操作系统 Lab3 解决方案文档

汪喆昊 516030910460

Exercise 1

这一部分基本上就是照着上个lab在**mem_init()**里面的内容写一下就行了。

Exercise 2 :: kern/env.c

env_init()

注释里已经把这个函数要干啥说得很明白了……

env_setup_vm()

也是看着注释写。设置env_pgdir，注意用page2kva转换。pp_ref要增加。然后按照注释的说明，设置每个pgdir的位置。从PDX(UTOP)到NPDENTRIES。

region_alloc()

从va到va+len（当然要对齐）分配并映射虚拟和物理地址。经大佬提醒，意识到实际上有对齐的宏的。

load_icode()

照着boot/main.c写了一部分。

```
struct Proghdr *ph, *eph;

struct Elf *elf = (struct Elf *)binary;

if (elf->e_magic != ELF_MAGIC)

    panic("kern/env.c: load_icode failed.");

// "Copy" from boot/main.c

ph = (struct Proghdr *)((uint8_t *)elf + elf->e_phoff);

eph = ph + elf->e_phnum;
```

然后装载elf的image。不要忘了切换cr3（血的教训）。另外不要忘了设置eip。

```
e->env_tf.tf_eip = elf->e_entry;
```

env_create()

按照注释里面写即可。不要忘记错误检查。

env_run()

实际上就是先检查是不是当前env，是的话就直接pop_tf，不是的话就更新相应的变量。注意这个时候也是要切换cr3的。

```
if (curenv != e)

{

    if (curenv && curenv->env_status == ENV_RUNNING)

    {

        curenv->env_status = ENV_RUNNABLE;

    }

    curenv = e;

    curenv->env_status = ENV_RUNNING;

    curenv->env_runs++;

    lcr3(PADDR(curenv->env_pgdir));

}

cprintf("esp %x\n", read_esp());

env_pop_tf(&(e->env_tf));
```

Exercise 3

None

Exercise 4 :: kern/trap.c & kern/trapentry.S

trap_init()

这里先声明需要的entry函数，然后用setgate设置idt中对用的内容。

trapentry.S

先去reference manual里面查每个exception会不会要err code。然后就普通地填上去就行了。_alltraps的实现的话，就是先在栈上弄出一个tf，然后设置ds、es，然后call trap，当然要先传参传参（因为esp现在直接指着tf，所以push esp就行了）。

另外，这里实际上有两个cprintf的字符串漏掉了0x，要加上才能过测试。

Exercise 5,6 :: kern/trap.c

trap_dispatch()

加一个switch就行了。如果trapno是T_PGFLT的话，就调用page_fault_handler。是T_DEBUG和T_BRKPT的话，就调用monitor。

Exercise 7 :: kern/trapentry.S & kern/trap.c & kern/syscall.c

trapentry.S

加上对应的项就行了。

trap_init()

同上。

syscall()

在syscall里面加一个switch，然后填入对应的sys函数。

trap_dispatch()

在switch里面加上T_SYSCALL一项，调用syscall，传入文档中写明的参数。把tf的eax一项设置为返回值。

Exercise 8 :: kern/trapentry.S & kern/init.c

目前这个部分没有成功……

Exercise 9 :: kern/libmain.c

libmain()

把thisenv设置为envs中对应的env的地址。对应的env的id可以调用sys_getenvid()函数得到，不过还要用ENVX宏获取到对应的index。

Exercise 10 :: kern/syscall.c

sys_sbrk()

这个理论上可以直接用region_alloc函数，但是我这里这么用会出现奇怪的编译错误。所以干脆就写了一遍region_alloc干的事情。

另外，在struct Env中增加了env_heap_marker一项，来指示对应的堆的顶部。该项在load_icode中初始化。每次成功sbrk后更新env_heap_marker。

Exercise 11,12 :: kern/trap.c & kern/pmap.c

page_fault_handler()

检测tf是否是kernel-mode（**tf_cs & 0x3 == 0**），是的话就panic。

user_mem_check()

取va到va+len之间的每个页地址，若超过ULIM则报错，若无法找到对应的pte或pte的权限和给定的权限不一致，也报错。

Exercise 13 :: user/evilhello2.c

这个部分我没搞懂（因为貌似已经写好了……