学习目标

1. 说出守护进程的特点
2. 独立完成守护进程的创建
3. 独立实现多个线程的创建
4. 独立实现线程的退出和资源回收

1-守护进程

1. 守护进程的特点
   - 后台服务进程
   - 独立于控制终端
   - 周期性执行某任务
   - 不受用户登录注销影响
   - 一般采用以d结尾的名字(服务)
2. 进程组
   - 进程的组长?
     - 组里边的第一进程
     - 进程组的ID == 进程组的组长的ID
   - 进程组组长的选则
     - 进程中的第一个进程
   - 进程组ID的设定
     - 进程组的id就是组长的进程ID
3. 会话 - 多个进程组
   - 创建一个会话注意事项:
     - 不能是进程组长
     - 创建会话的进程成为新进程组的组长
     - 有些linux版本需要root权限执行此操作(ubuntu不需要)
     - 创建出的新会话会丢弃原有的控制终端
     - 一般步骤:先fork, 父亲死, 儿子执行创建会话操作(setsid)
   - 获取进程所属的会话ID
     - pid_t getsid(pid_t pid);
   - 创建一个会话
     - pid_t setsid(void);
4. 创建守护进程模型
   - fork子进程, 父进程退出
     - 必须
   - 子进程创建新会话

        - 必须
        - setsid();
    - 改变当前工作目录chdir
        - 插了一个U盘，a.out,在U盘目录中启动a.out
        - a.out启动过程中，U盘拔掉了
        - 不是必须的。
    - 重设文件掩码
        - 子进程会继承父进程的掩码
        - 增加子进程程序操作的灵活性
        - umask(0);
        - 不是必须的
    - 关闭文件描述符

      close(0);
      close(1)
      close(2)
      释放资源
      不是必须的
    - 执行核心工作

      -- 必须的
5. 练习：

   写一个守护进程,每隔2s获取一次系统时间，
   将这个时间写入到磁盘文件.
    - 创建守护进程
    - 需要一个定时器，2s触发一次
        - setttimer
        - sleep
    - 信号捕捉

2 - 线程操作函数

线程相关概念

- 自行安装线程man page，命令：
  sudo apt-get install manpages-posix-dev
- 察看指定线程的LWP号：
  - 线程号和线程ID是有区别的
  - 线程号是给内核看的
  - 查看方式：
    - 找到程序的进程ID
    - ps -Lf pid

相同点：函数调用成功返回0，失败返回 错误号

1. 创建线程 -- pthread_create
   - 函数原型：

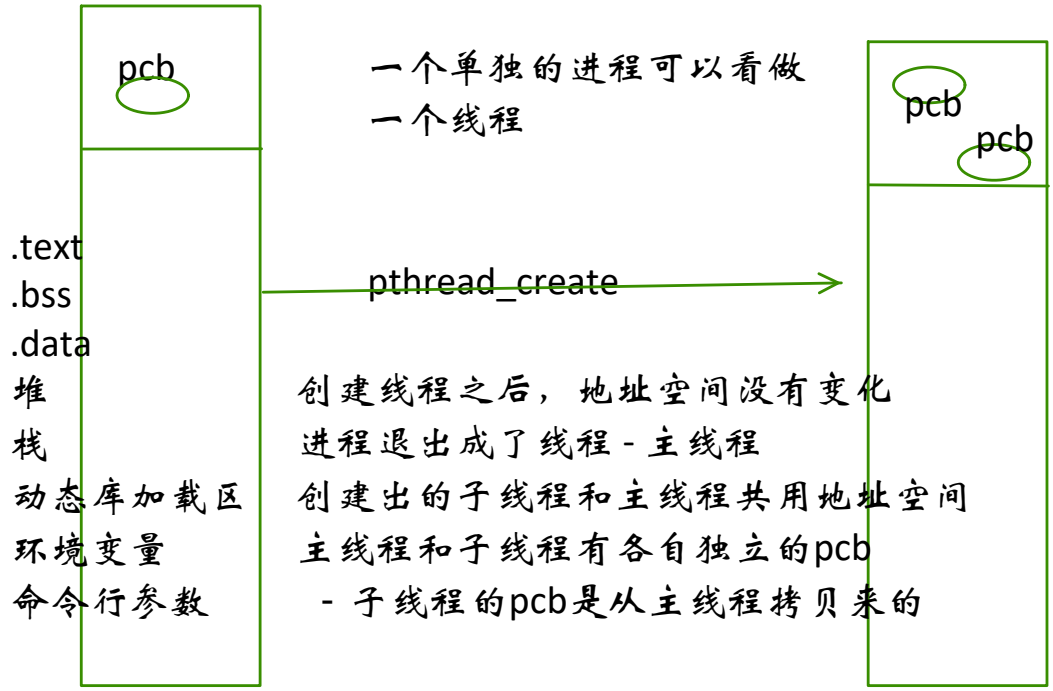
```
// 如果成功0，失败返回错误号
// perror() 不能使用该函数打印错误信息
int pthread_create(
    pthread_t *thread,   // 线程ID=无符号长整形
    const pthread_attr_t *attr, // 线程属性，NULL
    void *(*start_routine) (void *),   // 线程处理函数
    void *arg           // 线程处理函数参数
);
```

