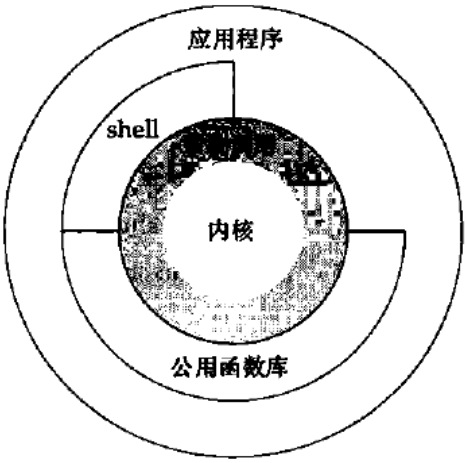
UNIX环境高级编程

UNIX基础知识

体系结构

内核的接口被称为系统调用（图里看不清的地方）。shell是一个特殊的应用程序，为运行其他应用程序提供了一个接口。



登录

用户登录UNIX系统时，先键入用户名，再键入口令。系统在其口令文件（/etc/passwd文件）

输入和输出

文件描述符：
通常是一个小的非负整数，内核用以标识一个特定进程正在访问的文件。当内核打开一个现有文件或创建一个文件时，它都返回一个文件描述符。在读、写文件时，可以使用这个文件描述符。

进程控制

进程标识符

进程ID虽然是唯一的，但是可以重用。大多数UNIX系统采用延迟重用算法，使得赋予新建进程的ID不同于最近终止进程所使用的ID。防止了将新进程误以为是使用同一ID的已终止的先前进程。 ID为0的进程通常是调度进程，常常被称为**交换进程(swapper)**，该进程是内核的一部分，它并不执行任何磁盘上的程序，因此
也被称为**系统进程。**进程ID1通常是init进程，在自举过程结束时由内核调用。此进程负责在自举内核后启动一个UNIX系统。init 通常读取与系统有关的初始化文件，并将系统引导到一个状态(如多用户)。**init进程决不会终止。**它是一个普通的用户进程(与交换进程不同，它不是内核中的系统进程)，但是它以**超级用户特权运行。**

fork函数

子进程中返回0，父进程中返回子进程的ID，出错返回-1。**fork函数被调用一次，但返回两次。**
（1）将子进程ID返回给父进程的理由是:因为一个进程的子进程可以有多个，并且没有一个函数使一个进程可以获得其所有子进程的进程ID；（2）fork 使子进程得到返回值0的理由是：一个进程只会有一个父进程，所以子进程总是可以调用getppid 以获得其父进程的进程ID (进程ID0总是由内核交换进程使

用，所以一个子进程的进程ID不可能为0).

在重定向父进程的标准输出时，子进程的标准输出也被重定向。实际上，fork的一个特性就是父进程的所有打开文件描述符都被复制到子进程中。父子进程的每个相同的打开描述符共享一个文件表项。

vfork函数

vfork用于创建一个新进程，而该新进程的目的是exec一个新程序。
vfork与fork一样都创建一个子进程，但是**它并不将父进程的地址空间完全复制到子进程中**，因为子进程会立即调用exec(或exit),于是也就不会引用该地址空间。vfork和fork之间的另一个区别是: vfork 保证子进程先运行，在它调用exec或exit之后父进程才可能被调度运行，当子进程调用这两个函数中的任意一个时，父进程会恢复运行。(如果在调用这两个函数之前子进程依赖于父进程的进一步动作，则会导致死锁。) vfork已保证在子进程调用exec或exit之前，内核会使父进程处于休眠状态。

exit函数

僵尸进程:
　内核为每个终止子进程保存了一定量的信息，所以当终止进程的父进程调用wait或waitpid时，可以得到这些信息。这些信息至少包括进程ID、该进程的终止状态以及该进程使用的CPU时间总量。内核可以释放终止进程所使用的所有存储区，关闭其所有打开文件。在UNIX术语中，一个已经终止、但是其父进程尚未对其进行善后处理(获取终止子进程的有关信息、释放它仍占用的资源)的进程被称为**僵死进程(zombie)**。

wait和waitpid函数

当一个进程正常或异常终止时，内核就向其父进程发送`SIGCHLD`信号。因为子进程终止是个**异步事件**(这可以在父进程运行的任何时候发生)，所以这种信号也是内核向父进程发的异步通知。父进程可以选择忽略该信号，或者提供一个该信号发生时即被调用执行的函数(信号处理程序)。对于这种信号的系统默认动作是忽略它。

```
#include <sys/wait.h>
pid_t wait(int *statloc);
pid_t waitpid(pid_t pid, int *statloc, int options);
// pid = -1，等待任一子进程
// pid > 0，等待其进程ID与pid相等的子进程
// pid == 0，等待其组ID调用进程组ID的任一子进程。
// pid < -1，等待其组ID等于pid绝对值的任一子进程。
```

如果一个进程有多个子进程，那么只要有一个子进程终止，wait就返回。

竞态条件

当多个进程都企图对共享数据进行某种处理，而最后的结果又取决于进程运行的顺序时，我们认为发生了竞争条件(race condition)。 如果在fork之后的某种逻辑显式或隐式地依赖于在fork之后是父进程先运行还是子进程先运行，那么fork函数就会是竞争条件活跃的滋生地。通常，我们不能预料哪一个进程先运行。即使我们知道哪一个 进程先运行，在该进程开始运行后所发生的事情也依赖于系统负载以及内核的调度算法。

exec函数

exec只是用磁盘上的一个新程序替换了当前进程的正文段、数据段、堆段和栈段。
用fork可以创建新进程，用exec可以初始执行新的程序。exit函数和wait函数处理终止和等待终止。