|  |
| --- |
|  |
| Lab4实习报告 |
|  |

|  |
| --- |
| 姓名：吴悦欣 学号：1900012946  日期：2021/10/18 |

目录

[内容一：实验工作总结 3](#_Toc66996796)

[内容二：遇到的困难以及收获 3](#_Toc66996797)

[内容三：对课程或Lab的意见和建议 3](#_Toc66996798)

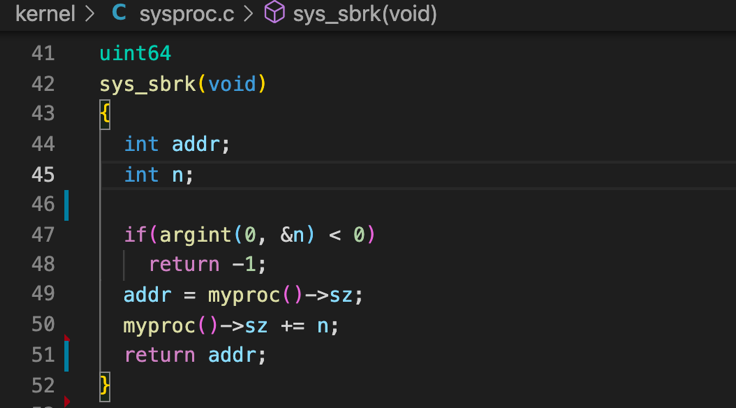
[内容四：参考文献 3](#_Toc66996799)

## 内容一：实验总结

1. Lab4

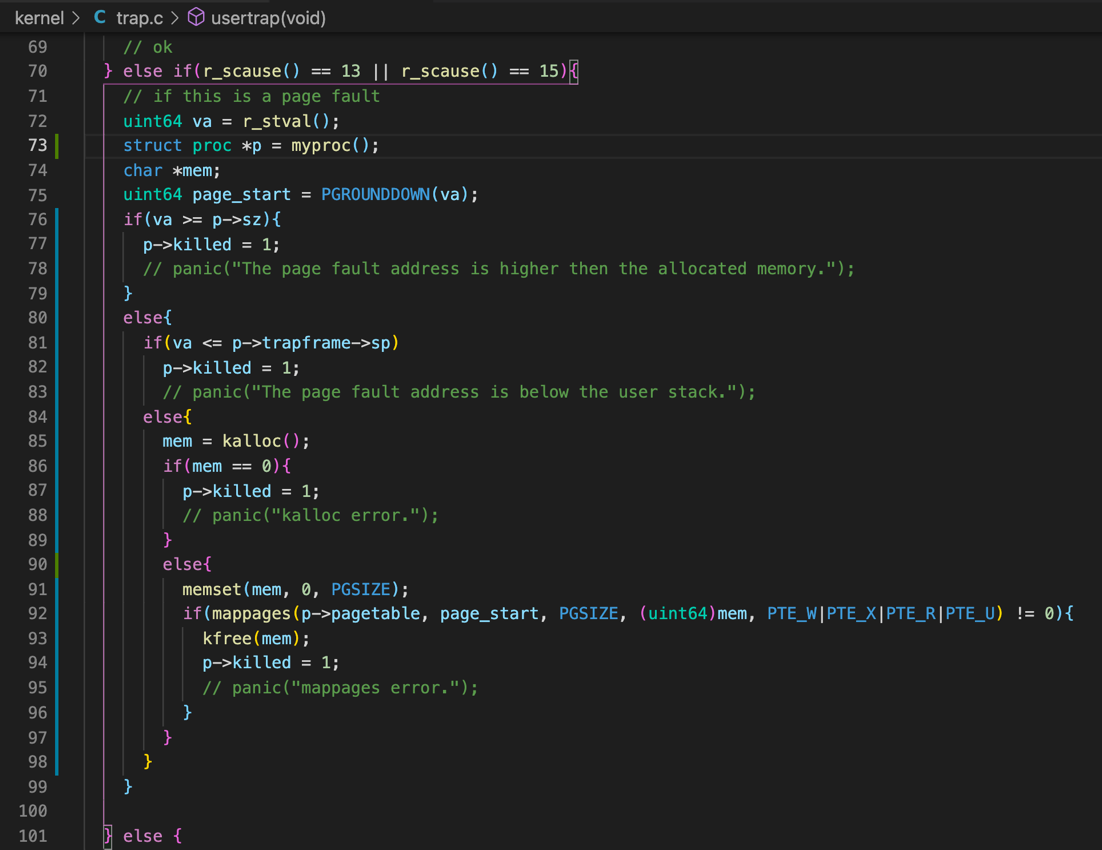
本次实验是关于使用page fault实现懒分配的过程。在xv6课本4.6部分介绍了页故障的类型，在fork后父子进程的写时复制中的作用；实验中则是实现内存的懒分配，有三个部分。

