**一、fork入门知识**

     一个进程，包括代码、数据和分配给进程的资源。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，也就是两个进程可以做完全相同的事，但如果初始参数或者传入的变量不同，两个进程也可以做不同的事。  
    一个进程调用fork（）函数后，系统先给新的进程分配资源，例如存储数据和代码的空间。然后把原来的进程的所有值都复制到新的新进程中，只有少数值与原来的进程的值不同。相当于克隆了一个自己。

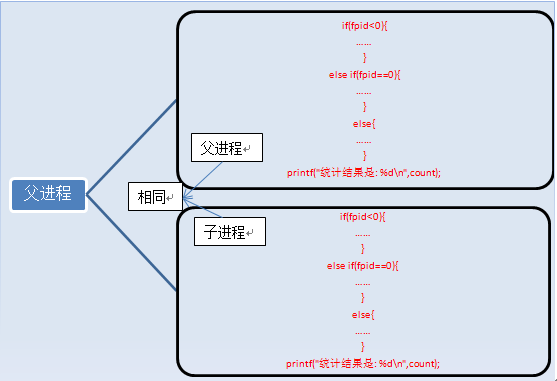
     我们来看一个例子：

1. /\*
2. \*  fork\_test.c
3. \*  version 1
4. \*  Created on: 2010-5-29
5. \*      Author: wangth
6. \*/
7. #include <unistd.h>
8. #include <stdio.h>
9. **int** main ()
10. {
11. pid\_t fpid; //fpid表示fork函数返回的值
12. **int** count=0;
13. fpid=fork();
14. **if** (fpid < 0)
15. printf("error in fork!");
16. **else** **if** (fpid == 0) {
17. printf("i am the child process, my process id is %d/n",getpid());
18. printf("我是爹的儿子/n");//对某些人来说中文看着更直白。
19. count++;
20. }
21. **else** {
22. printf("i am the parent process, my process id is %d/n",getpid());
23. printf("我是孩子他爹/n");
24. count++;
25. }
26. printf("统计结果是: %d/n",count);
27. **return** 0;
28. }

     运行结果是：  
    i am the child process, my process id is 5574  
    我是爹的儿子  
    统计结果是: 1  
    i am the parent process, my process id is 5573  
    我是孩子他爹  
    统计结果是: 1  
    在语句fpid=fork()之前，只有一个进程在执行这段代码，但在这条语句之后，就变成两个进程在执行了，这两个进程的几乎完全相同，将要执行的下一条语句都是if(fpid<0)……  
    为什么两个进程的fpid不同呢，这与fork函数的特性有关。fork调用的一个奇妙之处就是它仅仅被调用一次，却能够返回两次，它可能有三种不同的返回值：  
    1）在父进程中，fork返回新创建子进程的进程ID；  
    2）在子进程中，fork返回0；  
    3）如果出现错误，fork返回一个负值；

    在fork函数执行完毕后，如果创建新进程成功，则出现两个进程，一个是子进程，一个是父进程。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返回的值来判断当前进程是子进程还是父进程。

    引用一位网友的话来解释fpid的值为什么在父子进程中不同。“其实就相当于链表，进程形成了链表，父进程的fpid(p 意味point)指向子进程的进程id, 因为子进程没有子进程，所以其fpid为0.  
    fork出错可能有两种原因：  
    1）当前的进程数已经达到了系统规定的上限，这时errno的值被设置为EAGAIN。  
    2）系统内存不足，这时errno的值被设置为ENOMEM。  
    创建新进程成功后，系统中出现两个基本完全相同的进程，这两个进程执行没有固定的先后顺序，哪个进程先执行要看系统的进程调度策略。  
    每个进程都有一个独特（互不相同）的进程标识符（process ID），可以通过getpid（）函数获得，还有一个记录父进程pid的变量，可以通过getppid（）函数获得变量的值。  
    fork执行完毕后，出现两个进程，



    有人说两个进程的内容完全一样啊，怎么打印的结果不一样啊，那是因为判断条件的原因，上面列举的只是进程的代码和指令，还有变量啊。  
    执行完fork后，进程1的变量为count=0，fpid！=0（父进程）。进程2的变量为count=0，fpid=0（子进程），这两个进程的变量都是独立的，存在不同的地址中，不是共用的，这点要注意。可以说，我们就是通过fpid来识别和操作父子进程的。  
    还有人可能疑惑为什么不是从#include处开始复制代码的，这是因为fork是把进程当前的情况拷贝一份，执行fork时，进程已经执行完了int count=0;fork只拷贝下一个要执行的代码到新的进程。