OSlab3 实验报告

王亚楠　　141220107
邮箱　wyn1996@hotmail.com

实验进度

采取分段方式，已完成实验的所有要求，包括进程的组织和调度、Fork、Sleep、Exit。并通过“ping”“pong”检验了 fork、sleep、exit 系统调用的正确性。
使用的 gcc 版本：32 位

试验心得

1. 在 asm_do_irq 中，首先要通过 cli 指令来关中断，否则会在一个中断还没处理完进入下一个中断，这样就产生了中断的嵌套，从而导致中断响应和处理错误。
2. 在 asm_do_irq 中，要将 current->tf 的值赋给 esp，需要通过“movl current, %eax movl (%eax), %esp”来实现，一句话无法实现两级寻址，也就是无法通过一句指令将 current 指向的地址的内容传给 esp。
3. 在定义 TrapFrame 时，由于我在 do_irq.S 中设置中断和异常时压入了错误码，所以需要在 TrapFrame 中加入错误码的变量，否则会发生 13 号保护错。因为在 tf 压栈时会将各个变量的顺序或内容压错，从而导致程序运行出现问题，引发保护错。所以一定要保持程序内个部分的统一，否则会出现一些很奇怪的错误，这个愚蠢的 bug 我将近调了 1 天。
4. 关于初始化新的段描述符表，我是通过扩展全局描述符表来做的，就是每产生一个新的进程就分配两个段描述符表项，每个进程分配 0x400000 的大小。
5. 初始化进程信息：通过 PCB 来管理用户程序，其中采用双向链表 ListHead 来表示进程队列。因为程序有三种状态：就绪态、阻塞态、可用态，所以定义了三个队列：ready、block、free 来分别存放对应的程序。其中 idle 进程执行的就是 main 函数中的一个死循环，不会进行任何操作，也不会被加入到任何队列中去，一旦有其他进程可调度，就会立即调用其他进程，idle 不会占用时间片。在初始化第一个 PCB 时，需要伪造陷阱帧，获得用户进程的入口地址，然后将 PCB 中的 tf 进行初始化，然后将该 PCB 加入就绪队列。
6. 关于进程调度：当时钟中断到来、或者时间片用尽、执行 exit 或 sleep、当前执行 idle 进程时都要进行进程调度，调度就绪队列的第一个进程，若就绪队列为空则调度 idle。
7. fork 系统调用：需要给子进程分配资源，然后将父进程的数据段、用户段拷贝过来，同时设置子进程的陷阱帧以及返回值。
   在实现过程中，发生了程序异常退出的情况，后来通过观察程序运行结果发现，由于之前为用户程序分配的栈空间大小过大，使得分配子进程的时候内核栈和用户栈的大小之和已经超过了 128M，从而导致栈溢出，导致程序崩溃、异常退出。

8. 关于讲义中 fork 的趣味题

```c
#include <stdio.h>
#include <unistd.h>

int main()
{
  int i;
  for (i = 0; i < 2; i++) {
    fork();
    printf("X");
  }
}
```

初步一看，觉得本身程序应该输出 6 个“X”，因为 fork 出了 2 个子进程，然后每个进程会输出 2 个“X”，再加上父进程输出的 2 个“X”，所以一共是 6 个”X“。
但是经过实验发现，程序输出了 8 个”X“，后来查找资料发现，是因为 prinf 会将输出的信息存入到 buffer 中去，所以输出结果中会多 2 个缓存区中的”X“。
但如果改成”X\n“的话则是 6 个，因为“\n”会冲刷 buffer，所以是正常的 fork 结果。

9. sleep 系统调用：就是将当前的进程加入到阻塞态队列，即 block 队列，然后设定一下阻塞时间就行。
10. exit 系统调用：就是将当前进程加入到可用队列，即 free 队列。sleep 和 exit 系统调用实现都较为简单，并没有什么好赘述的。