C9 线程

线程概述

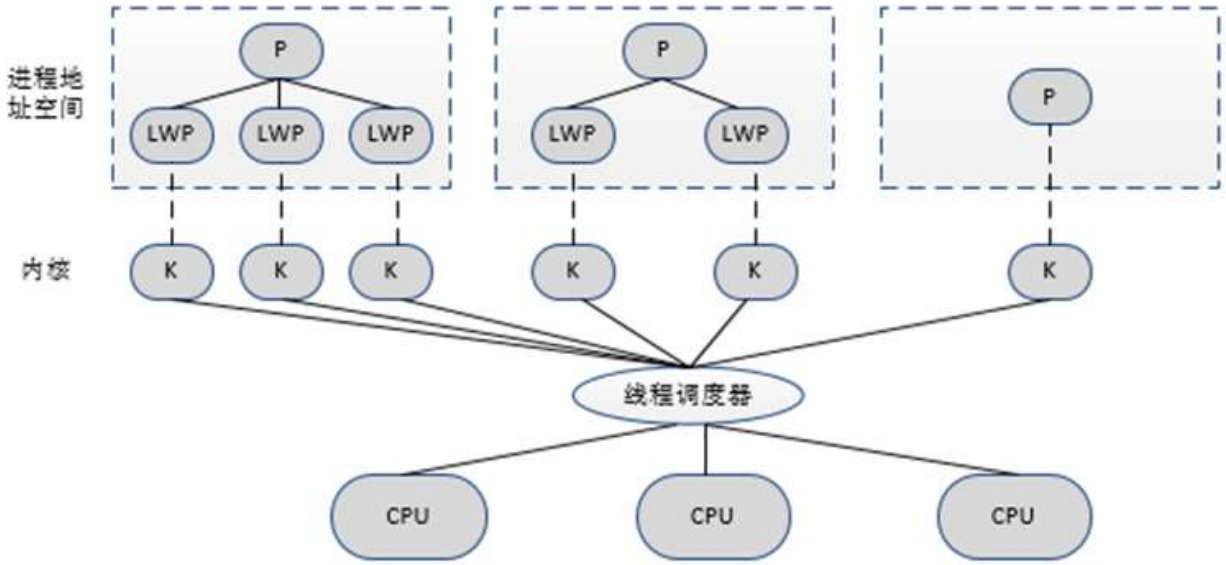
与进程不同，线程（Thread）是系统调度分派的最小单位，与进程相比，线程没有独立的地址空间，多个线程共享一段地址空间，因此线程消耗更少的内存资源，线程间通信也更为方便，有时线程也被称为轻量级进程（Light Weight Process，LWP）。

Linux系统中的线程借助进程机制实现，线程与进程联系密切：

进程可以蜕变成线程，当在一个进程中创建一个线程时，原有的进程就会变成线程，两个线程共用一段地址空间；

线程又被称为轻量级进程，线程的TCB（Thread Control Block，线程控制块）与进程的PCB相同，因此也可以将TCB视为PCB；

对内核而言，线程与进程没有区别，cpu会为每个线程与进程分配时间片，并通过PCB来调度不同的线程和进程。



线程操作

pthread_create()

进程拥有独立的地址空间，当使用fork()函数创建出新进程后，若其中一个进程要对fork()之前的数据进行修改，进程中会依据“写时复制”原则，先复制一份该数据到子进程的地址空间，再修改数据，因此即便是全局变量，在进程间也是不共享的。

但由于线程间共享地址空间，因此在一个线程中对全局区的数据进行修改，其它线程中访问到的也是修改后的数据。

pthread_exit()

return和exit()也有退出功能，但return用于退出函数，exit()用于退出进程。

pthread_cancel()

线程机制中用于终止线程的函数为pthread_cancel()，该函数对应线程机制中的kill()函数，pthread_cancel()函数可向指定线程发送信号CANCEL，使一个线程强行杀死另外一个线程。

pthread_join()

在进程中，父进程退出，子进程仍可继续执行；但在线程中，作为程序入口的主线程退出，属于同一进程中的所有线程都会退出。

为避免主线程提前退出对其它线程造成影响，可以使用pthread_join()函数将主线程挂起。

pthread_detach()

在线程终止后，其它线程调用pthread_join()函数获取该线程的终止状态前，该线程会一直保持终止状态，这种状态类似进程中的僵尸态。为避免处于终止状态的线程占用内存，线程机制中提供了pthread_detach()函数，可在线程被创建后设置线程分离，被分离的线程在执行结束后将会自动释放，不再等待其它线程回收。

线程属性

线程同步

互斥锁

使用互斥锁实现线程同步时，系统会为共享资源添加一个称为互斥锁的标记，防止多个线程在同一时刻访问相同的共用资源。

条件变量

使用互斥锁实现线程同步时，系统会为共享资源添加一个称为互斥锁的标记，防止多个线程在同一时刻访问相同的共用资源。

信号量

使用信号量实现线程同步时，线程在访问共享资源时会根据操作类型执行P/V操作：若有线程申请访问共享资源，系统会执行P操作使共享资源计数减一；若有线程释放共享资源，系统会执行V操作使共享资源计数加一。