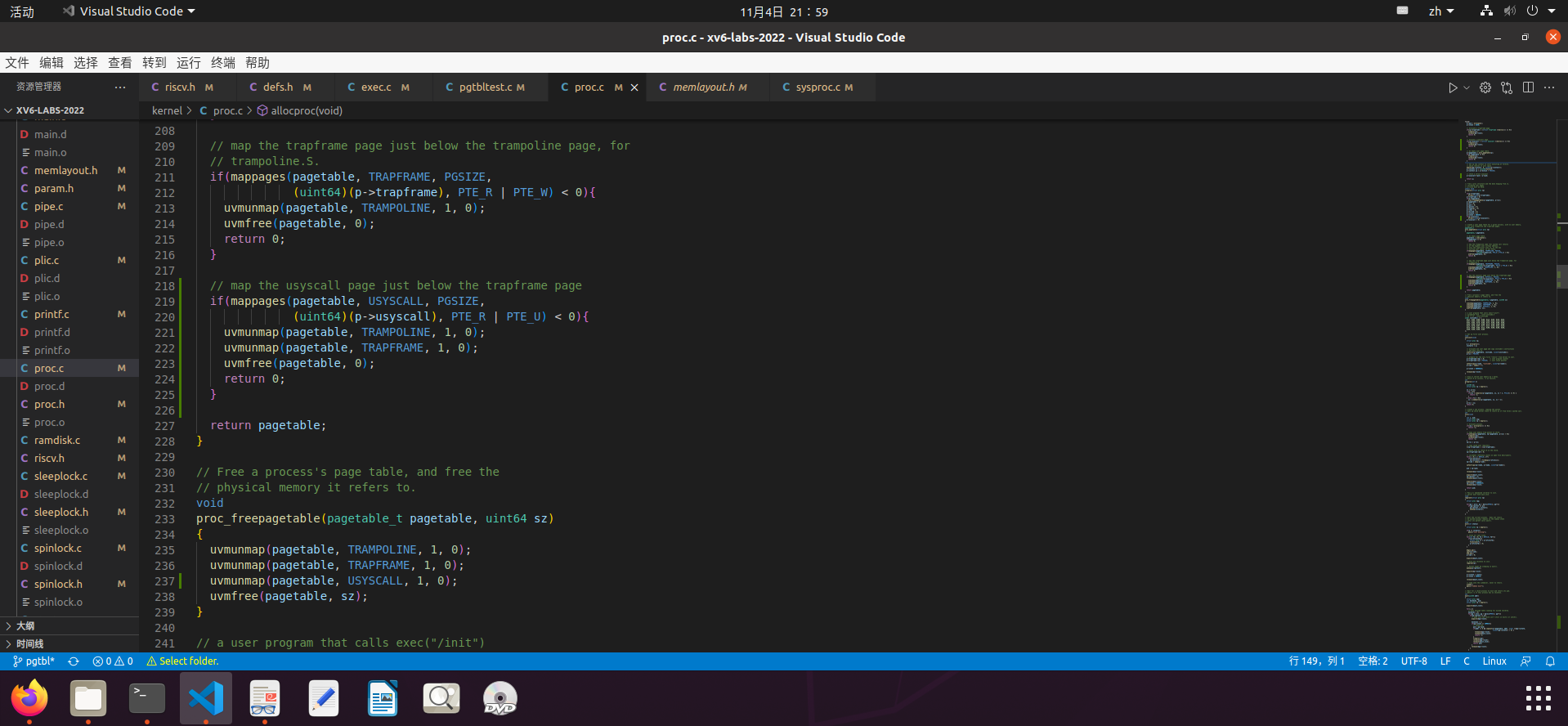
这个实验的目的就是要加速getpid()函数，根据文档提供的思路，首先为每一个进程多分配一个虚拟地址位于USYSCALL的页，然后这个页的开头保存一个usyscall结构体，结构体中存放这个进程的pid。