第一个部分是去掉在kernel/sysproc.c的sys\_sbrk函数每次直接分配内存的部分，而是采取先不分配这部分内存，只是在进程的大小中体现实际应当占用空间的大小，等到需要使用这部分内存时产生页故障后再将对应的物理页映射到内存中。这部分的实现很简单，只需要去掉growproc函数，只增加进程的大小。

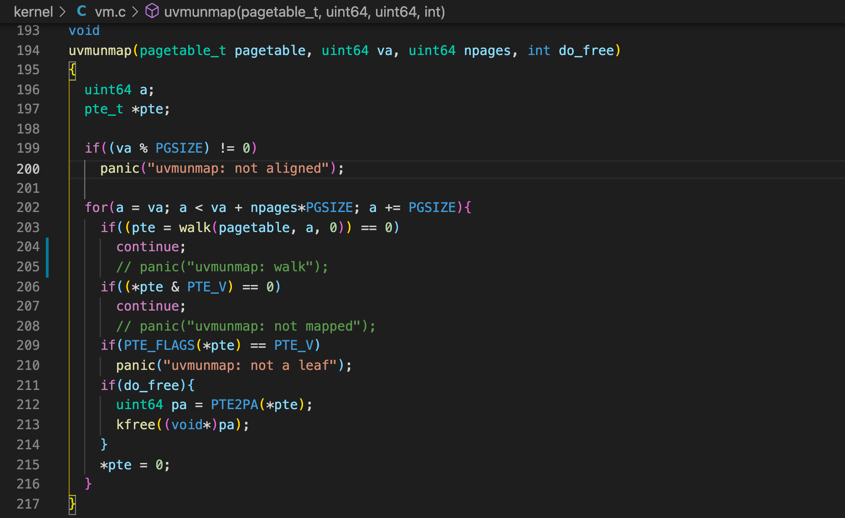


在第一部分完成后，这个系统显然是有问题的，因为并没有分配内存的步骤，在后续的内存的访问中会引发异常。因此在第二部分的工作就是在需要用到某个尚未分配的页的时候，进行内存的分配。由于在使用未分配的地址空间，会产生页故障，陷入内核，并且进入kernel/trap.c中的usertrap函数，而页故障的类型是通过判断在scause寄存器中存有的编号，引发故障的地址则保存在stval寄存器中。由此进行页故障的判断以及页面的分配。

