Lab5

#### **练习0：填写已有实验**

本实验依赖实验2/3/4。请把你做的实验2/3/4的代码填入本实验中代码中有“LAB2”/“LAB3”/“LAB4”的注释相应部分。注意：为了能够正确执行lab5的测试应用程序，可能需对已完成的实验2/3/4的代码进行进一步改进。

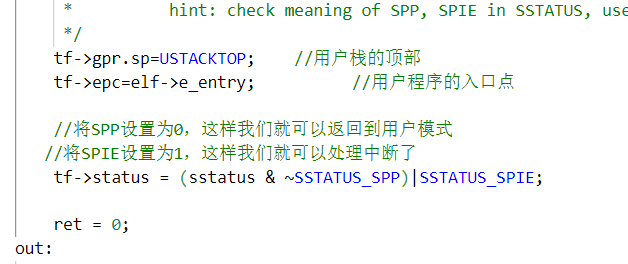
#### **练习1: 加载应用程序并执行（需要编码）**

****do\_execv****函数调用load\_icode（位于kern/process/proc.c中）来加载并解析一个处于内存中的ELF执行文件格式的应用程序。你需要补充load\_icode的第6步，建立相应的用户内存空间来放置应用程序的代码段、数据段等，且要设置好proc\_struct结构中的成员变量trapframe中的内容，确保在执行此进程后，能够从应用程序设定的起始执行地址开始执行。需设置正确的trapframe内容。

请在实验报告中简要说明你的设计实现过程。

* 请简要描述这个用户态进程被ucore选择占用CPU执行（RUNNING态）到具体执行应用程序第一条指令的整个经过。

proc.cpp/load\_icode



proc.cpp/do\_fork



### 系统调用（system call)，是用户态(U mode)的程序获取内核态（S mode)服务的方法，所以需要在用户态和内核态都加入对应的支持和处理。我们也可以认为用户态只是提供一个调用的接口，真正的处理都在内核态进行

用ecall从U mode进入S mode之后,对应的处理需要我们编写内核系统调用的代码来完成.

sys\_fork()：把当前的进程复制一份，创建一个子进程，原先的进程是父进程。接下来两个进程都会收到sys\_fork()的返回值，如果返回0说明当前位于子进程中，返回一个非0的值（子进程的PID）说明当前位于父进程中。然后就可以根据返回值的不同，在两个进程里进行不同的处理。

sys\_exec()：在当前的进程下，停止原先正在运行的程序，开始执行一个新程序。PID不变，但是内存空间要重新分配，执行的机器代码发生了改变。我们可以用fork()和exec()配合，在当前程序不停止的情况下，开始执行另一个程序。

sys\_exit()：退出当前的进程。

sys\_wait()：挂起当前的进程，等到特定条件满足的时候再继续执行。

在用户态进行系统调用的核心操作是，通过内联汇编进行ecall环境调用。这将产生一个trap, 进入S mode进行异常处理。

#### **练习2: 父进程复制自己的内存空间给子进程（需要编码）**

创建子进程的函数do\_fork在执行中将拷贝当前进程（即父进程）的用户内存地址空间中的合法内容到新进程中（子进程），完成内存资源的复制。具体是通过copy\_range函数（位于kern/mm/pmm.c中）实现的，请补充copy\_range的实现，确保能够正确执行。

请在实验报告中简要说明你的设计实现过程。

* 如何设计实现Copy on Write机制？给出概要设计，鼓励给出详细设计。

#### 

#### **练习3: 阅读分析源代码，理解进程执行 fork/exec/wait/exit 的实现，以及系统调用的实现（不需要编码）**

请在实验报告中简要说明你对 fork/exec/wait/exit函数的分析。并回答如下问题：

* 请分析fork/exec/wait/exit的执行流程。重点关注哪些操作是在用户态完成，哪些是在内核态完成？内核态与用户态程序是如何交错执行的？内核态执行结果是如何返回给用户程序的？
* 请给出ucore中一个用户态进程的执行状态生命周期图（包执行状态，执行状态之间的变换关系，以及产生变换的事件或函数调用）。（字符方式画即可）

执行：make grade。如果所显示的应用程序检测都输出ok，则基本正确。（使用的是qemu-1.0.1）