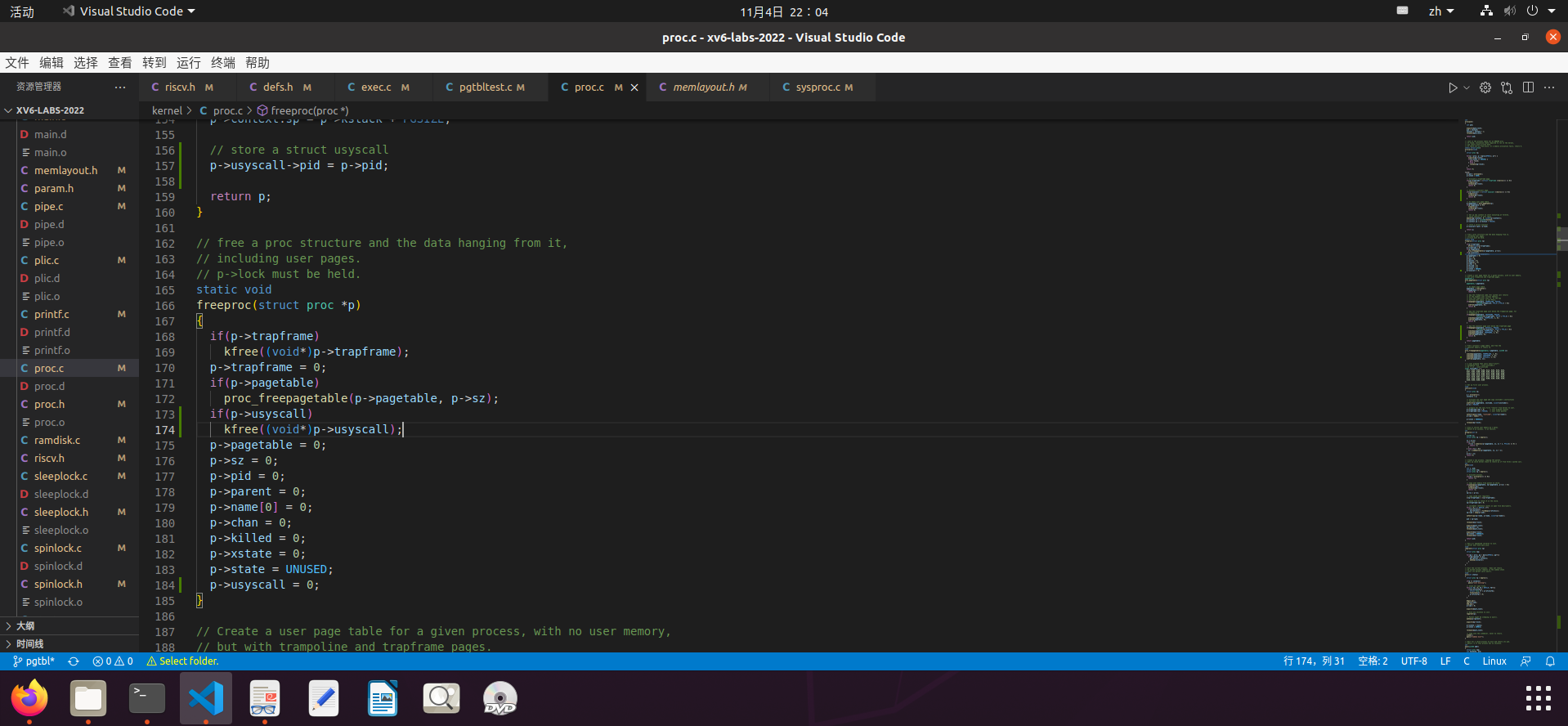
在proc.c中分配一个页给usyscall，并且对其中的参数pid进行初始化。



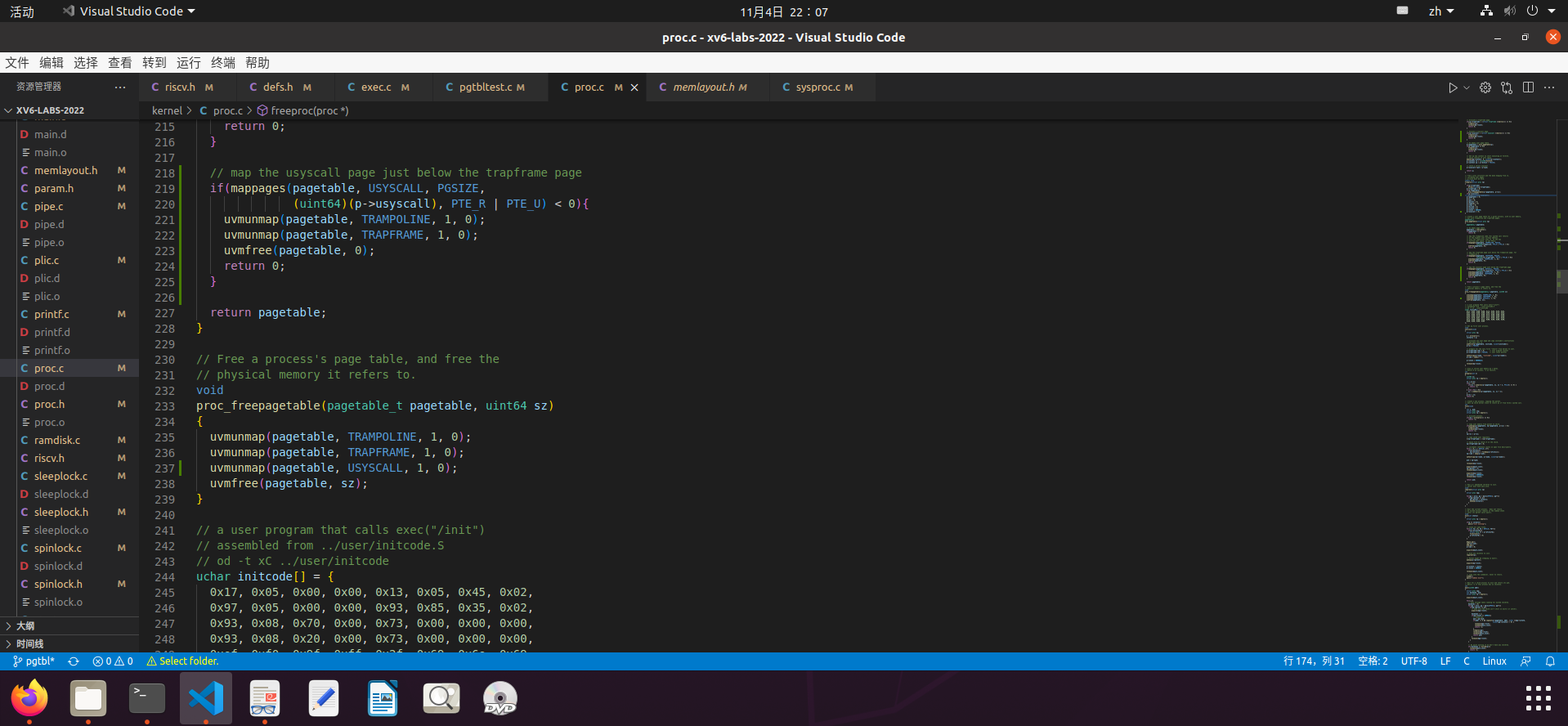
根据提示，在proc\_pagetable()中加入对usyscall的映射。（仿照TRAMPOLINE和TRAPFRAME的映射，首先取消其他页的映射，然后再设置对usyscall的映射）