以下是第二部分的代码实现（第三部分的部分实现也已经在里面）



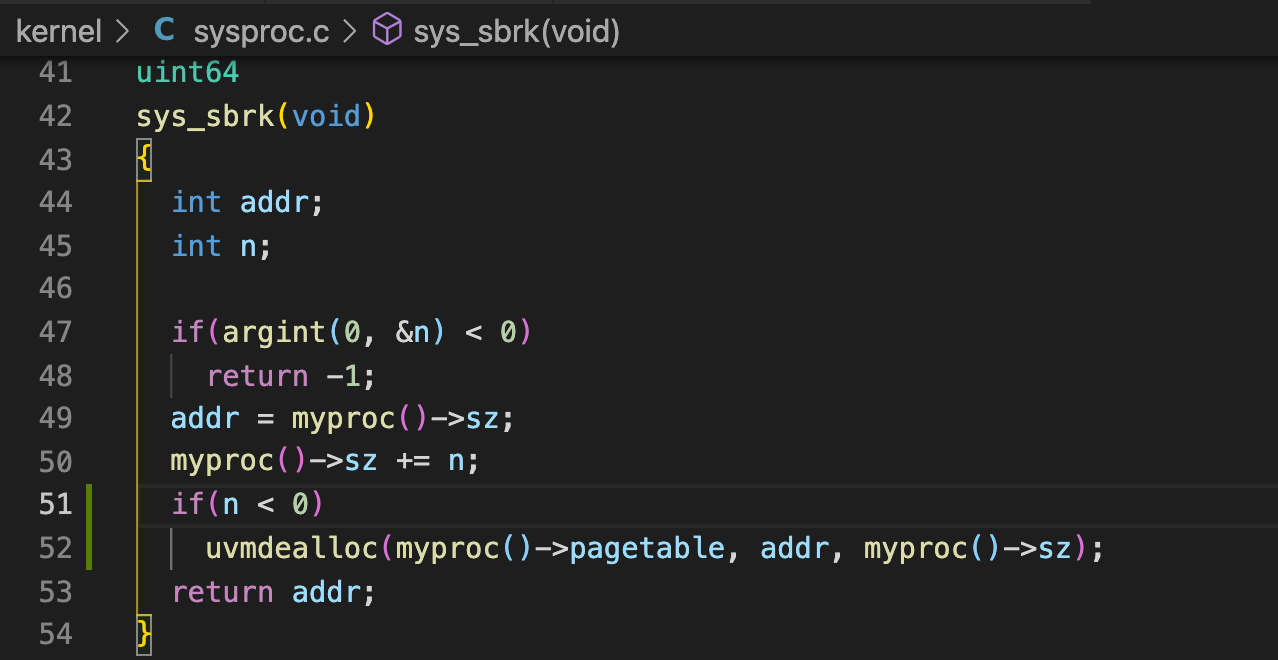
这一部分代码根据提示借鉴了kernel/vm.c中uvmalloc函数的实现，主要的不同是去掉了循环（因为懒分配只需要分配目前使用到的页面），基本和uvmalloc的实现一样。在运行了echo hi指令后uvmunmap函数会panic，因为它会检查是否有页表项，因此需要在kernel/vm.c的uvmunmap函数中取消相关的panic。



