**实验六：实现五状态的进程模型**

**实验者：张海涛，张晗宇，张浩然，张镓伟**

**学号：15352405,15352406,15352407,15352408**

**院系：数据科学与计算机学院**

**专业：15级软件工程（移动信息工程）**

**指导老师：凌应标**

**【实验题目】**

在设计的操作系统中实现五状态的进程模型

**【实验目的】**

保留原型原有特征的基础上，设计满足下列要求的新原型操作系统：

(1)实现控制的基本原语do\_fork()、 do\_wait()、do\_exit()、blocked()和wakeup()。

(2)内核实现三系统调用fork()、 wait()和exit() ，并在c库中封装相关的系统调用.

(3)编写一个c语言程序，实现多进程合作的应用程序。

**【实验要求】**

1. 创建8个空的PCB结构块
2. 实现控制的基本原语do\_fork(),do\_wait(),do\_exit(),blocked()和wakeup().
3. 内核实现三系统调用fork（），wait()和exit（）,并在c库中封装相关的系统调用。
4. 编写一个c语言程序，实现多进程合作的应用程序。
5. 多程序合作程序：由父进程生成一个字符串，交给子进程统计其中字母的个数，然后在父进程中输出这一统计结果

**【实验方案】**

**一.虚拟机配置方法**

无操作系统，10M硬盘，4MB内存，启动时连接软盘

**二.软件工具和作用**

Notepad++:用于生成.汇编语言文件

Nasm：用于编译.asm类型的汇编语言文件，生成bin文件

Vmware Workstation 12Player: 用于创建虚拟机，模拟裸机环境

TCC：用于由.c文件生成.obj文件

TNASM：用于由.asm文件生成.obj文件。

Tlink：用于链接obj文件生产.com可执行文件

DOSＢＯＸ：用于运行TCC，TASM与TLINK

**三.实验原理**

1. 五状态进程模型

在五状态进程模型中，进程状态被分成下列五种状态。进程在运行过程中主要是在就绪、运行和阻塞三种状态间进行转换。创建状态和退出状态描述进程创建过程和进程退出过程。

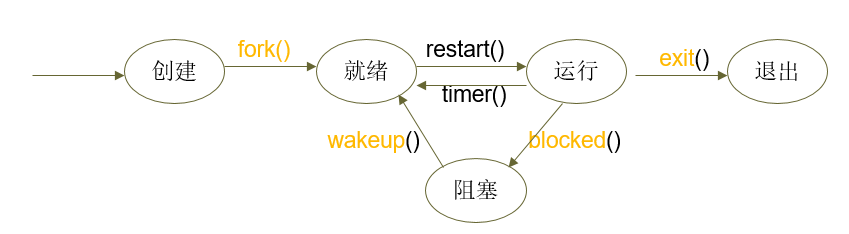
1)运行状态(Running)：进程占用处理器资源；处于此状态的进程的数目小于等于处理器的数目。在没有其他进程可以执行时(如所有进程都在阻塞状态)，通常会自动执行系统的空闲进程。

2)就绪状态(Ready)：进程已获得除处理器外的所需资源，等待分配处理器资源；只要分配了处理器进程就可执行。就绪进程可以按多个优先级来划分队列。例如，当一个进程由于时间片用完而进入就绪状态时，排人低优先级队列；当进程由I／O操作完成而进入就绪状态时，排入高优先级队列。

3)阻塞状态(Blocked)：当进程由于等待I/O操作或进程同步等条件而暂停运行时，它处于阻塞状态。

4)创建状态(New)：进程正在创建过程中，还不能运行。操作系统在创建状态要进行的工作包括分配和建立进程控制块表项、建立资源表格(如打开文件表)并分配资源、加载程序并建立地址空间表等。

5)退出状态(Exit)：进程已结束运行，回收除进程控制块之外的其他资源，并让其他进程从进程控制块中收集有关信息(如记帐和将退出代码传递给父进程)。



2. do\_fork()的功能描述

参考unix早期的fork()做法，我们实现的进程创建功能中，父子进程共享代码段和全局数据段。子进程的执行点(CS:IP)从父进程中继承过来，复制而得。创建成功后，