     - 参数：
       - thread: 传出参数，线程创建成功之后，

会被设置一个合适的值
        - attr: 默认传NULL
        - start_routine: 子线程的处理函数
        - arg: 回调函数的参数
    - 主线程先退出，子线程会被强制结束
    - 验证线程之间共享全局变量
2. 单个线程退出 -- pthread_exit
    - exit（0）；
    - 函数原型： void pthread_exit(void *retval);
        - retval指针：必须指向全局，堆
3. 阻塞等待线程退出，获取线程退出状态 -- pthread_join
    - 函数原型：
      int pthread_join(pthread_t thread, void **retval);
        - thread：要回收的子线程的线程id
        - retval：读取线程退出的时候携带的状态信息
            - 传出参数
            - void* ptr；
            - pthread_join(pthid, &ptr);
            - 指向的内存和pthread_exit参数指向同一块内存地址
4. 线程分离 -- pthread_detach
    - 函数原型： int pthread_detach(pthread_t thread);
    - 调用该函数之后不需要pthread_join
    - 子线程会自动回收自己的pcb

5. 杀死(取消)线程 -- pthread_cancel
    - 函数原型：int pthread_cancel(pthread_t thread);
    - 使用注意事项：
        - 在要杀死的子线程对应的处理的函数的内部，
          必须做过一次系统调用.
            - pthread_testcancel();
        - write read printf
        - int a；
        - a = 2；
        - int b = a+3；
6. 比较两个线程ID是否相等(预留函数) -- pthread_equal
    - 函数原型：
      int pthread_equal(pthread_t t1, pthread_t t2);

