

Homework: xv6 lazy page allocation

刘恒星 2022229044

March 16, 2023

1 Part One: Eliminate allocation from sbrk()

这一部分的要求是我们在 `sbrk()` 函数中, 只在变量上增加空间大小 `n`, 而不是实际分配空间。这一部分代码我们在 `sysproc.c` 里面的 `sys_sbrk()` 修改

```
1 int sys_sbrk(void)
2 {
3     int addr;
4     int n;
5
6     if (argint(0, &n) < 0)
7         return -1;
8     addr = myproc()->sz;
9     // if (growproc(n) < 0) // 不实际分配空间
10    // return -1;
11    myproc()->sz += n; // 增加空间大小
12    return addr;
13 }
```

我们编译之后, 运行 `echo hi` 指令, 效果如下:

```
Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ echo hi
pid 3 sh: trap 14 err 6 on cpu 1 eip 0x11c8 addr 0x4004--kill proc
```

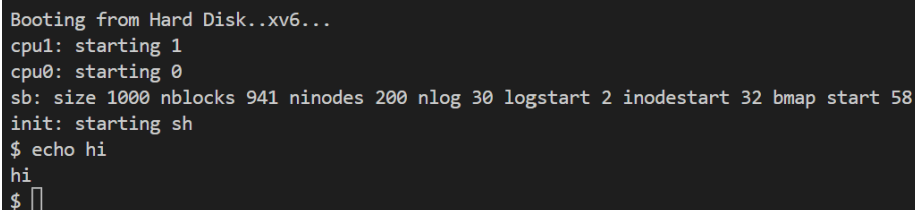
2 Part Two: Lazy allocation

这一部分的要求是实现懒分配，通过将新分配的物理内存页面映射到错误地址来响应用户空间的页面错误，然后返回到用户空间，让进程继续执行。

在 *trap.c* 中，当发现是 page fault 错误的时候，可以按照当前进程的 `proc->sz` 来实际的分配内存。所以首先应该获取发生 page fault 时刻的虚地址，根据 `proc->sz` 的大小来决定分配的空间大小。

```
1 case T_PGFLT:
2 {
3     uint newsz = myproc()->sz; // 应该有的空间大小
4     uint a = PGROUNDDOWN(rcr2());
5     if (a < newsz)
6     {
7         char *mem = kalloc(); //创建空间
8         if (mem == 0)
9         {
10             cprintf("out of memory\n");
11             exit();
12             break;
13         }
14         memset(mem, 0, PGSIZE);
15         mappages(myproc()->pgdir, (char *)a, PGSIZE, V2P(mem), PTE_W | PTE_U); //映射
16     }
17     break;
18 }
```

我们编译之后，运行 `echo hi` 指令，效果如下：



```
Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ echo hi
hi
$
```