# 操作系统

### 第一讲 计算机概述与操作系统引言

主要内容

1.1 基本构成

1.2 微处理器的发展演化

1.3 指令的执行

1.4 中断

1.5 存储器的层次结构

1.6 CPU与I/O设备协作

1.7 进程隔离和OS保护硬件支持----MMU

1.8 系统内核保护----CPU工作模式

1.9 系统调用机制

1.10 多处理器和多核计算机组织结构

1.4：

1. **中断：一种机制，即允许其他模块（I/O，存储器）在处理器正常处理过程中打断其工作**
2. **中断分类：**
3. **程序中断**
4. **时钟中断**
5. **i/o中断**
6. **硬件失效中断**
7. **中断是一种提高处理器效率的手段**
8. **利用中断功能，处理器可以在I/O操作的执行过程中 执行其他指令**
9. **I/O操作和用户程序中的指令的执行是并发的**

**1.7：进程隔离和OS保护硬件支持----MMU（存储管理部件）**

1. **进程隔离与OS保护的要求**

**多任务操作系统要求进程之间相互隔离：**

1. **一个进程的代码无论如何编写，CPU硬件应该提供一种机制，能防止它读出或更改其他程序的代码、数据**
2. **CPU硬件机制还应防护用户进程读出或更改操作系统内核的代码或数据**

**这种机制就是地址变换机制（MMU 存储管理部件）**

1. **MMU支持下进程隔离的方法**
   1. **有MUU环境下存储器地址的特点**

**在MUU支持下，CPU采用保护地址**

**程序代码，变量，堆栈的地址为逻辑地址（又称程序地址，虚拟地址VA），是用户能够看到的变量，程序代码所在的单元地址，它们所在的实际内存区域实际地址为物理地址PA**

**通常，逻辑地址≠物理地址**

* 1. **程序地址，物理地址和地址转换**

**程序地址空间低地址部分规定给用户进程使用称为用户地址，高地址部分分配给OS使用，称为内核空间**

* 1. **进程隔离方法**

**1.8：系统内核保护——CPU工作模式**

**1. 处理器工作模式，特权模式，非特权模式**

# ****第五章 进程控制****

**逻辑流与并发流**

1. 并发：指多个活动在在同一时间段内同时进行
2. 逻辑控制流（逻辑流）

逻辑流：在多进程运行环境下，一个程序（或进程）按程序员的意图，从main函数开始，一个个语句，一条条指令的执行，执行轨迹为一系列程序计数器（PC）的值，给人一种每个程序都独占处理器的假象，而一般整个系统可能仅有一个CPU，各进程轮流进行，我们称进程为一个逻辑流。

处理器在不同进程切换有两种原因：

进程主动放弃CPU：如进程执行耗时的I/O操作时

进程被动放弃CPU：此次分配的时间配额用完，或有更加紧迫的任务要执行，操作系统强行夺走CPU。

CPU控制发生转移的时间：一般都在中断响应发生之时，因为只有这个节骨眼操作系统能介入控制，中断有外部中断、时钟中断等

1. 并发流：一个逻辑流的执行在时间上与另一个流存在重叠情况，这两个流称为并发地执行，生命周期存在重叠地2个进程是并发流，或称为并发进程。
2. 并行流：如果同一时刻两个流运行在不同的处理器核或者计算机上，我们称他们为并行流。并行流在某段时间内同时执行，并发流的一个真子集。

**进程基本概念**

1. 进程：正在执行中的程序。
2. 进程的结构：程序代码，数据集和进程控制块（PCB）
3. 进程属性：
4. 进程描述信息：

进程号（PID）

用户标识

用户组标识

进程族亲信息：父进程标识，兄弟进程标识

1. 进程控制信息：

进程状态

调度信息（优先级，剩余时间片和调度策略）

计时信息：CPU使用时间

通信信息：未处理信号、管道、消息队列、共享内容等

1. 进程资源信息

存储器地址

打开文件信息

1. CPU现场

当前进程CPU寄存器副本：程序计数器PC、通用寄存器、标识寄存器FLACS

1. 进程基本状态以及变换

* 就绪状态(ready state)。进程已分配到除CPU以外的所有必要资源，只要获得CPU，便可立即执行，该状态为就绪状态
* 运行状态（也称执行状态， running state）：进程已包括CPU在内的所有运行所需资源，正在执行中
* 阻塞状态(blocked state)：正在执行的进程因请求资源、等待事件发生、等待I/O等原因，而暂时无法继续执行时，进入阻塞状态

