2152118 史君宝 进程创建和终止 订正

第一题：

代码 1.1。假定父进程的PID是 007，子进程的PID是008。写出程序的输出。

```
#include   <stdio.h>
#include <sys.h>
main( )
{
    int i=10, j =20;
    if( i=fork( ) )
    {
        printf("It is parent process. PID = %d, i = %d\n",getpid( ), i);
         i=wait(&j);
        printf("The finished child process is %d. \n", i);
        printf("The exit status is %d. \n", j);
    }
    else
    {
        printf("It is child process. PID = %d, i = %d\n",getpid( ), i);
        exit(1);
    }
}
```

自己的答案：

(1) 写出程序的输出。

程序的输出应该为：

```
It is parent process. PID = 007, i = 008
It is child process. PID = 008, i =10
```

(2) T0时刻，父进程创建子进程。子进程何时终止？终止后，子进程的PCB何时回收，由谁来回收。

解：

当子进程在被唤醒后执行完毕的时候，会调用exit()函数，终止子进程。

在将子进程的ppid改为1#进程之后，会唤醒1#进程，此时子程序的PCB过程块会由1#进程来回收。

有两种情况：

第一种：父进程先上台运行，然后入睡，子进程上台

It is parent process. PID = 7, i = 8

It is child process. PID =8, i = 0

The finished child process is 8.

The exit status is 256.

第二种：子进程先上台运行

It is child process. PID =8, i = 0

It is parent process. PID = 7, i = 8

The finished child process is 8.

The exit status is 256.

T0时刻，子进程终止。终止后，子进程的PCB何时回收，由谁来回收。

T0时刻，父进程007回收子进程008的PCB。

第二题：

自己的答案：

（2） T0时刻，父进程创建子进程。printf耗时忽略。子进程PCB何时回收，由谁来回收。

解：

当子进程2调用exit()函数来终止，之后会唤醒父进程，在父进程被唤醒之后，在wait系统调用时会返回，并回收子进程2的PCB进程管理块。

在父进程终止后，会将子进程的ppid改为1#进程，之后会唤醒1#进程，此时子程序1的PCB过程块会由1#进程来回收。

参考答案：

（2） T0时刻，父进程创建子进程。printf耗时忽略。子进程的PCB何时回收，由谁来回收。

子进程009的PCB，父进程007回收。T0时刻（之后的一小会）回收。

子进程008的PCB，1#进程回收。T0+100s回收。

（父进程007终止时，将未终止子进程的ppid改成1，子进程008终止的时候，1#进程回收其PCB）。

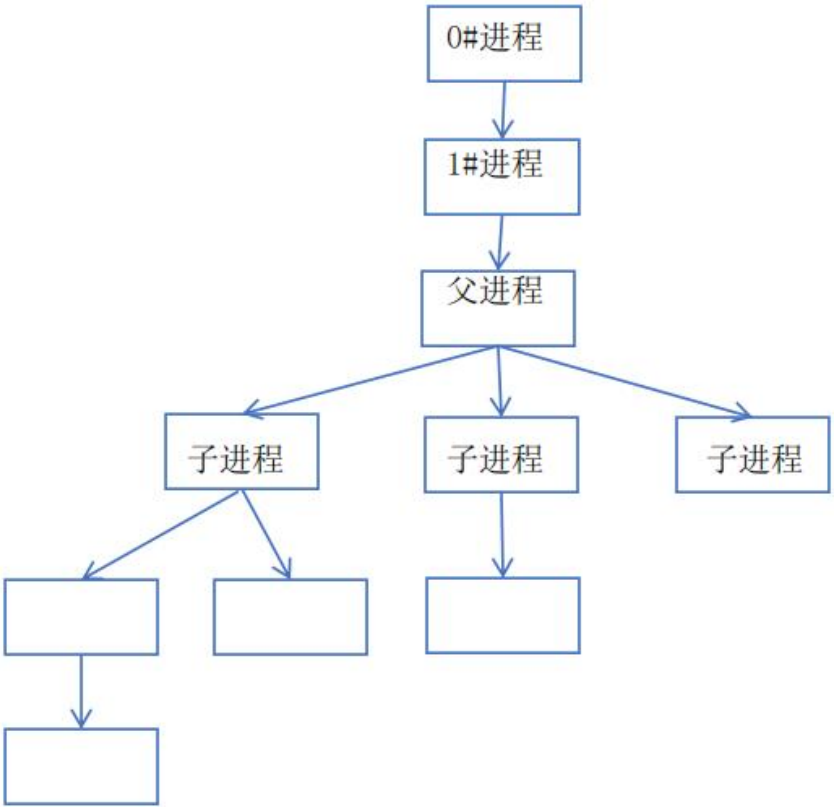
第三题

代码1.5 执行这个程序，系统需要使用几个进程？画与这个应用程序执行相关的进程树。

```
L1: #include <stdio.h>
L2: void main(void)
L3: {     int i;
L4:       printf ("%d   %d \n",   getpid ( ),   getppid ( ) );
L5:       for (i = 0; i < 3; ++i)
L6:             if ( fork( ) == 0 )
L7:                   printf ("%d   %d \n",   getpid ( ),   getppid ( ) );
L8: }
```

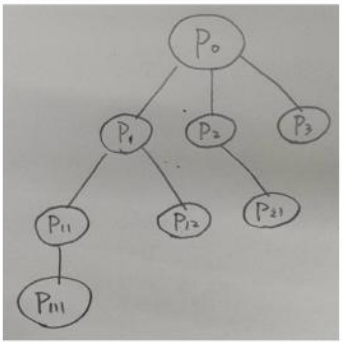
自己的答案：

解：我们可以仿照二叉树来画进程执行相关的进程树：



参考答案：

答：执行这个程序，系统需要使用8个进程。以Unix V6++系统为例，相关进程树如图所示：



第一个进程p0，从main函数入口开始执行，for循环3次，创建3个子进程p1,p2,p3。子进程复制父进程图像，继承 i 变量的值。

p1，i变量初值是0，加1变1。p2，i变量初值是1，加1变2。p3，i变量初值是2，加1变3。只要i变量值小于3，子进程还会创建子进程。所以，p1会创建2个子进程p11,p12。P2会创建1个子进程p21。其中，p11还会再创建1个子进程p111。

为表达清晰，以进程名代pid号，程序输出如下：

（1）如果子进程输出前，父进程未终止

p0　1(shell进程的pid号)

p1　p0

p2　p0

p3　p0

p11　p1

p12　p1

p21　p2

p111　p11

（2）如果子进程输出前，父进程已终止。输出的ppid#会是1，不准了。

注，上面的8行输出，除第一行一定先输出外，其余各行，输出顺序是不确定的，依赖于哪个进程先执行。另外，本例子进程创建不会导致进程入睡，所有进程会接连创建自己的所有子进程，之后终止。所以，子进程输出的ppid，很多是1。

继续思考

- 怎样防止子进程输出ppid=1？
- 将程序输出重定向至磁盘文件，输出会变为20行。解释这个现象。

  提示：屏幕输出是行缓存的，碰到回车就输出，写进tty输出缓存。磁盘文件是块缓存的（4096字节），不写满不执行write系统调用，数据还在进程的用户空间。