进程的描述与控制

2021年11月1日 21:30

1、前趋图与程序执行

1. 前趋图
    a. 有向无环图
    b. 描述进程之间执行的先后顺序
2. 程序顺序执行
    a. 程序的顺序执行
    b. 程序顺序执行时的特征
        i. 顺序性
        ii. 封闭性
        iii. 可再现性
3. 程序并发执行
    a. 程序的并发执行
        i. 不存在前驱关系才可以并发执行
    b. 程序并发执行时的特征
        i. 间断性
        ii. 失去封闭性
        iii. 不可再现性

2、进程的描述

1. 进程的定义和特征
    a. 进程的定义
        i. 进程是程序的一次执行
        ii. 进程是一个程序及其数据在处理机上顺序执行时所发生的活动
        iii. 进程是具有独立功能的程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位
    b. 进程的特征
        i. 动态性
        ii. 并发性
        iii. 独立性
        iv. 异步性
2. 进程的基本状态及其转换
    a. 进程的三种基本状态
        i. 就绪Ready
        ii. 执行Running
        iii. 阻塞Block
    b. 三种基本状态的转换
    c. 创建状态和终止状态

3. 挂起操作和进程状态的转换
    a. 挂起操作的引入
        i. 终端用户的需要
        ii. 父进程请求
        iii. 负荷调节的需要
        iv. 操作系统的需要
    b. 引入挂起操作后进程状态的转换
        i. 活动就绪，静止就绪
        ii. 活动阻塞，静止阻塞

        Suspend原语，Active原语
4. 进程管理中的数据结构
    a. 操作系统中用于管理控制的数据结构
        i. 内存表
        ii. 设备表
        iii. 文件表
        iv. 进程表
    b. 进程控制块PCB
        i. 作为独立运行基本单位的标志
        ii. 能实现间断性运行方式
        iii. 提供管理进程所需要的信息
        iv. 提供进程调度所需要的信息
        v. 实现与其他进程的同步与通信
    c. 进程控制块中的信息
        i. 进程标识符
            1) 外部标识符
            2) 内部标识符
        ii. 处理机状态
        iii. 进程调度信息
        iv. 进程控制信息
    d. 进程控制块的组织方式
        i. 线性方式
        ii. 链接方式
        iii. 索引方式

3、进程控制

1. 操作系统内核
    a. 支撑功能
        i. 中断处理
        ii. 时钟管理
        iii. 原语操作
    b. 资源管理功能
        i. 进程管理

ii. 存储器管理
iii. 设备管理

2. 进程的创建
   a. 进程的层次结构
      i. 子进程继承父进程所拥有的资源
      ii. 撤销父进程时，也要撤销其所有的子进程
      Windows：句柄（无层次结构，获得句柄即获得控制权）
      Linux：PID
   b. 进程图
   c. 引起创建进程的事件
      i. 用户登录
      ii. 作业调度
      iii. 提供服务
      iv. 用户请求
   d. 进程的创建
      i. 申请空白PCB，申请获得唯一数字标识符
      ii. 为新进程分配其所需要的资源
      iii. 初始化进程控制块
      iv. 如可以，插入就绪队列
3. 进程的终止
   a. 引起进程终止的事件
      i. 正常结束
      ii. 异常结束
      iii. 外界干预
   b. 进程的终止过程
      i. 根据进程标识符，从PCB集合中取出PCB，读取进程状态
      ii. 若进程正在执行则立即终止，设置调度标志为true
      iii. 若进程由子进程，终止所有子进程
      iv. 将被终止进程所有资源还给其父进程或系统
      v. 将被终止进程PCB从所在队列中取出，等待其他进程搜集信息
4. 进程的唤醒与阻塞
   a. 引起进程唤醒与阻塞的事件
      i. 向系统请求共享资源失败
      ii. 等待某种操作完成
      iii. 新数据尚未到达
      iv. 等待新任务的到达
   b. 进程阻塞过程
      i. 调用阻塞原语block将自己阻塞
      ii. 进程的主动行为
   c. 进程唤醒过程
      i. 当被阻塞进程所期待的事件发生时，有关进程调用唤醒原语wakeup将等待该事件的进程唤醒

ii. 进程被动唤醒

5. 进程的挂起与激活
   a. 进程的挂起

      系统中出现引起进程挂起的事件时，OS利用挂起原语suspend将指定进程挂起

      1) 活动就绪→静止就绪
      2) 活动阻塞→静止阻塞

      将进程从内存调入到外存

      将PCB复制到指定内存区域，方便用户或系统考察进程运行情况

   b. 进程的激活

      系统中出现引起进程激活的事件时，OS利用激活原语active将指定进程激活

      1) 活动就绪←静止就绪
      2) 活动阻塞←静止阻塞

      将进程从外存调入到内存

      重新调度该进程

4、进程同步

1. 进程同步的基本概念
   a. 两种形式的制约关系
      i. 间接相互制约关系
      ii. 直接相互制约关系
   b. 临界资源
      i. 许多硬件资源都是临界资源
      ii. 诸进程之间应该采用互斥方式实现共享
   c. 临界区

      无论是硬件临界资源还是软件临界资源，多个进程必须进行互斥访问。

      进程中访问临界资源的部分代码称为临界区

      1) 进入区
      2) 临界区
      3) 退出区
      4) 剩余区

   d. 同步机制应遵循的规则
      i. 空闲让进
      ii. 忙则等待
      iii. 有限等待
      iv. 让权等待
2. 硬件同步机制
   a. 关中断
      i. 简单
      ii. 可能导致严重后果
      iii. 影响效率
      iv. 不适用于多CPU系统

b. 利用Test-and-Set指令实现互斥
c. 利用Swap指令实现互斥

TS指令与Swap指令产生自旋锁。

i. 忙等，始终占用CPU测试锁
ii. 不符合让权等待
iii. 没有进程阻塞与唤醒的消耗

3. 信号量机制
   a. 整形信号量
      i. 原子操作P wait()
      ii. 原子操作V signal()

```
wait(S) {
     while (S <= 0) {};
     S--;
}
signal(S) {
     S++;
}
```