父子进程以后执行轨迹取决于各自的条件和机遇，即程序代码、内核的调度过程和同步要求。

系统调用的返值如何获得：C语言用ax传递，所以放在进程PCB的ax寄存器中

fork()调用功能如下：

寻找一个自由的PCB块，如果没有，创建失败，调用返回－1；

以调用fork()的当前进程为父进程，复制父进程的PCB内容到自由PCB中。

产生一个唯一的ID作为子进程的ID，存入至PCB的相应项中。

为子进程分配新栈区，从父进程的栈区中复制整个栈的内容到子进程的栈区中；

调整子进程的栈段和栈指针，子进程的父亲指针指向父进程。

Pcb\_list[s].fPCB= pcb\_list[CurrentPCBno];

在父进程的调用返回ax中送子进程的ID，子进程调用返回ax送0。

3. wait()的功能描述

父进程如果想等待子进程结束后再处理子进程的后事，需要一个系统调用实现同步。我们模仿UNIX的做法，设置wait()实现这一功能。

相应地，内核的进程应该增加一种阻塞状态，。当进程调用wait()系统调用时，内核将当前进程阻塞，并调用进程调度过程挑选另一个就绪进程接权。

相应调度模块也要修改，禁止将CPU交权给阻塞状态的进程。

4. exit()的功能描述

父进程如果想等待子进程结束后再处理子进程的后事，需要一个系统调用wait()，进程被阻塞。而子进程终止时，调用exit()，向父进程报告这一事件，可以传递一个字节的信息给父进程，并解除父进程的阻塞，并调用进程调度过程挑选另一个就绪进程接权。

**四.程序设计**

本次我们很早就开始了，开始时我们以上次的二状态操作系统为基础，加入五状态，即同时保留测试二状态的用户程序和测试五状态的用户程序。由于这次没有原型参考，只有ppt上的几小段代码，我们凭借自己的想法去添加，在实现的过程中我们遇到了很多问题：二状态和五状态的PCB需要独立开来，时间中断执行的功能需要根据不同情况作不同操作等等的问题，最后由于本身的代码已经很多，而我们添加进五状态之后代码显得很臃肿而且没办法正常运行相应功能。同时由于我们也只是在探索如何去编写五状态进程模型，所以庞大的代码造成了思维混乱，在我们尝试了多次DEBUG修复之后，五状态的操作系统还是没法运行。时间临近期末考试，无奈我们只能放弃之前的基础，只是实现了这次实验的要求。

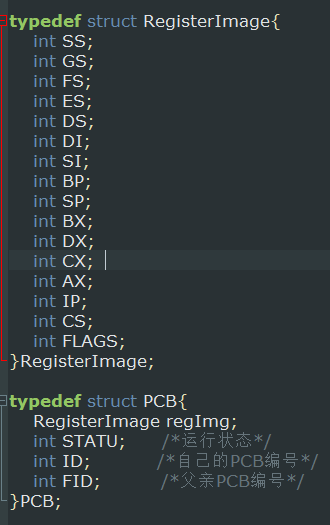
这次实验最后我们只实现了五状态进程模型的功能，即让父子进程协作去数字符串长度，在这里我们相比PPT中只数一个字符串，增加多了一个字符串，可以通过输入来控制数字符串1还是字符串2的长度。流程大概如下：

选择要数的字符串 🡪 父进程创建子进程 🡪 通过时间中断让父子进程不断轮转执行 🡪 父进程一开始先do\_wait，让子进程去数字符串的长度 🡪 子进程数完后执行do\_exit结束自己并把遗言(即字符串的长度告诉父进程)，在结束的同时将父进程唤醒 🡪父进程唤醒后输出从子进程得到的字符串长度

**五.程序关键代码及解释**

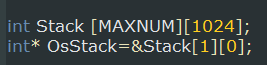
首先来看PCB的部分：

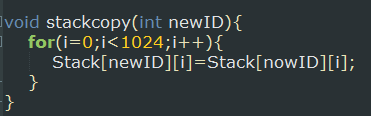
**数据结构：**

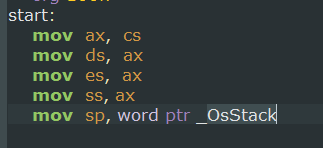


PCB存储的是模拟一个8086CPU所需的所有寄存器信息，相比实验5的二状态模型的数据结构，在寄存器方面没有任何变化，但是在PCB表中要新增两个数据项：ID(自己的PCB编号)，FID（父亲的PCB编号）。增加这两个数据项是为了以编号作为索引找到对应的进程，在子进程唤醒父进程的时候就起到了寻找父进程信息存储位置的作用。