线程



主线程和子线程：

共享：

.text
.bss
.data
堆
动态库加载区
环境变量
命令行参数

- 通信：全局变量，堆

不共享：

- 不共享
- 一个有5个线程
- 栈区被平均分为5份

在linux下：

- 线程就是进程 - 轻量级进程
- 对于内核来说，线程就是进程

多线程和多进程区别：

多进程：

- 始终共享的资源：
  - 代码
  - 文件描述符
  - 内存映射区 -- mmap
- 线程共享：
  - 堆
  - 全局变量
- 线程节省资源

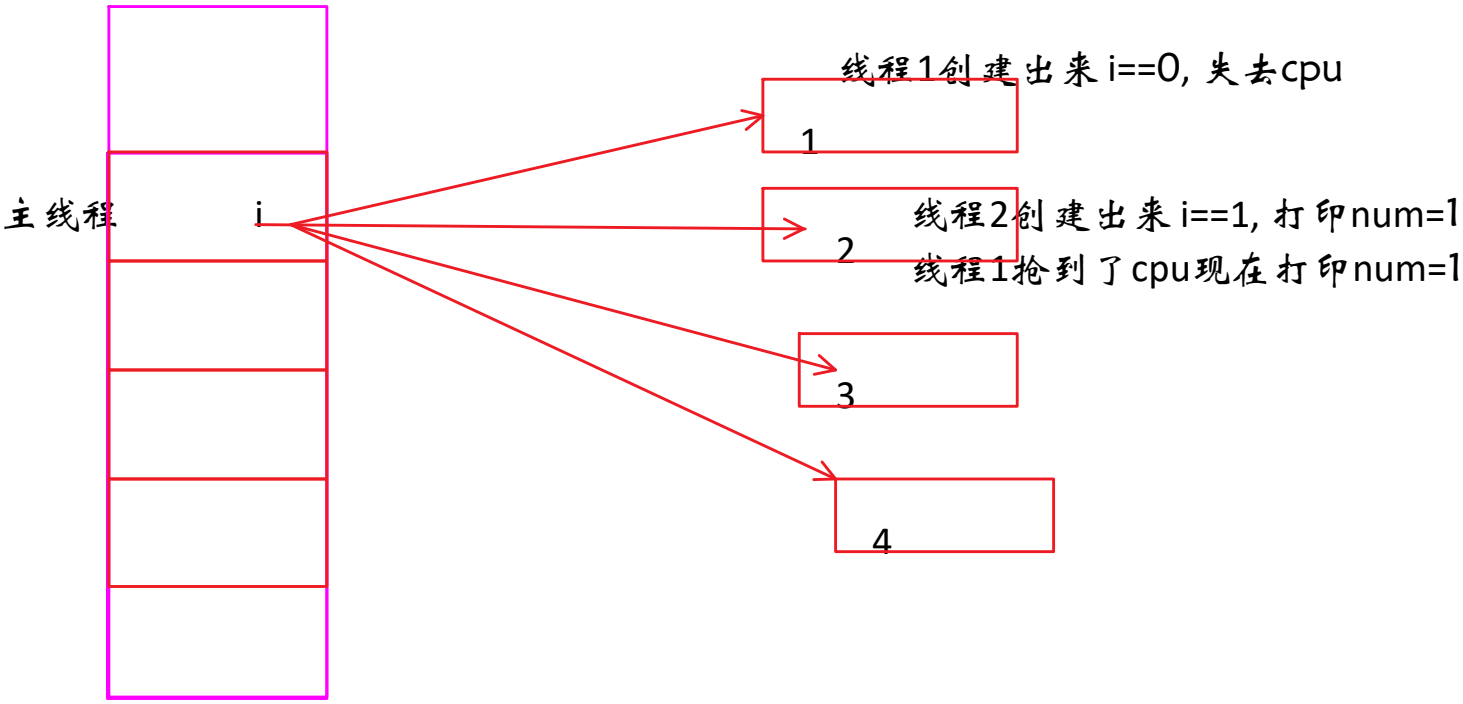
3-线程属性

通过属性设置线程的分离

1. 线程属性类型：pthread_attr_t attr;
2. 线程属性操作函数：
   - 对线程属性变量的初始化
     - int pthread_attr_init (pthread_attr_t* attr);
   - 设置线程分离属性
     - int pthread_attr_setdetachstate (
       pthread_attr_t* attr,
       int detachstate
       );
     - 参数：
       - attr: 线程属性
       - detachstate:
         - PTHREAD_CREATE_DETACHED(分离)
         - PTHREAD_CREATE_JOINABLE（非分离）
   - 释放线程资源函数
     - int pthread_attr_destroy(pthread_attr_t *attr);