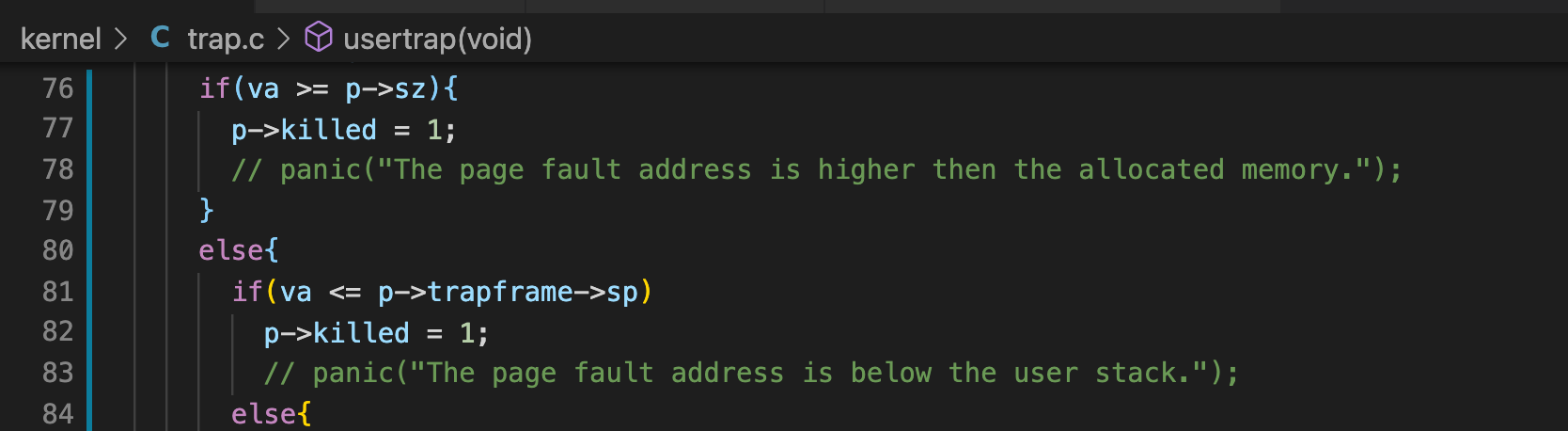
至此，echo hi可以正常运行和响应。

在第三部分中进一步完善了懒分配的机制：

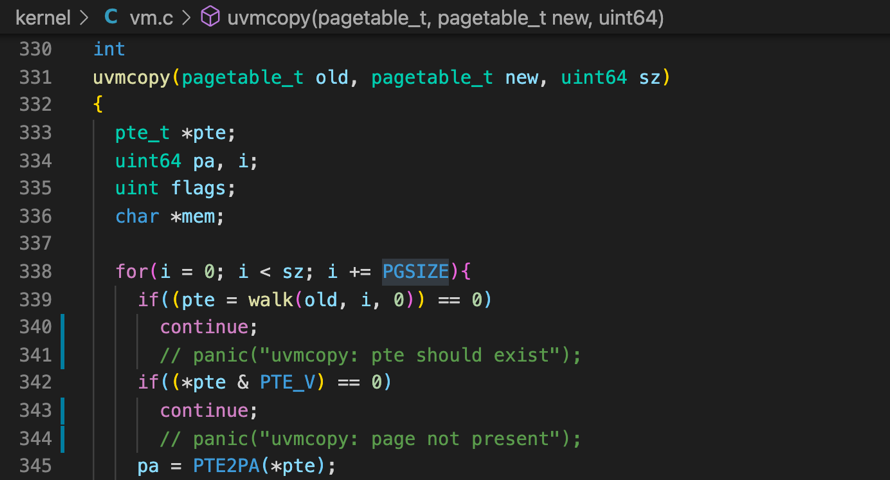
1. 处理sbrk中n为负数的情况，这一步是需要释放n字节的空间，只需要调用kernel/vm.c中的uvmdealloc函数



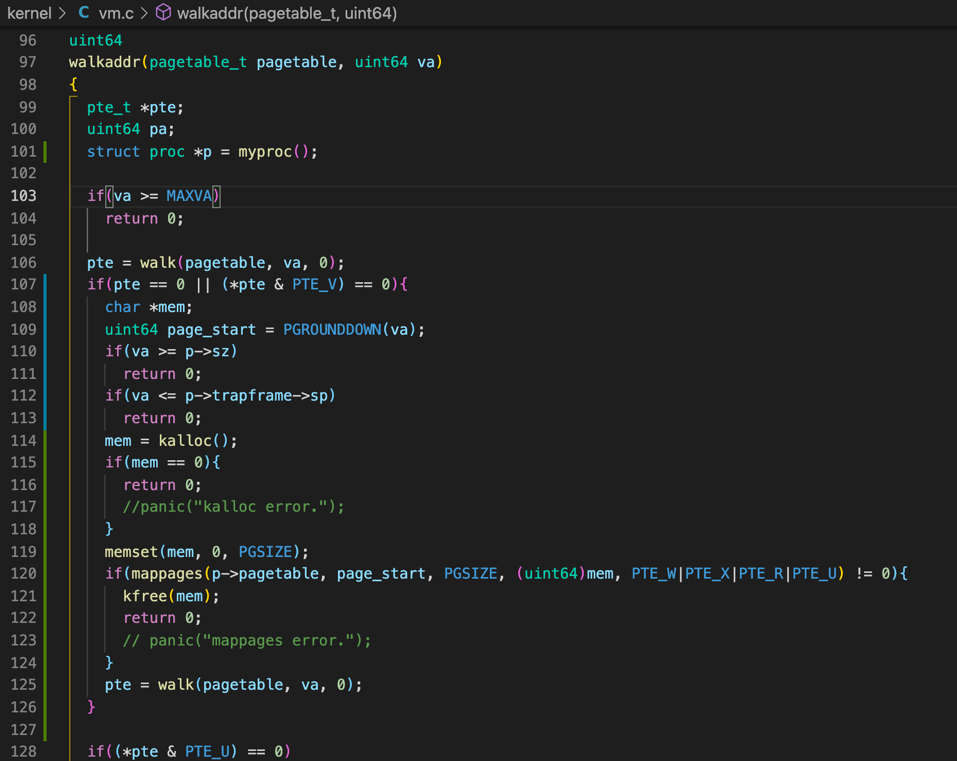
1. 对超过应该分配的空间范围的地址或是非法访问在用户栈之下的页面地址进行处理，杀掉这个进程（下图是在kernel/trap.c的usertrap中的实现）



