UC Day17

预习课

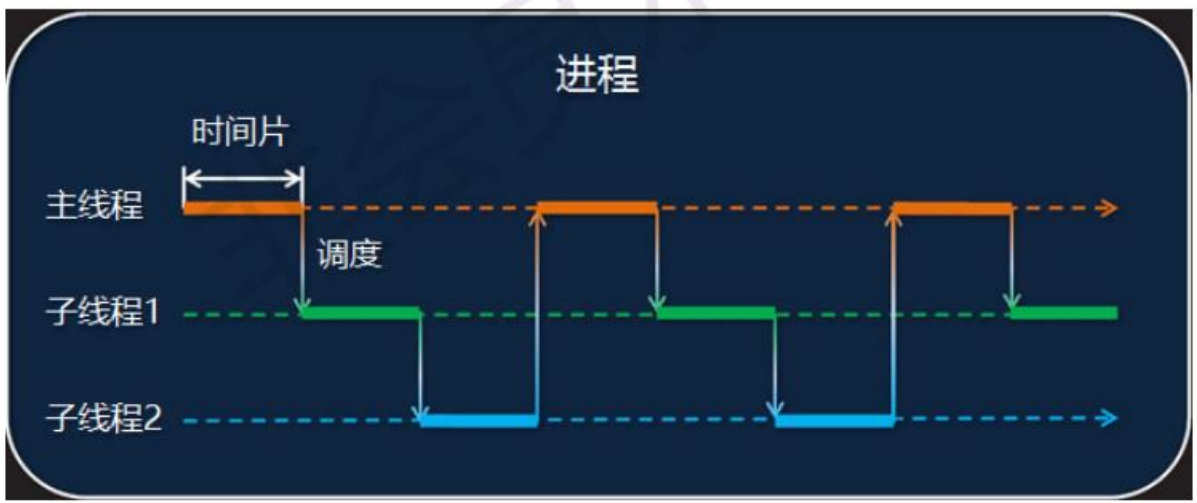
预习内容

线程的概念

线程的概念

线程的概念

- 线程就是程序的执行路线，即进程内部的控制序列，或者说是进程的子任务
- 一个进程可以同时拥有多个线程，即同时被系统调度的多条执行路线，但至少要有一个主线程



线程的概念

知识讲解

- 一个进程的所有线程都共享进程的代码区、数据区、BSS区、堆区、命令行参数和环境就变量，唯有栈区是一个线程一个
- 一个进程的所有线程共享系统内核中与进程有关的部分资源，如文件描述符表、信号处理函数、工作目录、各种ID等
- 一个进程的每个线程都有一个独立的ID、独立的寄存器上下文、独立的调度策略和优先级、独立的信号掩码、独立的errno以及线程私有数据

线程的概念

知识讲解

- 系统内核中专门负责线程调度的处理单元被称为调度器。调度器将所有处于就绪状态(没有阻塞于任何系统调用上)的线程排成一个队列，即所谓就绪队列
- 调度器从就绪队列中获取队首线程，为其分配一个时间片，并令处理机执行该线程，过了一段时间
- 该线程的时间片耗尽，调度器立即中止该线程，并将其排到就绪队列的尾端，接着从队首获取下一个线程

线程的概念

知识讲解

- 系统内核中专门负责线程调度的处理单元被称为调度器。调度器将所有处于就绪状态(没有阻塞于任何系统调用上)的线程排成一个队列，即所谓就绪队列
- 调度器从就绪队列中获取队首线程，为其分配一个时间片，并令处理机执行该线程，过了一段时间
- 该线程的时间片耗尽，调度器立即中止该线程，并将其排到就绪队列的尾端，接着从队首获取下一个线程

线程的概念

知识讲解

- 系统内核中专门负责线程调度的处理单元被称为调度器。调度器将所有处于就绪状态(没有阻塞于任何系统调用上)的线程排成一个队列，即所谓就绪队列
- 在低优先级线程执行期间，有高优先级线程就绪，后者会抢占前者的时间片
- 若就绪队列为空，则系统内核进入空闲状态，直至其非空

线程的概念

知识讲解

- 调度器分配给每个线程的时间片长短，对系统的行为和性能影响很大。如果时间片过长，线程必须等待很长时间才能重新获得处理机，这就降低了整个系统的并行性，用户会感觉到明显的响应延迟。如果时间片过短，大量时间会浪费在线程切换上，同时降低了虚拟内存的存储命中率，线程的时间局部性无法得到保证
- 某些Unix系统倾向于为线程分配较长的时间片，希望通过扩大系统吞吐率来改善其整体表现；而另一些Unix系统则更倾向于为线程分配较短的时间片，以提升系统的交互性。Linux系统根据线程在不同时间的具体表现，为其动态分配时间片，在吞吐率和交互性之间寻求最佳平衡点

