Operating System MIT 6.828 JOS Lab5 Report

Computer Science
ChenHao(1100012776)

2013 年 10 月 31 日

目录

1 Exercise 1

仅给 file system environment I/O 权限，如果同时有多个 environment 享有 I/O 权限，则会对于中断的分配到对应的 user-mode environment 造成很大困扰。

IOPL 有 4 种特权级，其中 0 级的特权最高，3 级最低，而此处是给予用户进程权限，因此给予 FL_IOPL_3。

```
    if (type == ENV_TYPE_FS)
        e->env_tf.tf_eflags |= FL_IOPL_3;
```

2 Question

e->env_tf 会在产生 trap 时，由硬件以及中断处理程序进行保存，在 env_pop_tf() 中恢复。

3 Exercise 2

本质上这就是一个 page fault handler，不同之处在于拷贝信息，一个是从内存中，这个是从硬盘中。

```
        addr = ROUNDDOWN(addr, PGSIZE);
    r = sys_page_alloc(0, addr, PTE_W | PTE_U | PTE_P);
    if (r < 0) panic("bc_pgfault sys_page_alloc error : %e\n", r);

    r = ide_read(blockno * BLKSECTS, addr, BLKSECTS);
    if (r < 0) panic("bc_pgfault ide_read error : %e\n", r);
```

4 Challenge 1

5 Exercise 3

这里 JOS 实现了一个简易的类似 exec 功能的过程——spaw。根据 Trapframe 来使某一个进程的状态变化，从而实现类似 exec 的效果。

```
static int
sys_env_set_trapframe(envid_t envid, struct Trapframe *tf)
{
    // LAB 5: Your code here.
    // Remember to check whether the user has supplied us with a good
    // address!
    struct Env * env;
    int r = envid2env(envid, &env, 1);
    if (r < 0) return -E_BAD_ENV;
```

```
    user_mem_assert (env, tf, sizeof(struct Trapframe), PTE_U);

    env->env_tf = *tf;
    env->env_tf.tf_cs |= 3;
    env->env_tf.tf_eflags |= FL_IF;

    return 0;
}
```

6 Exercise 4

对于 fork,spawn 之后的进程之间对于 file descriptor 是共享的，因此在 duppage 需要考虑这一段。

```
static int
duppage(envid_t envid, unsigned pn)
{
    // do not dup exception stack
    if (pn * PGSIZE == UXSTACKTOP - PGSIZE) return 0;

    int r;
    void * addr = (void *)(pn * PGSIZE);
    if (uvpt[pn] & PTE_SHARE) {
        r = sys_page_map(0, addr, envid, addr, uvpt[pn] & PTE_SYSCALL);
        if (r < 0) panic("duppage sys_page_map error : %e\n", r);
    } else
    if ((uvpt[pn] & PTE_W) || (uvpt[pn] & PTE_COW)) {
        // cow
        r = sys_page_map(0, addr, envid, addr, PTE_COW | PTE_P | PTE_U);
        if (r < 0) panic("duppage sys_page_map error : %e\n", r);

        r = sys_page_map(0, addr, 0, addr, PTE_COW | PTE_P | PTE_U);
        if (r < 0) panic("duppage sys_page_map error : %e\n", r);
    } else {
        // read only
        r = sys_page_map(0, addr, envid, addr, PTE_P | PTE_U);
        if (r < 0) panic("duppage sys_page_map error : %e\n", r);
    }

    return 0;
}
```

对于 spawn 也需要进行这部分的映射。

```
// Copy the mappings for shared pages into the child address space.
static int
copy_shared_pages(envid_t child)
{
    // LAB 5: Your code here.
    uint32_t i;
    int r;
    for (i = 0; i != UTOP; i += PGSIZE)
```

```
    if ((uvpd[PDX(i)] & PTE_P) && (uvpt[i / PGSIZE] & PTE_P) && (uvpt[i / PGSIZE
        ] & PTE_SHARE)) {
        r = sys_page_map(0, (void *)i, child, (void *)i, uvpt[i / PGSIZE] &
            PTE_SYSCALL);
        if (r < 0) return r;
    }
    return 0;
}
```

7 Exercise 5

这部分比较简单，增加 trap 处理判断即可。

8 Question

2. About 10 hours.

3. 这部分的 exercise 比较少，需要看的代码比较多，对于 file I/O 有了解，但是因为经过写的训练，总觉的有点陌生。