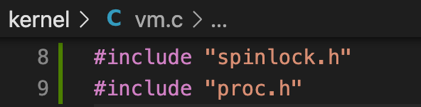
1. 处理进程fork后的内存的拷贝，即在fork时如果拷贝了还没有分配的页给子进程，要防止报错。由于sys\_fork复制内存时是调用的uvmcopy进行的复制，所以以下修改该函数删除panic



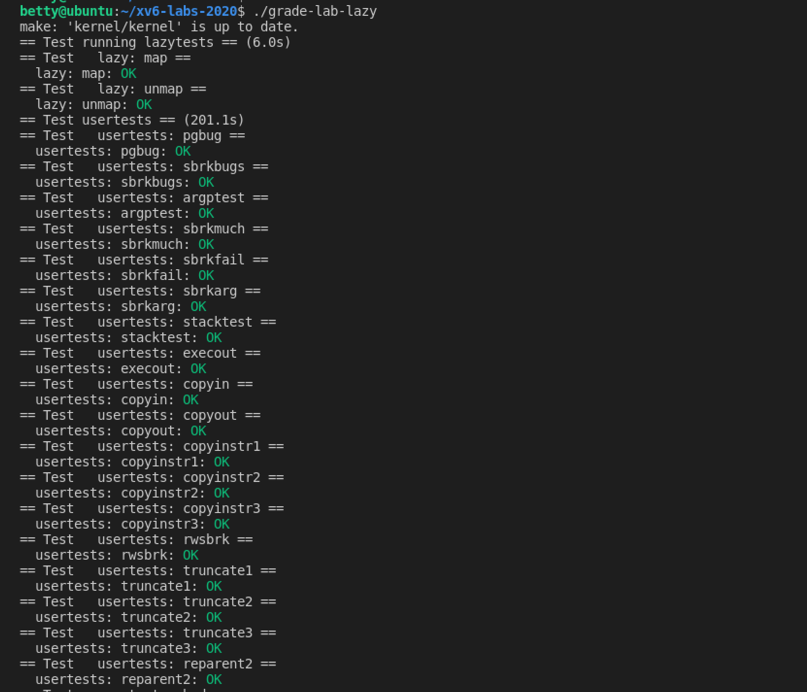
1. 处理系统调用中访问未分配页表地址的情况，并进行页分配。这一部分代码在kernel/vm.c的walkaddr中（被copyout调用）完成，该函数的主要功能是根据虚拟地址返回物理地址，需要将对于pte页表项判断为空或者无效时进行页的分配。



1. 在第三部分的测试的时候出现了“incomplete type proc”的报错信息，因此需要在vm.c中加上头文件spinlock.h和proc.h



至此，所有的测试都可以通过，实验结果如下





1. Lab
2. Lab

## 内容二：遇到的困难以及收获

1. Lab4

在kernel/trap.c的usertrap中对于无效的情况使用panic处理会陷入死循环，并不是要求期望的kill进程的结束方式。在lazytests的时候这里进入死循环还困惑了很久，后来就进到panic函数内部才意识到不应该使用panic。还有在walkaddr的修改中应该符合函数本身的行为，在无效的情况下返回函数值0.

主要的收获是认识到了页面分配的正确时机——在产生用户进程访问内存空间产生page fault陷入内核和系统调用访问未分配的内存空间时。=，进行页面的分配。

简要描述自己在实现Lab中遇到的难题和解决方法，以及在Lab实现中的收获与感想。内容要具体，按实际情况撰写。

## 内容三：对课程或Lab的意见和建议

欢迎同学们对本课程或Lab提出宝贵意见和建议。

## 内容四：参考文献

列出参考的文章或者网址等。

注意：在实习报告完成后，请同学们记得在目录部分按右键更新域。