练习1：分配并初始化一个进程控制块。

alloc\_proc函数：(这个函数里是对各个变量的初始化)

alloc\_proc(void) {

struct proc\_struct \*proc = kmalloc(sizeof(struct proc\_struct));

if (proc != NULL) {

proc->state = PROC\_UNINIT;

proc->pid = -1;

proc->runs = 0;

proc->kstack = 0;

proc->need\_resched = 0;

proc->parent = NULL;

proc->mm = NULL;

memset(&(proc->context), 0, sizeof(struct context));

proc->tf = NULL;

proc->cr3 = boot\_cr3;

proc->flags = 0;

memset(proc->name, 0, PROC\_NAME\_LEN);

}

return proc;

}

请说明proc\_struct中 struct context context 和 struct trapframe \*tf 成员变量含义和在 本实验中的作用是啥？

struct context context

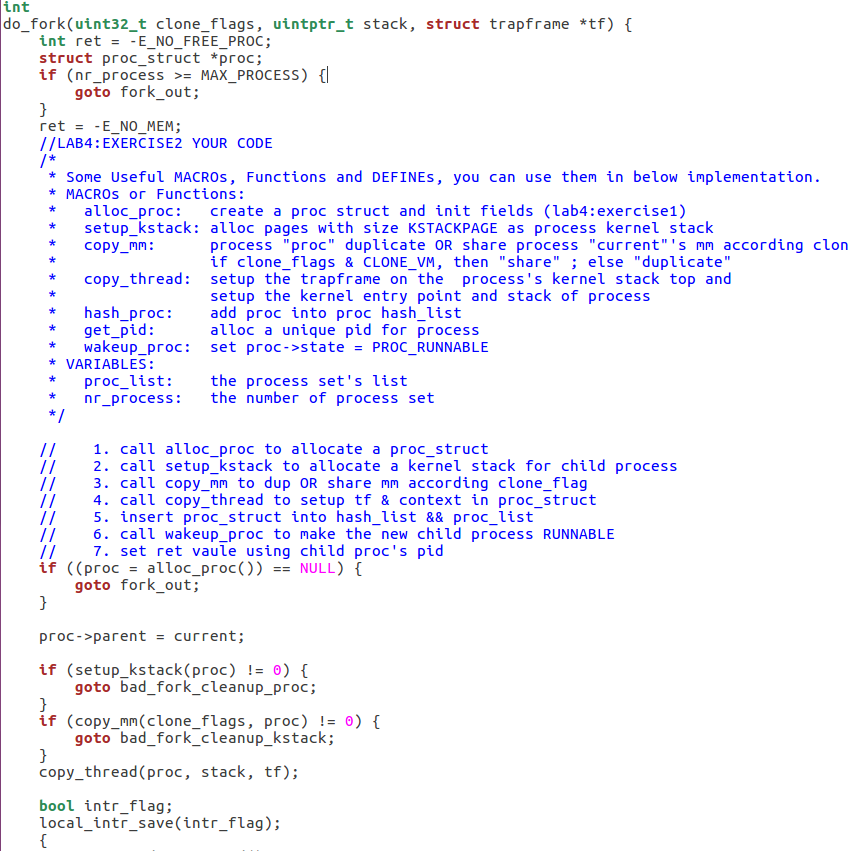
从截图中的代码注释可以看出这一变量是转移到这里开始另外的一个函数。

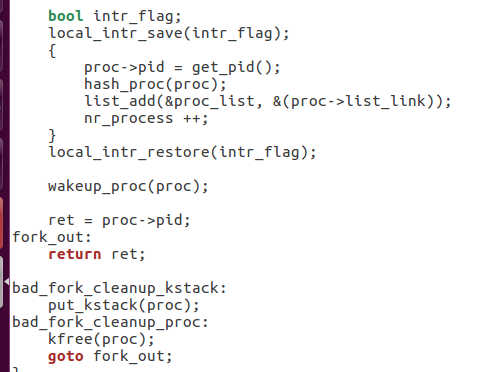
struct trapframe \*tf：



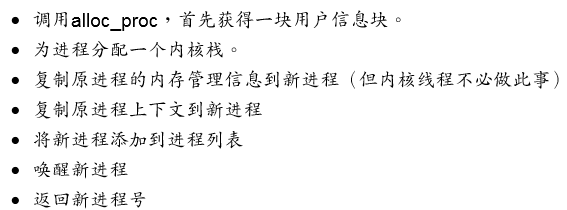
从截图中的代码注释可以看出这个是中断帧的指针。

练习2：为新创建的内核线程分配资源 do\_fork()的代码截图如下：





这个代码我是直接查看的answer中的内容，有一部分感觉好神奇，此代码首先用了alloc\_proc分配了用户信息块，然后用到了copy\_thread()复制了原线程的上下文的相关内容。反正大概上面的每一个函数就是对应的下图介绍的函数功能的内容：



思考题：请说明ucore是否做到给每个新fork的线程一个唯一的id？请说明你的分析和理由。

答：这个肯定是可以的，从我们课本中学到的内容就可以得到这个问题的肯定答案。这里应该是用到了同步与互斥保证了fork的线程有一个唯一的id。

练习3：阅读代码，理解 proc\_run 函数和它调用的函数如何完成 进程切换的。

proc\_run 用于使有个线程再cpu中运行，可以看到它的主要过程为：

1. 将当前进程设为传入的进程，
2. 修改esp的值
3. 修改页表项
4. 最后用switch\_to进行上下文额切换。

思考题(这个是借鉴网上的别人的博客)：

1. 本实验的执行过程中，创建并运行了几个内核线程？

答：2个，创建第0个内核线程idleproc和第1个内核线程initproc。

2. 语句 local\_intr\_save(intr\_flag);....local\_intr\_restore(intr\_flag); 在这里有何作用?

答：关闭中断和打开中断。有些过程是互斥的，只允许一个线程进入，因此需要关闭中断来处理临界区。