线程的大致轮廓

之前讲过,linux中只有进程的概念，共享某些资源的进程可以称为线程，进程创建子进程的过程中当共享某些资源如内存地址空间时，这个子进程可称为这个父进程的一个线程，每个线程都有一个线程ID,一组寄存器值，栈，调度优先级，信号屏蔽字，errno变量，其实这些都是子进程都有的，其中的线程ID是在父进程范围内的标识，而进程ID是整个系统范围内的标识

由于线程与父进程的内存地址空间共享，所以线程与父进程的内存区域（程序文本，全局变量，堆，栈）一样

另外线程打开的文件描述符与父进程也是一样的

线程的标识

每个线程都有一个线程ID,线程ID只在他的父进程中有效，线程ID用数据类型pthread\_t表示，是一个结构体，所以比较两个线程ID需要用如下接口

int pthread\_equal(pthread\_t tid1,pthread\_t tid2);

返回值：相等则返回为非0值，否则返回0

线程获取自身的线程ID用如下接口

pthread\_t pthread\_self(void);

返回值：返回线程ID

线程的创建

创建线程用如下接口：

int pthread\_create(pthread\_t \*restrict tidp,//创建线程的线程ID

const pthread\_attr\_t \*rsetrict addr,//定制线程的属性

void \*(\*start\_rtn)(void \*),//线程运行的函数

void \*restrict arg);//传给线程运行函数的参数

返回值：成功返回0，否则返回错误编号

创建的线程与父进程运行顺序并不能保证

线程的终止

如果进程的任意线程调用了exit \_Exit \_exit或者任意线程接收到终止进程的信号则整个进程都会终止

如何退出单个线程，有如下三种情况

1：被同一父进程下的其他线程取消,调用如下接口

int pthread\_cancel(pthread\_t tid);

2：线程运行完函数自动退出或者线程自己调用pthread\_exit退出

void pthread\_exit(void \*rval\_ptr);//rval\_ptr为返回值得指针,如不需要可以设为NULL

其他线程可以调用如下接口来获取pthread\_exit中的返回参数rval\_ptr,pthread\_join会一直阻塞,直到指定的线程调用pthread\_exit,自动退出,或者被其他线程取消

int pthread\_join(phtread\_t thread, void \*\*rval\_ptr);

**可以安排线程终止时调用的函数,这个函数叫线程清除处理程序,可有多个**

**线程同步**

线程同步的方法有三种

1:互斥量

互斥量本质上是一把锁,访问临界资源时,先获取锁,访问完后释放锁,如果获取不到锁,则线程会阻塞直到获取到锁,

互斥量用pthread\_mutex\_t数据类型表示,使用前需要初始化

int pthread\_mutex\_init(pthread\_mutex\_t \*restric mutex,

const pthread\_mutexattr\_t \*restrict attr);

对互斥量加锁

int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex);//会引起阻塞

int pthread\_mutex\_trylock(pthread\_mutext\_t \*mutex);//不会引起阻塞,立即返回

对互斥量解锁

int pthread\_mutex\_unlock(pthread\_mutext\_t \*mutex);

2:条件变量

条件变量用pthread\_cond\_t数据类型表示,使用前需要初始化

int pthread\_cond\_init(pthread\_cond\_t \*restrict cond,

pthread\_condattr\_t \*restrict attr);