LAB5 田新韬 2012011268

Ex0

先diff修改后的lab4与原lab4，再patch到lab5中

Ex1 加载应用程序并执行

更新了LAB4中的代码，在proc\_alloc方法中额外初始化了成员wait\_state, cptr, yptr, optr；

在do\_fork方法中调用set\_links

初始化tf中的变量，按照注释编码即可。

然后跑不通，发现需要更新LAB1中代码，使系统调用对应的中断描述符可以在用户态被调用。同时需要更新LAB3中代码，使处理时钟中断时每过TICKNUM将当前进程设置为need\_reschedule。

请在实验报告中描述当创建一个用户态进程并加载了应用程序后，CPU是如何让这个应用程序最终在用户态执行起来的。即这个用户态进程被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。

用户态进程调用链：

exec(中断处理,修改eip)->syscall(查找系统调用号)->sys\_exec->do\_execve(换PDT，换页表)->load\_icode(初始化内存空间，加载elf，current->tf)

中断返回，切换堆栈，转换特权级，跳转到用户程序入口。

Ex2 实现Copy\_range

按照注释提示编码即可。

请在实验报告中简要说明如何设计实现”Copy on Write 机制“，给出概要设计，鼓励给出详细设计。

在拷贝页时，如果是读操作，将原来的页W置0，刷新TLB，返回原来的页；如果是写操作，进行拷贝。

对于引用计数1的页面，写操作则直接写，不复制。

若尝试对只读页面进行写操作，抛出异常，并在异常处理中拷贝页面给试图写操作的进程，修改页表，重新映射虚拟地址，W置1。

Ex3 fork/exec/wait/exit

请分析fork/exec/wait/exit在实现中是如何影响进程的执行状态的？

fork不改变当前进程的执行状态，对于fork出来的新进程，先是uninit，在wakeup\_proc过后变为runnable。

exec不改变进程执行状态。

wait，若存在zombie状态子进程，不会改变执行状态，若不存在，将当前进程置为sleeping，直至出现zombie状态的子进程，exit掉子进程，将父进程置为runnable。

exit,先将该进程置为zombie，如有父进程则唤醒（由sleeping置为runnable）。

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）

--alloc\_page()--> UNINIT --wakeup\_proc()--> RUNNABLE – proc\_run()-->RUNNING

--do\_exit()--> ZOMBIE --do\_wait()--> exit

实验结果

 