BUAA-OS-lab3

一、实验思考题

Thinking 3.1

> 请结合MOS中的页目录自映射应用解释代码中e->env_pgdir[PDX(UVPT)] = PADDR(e->env_pgdir) | PTE_V 的含义。

进程e中页目录的第PDX(UVPT)个页目录项等于页目录所在页表的基地址加上页表项有效位。

Thinking 3.2

> elf_load_seg以函数指针的形式，接受外部自定义的回调函数map_page。 请你找到与之相关的data这一参数在此处的来源，并思考它的作用。没有这个参数可不可 以？为什么？

data的来源是Env结构体指针，它的作用是传给elf_load_seg函数，并作为map_page函数的参数，获得结构体的页目录。

Thinking 3.3

> 结合elf_load_seg的参数和实现，考虑该函数需要处理哪些页面加载的情况。

该函数需要考虑bin_size起始点、bin_size大小和sgsize大小以及地址对齐情况。

Thinking 3.4

> 思考上面这一段话，并根据自己在Lab2中的理解，回答：
>
> 你认为这里的env_tf.cp0_epc存储的是物理地址还是虚拟地址?

我认为env_tf.cp0_epc存储的是物理地址。

Thinking 3.5

> 试找出0、1、2、3号异常处理函数的具体实现位置。8号异常（系统调用） 涉及的do_syscall()函数将在Lab4中实现。

0号异常handle_int在genex.S实现，1号异常的处理函数为handle_mod在tlbex.c中实现，2号异常的处理函数handle_tlb和3号异常的处理函数为handle_tlb在tlb_asm.S中实现。

Thinking 3.6

> 阅读 entry.S、genex.S 和 env_asm.S 这几个文件，并尝试说出时钟中断 在哪些时候开启，在哪些时候关闭。

时钟中断在调用 env_pop_tf 函数时开启，在恢复现场、异常返回后关闭。

二、实验难点

env_init 函数

需要按倒序将所有控制块插入到空闲链表的头部，使得编号更小的进程控制块被优先分配。

```
for (i = NENV - 1; i >= 0; i--) {
        envs[i].env_status = ENV_FREE;
        LIST_INSERT_HEAD(&env_free_list, &envs[i], env_link);
    }
```

env_setup_vm 函数

难点在于为进程页目录分配地址时需要进行的地址转换，此过程需要调用**page2kva**函数将页面转变为虚拟地址，再将其赋值给e->env_pgdir。

```
e->env_pgdir = (Pde *)page2kva(p);
```

env_alloc函数

先从env_free_list中获得新的Env，并判断是否为空。

```
if((e = LIST_FIRST(&env_free_list)) == NULL) {
        return -E_NO_FREE_ENV;
    }
```

再使用env_setup_vm函数初始化Env。

```
env_setup_vm(e);
```

初始化Env中的一系列需要初始化的参数。

```
asid_alloc(&(e->env_asid));
e->env_id = mkenvid(e);
e->env_parent_id = parent_id;
```

再将Env从env_free_list中移除。

```
LIST_REMOVE(e, env_link);
```

load_icode_mapper函数

难点在于如何使用**memcpy**函数将从src开始的len大小的字节复制到这一页的偏移量地址上。注意需要使用**page2kva**将页面地址转化为虚拟地址。

```
if (offset + len <= PAGE_SIZE) {
            memcpy(page2kva(p) + offset, src, len);
        }
```

schedule函数

这是本次实验最难的一部分，需要完成对进程的调度函数。根据注释的提示来完成，我们总是将“计数”减少1。如果设置了'yield'，或者'count'已减少为0，或者'e'(之前的' currenv ')为'NULL'，或者'e'不可运行，那么我们从'env_sched_list'(所有可运行的env列表)中拾取一个新的env，将'count'设置为其优先级，并将其与'env_run'调度。**如果列表是空的**。(注意，如果'e'仍然是一个可运行的环境，我们应该将它移到'env_sched_list'的尾部，然后再从它的头部获取另一个环境，否则我们将重复调度头部环境。)否则，我们只需再次安排“e”。

```
if (e == NULL || count == 0 || e->env_status != ENV_RUNNABLE || yield != 0) {
            if (e != NULL && e->env_status == ENV_RUNNABLE) {
                if (yield && TAILQ_FIRST(&env_sched_list) != NULL) {

                } else {
                    TAILQ_INSERT_TAIL(&env_sched_list, e, env_sched_link);
                }
```

```
            }

            if ((e = TAILQ_FIRST(&env_sched_list)) == NULL) {
                panic("schedule: no runnable envs");
            }
            TAILQ_REMOVE(&env_sched_list, e, env_sched_link);
            count = e->env_pri;
            count--;
            env_run(e);

    } else {
        count--;
        env_run(e);
    }
```

三、体会与感想

个人认为lab3的总体难度是大于之前的lab的，难点在于进程的调度、地址转换以及一些变量含义的理解。从Lab3遇到Bug会比较瞻前顾后，不知道是本次Lab还是之前Lab的bug，排查Bug所在范围的过程比较头疼。