练习 1

按照注释中给出的提示依次完成即可。

练习 2

该函数代码如下：

```
void process_create_root(char *bin_name)
{
        struct process *root_process;
        int thread_cap;
        struct thread *root_thread;
        char *binary = NULL;
        int ret;

        ret = ramdisk_read_file(bin_name, &binary);
        BUG_ON(ret < 0);
        BUG_ON(binary == NULL);

        root_process = process_create();

        thread_cap = thread_create_main(root_process, ROOT_THREAD_STACK_BASE,
                                        ROOT_THREAD_STACK_SIZE,
                                        ROOT_THREAD_PRIO, TYPE_ROOT,
                                        0, binary, bin_name);

        root_thread = obj_get(root_process, thread_cap, TYPE_THREAD);
        /* Enqueue: put init thread into the ready queue */
        obj_put(root_thread);
        current_thread = root_thread;
}
```

函数调用图如下：

1. ramdisk_read_file
    a. cpio_extract_single
2. process_create
    a. process_init
    b. vmspace_init
3. thread_create_main
    a. vmspace_map_range
    b. load_binary
        i. elf_parse_file
        ii. vmspace_map_range
    c. prepare_env
    d. thread_init
        i. create_thread_ctx
        ii. init_thread_ctx

首先通过 ramdisk_read_file 函数读取二进制文件，然后创建一个进程，再使用 thread_create_main 函数为这个进程创建一个主线程。thread_create_main 函数首先获取进程的 vmspace，然后为这个线程分配和设置一个用户栈，接下来把这个栈映射到特定的虚拟内存地址，然后解析 ELF 文件，并将其内容加载到这个线程的用户内存空间中，最后一步是准备环境、创建并初始化线程的上下文。

练习 3

修改 exception_table.S 的内容：PPT 中有参考代码，照搬即可。
exception_init 函数：调用 set_exception_vector()函数即可。
修改 handle_entry_c：该练习只要求在 ESR_EL1_EC_UNKNOWN 发生是打印信息并中止用户进程，其他情形无需处理。

练习 4

系统调用对应于异常向量表中的 sync_el0_64，当发生系统调用时，就会执行异常向量表中 sync_el0_64 对应的过程，该过程比较 esr_el1 的值是否等于 ESR_EL1_EC_SVC_64，结果是等于，然后就会跳转到 el0_syscall 过程进行系统调用。

练习 5

参考代码可以在 PPT 中找到。

练习 6

宏已经在 syscall.h 被定义，调用 syscall 函数并填入正确的 sys_no 及参数即可。
vm_syscall.c 中的 sys_handle_brk 函数需要被完善，按照注释给出的提示照做即可。

练习 7

在 START 函数入口设置断点，连续使用“n”指令可以发现下面的结果：

```
Single stepping until exit from function main,
which has no line number information.
0x0000000000000000 in ?? ()
```

此时程序计数器的值为 0x0，这是因为 main 函数具有特殊性，在调用 main 函数时并没有对 x30 进行设置，所以当 main 函数返回时 x30 的值为 0，这意味着下一个要执行的指令地址为 0，而显然这是一个无效的地址。

练习 8

可以在_start_c 函数末尾添上一行系统调用：

```
usys_exit(ret);
```

练习 9

对于 ESR_EL1_EC_DABT_LEL 和 ESR_EL1_EC_DABT_CEL 类型的异常，要调用 do_page_fault 函数进行处理，然后按照注释给出的提示完善 handle_trans_fault()函数即可。