CS61第九次课程记录

傅海平

Institute Of Computing Technology,

Chinese Academy Of Sciences

haipingf@gmail.com

December 3, 2011

Contents

1 Topics

线程

2 progress

晚上7:30点开始，7:30 - 10:00 学习 lec17-threads.pdf 课程讲义，然后10:00开始讨论学习过程中遇到的问题。

3 learning details

3.1 course sketch

3.1.1 Threads: Allowing a single program to do multiple things concurrently.

- 并行编程
  - 许多程序都需要并行执行：
  - 浏览器
  - 服务器
  - 科学计算
- 利用进程可以达到程序的并行执行
- 但是，进程不太高效
  - 每个进程有自己的也表，文件表，sockets...
  - 所以进程切换时代价较大，1.7KB的 *task_struct*.
  - 进程的创建代价大。
- 进程并不直接共享内存

  - 并行程序需要对相同的内存区域进行操作。
  - 许多 OS 提供了进程间通信的手段，如共享内存，$shmget(), shmat()$

- 更好的办法
  - 任务间共享了什么?
    - 相同的代码
    - 相同的数据
    - 优先级
    - 某些系统资源：文件，sockets...
  - 每个任务间哪些东西是私有的?
    - 执行状态：CPU 寄存器，栈，和程序计数器PC
  - 线程
- 线程和进程
  - 线程与进程的相同点和区别
  - 每个线程有自己的栈，栈指针，PC，和一些 CPU 寄存器,但共享相同 的地址空间和系统资源，由于线程共享内存，所以方便线程间通信

3.1.2 Implementing

- 线程控制块TCB vs. 进程控制块PCB
- 每个线程控制块内部都有指针指向相关的进程控制块。
- TCB包含了与线程相关的信息，处理器状态和指向PCB的指针
- PCB 包含了进程的相关信息，如地址空间，系统资源，但是没有处理器状态
- Linux：TCB(thread_struct), PCB(task_struct)

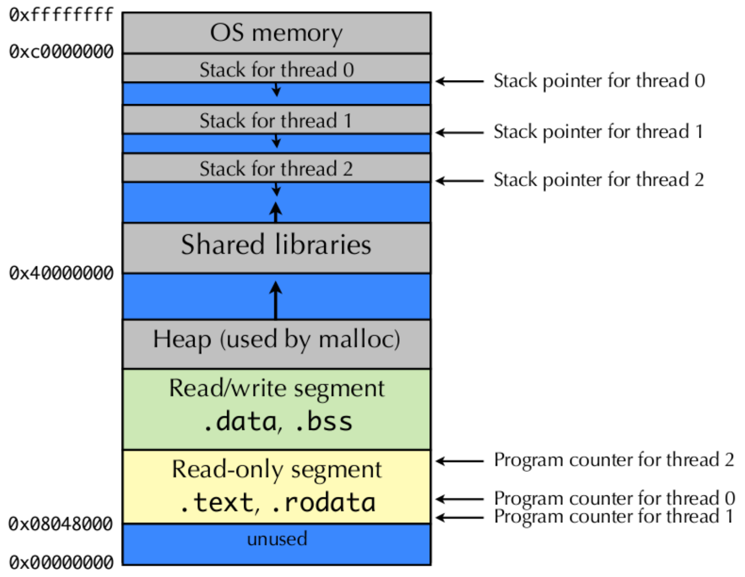

图1: 引入线程后，程序的进程空间

- 用户空间的线程实现
  - setcontext, getcontext, makecontext, swapcontext
  - int setcontext(const ucontext_t *ucp)
    - This function transfers control to the context in ucp. Execution continues from the point at which the context was stored in ucp. setcontext does not return.
  - int getcontext(ucontext_t *ucp)
    - Saves current context into ucp. This function returns in two possible cases: after the initial call, or when a thread switches to the context in ucp via setcontext or swapcontext. The getcontext function does not provide a return value to distinguish the cases (its return value is used solely to signal error), so the programmer must use an explicit flag variable, which must not be a register variable and must be declared

    volatile to avoid constant propagation or other compiler optimisations.

  - void makecontext(ucontext_t *ucp, void *func(), int argc, ...)
    * The makecontext function sets up an alternate thread of control in ucp, which has previously been initialised using getcontext. The ucp.uc_stack member should be pointed to an appropriately sized stack; the constant SIGSTKSZ is commonly used. When ucp is jumped to using setcontext or swapcontext, execution will begin at the entry point to the function pointed to by func, with argc arguments as specified. When func terminates, control is returned to ucp.uc_link.
  - int swapcontext(ucontext_t *oucp, ucontext_t *ucp)
    * Transfers control to ucp and saves the current execution state into oucp.

3.1.3 Scheduling

- 线程是CPU调度的基本单位。
- 上下文切换
- 抢占调度
- 协同调度

3.1.4 Programming with threads (pthreads library)

- Pthread
  - 线程的创建，终结
  - 同步
  - 通信

- Pth
- Pthread: API
  - pthread_create
  - pthread_self
  - pthread_exit
  - pthread_cancel
  - pthread_join

3.1.5 Shared vs. private resources

3.1.6 The need for synchronization

3.2 Problems

3.3 Solutions