惰性页表分配

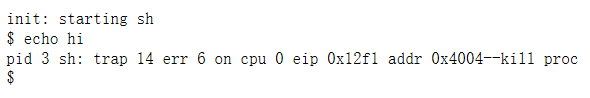
操作系统的一个小技巧是栈内存的惰性页表。Xv6的应用使用sbrk系统调用来向核索取栈内存。Sbrk分配物理内存，并加到进程的虚拟地址空间中。有些程序分配了这种内存，但是并不使用，比如大的稀疏矩阵。复杂的核延迟内存的每页分配，直到应用开始使用时分配。

**第一部分：估计sbrk的分配**

第一个任务是在系统调用sbrk中删除页表分配，sbrk增加n bytes的内存，然后返回新分配区域的开始地址（即老的size）。你的新sbrk应该给进程（）增加n bytes，然后返回老的大小。Sbrk（n）不应该分配内存，所以你应该删除调用growproc（），但是你任然需要增加进程的size

试着猜想这个修改的结果是什么？

做这个修改，然后输入echo hi到shell，你应该看到这样的内容



Pid3…….是trap.c输出的内容，traphandler获取了页表错误，xv6核不知道如何处理.确保你知道为什么会产生错误。

地址0x4004意味着导致错误的虚拟地址是0x4004

**第二部分：页表分配**

在trap.c中修改对应页表错误的代码，把导致错误的内存地址映射在错误的地址，然后返回用户空间，让进程继续执行，你应该在cprintf前面增加代码。你的代码不需要包含所有的极端情况和错误情况，只需要让sh能够运行像echo和ls这样的简单命令即可。

提示：

查看cprintf变量，看怎样找到导致错误的内存地址的

从vm.c中copy一些代码，

使用PGROUNDDOWN(va)来对应页表边界

Break和return来避免cprintf和myproc()->killed = 1.

你需要调用mappages()，需要删除vm.c中的static，并在trap.c中声明，在trap.c添加

通过tf->trapno是否等于T\_PGFLT来查看是否是一个页表错误

如果修改正确，echo hi便可以正常运行，你最少会得到一个页表错误，也可能两个