lab3实验报告

思考题

Thinking 3.1

e->env_pgdir[PDX(UVPT)] = PADDR(e->env_pgdir) | PTE_V

理解这段代码需要先明确env_pgdir指的是改进程页目录的内核虚拟地址，

所以PADDR(e->env_pgdir)是该进程页目录的物理地址

所以这段代码就是要将该进程地址空间的UVPT映射到页目录的物理地址，使得UVPT开始是页目录

而这一段UVPT其实处于用户空间，也就是说，进程可以在用户态就访问到页目录，只需要访问UVPT这段空间

Thinking 3.2

宏实现链表的好处

data是该进程控制块的地址，不能没有，不然无法确定加载程序的进程

Thinking 3.3

- 完全被包裹的页面
- 从中间开始的页面
- 在中间结束的页面

Thinking 3.4

虚拟地址

Thinking 3.5

在kern/genex.S中

Thinking 3.6

开启异常处理的时候关闭

从异常处理返回的时候开启

Thinking 3.7

每次时钟中断记录当前进程的信息，比如经过了多少个时间中断，然后根据特定的调度算法，决定下一个运行的进程

难点分析

lab3的mips_init

执行了以下函数

- env_init()

        - LIST_INIT(&env_free_list)
        - TAILQ_INIT(&env_sched_list)
        - 把所有pcb控制块加入到空闲链表
        - 分配一个物理页用来做base_pgdir
            - base_pgdir是全局变量，所代表的地址空间有着所有进程共享的空间，UTOP以上到ULIM以下就是共享空间，UTOP又称UENVS，以上是envs控制块数据，UPAGES以上是pages控制块数据，UVPT以上是页表空间
        - 初始化所有进程的共享空间的数据：
            - 把pages的物理页映射到UPAGES（user pages)
            - 把envs的物理页映射到UENVS(user envs)
            - 使用map_segment,其实就是page_insert的升级版
- ENV_CREATE_PRIORITY(user_bare_loop, 1);
- ENV_CREATE_PRIORITY(user_bare_loop, 2);
    - env_create(const void *binary, size_t size, int priority)
        - 用env_alloc分配一个进程块，从空闲链表中移除，从
        - 设置env_prority和env_status
        - 使用load_icode加载程序
        - TAILQ_INSERT_HEAD将进程插入env_sched_list中
    - 这里的数字是优先级

为什么不用移除空闲链表？因为env_alloc的时候就已经移出了

这里我发现init只是创建了进程，但是进程没有开始运行，只是返回了一个进程控制块，那么什么时候这个创建的进程才开始运行呢?

其实是需要中断来完成的。

特定硬件会定时发出时钟中断，这种中断对于CPU来说是一种异常

而CPU处理异常有一套标准的流程:有异常的时候跳转到固定地址进行处理，

- 设置EPC指向从异常返回的地址
- 设置EXL位
- 设置Cause
- CPU开始从异常入口位置取值，进入异常分发程序：
    - 使用SAVE_ALL宏将当前上下文保存到内核异常栈中
    - 清楚Status寄存器中的UM、EXL、IE位，以保持处理器处于内核态
    - 使用Cause中的2-6位判断异常类型，找到对应的中断处理函数，这里需要异常向量组exception_handlers
    - 跳转到对应的中断处理函数中，我们时钟中断的中断处理程序的最后调用了schedule,就是调度函数，而schedule最后调用了env_run开始运行特定的进程

实验体会

lab3实现了进程，需要结合异常和中断理解进程的切换和调度，以及需要理解用户空间中内核的部分，用户空间有pages和envs和页表