

嵌入式技术 EC 2015



嵌入式技术

邢超

嵌入式技术

GNU/Linux 线程

线程

互斥同步

思考

邢超

西北工业大学航天学院



- 线程是包含在进程中的一种实体，线程有自己的运行线索，可以完成一定的任务，可与其他线程共享进程中的共享变量及部分环境、相互之间协同来完成进程所要完成的任务。
- 从调度的角度，线程可以分为用户线程和内核线程。
 - 用户线程：用户线程的实现是通过运行时间系统代码来实现的，线程的切换实际上并不是核心调度器来实现的，而是通过进程内的代码来实现的
 - 内核线程：内核线程切换的实现是通过内核调度器来实现的，内核线程同进程是一样的，都是核心调度器的调度实体。



- 线程和进程相比有以下优点：
 - “节俭”的多任务操作方式
 - 线程间方便的通信机制
 - 提高应用程序响应
 - 使多 CPU 系统更加有效
 - 改善程序结构



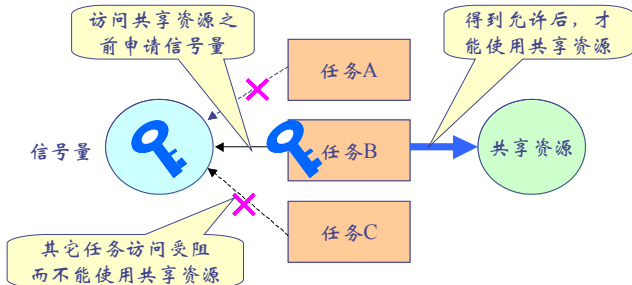
- 多个线程将共享同一个进程虚拟空间。线程共享的环境包括：
 - 进程代码段
 - 进程的公有数据 (有利于实现线程相互之间通讯)
 - 进程打开的文件描述符
 - 信号的处理程序
 - 进程的当前目录
 - 进程用户 ID 与进程组 ID。
- 每个线程都具有：
 - 线程 ID
 - 寄存器组的值
 - 线程的堆栈
 - 错误返回码
 - 线程的信号屏蔽码
 - 线程的优先级



- Linux 系统下的多线程遵循 POSIX 线程接口，称为 pthread
- 编写 Linux 下的多线程程序，需要使用头文件 pthread.h，库文件 libpthread.a
- Linux 下 pthread 通过系统调用 clone() 来实现的



```
#include<stdio.h>
#include<pthread.h>
void *thread(void){
    int i;
    for (i=0;i<3;i++)
        printf("This is a pthread.\n");
}
int main(void){
    pthread_t id;
    int i,ret;
    ret=pthread_create(&id,NULL,thread,NULL);
    if (ret!=0){
        printf("Create pthread error!\n");
        exit();
    }
    for (i=0;i<3;i++) printf( \
        "This is the main process.\n");
    pthread_join(id,NULL);
    return (0);
}
```





- 原子操作 (Atomic Operation): 需要硬件的支持, 架构相关; 通常用于实现资源的引用计数
- 自旋锁 (spinlock):
 - 不会引起调用者睡眠
 - 若自旋锁已经被别的执行单元保持, 调用者就一直循环在那里看是否该自旋锁已释放
- 信号量 (semaphore): 创建时设置初始值, 表示同时可以有几个任务可以访问该信号量保护的共享资源,
- 读写信号量 (rw_semaphore): 可有任意个读者同时拥有一个读写信号量。在内核配置时, 可以通过选项去控制使用哪一种实现:
 - 架构无关: 新的架构不需要重新实现它。性能低, 获得和释放读写信号量的开销大;
 - 架构相关: 性能高, 获取和释放读写信号量的开销小, 但增加新的架构需要重新实现。



- 许多函数是不可重入的，即同时不能运行一个函数的多个拷贝
- 在函数中声明的静态变量常常会带来一些问题，函数的返回值也会有问题
- 共享的变量必须用关键字 `volatile` 来定义
- 为了保护变量，必须使用信号量、互斥等方法来保证对变量的正确使用



嵌入式技术

邢超

线程

互斥同步

思考

- 进程与线程的区别
- 线程的互斥与同步