实际操作系统设计中，往往还需增加2个状态：

* 创建状态：正在创建且尚未完成创建过程的进程(处于胎儿期）所处的状态称为创建状态

终止状态：进程终止后并不立即将其清理，而是让其进入终止状态

1. 进程PCB组织

双向链表队列

双向链表加树形结构

1. 操作进程的工具

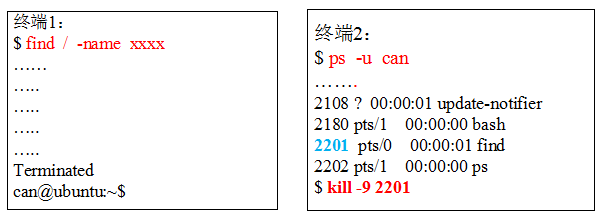
显示全部进程信息：ps —ef

从进程中过滤进程信息：ps —ef|grep （bash）

显示当前进程的消耗资源信息：ps -u

显示当前用户拥有进程的进程信息：ps l

终止进程：kill



**后台执行进程：命令后加“&”**好处：输入命令串 后，立即显示命令提示符，前面命令的执行和下一条命令的输入可并发执行

### 第六章 线程管理与同步互斥

* 1. 线程概念

线程：线程定义为进程内一个可执行单元或可调度实体，每个执行单元可执行进程内的一段代码（如函数），在一个进程里面可以创建多个线程，每个线程执行一块功能或活动（线程都是并发逻辑流的）

线程共享资源：一个进程内的线程都是在同一个地址空间的，共享线程的资源，如数据（全局变量）、代码、堆和打开文件等。

线程的结构：每个线程都拥有一个线程控制块（TCB thread control block）和一个线程堆栈。TCB用于存放线程的属性，堆栈用于给函数调用局部变量分配内存

TCB内容：线程ID，线程状态，栈指针、通用寄存器、程序计算器（PC）、标志寄存器。

* 1. **多线程并发特征与编程方法**

线程提供的API：pthread\_create（创建）, pthread\_exit（退出）, pthread\_join（线程）, pthread\_detach（分离）、pthread\_cancel（取消）;

* 1. 线程互斥
  2. 线程同步

同步：对并发线程操作的执行顺序进行控制。才能保证程序运行的正确性

1. 单向同步：
2. 双向同步问题模型：线程a 产生数据，将数据放入缓冲区（全局变量），线程b 从缓冲区读取数据，进行处理：



·线程a 的操作“buffer = v1”必须在线程b 的操作“v2 = buffer”前执行，保证线程b 总能读取到新数据

·线程a 的第二次操作“buffer = v1”必须在线程b 的第一次操作“v2 = buffer”之后，以保证不会覆盖还没被取走的数据

·以此类推

* 1. 用Pthreads同步机制实现线程互斥与同步

Linux 支持posix threads（简称Pthreads）线程规范，Pthreads提供四种同步设施：信号量，互斥量，读写锁，条件变量

Pthreads规范实现的信号量设施是Pthreads信号量，类型是sem\_t，初始化及对应的P操作、v操作函数分别为sem\_init, sem\_wait, sem\_post。

# 第七章

1.

2. 消息队列

基本思想：进程间以数据块（消息）为单位来传递数据，是进程之间传递数据的简单有效方法

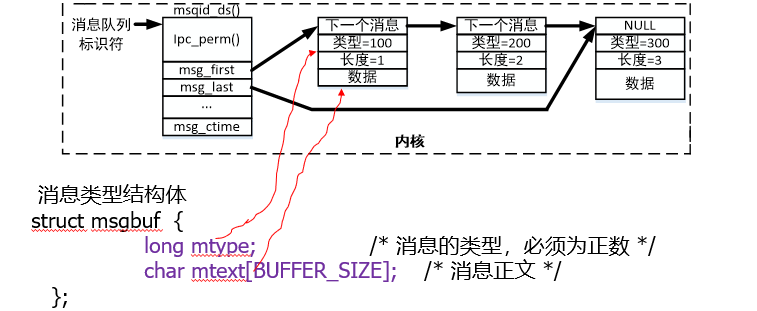
·它独立于接收和发送进程的存在，消除了同步命名管道的打开和关闭时遇见的麻烦

·每个数据块都可以看成某种类型，可发送任何数据类型

·MSGMAX：消息的最大长度

·MSGMNB：消息队列的最大长度

1. 消息队列结构（队列、消息）



1. 消息队列通信实例

·msgcreate.c：创建消息队列

·msgsend.c：用于发送消息

·msgrcv.c：接收消息

·msgdel.c：删除消息队列

1. 通过消息队列传输任意类型的数据

我们将传送类型为T的变量var，其定义为T var，消息缓冲区是一个地址为buffer的缓冲区

1. 传输方法



1. 共享内存
2. 基本原理：根据进程请求分配一块合适大小的存储块，各请求进程在其地址空间为该共享内存块分配程序地址，将程序地址赋值给变量指针，然后通过变量指针读写共享内存单元从而数据传输

·一般情况下，os分配给进程的物理内存块都是独享的因此不同进程的虚拟地址（程序地址）页对应不同的物理内存块

·只有共享是两个进程共有的内存块，两个进程中对应共享内存的地址区域对应着相同的物理内存块

·各进程分配给共享内存块的逻辑地址范围可能不相同

5 用ipc信号量实施进程同步