练习1、

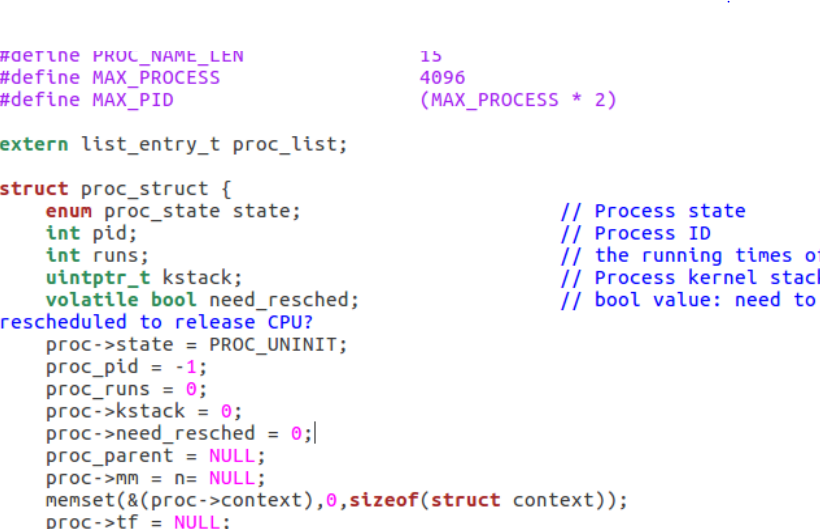
proc->state = PROC\_UNINIT;

proc->pid = =-1;

proc->runs = 0;

proc->kstack = 0;

proc->need\_resched = 0;



题目：分配并初始化一个进程控制块

Alloc\_proc函数负责分配并返回一份新的struct\_struct结构，用于存储新建立的内核线程的管理星系。Ucore需要对这个机构进行最基本的初始，你需要完成这个初始化的过程。

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结构体的定义，可以发现在结构体中存储这除了eax之外的所有通用寄存器以及eip的数值，这就提示我们这个线程控制块中的context很有可能是保存的线程运行的上下文信息；

接下来同样在代码中寻找对tf变量进行了定义的地方，最后可以发现在copy\_thread函数中对tf进行了设置

练习2、

为新创建的内核线程分配资源

代码:

Int

do\_fork(uint32\_t clone\_flags,uintptr\_t stack,struct trapframe \*tf){

int ret = =-E\_NO\_FREE\_PROC;

struct proc\_struct \*prorc;

if(nr\_process >= MAX\_PROCESS){

goto fork\_out;

}

ret = -E\_NO\_MEM;

proc = alloc\_proc();

if(proc == NULL){

goto fork\_out;

}

proc ->parent = current;

if(setup\_kstack(proc) != 0){

goto bad\_fork\_cleanup\_proc;

}

If(copy\_mm(clone\_flags,proc) != 0){

goto bad\_fork\_cleanup\_kstack;

}

copy+thread(proc,stack,tf);

bool intr\_flag;

local\_intr\_save(intr\_flag){

proc -> pid = get\_pid();

hash\_proc(proc);

nr\_process++;

list\_add\_before(&proc\_list,&proc->list\_link);

}

local\_intr\_restore(intr\_flag);

wakeup\_proc(proc);

ret = proc->pid;

fork\_out:

return ret;

bad\_fork\_cleanup\_kstack:

put\_ksatck(proc);

bad\_fork\_cleanup\_proc:

kfree(proc);

goto forl\_out;

}

static int

get\_pid(void){

static\_assert(MAX\_PID > MAX\_PROCESS);

struct proc\_struct \*proc;

list\_entry\_t \*list = &proc\_list,\*le;

static int next\_safe = =MAX\_PID,last\_pid = MAX+PID;

if(++last\_pid >= MAX\_PID){

last+pid = 1;

goto inside;

}

If(last+pid >= next\_safe){

Inside:

next\_safe = MAX+PID;

repeat:

le = list;

while((le = list\_next(le)) != list){

proc = le2proc(le,list\_link);

if(proc->pid == last\_pid){

if(++last+pid >= next+safe){

if(last\_pid >= MAX\_PID){

last+pid = 1;

}

next\_safe = MAX\_PID;

goto repeat;

}

}

else if(proc->pid > last\_pid &&next\_safe > proc->pid){

next\_safe = =proc->pid;

}

}

}

Return last\_pid;

}

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。

由于进程要求唯一性，进程号分配时可能需要查看进程 列表中高的全部进程以避免发生冲突。若进程号已分配而进程尚未添加进进程列表时被终端，则该进程号可能会被重复分配，故进程号分配与进程添加应为原子操作。因而在进行上述操作时需关闭中断。