**栈分配：**

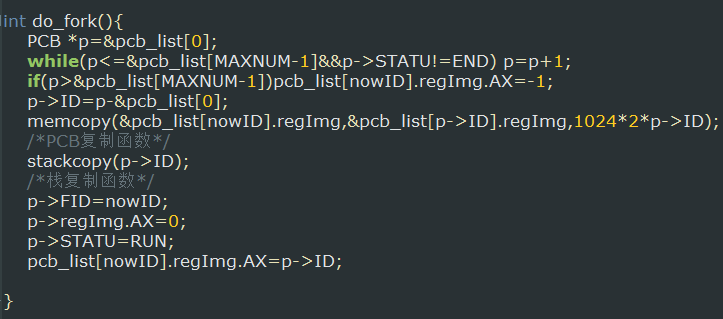






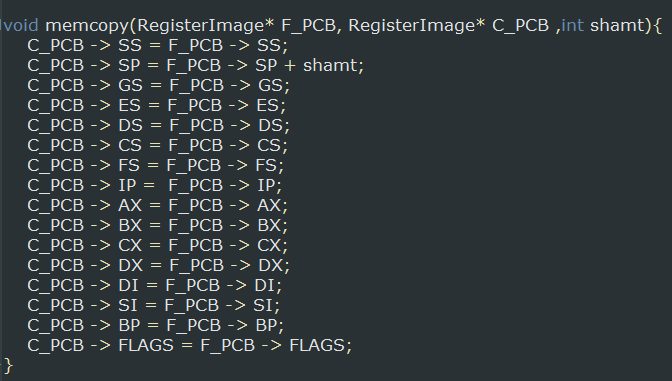
这一段是这次实验困扰了我们最久的地方，我们知道每个进程都有栈信息，那么我们就开个二维数组来保存栈信息，到这里为止还没有问题，问题在于保存栈信息的地方，原本我们与另一组同学讨论得出的保存信息的方法是：在汇编代码中写一个循环将sp不断减一，每减一次再调用C代码将当前sp的信息保存下来，但是这样做就非常的麻烦，还容易出错，后来我们请教了另一组同学，他们告诉我们只要将二维数组的其中一维以指针方式记录，然后拿给内核使用就可以了，这样做的好处就在于复制栈的信息变得简单粗暴。另外需要说的一点就是PCB[i]使用stack[i]来保存相应的栈信息。

**进程创建原语do\_fork():**



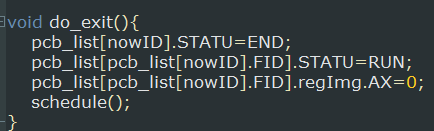
do\_fork()函数先在PCB查找空闲的表项，若没有找到，则爬生失败，将当前进程的ax寄存器的值设为-1；如果找到了，那么将父进程的PCB复制给子进程，并将父进程的堆栈内容复制给子进程的堆栈，然后为子进程分配一个ID。该函数。

**PCB复制函数：**



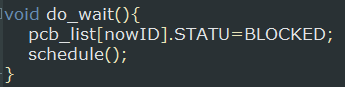
这一段就是复制PCB表的内容，shamt是栈段位置的偏移量，由于父子进程是共享一个栈段，父进程使用的是运行操作系统的那个栈，即stack[1]，子进程是使用stack[p->ID]，那么这里应该相隔了p-ID\*1024个int(我每个stack开了1024个位置)，但是由于int的是两个字节，所以偏移量是p->ID\*2048，在do\_fork中计算好当作参数传进来即可。这个乘2的问题我们一开始也被坑了，也是在一大佬的提点下才知道。

**进程终止原语do\_exit():**



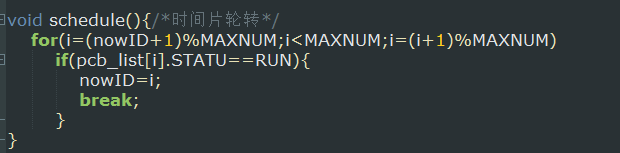
这个函数就是结束自己，唤醒父进程，将父进程的ax寄存器设为0，然后进行时间片轮转到下一个在运行状态的进程。

