****



**MIT xv6实验报告**

**——lab5 : lazy**

**学生姓名 胡轶然**

**学 号 3019244355**

1. **实验目的**
2. 熟悉xv6系统的内存分配和缺页中断机制。
3. 改变分配页面的方式，不再使用growproc函数一次性将页面分配给进程，而是在发生缺页异常时再向进程分配页面
4. **前期准备**
5. 切换git分支。
6. 阅读指导书第4章，尤其是第6小节。
7. 阅读kernel文件夹下的trap.c、vm.c和sysproc.c。

**三、 实验内容及实现步骤**

**任务1 Eliminate allocation from sbrk**

1. **问题描述**

本任务要求删去sys\_sbrk函数中分配内存页的逻辑，改为只增大进程空间的大小。同时返回旧的进程空间大小。

1. **思路与实现步骤**

本任务是接后两个任务的铺垫，较为简单，只要在sysproc.c文件的sys\_sbrk函数中注释掉growproc函数的调用，并增大->sz即可。

具体修改内容如下：

1. 不再于sys\_sbrk函数中分配内存页，修改sysproc.c文件中的sys\_sbrk函数，注释掉growproc函数的调用。



1. 增大进程堆空间大小，为缺页异常时的内存页分配做准备。

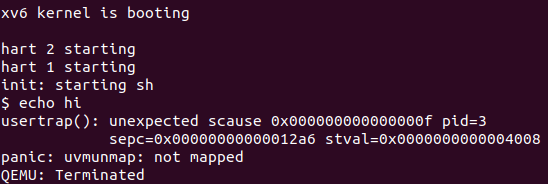


1. **问题与解决方法**

本任务较为简单，只需修改3行代码，暂无问题。

1. **结果**

地址映射失败，符合预期



**任务2 Lazy allocation**

1. **问题描述**

Xv6系统中，当进程申请大量内存时，内核会立刻为进程分配相应数量的内存页，这会导致申请过程耗费很长的时间。且当申请的内存大小过大时，会因内存不足而分配失败，导致程序崩溃。

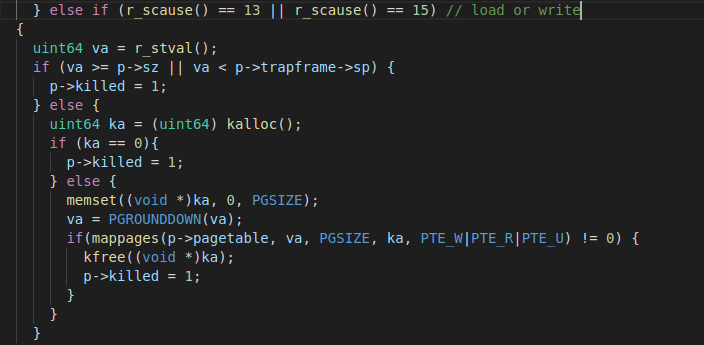
本任务要求实现lazy allocation，**当进程调用sys\_sbrk时，只扩大其进程空间，而不分配内存页。当进程因读内存或写内存触发缺页异常时，为其分配一页内存。**Lazy allocation**将分配内存页的工作从进程申请内存时集中完成改为进程使用内存时按需分配**，很好地解决了进程大量申请内存时导致的卡顿现象，而且可为进程分配超过剩余内存大小的内存空间。

1. **思路与实现步骤**

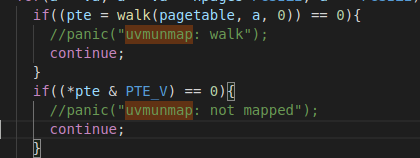
要实现lazy allocation，应修改缺页异常的处理逻辑，并允许缺页异常的发生，即允许walk函数映射失败，或目标页无效（未持有PTE\_V权限）的情况。

具体思路和实现如下：

1. 在usertrap函数中修改缺页异常的处理逻辑，内容包括：
   1. 当发生读内存、写内存缺页异常（对应标识13、15）时，进入内存分配逻辑。（注：下图包含了部分任务3的内容）
   2. 调用r\_stval获取触发缺页异常的虚拟地址va，并使用PGROUNDDOWN定位到对应虚拟内存页的首部。
   3. 调用kalloc函数分配一个物理页，使用memset函数清空其内容。
   4. 调用mappages函数，将虚拟地址va与新物理页相关联。