线程的概念

- 线程是进程的一个实体，可作为系统独立调度和分派的基本单位
- 线程有不同的状态，系统提供了多种线程控制原语，如创建、终止、取消等等
- 线程可以使用的大部分资源都是隶属于进程的，即使是在特定线程中动态分配的资源，也同样为进程所拥有
- 一个进程中可以有多个线程并发地运行。它们可以执行相同的代码，也可以执行不同的代码

知识讲解

线程的概念

知识讲解

- 同一个进程的多个线程都在同一个地址空间内活动，因此相较于进程，线程的系统开销小，任务切换快
- 进程空间内的代码和数据对于该进程的每个线程而言都是共享的。因此同一个进程的不同线程之间不存在通信问题，当然也就不需要类似IPC的通信机制
- 线程之间虽然不存在通信问题但是存在冲突问题。同样是因为数据共享，当一个进程的多个线程“同时”访问一份数据时，线程间的冲突可能造成逻辑甚至系统错误

直播课见

UC

C/C++教学体系

目录

POSIX线程

POSIX线程

知识讲解

- #include <pthread.h>
- int pthread_create(pthread_t* tid, pthread_attr_t const* attr,
  void* (*start_routine)(void*), void* arg);
  - 功能：创建新线程
  - 参数：tid：输出线程ID。pthread_t即unsigned long int
    attr：线程属性，NULL表示缺省属性
    start_routine：线程过程函数指针，所指向的函数将在被创建的线程中执行
    arg：传递给线程过程函数的参数
  - 返回值：成功返回0,失败返回错误码！

POSIX线程

知识讲解

- 线程过程函数
  - void* thread_proc(void* arg) { ... }
  - 定义线程过程函数，该函数会在所创建的线程中被执行，代表了线程的任务。
  - 而main函数其实就是主线程的线程过程函数。
  - main函数一旦返回，主线程即告结束。主线程一旦结束，进程即告结束。进程一旦结束，其所有的子线程统统结束

POSIX线程

知识讲解

- 补充说明：
  - A.pthread_create函数本身并不调用线程过程函数，而是在系统内核中开启独立的线程，并立即返回，在该线程中执行线程过程函数。
  - B.pthread_create函数返回时，所指定的线程过程函数可能尚未执行，也可能正在执行，甚至可能已经执行完毕。
  - C.主线程和通过pthread_create函数创建的多个子线程，在时间上"同时"运行，如果不附加任何同步条件，则它们每一个执行步骤的先后顺序是完全无法确定的，这就叫做自由并发。

POSIX线程

知识讲解

- 传递给线程过程函数的参数是一个泛型指针void*，它可以指向任何类型的数据：基本类型的变量、结构体类型的变量或者数组类型的变量等等，但必须保证在线程过程函数执行期间，该指针所指向的内存空间持久有效，直到线程过程函数不再使用它为止。
- 调用pthread_create函数的代码在用户空间，线程过程函数的代码也在用户空间，但偏偏创建线程的动作由系统内核完成。因此传递给线程过程函数的参数也不得不经由系统内核传递给线程过程函数。
  pthread_create函数的arg参数负责将线程过程函数的参数带入系统内核。

汇合线程

汇合线程

知识讲解

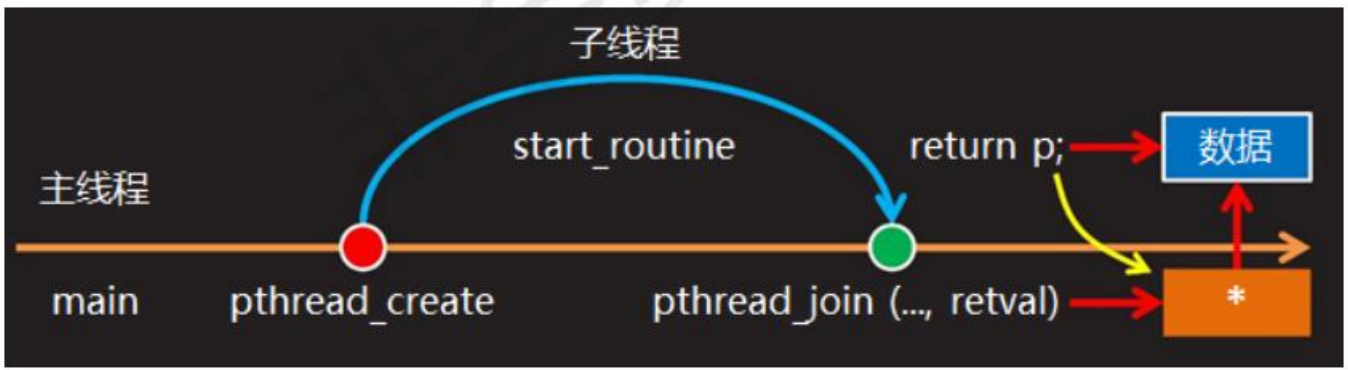
- #include <pthread.h>
- int pthread_join(pthread_t tid, void** retval);
  - 功能：等待子线程终止，即其线程过程函数返回，并与之会合，同时回收该线程的资源
  - 参数：tid：线程ID。

    retval：输出线程过程函数的返回值
  - 返回值：成功返回0，失败返回错误码。

汇合线程

知识讲解

- 在父线程中调用pthread_join函数等待子线程的终止，并回收其资源。如果调用该函数时子线程已经终止，该函数会立即返回，如果调用该函数时子线程尚在运行中，该函数会阻塞，直至子线程终止。该函数在返回成功的同时通过其retval参数向调用者输出子线程线程过程函数的返回值。



