Lab Report 3

Wang Haotian

519021910685

Grizzy@sjtu.edu.cn

练习1

首先运行 `_start`，跳转到 `init_c` 执行必要的初始化操作，然后跳转到 `start_kernel` 函数，这些都是在内核态运行的

在 `start_kernel` 最后跳转到 `main` 函数，初始化uart，初始化内存管理系统，配置异常向量表，然后调用 `create_root_thread` 创建第一个用户态进程，并设置当前线程为root thread

在 `main` 函数的最后，获取目标线程的上下文，通过 `eret_to_thread` 实现上下文切换，完成了内核态到第一个用户态线程的切换。

练习2

根据提示完成代码

练习3

```
/* load binary into some process (cap_group) */
static u64 load_binary(struct cap_group *cap_group, struct vmspace *vmspace,
                       const char *bin, struct process_metadata *metadata)
{
        struct elf_file *elf;
        vmr_prop_t flags;
        int i, r;
        size_t mem_sz, file_sz, seg_map_sz;
        u64 p_vaddr;

        int *pmo_cap;
        struct pmobject *pmo;
        u64 ret;

        elf = elf_parse_file(bin);
        pmo_cap = kmalloc(elf->header.e_phnum * sizeof(*pmo_cap));
        if (!pmo_cap) {
                r = -ENOMEM;
                goto out_fail;
```

```
        }

        /* load each segment in the elf binary */
        for (i = 0; i < elf->header.e_phnum; ++i) {
                pmo_cap[i] = -1;
                if (elf->p_headers[i].p_type == PT_LOAD) {
                        mem_sz = elf->p_headers[i].p_memsz;
                        p_vaddr = elf->p_headers[i].p_vaddr;
                        /* LAB 3 TODO BEGIN */
                        file_sz = elf->p_headers[i].p_filesz;
                        flags = PFLAGS2VMRFLAGS(elf->p_headers[i].p_flags);
                        seg_map_sz = ROUND_UP(p_vaddr + mem_sz, PAGE_SIZE) -
ROUND_DOWN(p_vaddr, PAGE_SIZE);
                        create_pmo(seg_map_sz, PMO_DATA, cap_group, &pmo);
                        ret = vmspace_map_range(vmspace, p_vaddr, seg_map_sz, flags, pmo);
                        /* load data */
                        memcpy(phys_to_virt(pmo->start), bin + elf->p_headers[i].p_offset,
file_sz);
                        /* fill bss with zero */
                        memset(phys_to_virt(pmo->start) + file_sz, 0, mem_sz - file_sz);
                        /* LAB 3 TODO END */
                        BUG_ON(ret != 0);
                }
        }

        /* return binary metadata */
        if (metadata != NULL) {
                metadata->phdr_addr =
                        elf->p_headers[0].p_vaddr + elf->header.e_phoff;
                metadata->phentsize = elf->header.e_phentsize;
                metadata->phnum = elf->header.e_phnum;
                metadata->flags = elf->header.e_flags;
                metadata->entry = elf->header.e_entry;
        }

        /* PC: the entry point */
        return elf->header.e_entry;
out_free_cap:
        for (--i; i >= 0; i--) {
                if (pmo_cap[i] != 0)
                        cap_free(cap_group, pmo_cap[i]);
        }
out_fail:
        return r;
}
```

练习4

按照课程讲解配置异常向量表

练习5

```
ret = handle_trans_fault(current_thread->vmspace, fault_addr);
```

练习6

分配page，并初始化值为0

将page提交给pmo

建立页表映射并设置flag

练习7

保存和恢复各种寄存器

```
.macro  exception_enter
  /* LAB 3 TODO BEGIN */
  sub sp, sp, #ARCH_EXEC_CONT_SIZE
  stp x0, x1, [sp, #16 * 0]
  stp x2, x3, [sp, #16 * 1]
  stp x4, x5, [sp, #16 * 2]
  stp x6, x7, [sp, #16 * 3]
  stp x8, x9, [sp, #16 * 4]
  stp x10, x11, [sp, #16 * 5]
  stp x12, x13, [sp, #16 * 6]
  stp x14, x15, [sp, #16 * 7]
  stp x16, x17, [sp, #16 * 8]
  stp x18, x19, [sp, #16 * 9]
  stp x20, x21, [sp, #16 * 10]
  stp x22, x23, [sp, #16 * 11]
  stp x24, x25, [sp, #16 * 12]
  stp x26, x27, [sp, #16 * 13]
  stp x28, x29, [sp, #16 * 14]
  /* LAB 3 TODO END */
  mrs x21, sp_el0
  mrs x22, elr_el1
  mrs x23, spsr_el1
  /* LAB 3 TODO BEGIN */
  stp x30, x21, [sp, #16 * 15]
  stp x22, x23, [sp, #16 * 16]
  /* LAB 3 TODO END */
.endm

.macro  exception_exit
  /* LAB 3 TODO BEGIN */
  ldp x22, x23, [sp, #16 * 16]
```

```
    ldp x30, x21, [sp, #16 * 15]
    /* LAB 3 TODO END */
    msr sp_el0, x21
    msr elr_el1, x22
    msr spsr_el1, x23
    /* LAB 3 TODO BEGIN */
    ldp x0, x1, [sp, #16 * 0]
    ldp x2, x3, [sp, #16 * 1]
    ldp x4, x5, [sp, #16 * 2]
    ldp x6, x7, [sp, #16 * 3]
    ldp x8, x9, [sp, #16 * 4]
    ldp x10, x11, [sp, #16 * 5]
    ldp x12, x13, [sp, #16 * 6]
    ldp x14, x15, [sp, #16 * 7]
    ldp x16, x17, [sp, #16 * 8]
    ldp x18, x19, [sp, #16 * 9]
    ldp x20, x21, [sp, #16 * 10]
    ldp x22, x23, [sp, #16 * 11]
    ldp x24, x25, [sp, #16 * 12]
    ldp x26, x27, [sp, #16 * 13]
    ldp x28, x29, [sp, #16 * 14]
    add sp, sp, #ARCH_EXEC_CONT_SIZE
    /* LAB 3 TODO END */
    eret
.endm
```

练习8

见代码