根据提示，在freeproc（）中也应该释放页。



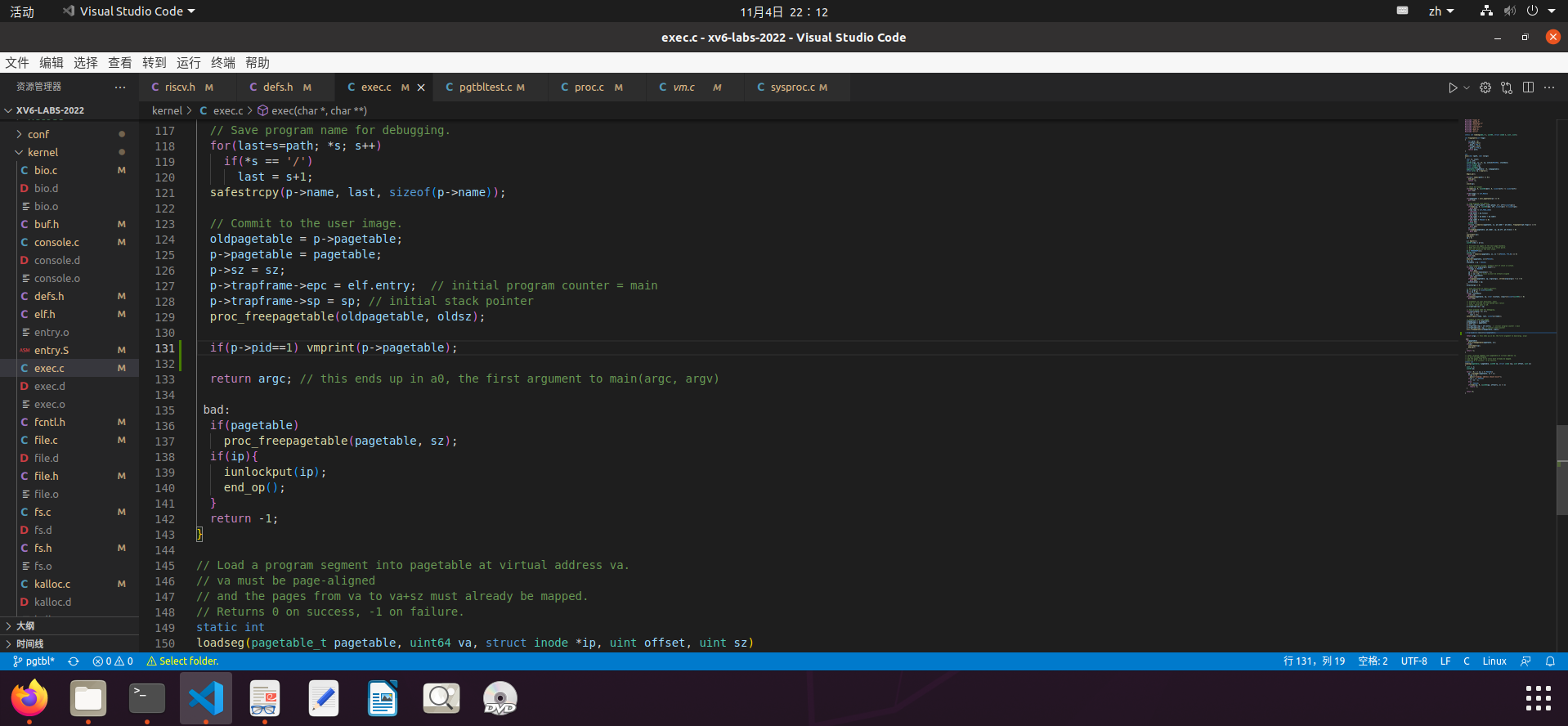
同理，在proc\_freepagetable()中也应该解除映射，否则会报错freewalk panic。



