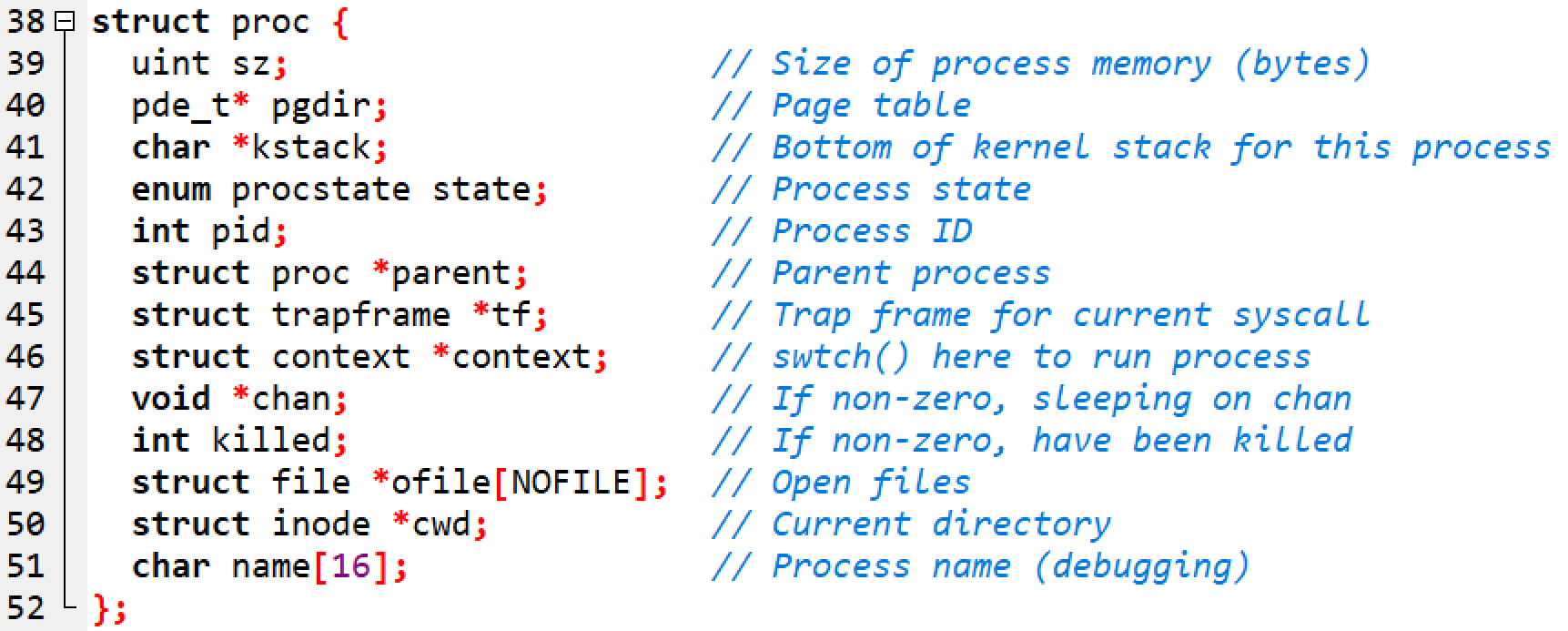
**操作系统 第三次作业**

魏旭晨

学号2016K8009908010

**一、**

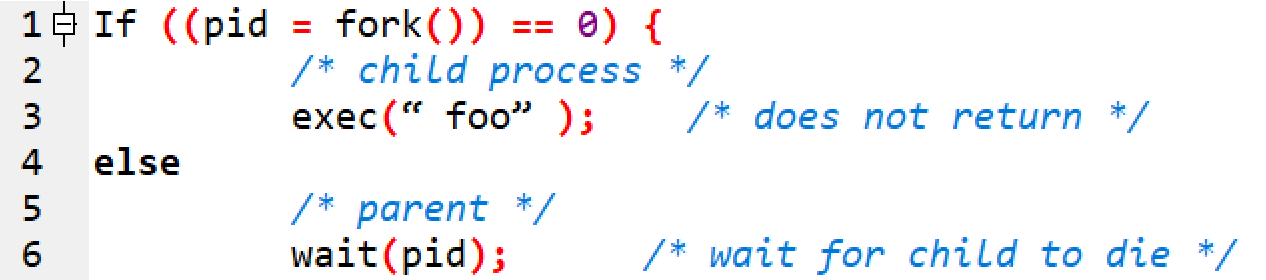
通过阅读原代码和查询网上资料，我了解到PCB是通过数据结构struct pro来表示的，存放在文件pro.h中，PCB具体变量及其注释如下图所示：



**二、**

fork（）函数通过系统调用创建一个与原来进程几乎完全相同的进程，在完成对其调用之后，会产生两个进程，且每个进程都会从fork()的返回处开始执行。这两个进程将执行相同的程序段，但是拥有各自不同的堆段，栈段，数据段，每个子程序都可修改各自的数据段，堆段，和栈段。

