fork（） 是多进程 不同于多线程<http://blog.csdn.net/cywosp/article/details/27316803>

一、fork()函数

在操作系统的基本概念中进程是程序的一次执行，且是拥有资源的最小单位和调度单位（在引入线程的操作系统中，线程是最小的调度单位）。在Linux系统中创建进程有两种方式：一是由操作系统创建，二是由父进程创建进程（通常为子进程）。系统调用函数fork()是创建一个新进程的唯一方式，当然vfork()也可以创建进程，但是实际上其还是调用了fork()函数。fork()函数是Linux系统中一个比较特殊的函数，其一次调用会有两个返回值，下面是fork()函数的声明：

#include <unistd.h>

// On success, The PID of the process is returned in the parent, and 0 is returned in the child. On failure,

// -1 is returned in the parent, no child process is created, and errno is set appropriately.

pid\_t fork (void);

当程序调用fork()函数并返回成功之后，程序就将变成两个进程，调用fork()者为父进程，后来生成者为子进程。这两个进程将执行相同的程序文本，但却各自拥有不同的栈段、数据段以及堆栈拷贝。子进程的栈、数据以及栈段开始时是父进程内存相应各部分的完全拷贝，因此它们互不影响。从性能方面考虑，父进程到子进程的数据拷贝并不是创建时就拷贝了的，而是采用了写时拷贝（copy-on -write）技术来处理。调用fork()之后，父进程与子进程的执行顺序是我们无法确定的（即调度进程使用CPU），意识到这一点极为重要，因为在一些设计不好的程序中会导致资源竞争，从而出现不可预知的问题。

二 线程

与进程类似，线程（thread）是允许应用程序并发执行多个任务的一种机制。一个进程中可以包含多个线程，同一个程序中的所有线程均会独立执行，且共享同一份全局内存区域，其中包括初始化数据段（initialized data），未初始化数据段（uninitialized data），以及堆内存段（heap segment）。在多处理器环境下，多个线程可以同时执行，如果线程数超过了CPU的个数，那么每个线程的执行顺序将是无法确定的，因此对于一些全局共享数据据需要使用同步机制来确保其的正确性。

在系统中，线程也是稀缺资源，一个进程能同时创建多少个线程这取决于地址空间的大小和内核参数，一台机器可以同时并发运行多少个线程也受限于CPU的数目。在进行程序设计时，我们应该精心规划线程的个数，特别是根据机器CPU的数目来设置工作线程的数目，并为关键任务保留足够的计算资源。如果你设计的程序在背地里启动了额外的线程来执行任务，那这也属于资源规划漏算的情况，从而影响关键任务的执行，最终导致无法达到预期的性能。很多程序中都存在全局对象，这些全局对象的初始化工作都是在进入main()函数之前进行的，为了能保证全局对象的安全初始化（按顺序的），因此在程序进入main()函数之前应该避免线程的创建，从而杜绝未知错误的发生。

三、fork()与多线程

推荐在多线程程序中调用fork()的唯一情况是：其后立即调用exec()函数执行另一个程序，彻底隔断子进程与父进程的关系。由新的进程覆盖掉原有的内存，使得子进程中的所有pthreads对象消失。

对于那些必须执行fork()，而其后又无exec()紧随其后的程序来说，pthreads API提供了一种机制：fork()处理函数。利用函数pthread\_atfork()来创建fork()处理函数。pthread\_atfork()声明如下：

#include <pthread.h>

// Upon successful completion, pthread\_atfork() shall return a value of zero; otherwise, an error number shall be returned to indicate the error.

// @prepare 新进程产生之前被调用

// @parent 新进程产生之后在父进程被调用

// @child 新进程产生之后，在子进程被调用

int pthread\_atfork (void (\*prepare) (void), void (\*parent) (void), void (\*child) (void));

该函数的作用就是往进程中注册三个函数，以便在不同的阶段调用，有了这三个参数，我们就可以在对应的函数中加入对应的处理功能。同时需要注意的是，每次调用pthread\_atfork()函数会将prepare添加到一个函数列表中，创建子进程之前会（按与注册次序相反的顺序）自动执行该函数列表中函数。parent与child也会被添加到一个函数列表中，在fork()返回前，分别在父子进程中自动执行（按注册的顺序）。具体事例可参考：<http://blog.chinaunix.net/uid-26885237-id-3210394.html>

四、总结

fork()函数的调用会导致在子进程中除调用线程外的其它线程全都终止执行并消失，因此在多线程的情况下会导致死锁和内存泄露的情况。在进行多线程编程的时候尽量避免fork()的调用，同时在程序在进入main函数之前应避免创建线程，因为这会影响到全局对象的安全初始化。线程不应该被强行终止，因为这样它就没有机会调用清理函数来做相应的操作，同时也就没有机会来释放已被锁住的锁，如果另一线程对未被解锁的锁进行加锁，那么将会立即发生死锁，从而导致程序无法正常运行。