PartB：Print a page table

这个实验主要就是层次遍历页表，并输出每一层的页表。

首先，根据文档提示，在exec.c中加入：



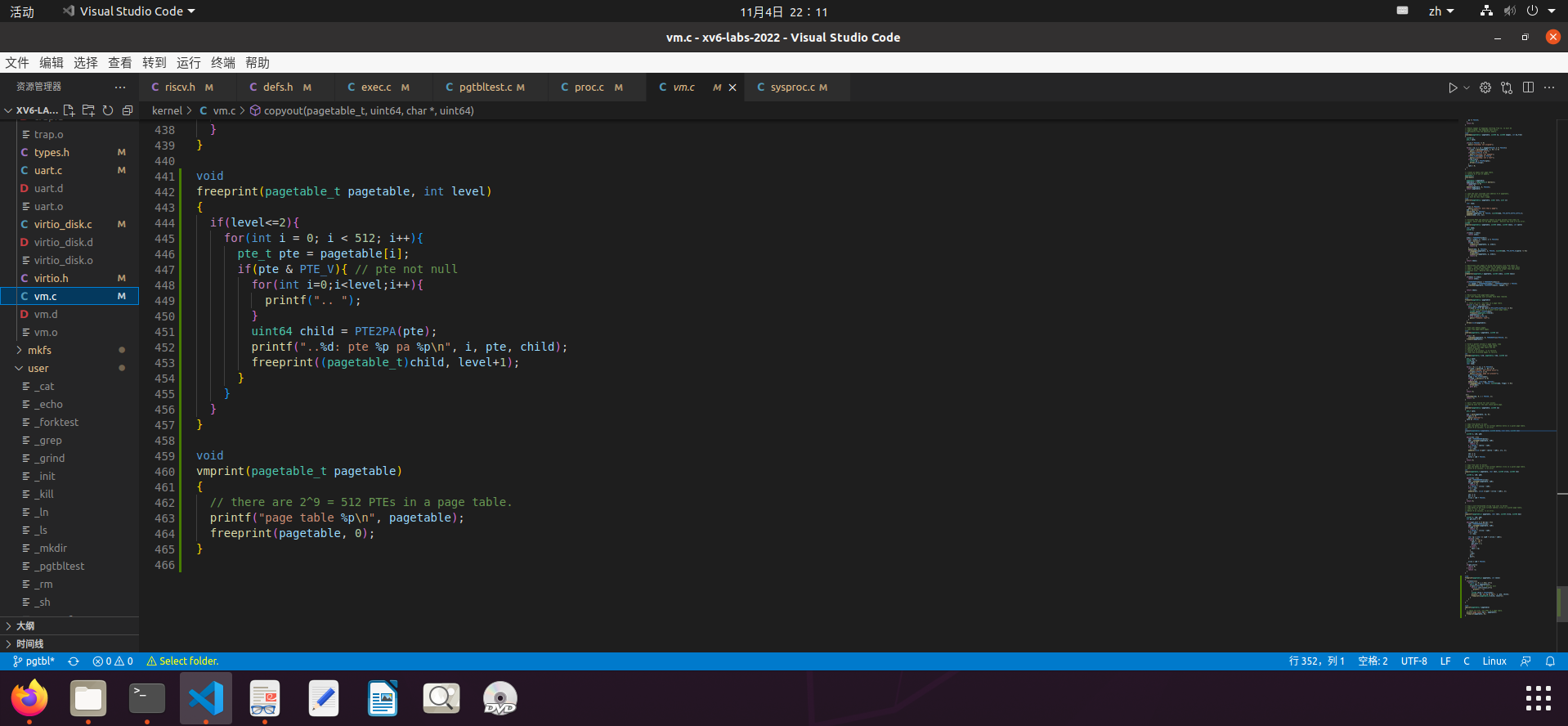
保证可以在pid为1时就输出，即首先输出页表，然后再进行其他进程。

观察freewalk函数，发现其的遍历方式是对每一个页表进行深度遍历，可以借鉴其思路，然后实现了函数freeprint（）：