那么如何去区分父进程与子进程呢？我们采取以下方式：在子进程中，fork函数返回0，在父进程中，fork返回新创建子进程的进程ID。我们可以通过fork返回的值pid来判断当前进程是子进程还是父进程。



如上图所示：

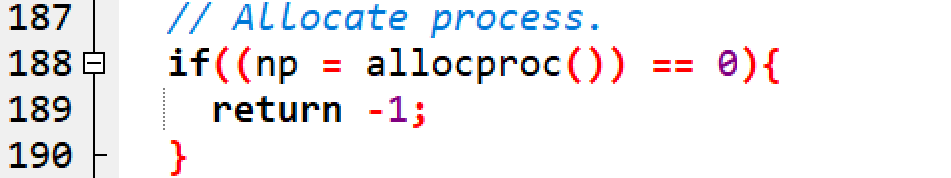
如果返回值pid=0，说明执行的是子进程，下一步该子进程执行exec(“foo”)操作。exec函数将找到可执行文件“foo”，并用它来取代原调用进程的数据段、代码段和堆栈段，仅保留进程号。

如果返回值pid不等于0，说明执行的是父进程，下一步该父进程执行wait(pid)操作。即等待进程号为pid的进程（即子进程）执行完毕然后死亡之后父进程才会继续执行。

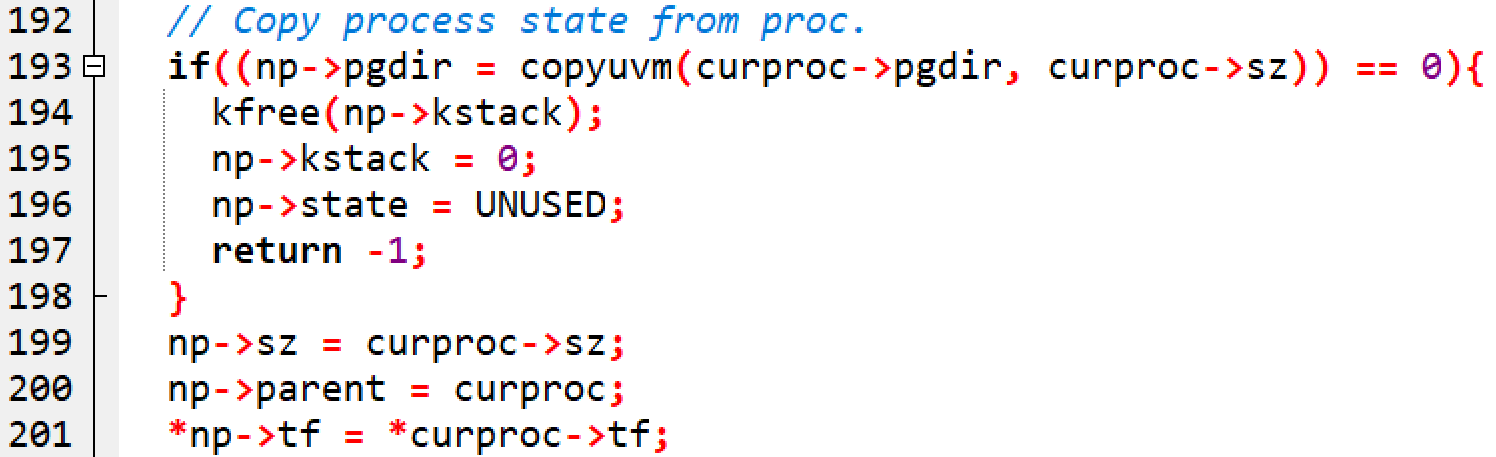
下面是代码中函数的详细分析：

1. folk函数

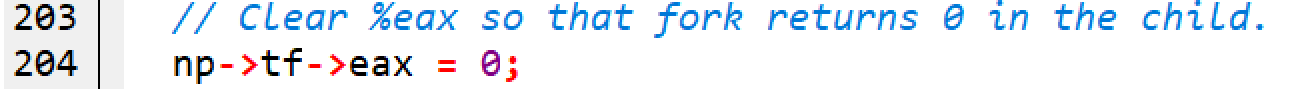
（1）通过allocproc函数创建了一个新的进程。并由可知新进程的pid为现有进程的pid再加1。如下图所示：



（2）把父进程的几乎所有的struct proc中的信息复制到子进程，除了parent属性指向了父进程。如下图所示：



（3）清除子进程的tf的eax以确保子进程的返回值是0。



1. exec函数

exec函数存在exec.c文件中，内容较多，读过几遍后了解到在执行过程中改变的struct proc中的属性有：

sz // Size of process memory (bytes)

pgdir // Page table

tf // Trap frame for current syscall

name // Process name (debugging)

1. wait函数
2. 扫描所有进程表中的子进程，找到目前正在执行的父进程的儿子
3. 如果它的状态为ZOMBIE，则释放它所占用的栈和空间，再将其进程号、父进程、name、killed变为0，将state变为unused状态。即进行如下图的操作：

