

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 操作系统 |
| 姓 名： | 董佳鑫 |
| 学 院： | 计算机学院 |
| 系： | 计算机系 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3210102181 |
| 指导教师： | 寿黎但 |

# Lab 5: RV64 缺页异常处理

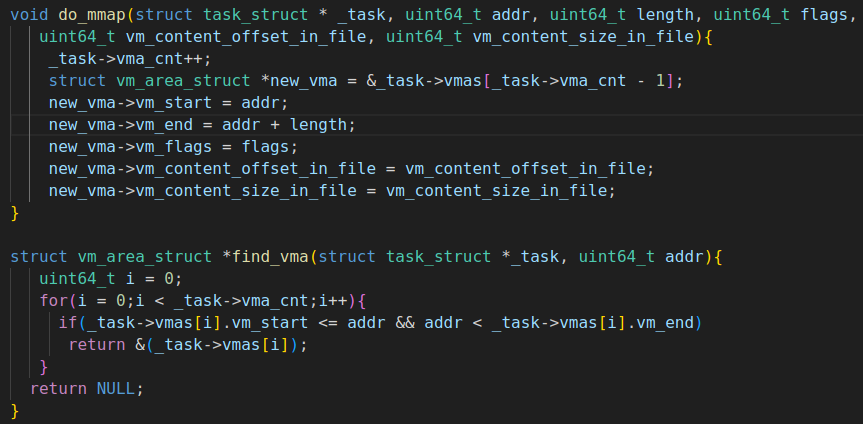
实验目的

1. 通过 vm\_area\_struct 数据结构实现对 task 多区域虚拟内存的管理。
2. 在 Lab4 实现用户态程序的基础上，添加缺页异常处理 Page Fault Handler。

实验过程和操作步骤

1. 实现VMA

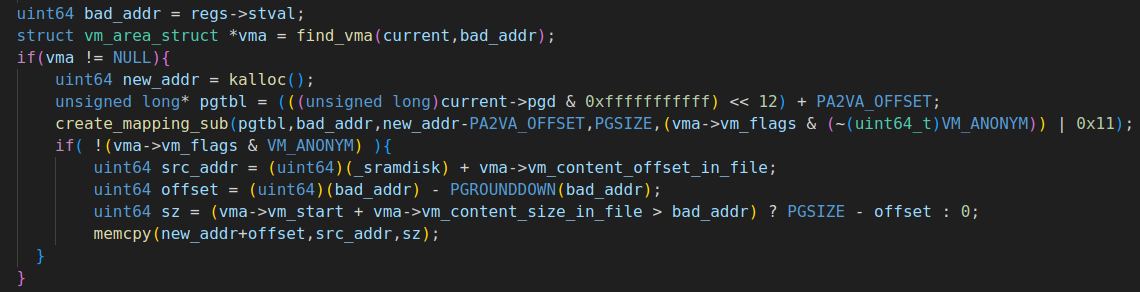
阅读文档以后，实现find\_vma和do\_mmap两个函数，这两个函数的实现较为直观。



1. 修改PAGE FAULT HANDLER

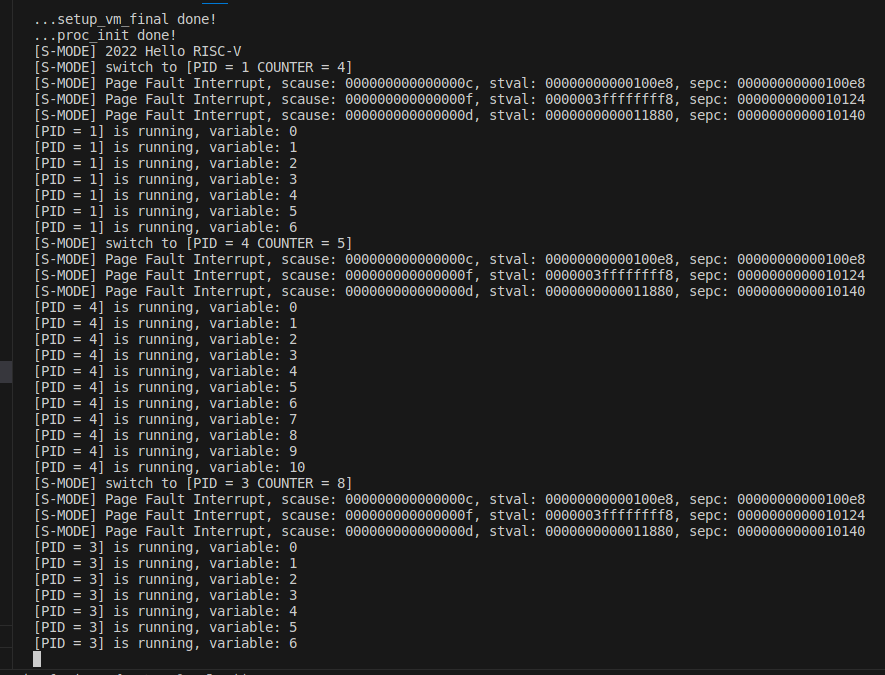
本次实验需要添加新的异常处理：page fault。对应的scause有三个值。因此在trap\_handler中要添加对这三种异常的捕获，并在捕获后调用do\_page\_fault函数，处理缺页异常。