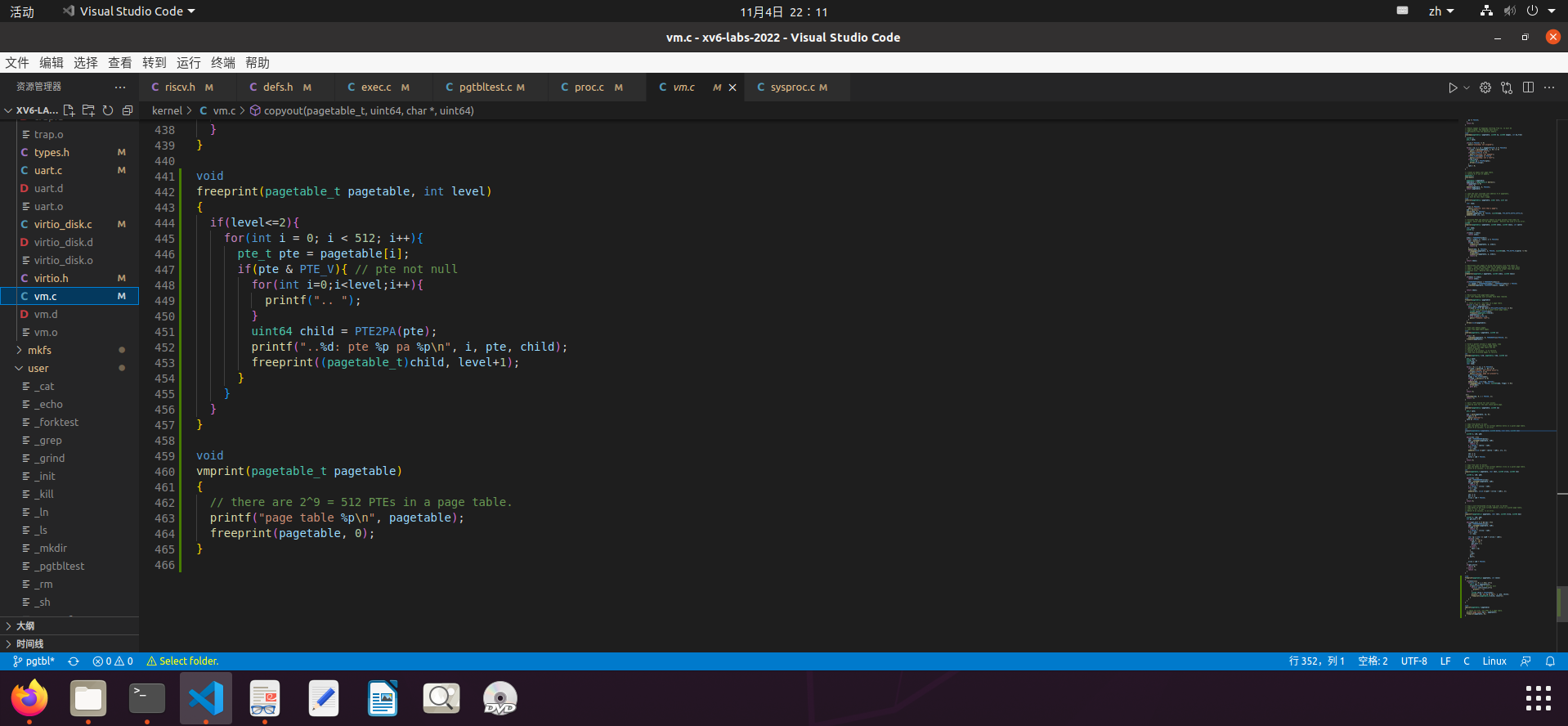
其中pte&PTE\_V为true保证pte页永远是有效的，然后才进行输出。

增加限制条件：只输出两层，否则会一直遍历

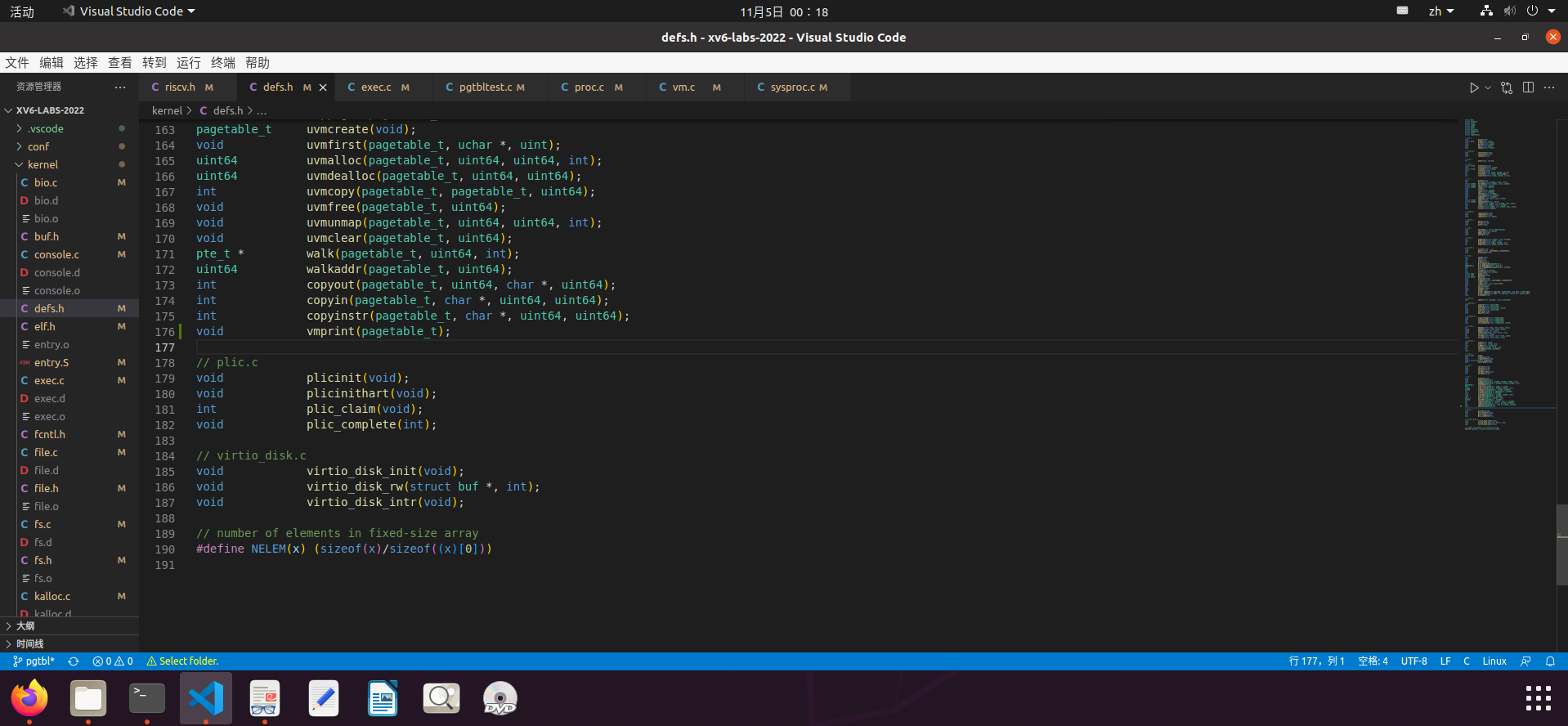
输出为从第0层开始向下遍历，每一层会比上一层前的格式多输出“..”，第0层只有一个“..”。且输出内容包含页表序号、页表中的pte的地址、物理地址。（用16进制表示）