**进程等待子进程结束do\_wait()原语:**



将自己设置为阻塞状态，进行时间片轮转到下一个在运行状态的进程。

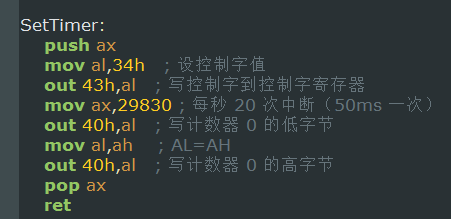
**时间片轮转：**

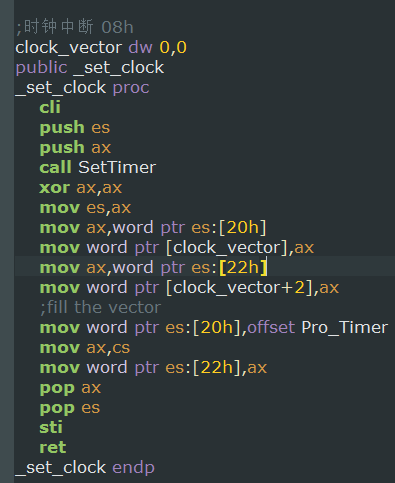
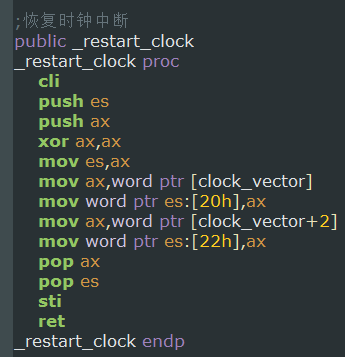


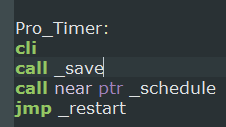
不断在pcb\_list[0]~ pcb\_list[MAXNUM-1]中循环寻找下一个在运行状态的进程。

**时间中断：**

时间中断是把上次实验的时间中断给搬了过来，然后修改一下中断调用的程序Pro\_Timer的内容

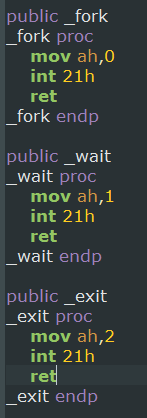
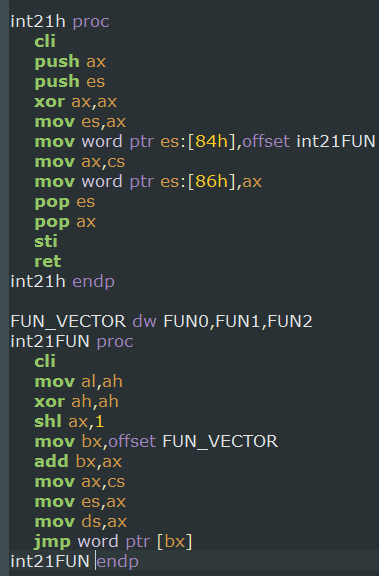
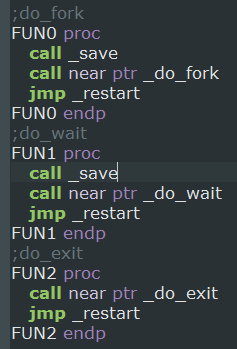