分离线程

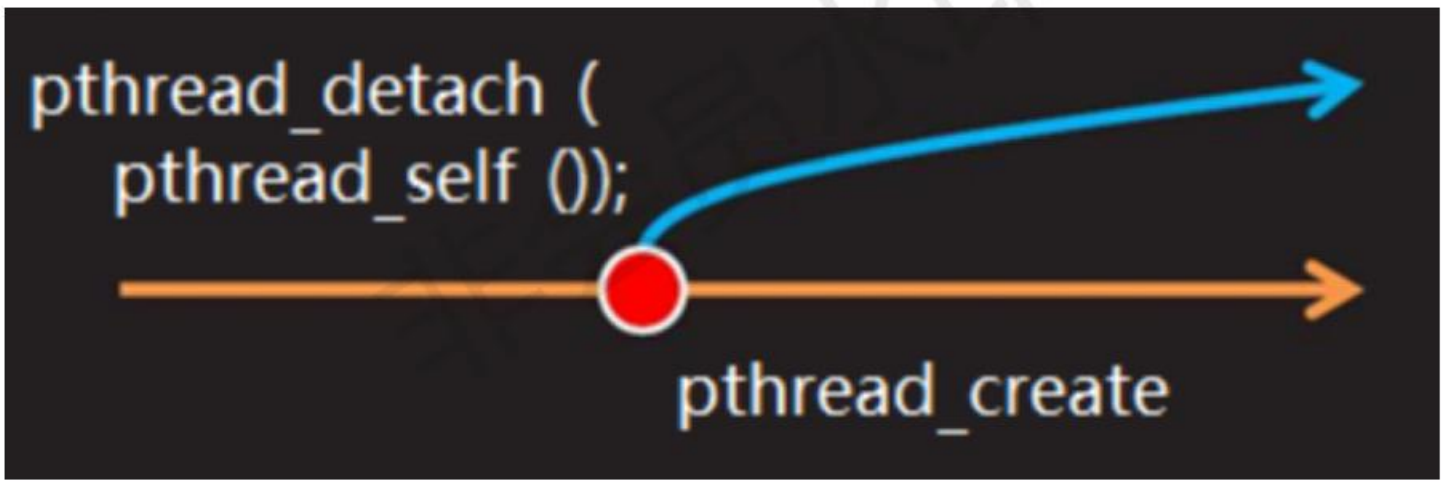
分离线程

知识讲解

- 什么是分离线程
  - 一个线程在默认情况下都是可汇合线程，这样的线程在终止以后其资源会被保留，其中含有线程过程函数的返回值及线程ID等信息，父线程可以通过pthread_join函数获得线程过程函数的返回值，同时释放这些资源。如果在创建线程以后，对其调用pthread_detach函数并返回成功，该线程即成为分离线程，这样的线程终止以后其资源会被系统自动回收，不需要也无法通过pthread_join函数汇合它

分离线程

- 什么是分离线程



谢谢

UC Day17

复习课

线程的属性

线程的属性

- pthread_create()函数的第二个参数为pthread_attr_t类型，代表线程的属性，我们习惯性赋值为NULL，表示默认属性
- 通过对第二个参数的设置，可以在创建线程时，即为所创建的线程执行不同的属性
- 那么线程都有哪些属性？线程的属性又该如何设置呢？

知识讲解

线程的属性

- 常用的线程属性包括
  - __detachstate 分离状态
  - __schedpolicy 线程调度算法
  - __schedparam 线程的优先级
  - __scope 争夺CPU资源范围
  - __stacksize 线程栈大小
  - __guardsize 警戒缓冲区大小

知识讲解

线程的属性

- 我们以分离属性为例，演示线程属性的设置
  - pthread_attr_t myAttr；//表示线程属性的变量
  - pthread_attr_init(&myAttr)；//初始化线程属性
  - //设置myAttr的__detachstate属性值为PTHREAD_CREATE_DETACHED
  - pthread_attr_setdetachstate(&myAttr, PTHREAD_CREATE_DETACHED);
  - //以myAttr创建新的线程
  - pthread_create(&tid,&myAttr,pthread_fun,NULL);

知识讲解

下节课见