函数vmprint（）则是打印页表的起始地址，然后对页表进行输出。



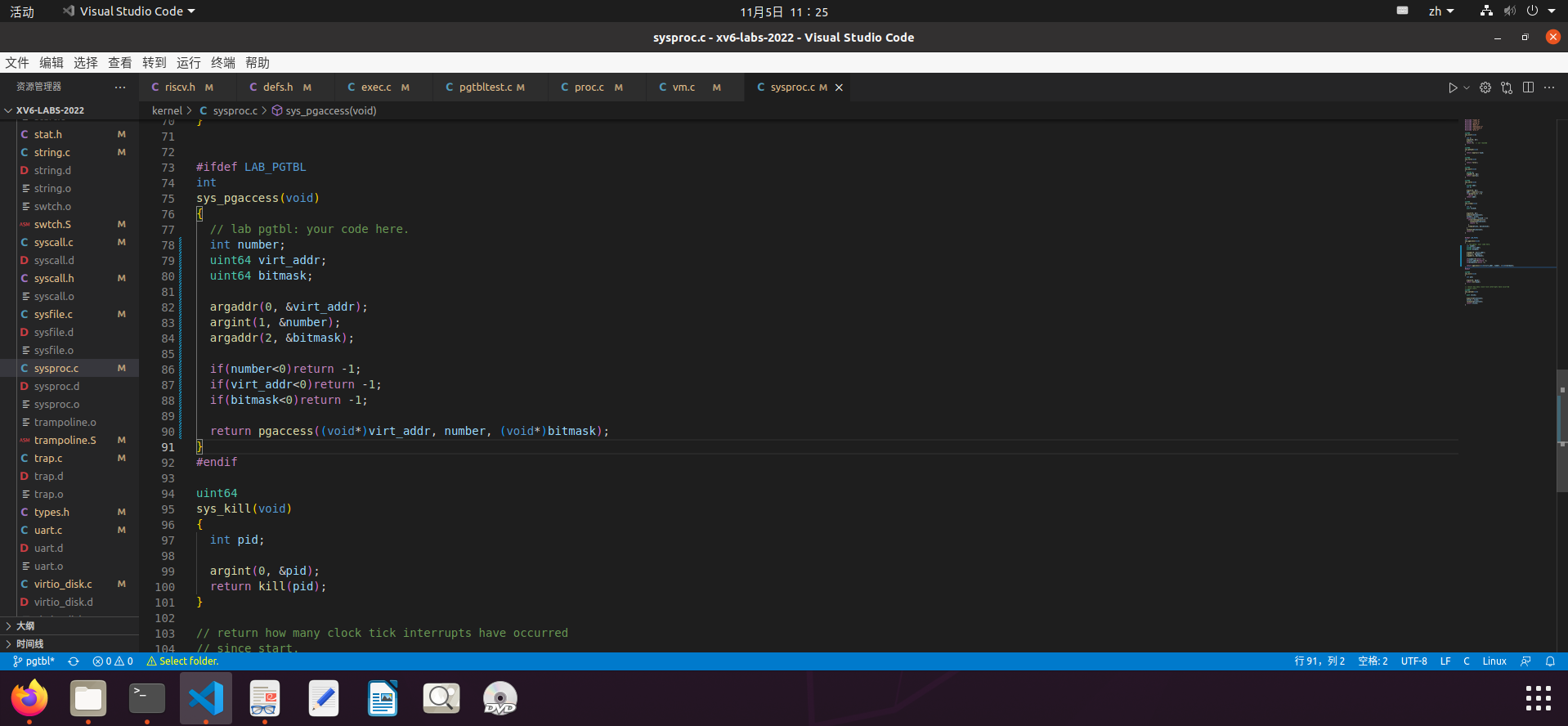
最后对vmprint进行声明，是exec.c可以调用。



PartC：Detect which pages have been accessed

