# shell

- 姓名：陈姝宇
- 学号：3017218119
- 班级：软工3班

## 实验内容

O/S可以对页表硬件使用的许多巧妙的技巧之一是延迟分配堆内存。Xv6应用程序使用sbrk（）系统调用向内核请求堆内存。在我们提供给您的内核中，sbrk（）分配物理内存并将其映射到进程的虚拟地址空间。有些程序分配内存，但从不使用它，例如实现大型稀疏数组。复杂的内核会延迟每一页内存的分配，直到应用程序尝试使用该页——这是由页错误发出的信号。在本练习中，将把这个延迟分配特性添加到xv6中。

## 实验步骤

## 1. 第一部分：从sbrk（）中消除分配

要求修改 `sbrk syscall`，去掉实际分配内存的代码，进入 shell 后运行命令会发送错误。

`sbrk(n)`应该只用n增加进程的大小`(myproc() - > sz)`并返回旧的大小。

```
46 sys_sbrk(void)
47 {
48   int addr;
49   int n;
50
51   if(argint(0, &n) < 0)
52     return -1;
53   addr = myproc()->sz;
54   myproc()->sz += n;
55 //  if(growproc(n) < 0)
56 //    return -1;
57   return addr;
58 }
```

返回的地址是新分配的地址空间的开头，在此处就是原来地址空间的末尾，此处都是虚地址。再运行的结果就是：

```
init: starting sh
$ echo hi
pid 3 sh: trap 14 err 6 on cpu 0 eip 0x112c addr 0x4004--kill proc
$
```

也就是在运行malloc的时候，虽然返回是成功了的，但是当程序试图操作cmd指向的内存区域的时候，发现该内存区域不是当前进程所有的，因为在sys_sbrk中根本没分配

## 2. 第二部分：Lazy allocation

由于我们需要在trap.c中调用vm.c中的int mappages()函数，所以要去除原本的static关键字。

```
60 int
61 mappages(pde_t *pgdir, void *va, uint size, uint pa, int perm)
62 {
63   char *a, *last;
```

在trap.c中在调用之前使用extern关键字声明int mappages()函数:

```
17 extern int mappages(pde_t *pgdir, void *va, uint size, uint pa, int perm);
```

在trap.c中的void trap(struct trapframe *tf)的defaut部分添加以下代码，放置的位置为if模块后:

```
100     char *mem;
101     uint a;
102     a = PGROUNDDOWN(rcr2());
103     uint newsz = myproc() -> sz;
104     for(;a < newsz;a += PGSIZE){
105       mem = kalloc();
106       memset(mem,0,PGSIZE);
107       mappages(myproc()->pgdir,(char*)a,PGSIZE,V2P(mem),PTE_W|PTE_U);
108     }
109     return;
```

再次运行qemu:

```
init: starting sh
$ echo i
i
$
```