**fork、wait、exit函数：**

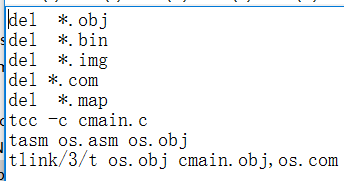
**将fork wait exit 函数封装在21号中断中，然后再c中调用**

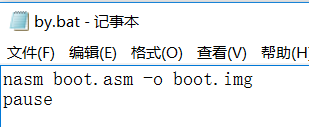
**六.编译.**

使用bat批处理文件进行自动化编译。

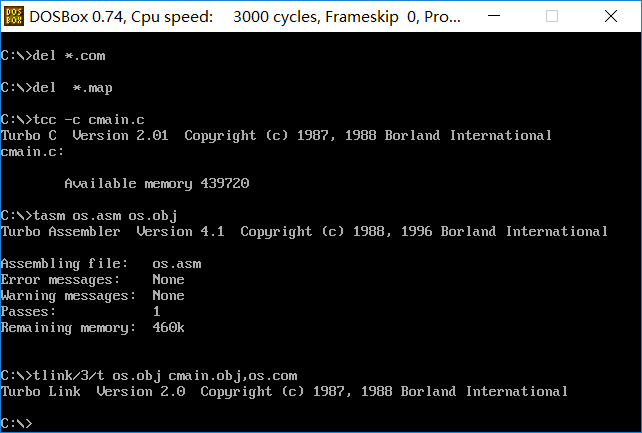
make.bat:



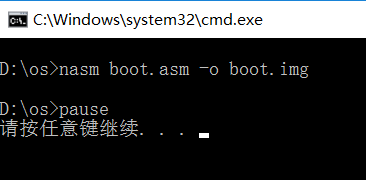
by.bat：



在DOSBOX中运行make：

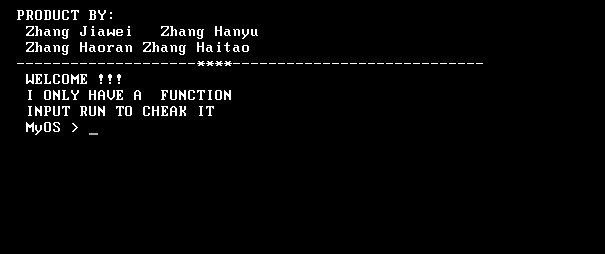


在本机运行by.bat：

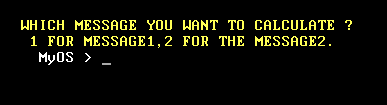


**七.程序运行效果**

**内核界面：**

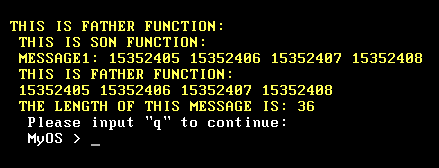


**键入‘RUN’后:**



之后可以键入1可以数字符串1，键入2可以数字符串2

**数字符串1：**



字符串1是4个队员学号相连，加上空格共36个字符。

**数字符串2：**



字符串2是4个队员名字拼音相连，加上空格共51个字符。

**【实验总结】**

本次实验我们很早就开始了尝试，一面理解PPT内容，一面在之前的操作系统上修改添加内容，前前后后我们大概花费了一周时间的实现了初次成果，初次尝试还是一如既往的让人失望：虚拟机上什么都没有显示一片漆黑。哎！这就是做操作系统最心累的地方了，一旦有几个小细节没有处理好，运行结果就会什么都没有。出现这样的问题，原因主要有两方面：一是没有汇编编程的基础，对寄存器之类的使用不熟悉，二是对操作系统的认识不够深入，很多概念都还是停留在知道的层面，真正要实现起来，真要花费很多精力去弥补知识的漏洞。

这次试验是要实现五状态的操作系统，在上课听老师讲了之后其实我们脑海中没多少深刻的印象，大概是因为我们本身还是水平有限吧。但是古语有云，三个臭皮匠，赛过诸葛亮，既然我们自己懵懵懂懂，那就拉多一点人一起研究啊，于是我们开始和另外几组人一起讨论，在讨论的过程中，我们得出了一个一个地方的写法，当然，有的写法比较蠢，在另外一些大神的指点下，我们又将一些比较蠢的地方作了改进，比如本次实验中最重要的部分一直--栈的使用部分。除此之外，我们也遇到了其他一些问题，比如操作系统会在每次调用中断的时候改变当前的FLAG,CS,IP三个寄存器的值，所以我们要将fork，wait，exit封装到系统调用里，在里面先用save保存好相关信息，再去执行相应操作。

栈那部分是这次实验的另一个难点，在保存栈的方法上我们一开始陷入了思维上的死局，总想着在汇编中对sp不断减一然后不断调用c代码去保存，殊不知大神们就使用了一下数组引用就解决了这个难点！学习到这个方法的时候我们觉得自己顿时豁然开朗。至于不同进程栈的切换，就只要在汇编中对sp的值修改一下就可以了。

做操作系统真是一件过程痛苦，结果很快乐的事。