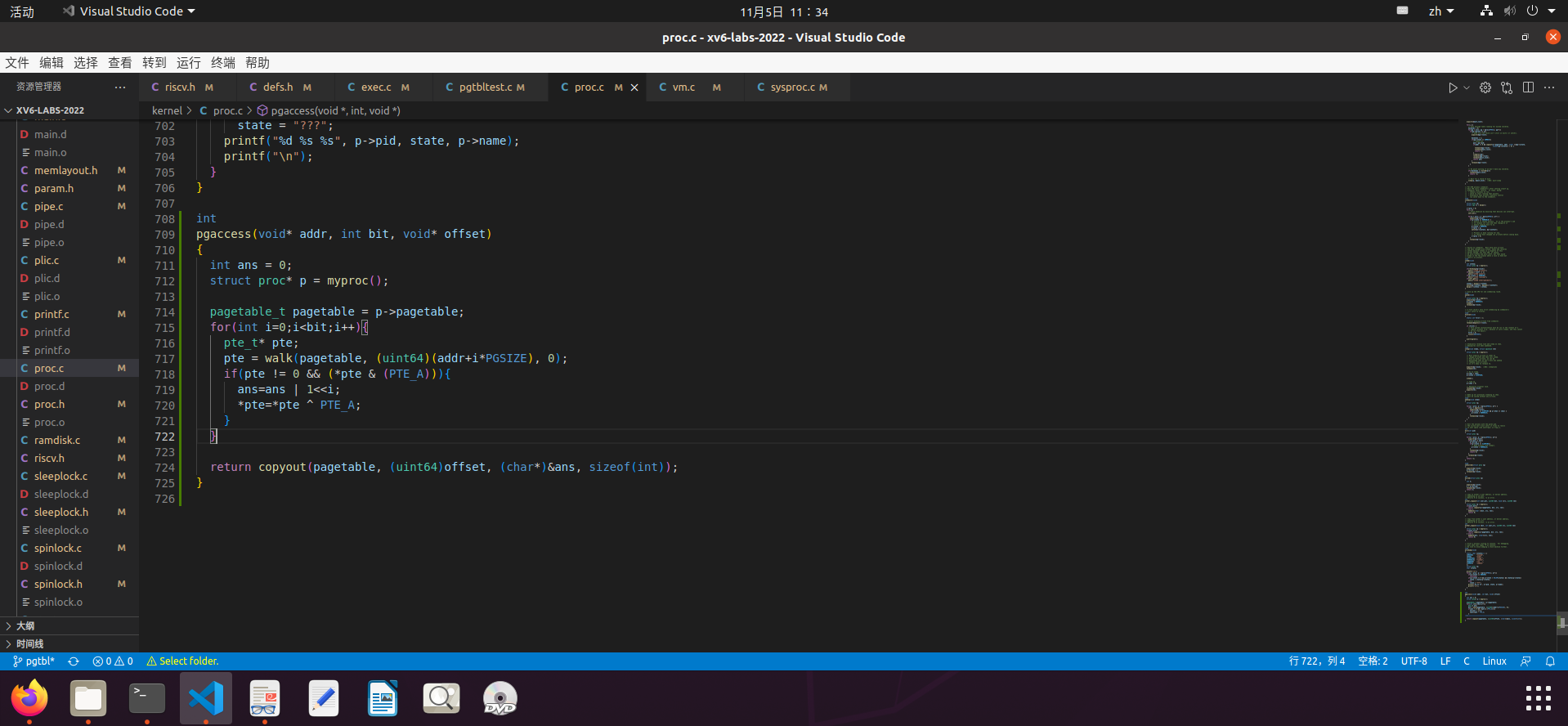
主要思路就是用户调用系统调用的时候，我们去查找页表，获得对应的PTE，然后检查PTE\_A(需要自己定义)，然后决定是否在答案设置对应有效位。

首先在sysproc.c中完善sys\_pgaccess()