不遵循让权等待

   b. 记录型信号量
      i. 不存在忙等

```
Typedef struct {
     int value;
     struct  pcb_list *list;
}semaphore;
wait(semaphore *S) {
     S->value--;
     if (S->value < 0)
          block(S->list);
}
signal(S) {
     S->value++;
     if (S->value <= 0)
          wakeup(S->list);
}
```

   c. AND型信号量
      i. 将程序运行期间所需所有资源一次性全部分配给进程
      ii. 避免死锁（有限等待）
      iii. Swait()
   d. 信号量集
      i. 一次性申请多个单位资源
4. 信号量的应用
   a. 实现进程互斥
      i. 信号量机制产生互斥锁
      ii. wait和signal成对出现
   b. 实现前趋关系
5. 管程机制

5、经典进程的同步问题

1. 生产者-消费者问题

a. 缓冲区满、缓冲区空
b. 阻塞、唤醒生产者消费者

2. 哲学家进餐问题
   a. 竞态资源访问
3. 读者-写者问题
   a. 允许多进程同时读，不允许同时写

6、进程通信

1. 进程通信的类型
   a. 共享存储器系统
      i. 基于共享数据结构
      ii. 基于共享存储区
   b. 管道通信系统
      i. 互斥读写管道
      ii. 同步
      iii. 确定对方是否存在
   c. 消息传递系统
      i. 直接通信，消息发送原语
      ii. 间接通信方式
   d. 客户机-服务器方式
      i. 套接字Socket
         1) 基于文件型
         2) 基于网络型
      ii. 远程过程调用
      iii. 远程方法调用
2. 消息传递通信的实现方式

7、线程的基本概念

1. 线程的引入
   a. 进程的基本属性
      i. 进程是一个可用有资源的独立单位
      ii. 进程是可独立调度和分派的基本单位
   b. 进程并发所需付出的时空开销
      i. 创建进程
      ii. 撤销进程
      iii. 进程切换
   c. 线程作为调度和分配的基本单位
2. 线程与进程的比较
   a. 调度的基本单位
      i. 进程调度需要进行上下文切换
      ii. 线程调度仅需保存与设置少量寄存器内容，不会引起进程切换
   b. 并发性
      i. 线程可以并发执行

c. 拥有资源
    i. 进程拥有资源
    ii. 线程本身不拥有系统资源
d. 独立性
    i. 进程独立性较高，除全局变量不允许互相访问
    ii. 线程独立性低，线程共享内存与地址空间
e. 系统开销
    i. 系统创建与撤销进程所付出的开销远大于线程
    ii. 进程切换的代价远高于线程

3. 线程的状态与线程控制块
a. 线程运行的三个状态
    i. 执行状态
    ii. 就绪状态
    iii. 阻塞状态
b. 线程控制块TCB
    i. 线程标识符
    ii. 寄存器，包括程序计数器，状态寄存器，通用寄存器
    iii. 线程运行状态
    iv. 优先级
    v. 线程专有存储区
    vi. 信号屏蔽
    vii. 堆栈
c. 多线程OS中的进程属性
    i. 进程是一个可拥有资源的基本单位
    ii. 多个线程可并发执行
    iii. 进程不再是可执行的实体，将线程作为独立运行或调度的基本单位

8、线程的实现

1. 线程的实现方式
a. 内核支持线程KST
    i. 在内核的支持下运行
    ii. 在内核空间实现，每个线程一个TCB
    iii. 优点
        1) 多处理机系统下可同时调度一个进程中的多个线程并行执行
        2) 进程中的一个线程被阻塞，内核可以调度改进程中的其他线程，或其他进程中的线程
        3) 线程切换速度快，切换开销小
        4) 内核本身可以采用多线程，可提高系统执行速度和效率
    iv. 缺点
        1) 对用户的线程切换而言，模式切换的开销较大
b. 用户级线程ULT
    i. 在用户空间实现，无需内核支持

ii. 内核不知道用户级线程的存在，调度仍旧以进程为单位
iii. 优点
    1) 线程切换不需要转换到内核空间
    2) 调度算法可以是进程专用的
    3) 用户级线程的实现与OS平台无关
iv. 缺点
    1) 系统调用将使进程阻塞，所有线程均不可执行
    2) 每次分配给进程的只有单个CPU，仅有一个线程可以执行

c. 组合方式
i. 多对一模型
    1) 将用户级线程映射到一个内核控制线程
    2) 开销小，效率高
    3) 一个线程阻塞则整个进程阻塞
    4) 多个线程不能同时在多个处理机上运行
ii. 一对一模型
    1) 一个用户级线程映射到一个内核支持线程
    2) 并发性更好
    3) 一个线程阻塞，其他线程仍可以执行
    4) 开销较大
iii. 多对多模型
    1) 将多个用户级线程映射到更少数量的内核级线程
    2) 结合二者优点

2. 线程的实现
a. KST的实现
i. 与进程的实现非常类似
ii. 创建、回收TCB
b. ULT的实现
i. 运行时系统
    1) 线程切换时不切换至核心态，直接通过线程切换过程函数修改CPU寄存器内容
    2) 系统调用通过软中断实现，内核分配资源
ii. 内核控制线程
    1) 线程池

3. 进程的创建与终止