操作系统

叶茂林 2024年3月28日星期四

**操作系统四大功能**

处理机管理功能：进程控制、进程同步、进程通信、作业调度和进程调度

存储器管理功能：内存分配、内存保护、地址映射、虚拟内存

设备管理功能：缓存管理、设备分配、设备管理

文件管理功能：文件存储空间管理、目录管理、文件读写管理、文件共享保护

**物理地址和逻辑地址**

内存单元真正的地址，进程执行指令和访问数据的真正地址，程序暴露给程序员的地址

**虚拟内存**

当物理内存不足时操作系统会将一部分数据暂时存储在磁盘上，使得应用程序认为它拥有连续完整可用的地址

**进程和线程**

进程是资源分配的基本单位，线程是处理机任务调度和执行的基本单位，一个进程可以有多个线程

不同的进程之间的地址空间和资源是相互独立的，同一进程的线程共享进程的地址空间和资源

通信方式不同，创建的过程开销不同，在Linux上，创建进程会创建一个进程控制块PCB，也就是task\_struct，描述进程的属性信息，然后会分配一个mm\_struct描述进程的内存空间，其中有一个vm\_area\_struct指针指向一个vm\_area链表描述进程的虚拟内存区域，如果创建线程，同样会有一个task\_struct描述线程的状态信息，但是会和进程，也就是主线程共享一个mm\_struct，会增加一个vm\_area\_struct结构体描述线程的堆栈空间

**进程状态**处理机调度的时候：执行、就绪

IO资源需求：阻塞

挂起激活：活动（就绪、阻塞），静止（就绪、阻塞）

创建、终止

**进程控制块**进程标识符：外部标识符、内部标识符（进程号）

处理机状态：：通用寄存器、指令计数器PC、程序状态字、用户栈指针

进程调度信息：进程状态、进程优先级、调度算法所需数据、阻塞事件

进程控制信息：程序数据地址、同步通信机制、资源清单、链接指针

**进程间通信（IPC）**

管道（匿名管道、命名管道）、信号、信号量、消息队列、套接字、信号量

信号量：计数器，控制并发进程对共享资源的访问，用来上锁和解锁

共享内存：

共享数据结构：生产者-消费者问题，程序定义数据结构申请内存同步访问

共享存储区：程序申请一块共享内存，进程通过对共享区读写来交换信息

消息队列：消息队列是先进先出的队列型数据结构，实际上是系统内核的一个链表，进程发送消息入队尾，进程从队首读取消息

**进程调度**

**先来先**服务：非抢占式，按照请求顺序进行调度，有利于长任务，对短任务不利

**短作业**优先：预估执行时间短的先执行，有利于短作业，如何正确评估时间，如果是抢占式变成最短剩余时间优先

**高响应比**优先：先来先服务只考虑了等待时间，短作业优先只考虑了执行时间，高响应比考虑响应时间和执行时间的比值，响应时间就是等待时间和执行时间之和，如果等待时间相同，则短作业的响应比更高，如果执行时间相同，则等待时间长的响应比更高，随着等待时间的增加，短作业的响应比提高更快，同时长作业的响应比也会随着等待时间的增加而提高

**优先级**调度：优先级高的先执行

**时间片**轮转：按照先来先服务排成就绪队列，让队首执行一个时间片的时间后中断送往队尾，受时间片大小影响，如果时间片过小就会频繁切换线程，过长就会退化为先来先服务

**多级反馈队列**：准备多个就绪队列，每个队列优先级不同，第一个队列优先级最高，其余队列优先级逐个降低，每个队列时间片大小也不同，后一个队列时间片是前一个的两倍，每个队列采用先来先服务调度，如果进程可以在当前队列的一个时间片内完成就离开，不能完成就进入下一个队列的末尾

**协程**

用户态的轻量级线程，线程内部调度的基本单位

**管程**

定义公共数据结构为并发进程执行的一组操作，进程调用管程访问临界资源

**临界资源**

同一时刻只能有一个进程使用的资源

**临界区**

进程中访问临界资源的代码段

**守护进程**

后台运行的独立于终端的进程

**创建守护进程**

后台运行程序，创建子进程后结束父进程，调用setsid创建新对话期成为会话组长脱离原终端、原进程组和登陆会话，再次创建子进程退出父进程卸任会话组长，这样就无法打开终端，关闭文件描述符，将当前目录改为根目录，将屏蔽字清零，忽略SIGCHLD信号避免产生僵尸进程

**孤儿进程**

父进程先结束，子进程成为孤儿进程，父进程变成init（1号）进程

**僵尸进程**

子进程结束父进程没有捕获它的退出状态，子进程变成僵尸进程，可以忽略SIGCHLD信号避免僵尸进程

**局部性原理**

时间局部性：最近访问的内存不久后可能还会被访问

空间局部性：当前访问的附近内存不久后可能会被访问

**操作系统中的task\_struct、mm\_struct和vm\_area\_struct有什么联系**

Task\_struct是进程控制块PCB的数据结构，用于管理和控制进程，存储了进程的信息，其中有一个mm\_struct指针指向，mm\_struct是描述进程内存布局的数据结构，其中有一个vm\_area\_struct指针指向一个链表，vm\_area\_struct描述了虚拟地址空间的一个区间，区间少时使用链表管理，区间多时使用红黑树管理

**进程地址空间**

代码段、数据段、BSS段、堆、内存映射段、栈

**大小端**

小端：数据的低字节存储在低地址

大端：数据的高字节存储在低地址

**解决死锁**

忽略死锁：当死锁影响不大，或者发生死锁的概率很低，可以不管它

死锁检测与死锁恢复：

死锁检测：检测是否存在环路等待

死锁恢复：抢占、回滚、杀死

死锁预防：在程序运行前破坏死锁发送的条件

死锁避免：在程序运行时避免发送死锁

**逻辑地址到物理地址的转换**

页内地址和物理块内的地址一一对应，地址转换只需要将页号转换为物理块号，通过页表完成，页表里面记录了从页号到物理块号的地址映射

由于地址转换的频率非常高，页表的功能需要通过硬件来实现，系统中设置了一个页表寄存器，存放页表在内存中的起始地址和页表的长度

当进程没有执行的时候，页表的起始地址和页表的长度存放在PCB里面，当进程进入执行状态的时候，页表的起始地址和页表长度被装进页表寄存器里面

当程序访问逻辑地址的数据的时候，逻辑地址会被分为页号和页内地址偏移两部分，先把页号和页表长度作比较，如果页号大于等于页表长度，说明本次访问越界，如果没有越界，以页号为索引，页表起始地址为基址找到页号对应的物理块号，再和页内地址组合得到物理地址

**快表**

没有快表之前，程序访问逻辑地址的数据需要两次访问内存，第一次访问内存中的页表得到物理块号，再和页内偏移地址拼接得到物理地址，以此地址第二次访问内存才得到数据

因此在原来的基础上增加了一个高速缓存寄存器，存放着最近访问过的页表项，地址变换的时候先在快表里面找看看有没有，有的话直接读取页号对应的物理块号，没有再去内存里面的页表里面找到，然后往快表里面添加新的页表项，如果快表里面满了，就去掉一个最少访问的页表项，类似于LRU缓存