# linux中的进程控制

## 进程控制理论基础

程序是一组可执行的静态指令集，而进程(process)是一个执行中的程序实例。利用分时技术，在Linux[操作系统](http://lib.csdn.net/base/operatingsystem)上同时可以运行多个进程。**分时技术的基本原理是把CPU的运行时间划分成一个个规定长度的时间片，让每个进程在一个时间片内运行。**当进程的时间片用完时系统就利用调度程序切换到另一个进程去运行。因此实际上对于具体单个CPU的机器来说某一个时刻只能运行一个进程。但由于每个进程运行的时间片很短(例如15个系统滴答=150ms)，所以表面看起来好像所有进程在同时运行着。

　　对于Linux0.11内核来讲，系统最多可由64个进程同时存在。除了第一个进程是"手工"建立以外，其余的都是进程使用系统调用fork创建的新进程，被创建的进程成为子进程(Child Process)，创建者，则称为父进程(parent process)。内核程序使用进程标识号(process ID,pid)来标识每个进程。**进程由可执行的指令代码，数据和堆栈区组成**。进程中的代码和数据部分分别对应一个可执行文件中的代码段，数据段。每个进程只能执行自己的代码和访问自己的数据及堆栈区。进程之间相互之间的通信需要通过系统调用来进行。对于只有一个CPU的系统，在某一个时刻只能有一个进程正在运行。**内核通过进程调度程序分时调度各个进程运行**。

### 定义

进程是一个具有一定独立功能的程序的一次运行活动。

### 特点

动态性：进程的实质是程序在[多道程序系统](http://baike.baidu.com/item/%E5%A4%9A%E9%81%93%E7%A8%8B%E5%BA%8F%E7%B3%BB%E7%BB%9F)中的一次执行过程，进程是动态产生，动态消亡的。

并发性：任何进程都可以同其他进程一起并发执行

独立性：进程是一个能独立运行的基本单位，同时也是系统分配资源和调度的独立单位；

[异步性](http://baike.baidu.com/item/%E5%BC%82%E6%AD%A5%E6%80%A7)：由于进程间的相互制约，使进程具有执行的间断性，即进程按各自独立的、不可预知的速度向前推进

### 状态

进程执行时的间断性，决定了进程可能具有多种状态。事实上，运行中的进程可能具有以下三种基本状态。

1）[就绪状态](http://baike.baidu.com/item/%E5%B0%B1%E7%BB%AA%E7%8A%B6%E6%80%81)（Ready）：

进程已获得除处理器外的所需[资源](http://baike.baidu.com/item/%E8%B5%84%E6%BA%90)，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](http://baike.baidu.com/item/%E4%BC%98%E5%85%88%E7%BA%A7)来划分[队列](http://baike.baidu.com/item/%E9%98%9F%E5%88%97)。例如，当一个进程由于时间片用完而进入[就绪状态](http://baike.baidu.com/item/%E5%B0%B1%E7%BB%AA%E7%8A%B6%E6%80%81)时，排入低[优先级队列](http://baike.baidu.com/item/%E4%BC%98%E5%85%88%E7%BA%A7%E9%98%9F%E5%88%97)；当进程由[I/O操作](http://baike.baidu.com/item/I%2FO%E6%93%8D%E4%BD%9C)完成而进入[就绪状态](http://baike.baidu.com/item/%E5%B0%B1%E7%BB%AA%E7%8A%B6%E6%80%81)时，排入高优先级队列。

2）运行状态(Running)：

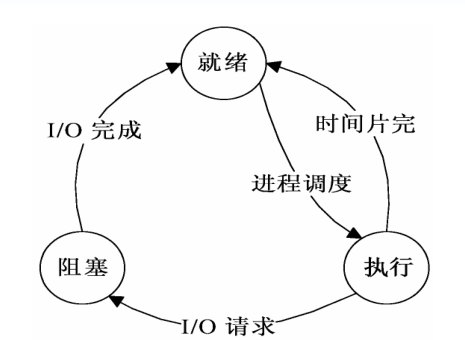
进程占用处理器资源；处于此状态的进程的数目小于等于处理器的数目。在没有其他进程可以执行时(如所有进程都在[阻塞状态](http://baike.baidu.com/item/%E9%98%BB%E5%A1%9E%E7%8A%B6%E6%80%81))，通常会自动执行系统的空闲进程。