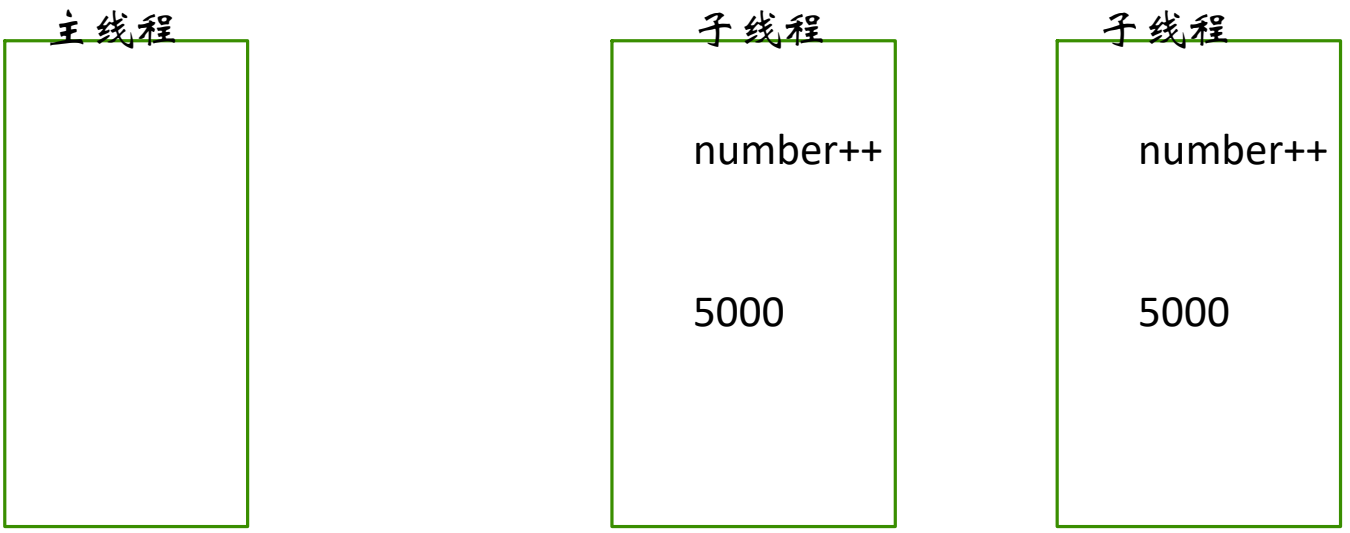
2017年1月6日 16:20

1. 传参数时候，传递是一个地址，地址中的值是变化的



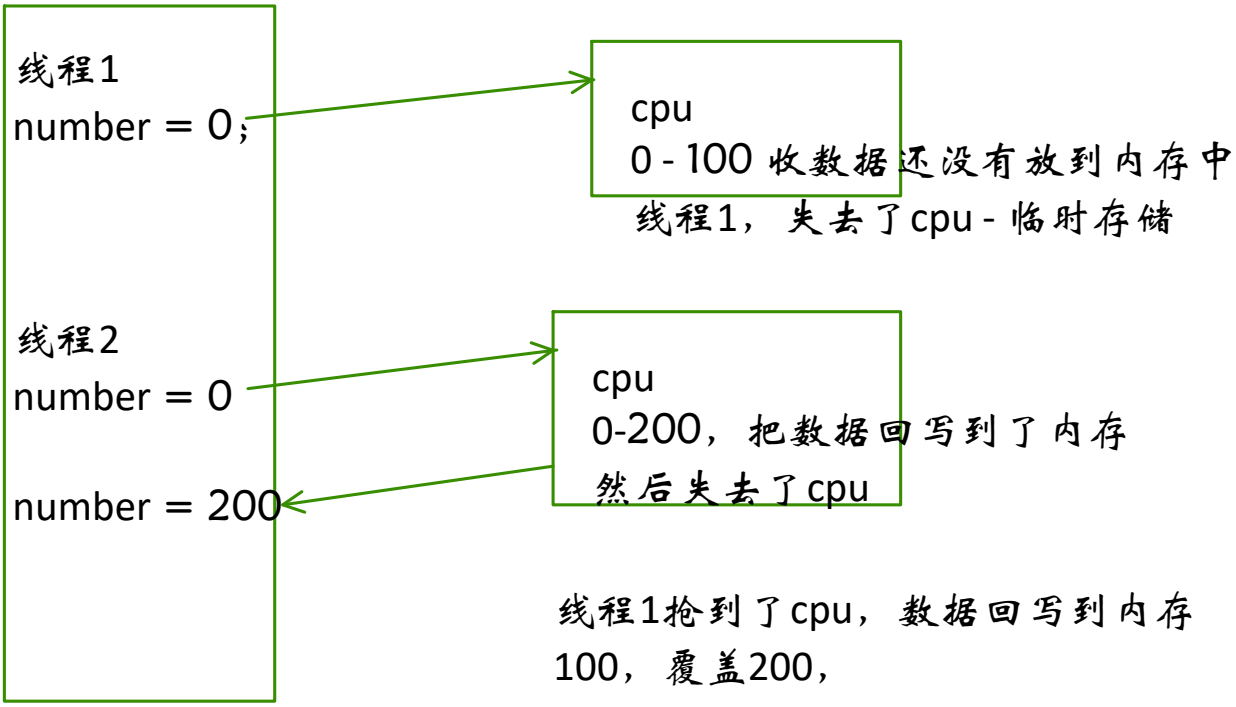
2017年3月25日 16:07

int number； - 全局



数数 - cpu
- 寄存器，存储少量数据
- 时时的与物理内存做数据交换

- 取值的时候：
- 通过内存地址