1. 允许缺页异常的出现，注释掉uvmunmap函数中对缺页现象的panic：
   1. 未分配的物理内存页可能没有PTE\_V（被使用标记），故应注释掉相关的panic。



1. **问题与解决方法**

**问题一**：出现了incomplete type proc错误。

**解决方法**：说明存在**头文件循环引用**的情况，修改引用“spinlock.h”和“proc.h”的顺序即可。

**4. 结果**

地址映射恢复正常



**任务3 Lazytests and Usertests**

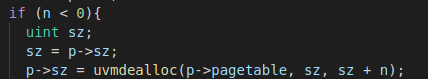
1. **问题描述**

本任务要求完善lazy allocation，使程序能够发现和处理一些特殊情况和异常。

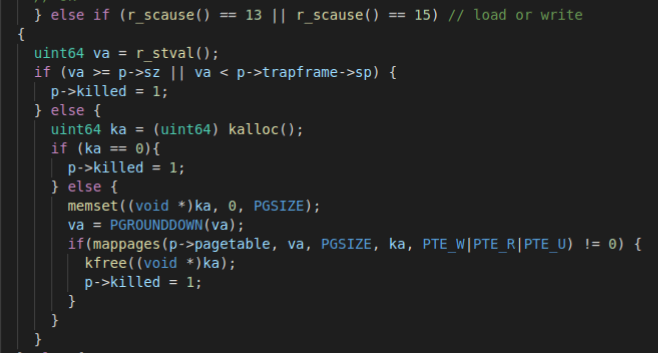
1. **思路与实现步骤**

按照指导书中的提示（hint）逐个修改即可。

1. 处理sys\_sbrk中参数n<0的情况。该函数中，n>0代表分配内存，相应地n<0代表释放内存。当n<0时应调用uvmdealloc释放一部分内存。

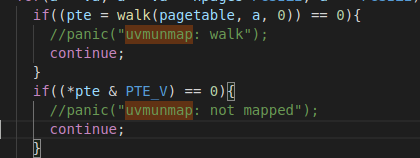


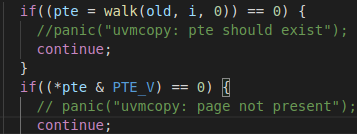
1. 修改usertrap函数，具体内容如下：
   1. 当进程尝试访问进程空间以外地址，即”va>=p->sz”时，kill进程。
   2. 当虚拟地址小于用户栈的栈顶地址时，kill进程。
   3. 当物理地址分配失败时（kalloc返回0），kill进程。

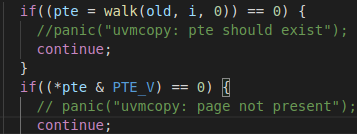




1. 调整将父进程内存复制到子进程的逻辑，对应修改uvmunmap函数和uvmcopy函数，注释掉其中对虚实地址映射失败和物理页无效的panic。
2. **因为sys\_sbrk仅增大了内存空间大小，而没有分配物理内存页，将虚拟地址映射到物理地址失败是正常现象，所以注释掉walk失败的panic。**







1. 系统在内核空间和用户空间之间拷贝数据时，使用walkaddr函数映射虚实地址。因为lazy alloc只增大进程空间而不立刻分配物理页，walkaddr函数可能因物理页未分配而异常返回。此时，需要模仿usertrap函数中的逻辑，在copyout、copyin函数中分配物理内存页。

具体的实现方法与usertrap函数相似，不再赘述。



1. 阅读相关资料发现，**还应当增加对guard page的保护（不保护也可通过测试）**，guard page位于用户栈和数据之间，起到分隔二者的作用。当进程要求覆盖guard page时，应予拒绝并kill进程。
2. **问题与解决方法**

**问题一**：不理解如何在fork函数中处理“为子进程复制lazy allcoation未分配的物理内存”。

**解决方法**：

阅读fork代码发现，该函数通过调用uvmcopy函数实现对进程内存的复制。而**父进程已将进程空间大小p->sz复制给了子进程**，因此只需要把已分配的物理内存复制给子进程，**对于父进程中声明而未分配的物理内存，在复制时可以直接忽略，让子进程自己触发缺页异常并由系统lazy allocate物理页**。

**问题二**：指导书中未提及对guard page的保护。

**解决方法**：

本任务不要求保护guard page，只是在查阅资料，阅读指导书时发现了这一知识点，故额外实现了该功能，并通过了测试。

**四、 实验结果**