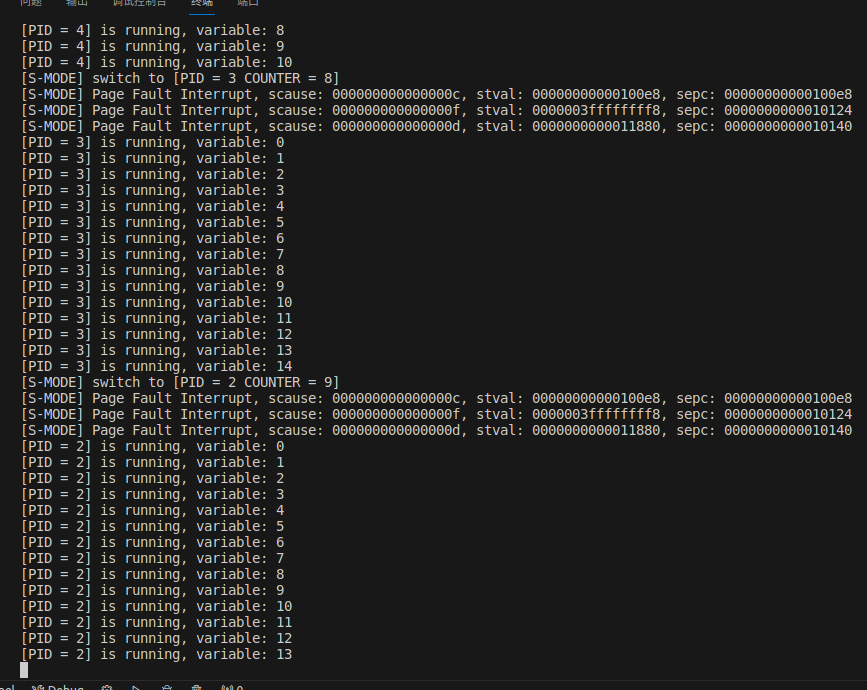
具体缺页异常处理如图。

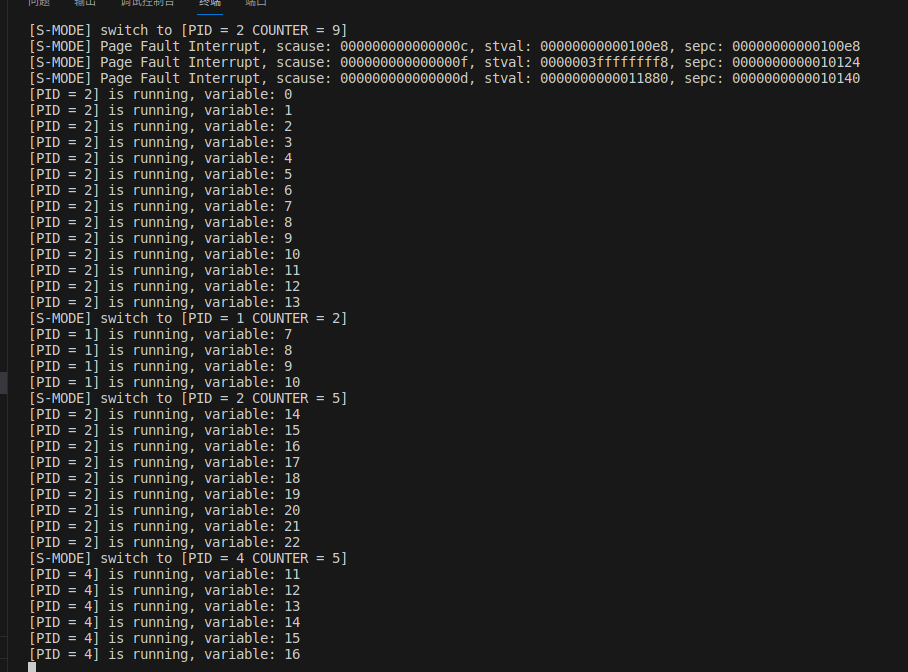


首先查找出现缺页异常的地址有没有被记录到VMA中，如果有，就要对bad\_addr所在页建立映射。并且如果该区域不是匿名区域，还应该拷贝文件的内容。这里要注意在拷贝的时候，由于bad\_addr可能会落在未初始化的区域，这块区域并没有占用文件大小，也就是vm\_content\_offset\_in\_file。因此如果bad\_addr超过了vm\_content\_offset\_in\_file的范围，则不应该拷贝那些未初始化的值，而应该直接默认使用0值。

完成上述代码后，运行程序，可以看到，在处理了缺页异常后，程序最终找到了指令所在地址，程序正确运行。







思考题

1. uint64\_t vm\_content\_size\_in\_file; 对应的文件内容的长度。为什么还需要这个域?

由于在内存中的长度往往会大于在内存中的长度，因为存在一些未初始化的变量不占用文件空间。我们的[vma\_start,vma\_end)的范围可能会大于vm\_content\_size\_in\_file，这就导致我们的bad\_addr可能会落在未初始化区域，而这部分区域在demand paging中是不需要拷贝的，因此我们需要使用 vm\_content\_size\_in\_file来决定是否对刚映射的一个page进行拷贝文件内容。

2.struct vm\_area\_struct vmas[0]; 为什么可以开大小为 0 的数组? 这个定义可以和前面的 vma\_cnt 换个位置吗?

大小为0的数组是可变数组。这样处理的好处是不会占用更多的内存空间，只有当需要的时候才动态申请内存空间（使用kalloc()函数），并且比链表实现更为方便。不可以和vma\_cnt换位置,它必须处在结构体的末尾，才能动态决定结构体的size以节省空间。

讨论心得

本次实验的难度相比前两个有了很大的降低，基本没有遇到太多问题。唯一需要注意的就是do\_page\_fault函数的实现，需要理清各种细节。关键点就是在对新的页建立映射后是否要进行内容拷贝，bad\_addr如果没有超过vm\_content\_size\_in\_file的范围则正常拷贝，否则不进行拷贝。对于文件中的大小和内存中的大小的区别一定要掌握清楚。另外一个值得关注的地方就是在建立映射或者do\_mmap的时候，权限设置要谨慎，由于权限不正确而导致缺页异常处理失败的错误也非常致命。