3）[阻塞状态](http://baike.baidu.com/item/%E9%98%BB%E5%A1%9E%E7%8A%B6%E6%80%81)(Blocked)：

由于进程等待某种条件（如I/O操作或[进程同步](http://baike.baidu.com/item/%E8%BF%9B%E7%A8%8B%E5%90%8C%E6%AD%A5)），在条件满足之前无法继续执行。该事件发生前即使把处理器资源分配给该进程，也无法运行。

结构特征：进程由程序、数据和[进程控制块](http://baike.baidu.com/item/%E8%BF%9B%E7%A8%8B%E6%8E%A7%E5%88%B6%E5%9D%97)三部分组成。

多个不同的进程可以包含相同的程序：一个程序在不同的数据集里就构成不同的进程，能得到不同的结果；但是执行过程中，程序不能发生改变。



### 进程ID

进程ID(PID):表示进程的唯一数字

父进程的ID(PPID)

启动进程的用户ID(UID)

### 进程互斥

进程互斥是指当有若干进程都要使用某一共享资源时，任何时刻最多允许一个进程使用，其他要使用该资源的进程必须等待，直到占用该资源者释放了该资源为止。

### 临界资源

多道程序系统中存在许多进程，它们共享各种资源，然而有很多资源一次只能供一个进程使用。一次仅允许一个进程使用的资源称为临界资源。许多物理设备都属于临界资源，如输入机、打印机、磁带机等。

共享资源和临界资源

如果一个资源在一段时间内只能为一个进程使用，那么它就是临界资源，比如打印机。如果一个资源在一段时间内内为多个进程分时使用（并发），那么它就是共享资源。接下来说说它们和共享设备，独占设备的关系。共享资源对应的是共享设备，即共享设备就是一个时刻只能为一个进程使用，一段时间却能为多个进程并发使用的设备，比如磁盘，从资源类型角度讲它就是共享资源。临界资源对应的是独占设备，独占设备是那种一段时间只能为一个进程使用，一直要到该设备被释放才允许其他进程使用，比如打印机，对应资源就是临界资源。

临界区

进程中访问临界资源的那段程序代码成为临界区。为实现对临界资源的互斥访问，应保证诸进程互斥的进入各自的临界区。

进程同步

一组并发进程按一定的顺序执行的过程称为进程间的同步。具有同步关系的一组并发进程称为合作进程，合作进程互相发送的信号称为消息或者事件。

### 进程调度

按一定算法，从一组待运行的进程中选出一个来占有CPU运行。

调度方式：抢占式和非抢占式。

调度算法

先来先服务调度算法

短进程优先算法

高优先级优先调度算法

时间片轮转法

死锁

多个进程因竞争资源而形成一种僵局，若无外力作用，这些进程都将永远不能再向前推进。

## 进程控制编程

### 获取ID

#include<sys/types.h>

#include<unistd.h>

pid\_t getpid()

获取本进程ID。

pid\_t getpid(void)

获取父进程ID。

### 进程创建

#include<unistd.h>

pid\_d fork(void)

功能：创建子进程

fork的奇妙之处在于它被调用一次，却返回两次，他可能有三种不同的返回值：

1. 在父进程中，返回新创建的子进程的PID。
2. 在子进程中，返回0.
3. 如果出现错误，返回一个负值。

子进程的数据空间、堆栈空间都会从父进程的到一个拷贝，而不是共享。

pid\_t vfork(void)

fork VS vfork

区别：

1. fork：子进程拷贝父进程的数据段。

vfork：子进程与父进程共享数据段。

1. fork：父、子紧凑型的执行次序不确定

vfork:子进程先运行，父进程后运行。

### exec 函数族

exec启动一个新的进程，产生一个新的PID。

exec启动一个新程序，替换原有的进程，因此进程的PID不会改变。

### 进程等待

pid\_t wait(int \*status)

功能：阻塞该进程，直到其某个子进程退出。

exit(1)时异常退出，exit(0)是正常退出。