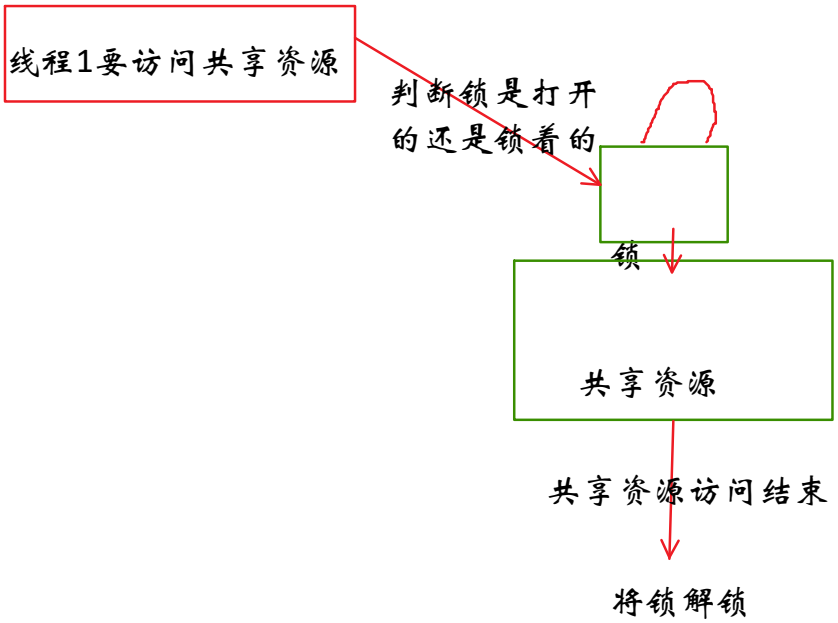


数据混乱原因：
- 操作了共享资源
- cpu调度问题

解决：
- 线程同步
- 什么叫同步：
  - 协同步调，按照先后顺序执行操作

线程同步思想

2017年3月25日 17:00



如果锁是锁着的：
- 线程阻塞，阻塞在这把锁上

如果锁是打开的：
- 线程访问共享资源
- 会将这把锁上锁

本来多线程访问共享资源的时候，可以并行
通过加锁机制：
并行 -> 串行