浙江大学实验报告

课程名称：操作系统
实验项目名称：RV64 用户态程序
学生姓名：展翼飞 学号：3190102196
电子邮件地址：1007921963@qq.com
实验日期： 2024年 12 月 1 日

一、实验内容

1. 准备工程

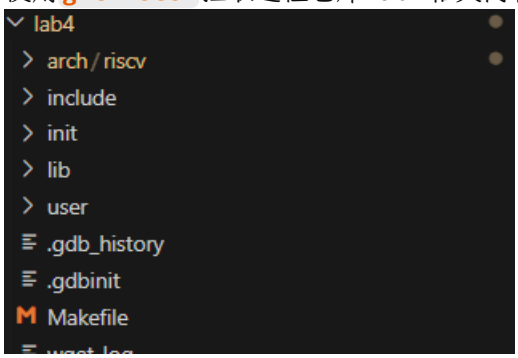
- 修改 `vmlinux.lds`，将用户态程序 `uapp` 加载至 `.data` 段：

```
    .data : ALIGN(0x1000) {
        _sdata = .;

        *(.sdata .sdata*)
        *(.data .data.*)

        _edata = .;

        . = ALIGN(0x1000);
        _sramdisk = .;
        *(.uapp .uapp*)
        _eramdisk = .;
        . = ALIGN(0x1000);
    } >ramv AT>ram
```

- 修改 `defs.h`，在 `defs.h` **添加**如下内容：

```
#define USER_START (0x0000000000000000) // user space start virtual address
#define USER_END (0x0000004000000000) // user space end virtual address
```

- 使用`git fetch`拉取远程仓库Lab4相关内容，同步并接受新文件：



- 修改**根目录**下的 Makefile, 将 `user` 文件夹下的内容纳入工程管理：

```
all: clean
    $(MAKE) -C lib all
    $(MAKE) -C init all
    $(MAKE) -C user all
    $(MAKE) -C arch/riscv all
    @echo -e '\n'Build Finished OK

run: all
    @echo Launch qemu...
    @qemu-system-riscv64 -nographic -machine virt -kern

debug: all
    @echo Launch qemu for debug...
    @qemu-system-riscv64 -nographic -machine virt -kern

clean:
    $(MAKE) -C lib clean
    $(MAKE) -C init clean
    $(MAKE) -C user clean
    $(MAKE) -C arch/riscv clean
```

2.创建用户态进程

- 修改 `proc.h` 中的 `NR_TASKS` ：
  由于创建用户态进程要对`sepc`，`sstatus`，`sscratch`做设置，我们需要将其加入 `thread_struct` 中且多个用户态进程需要保证相对隔离，因此不可以共用页表，我们需要为每个用户态进程都创建一个页表并记录在 `task_struct` 中

```
struct thread_struct {
    uint64_t ra;
    uint64_t sp;
    uint64_t s[12];
    uint64_t sepc, sstatus, sscratch;
};

struct task_struct {
    uint64_t state;
    uint64_t counter;
    uint64_t priority;
    uint64_t pid;

    struct thread_struct thread;
    uint64_t *pgd;  // 用户态页表
};
```

3.修改`__switch_to`

前面新增了 sepc、sstatus、sscratch 之后，需要将这些变量在切换进程时保存在栈上，因此需要更新 `__switch_to` 中的逻辑，同时需要增加切换页表的逻辑。在切换了页表之后，需要通过 `sfence.vma` 来刷新 TLB 和 ICache
尚未debug完成

4.更新中断处理逻辑

由于我们的用户态进程运行在 U-Mode 下，使用的运行栈也是用户栈，因此当触发异常时，我们首先要对栈进行切换（从用户栈切换到内核栈）。同理，当我们完成了异常处理，从 S-Mode 返回至 U-Mode 时，也需要进行栈切换（从内核栈切换到用户栈）

5.添加系统调用

6.调整时钟中断

二、思考题

1. 我们在实验中使用的用户态线程和内核态线程的对应关系是怎样的？（一对一，一对多，多对一还是多对多）
   一对一
2. 系统调用返回为什么不能直接修改寄存器？
3. 针对系统调用，为什么要手动将 sepc + 4？
4. 为什么 Phdr 中，`p_filesz` 和 `p_memsz` 是不一样大的，它们分别表示什么？
5. 为什么多个进程的栈虚拟地址可以是相同的？用户有没有常规的方法知道自己栈所在的物理地址？

三、讨论心得

本次实验难度较大，因为Lab3debug完成较晚，未留出充足时间理解并完成实验