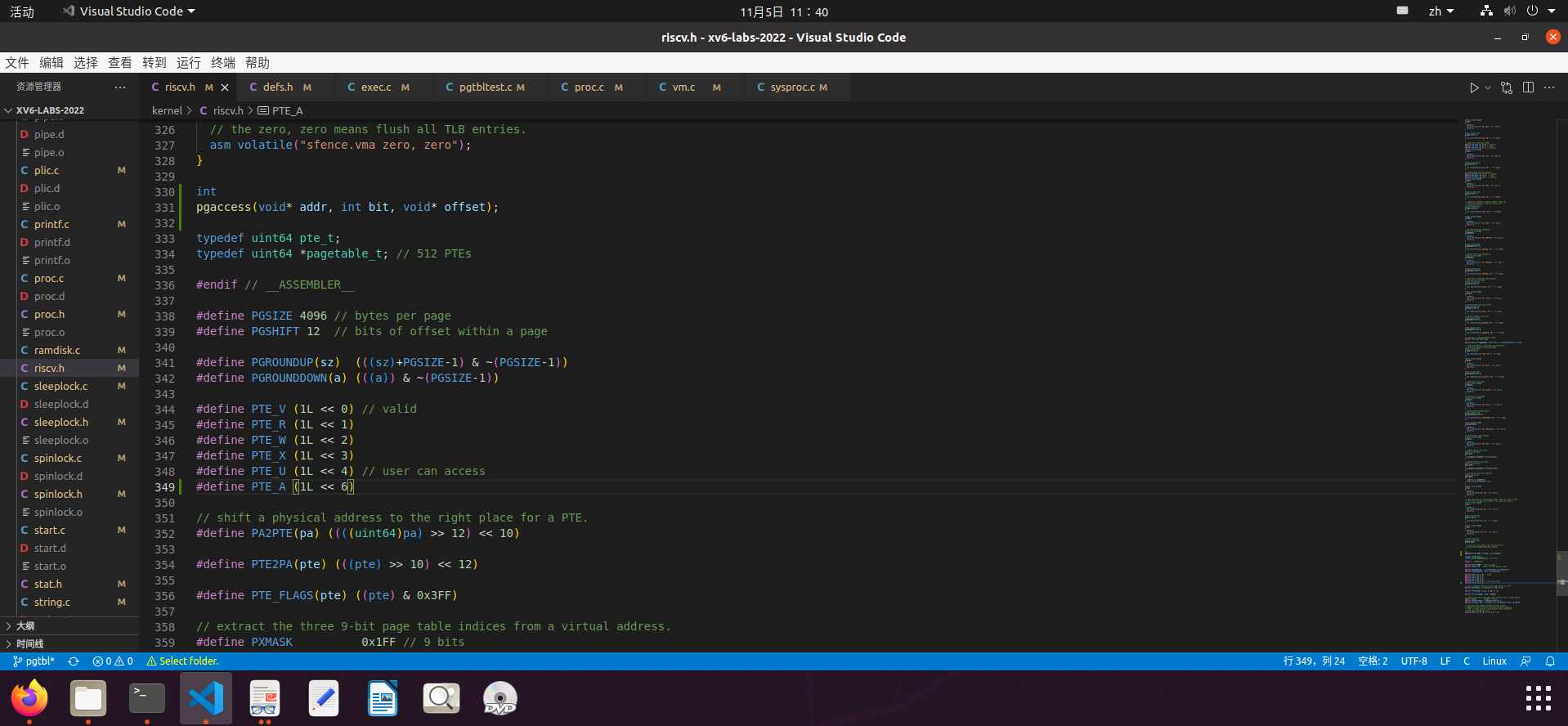
设置输入的第一个参数为虚拟地址，第二个参数为页数，第三个参数为偏移量。

然后调用pgaccess（）函数：

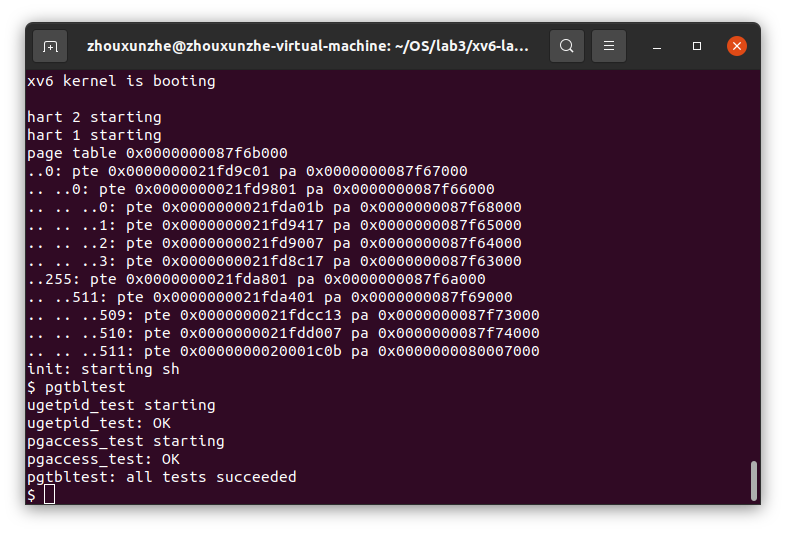


利用walk函数对每一个页表进行遍历搜寻，然后对被access的页表进行记录，最后通过copyout函数进行输出。

查询页表，发现第6位就是access的标识符，所以判断是否被access即判断第PTE\_A是否为1即可。



最后的运行结果如下图：



二、问题回答

（1）在 Part A 加速系统调用部分，除了 getpid()系统调用函数，你还能想到哪

些系统调用函数可以如此加速？

Fork（）函数也是返回pid，似乎也可以利用页表结构存储pid，使得系统调用加速。

（2）虚拟内存有什么用处？

虚拟内存可以扩大内存空间，使得程序能够有效运行；除此以外，还可以利用虚拟内存进行数据的存储，然后调用时又无需花费访问主存那么多的时间，加速系统调用进程。

（3）为什么现代操作系统采用多级页表？

多级页表可以有效节省物理内存空间，使页表可以在内存中离散存储，不需要连续存储消耗极大的空间。

（4）简述 Part C 的 detect 流程。

首先设置起始查询地址，然后对每一个页表进行深度遍历查询，如果有页表被access并且是有效的，则将那一页所对应的标号进行记录，最后进行输出返回结果。

三、实验中碰到的问题。

1.一开始输出页表的格式不正确，并且会超出要求范围，将遍历顺序改为由上至下即可，并且设置范围为两层，就不会出现之前的问题。

2.一开始，在proc\_freepagetable()中没有解除映射，报错freewalk panic，解除映射之后问题就能解决。

四、实验感想

在本次实验中，我学会了页表的一些基本原理，以及如何对页表进行访问以及输出。我还了解了一种系统调用加速的方法——访问页表进行存储。本次的三个实验需要自己思考的地方很多，让我能够自主研究了解页表的基本构造，以及在程序中应该如何表达，收获颇丰。