2020年7月29日 13:41

《操作系统》

一、 操作系统引论

1. 单道批处理系统 多道批处理系统、
   分时系统 时间片
   实时系统
2. **基本特性**
   并行与并发
   引入进程 Process
   共享
   虚拟 时分复用 空分复用
   异步
3. **主要功能**
   处理机管理功能 进程控制 进程同步 进程通信 调度
   储存器管理功能 内存分配 内存保护 地址映射 内存扩充
   设备管理功能 缓冲管理 设备分配 设备处理
   文件管理系统 文件储存空间管理 目录管理 文件的读/写管理和保护

二、进程的描述与控制

1. **前趋图 DAG**
   顺序执行： 顺序性 封闭性 可再现性
   并发执行： 间断性 失去封闭性 不可再现性
2. **进程 进程控制块PCB**
   进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。
   动态性 并发性 独立性 异步性
   三种基本状态 就绪Ready 执行Running 阻塞Block

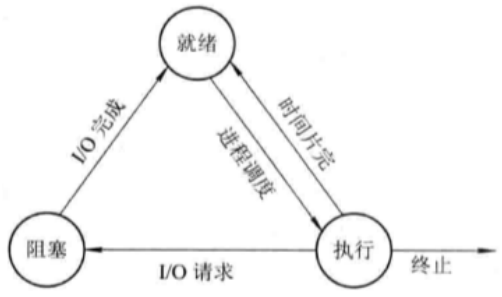


图 2-5 进程的三种基本状态及其转换

引入两种常见状态：创建状态 终止状态

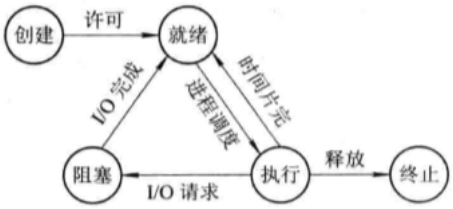


图 2-6　进程的五种基本状态及转换

挂起操作

终端用户的需要 父进程请求 负荷调节需要 操作系统的需要

活动就绪 -> 静止就绪

活动阻塞 -> 静止阻塞

静止就绪 -> 活动就绪

静止阻塞 -> 活动阻塞

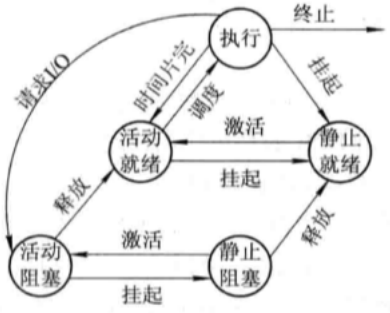


图 2-7　具有挂起状态的进程状态图

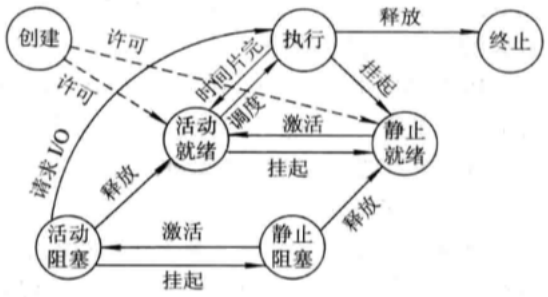


图 2-8　具有创建、终止和挂起状态的进程状态图

3. 进程管理中的数据结构

内存表 设备表 文件表 进程表PCB

PCB

1）作为独立运行基本单位的标志

2）能实现间歇性运行方式

3）提供进程管理所需要的信息

4）提供进程调度所需要的信息

5）实现与其他进程的同步与通信

PCB控制信息

1）进程标识符 外部标识符（人为命名） 内部标识符（序号）

2）处理机状态 各种寄存器内容 通用寄存器 指令寄存器 程序状态字PSW 用户栈指针

3）进程调度信息 进程状态 进程优先级 进程调度所需的其他信息 事件

4）进程控制信息 程序和数据的地址 进程同步和通信机制 资源清单 链接指针

4. 进程控制块的组织方式

线性方式

链接方式 队列

索引方式 建立索引表

5. 进程控制

处理机执行状态： 系统态 用户态

支撑功能 中断处理 时钟管理 原语操作

资源管理功能 进程管理 储存器管理 设备管理

6. **进程的层次结构**

进程内创建进程 形成父子关系 共同构成进城家族

Windows中不存在进程层次结构 获得句柄与否 控制与被控制

进程图

7. **创建进程有关的事件**：用户登录 作业调度 提供服务 应用请求
8. **创建过程：**

1）申请空白PCB

2）为新进程分配其运行所需的资源，包括各种物理和逻辑资源

3）初始化PCB 初始化标识信息 初始化处理机状态信息 初始化处理机控制信息

4）如果进程就绪队列能够接纳新进程，便将进程插入就绪队列

9. **进程终止**

正常结束

异常结束 越界错 保护错 非法指令 特权指令错 运行超时 等待超时 算数运算错 I/O故障

外界干预 操作员/操作系统干预 父进程请求 父进程终止

10. **终止过程**

1）根据被终止进程的标识符，检索PCB，读出状态

2）若正在执行，立即终止执行，调度标志置为真

3）所有子孙进程都予终止

4）归还资源

5）将终止PCB从队列中移除

11. **进程阻塞和唤醒事件**

1）向系统请求共享资源失败

2）等待某种操作的完成

3）新数据尚未到达

4）等待新任务的到达

创建 create源语

阻塞 block源语

唤醒 wakeup源语

挂起 suspend源语

激活 active源语

12. **进程同步**

间接相互制约关系 直接相互制约关系

临界资源

临界区 访问临界资源的代码

规则：空闲让进 忙则等待 有限等待 让权等待

13. **硬件同步**

关中断

TS指令 Test-and-Set

```
do {
  ...
  while TS(&lock);              /*do skip */
  critical   section;
  lock := FALSE;
  remainder section;
}while(TRUE);
```

Swap指令实现进程互斥

```
do {
    key=TRUE;
    do {
      swap(&lock，&key);
    }while (key!=FALSE);
    临界区操作;
    lock = FALSE;
    ...
} while (TRUE);
```

14. 信号量机制

整型信号量 S

wait(S) signal(S) 称为P,V操作

```
wait(S){
      while (S<=0);              /*do no-op*/
      S--;
}
signal(S)
{
      S++;
}
```

wait中，若S<=0会不停的测试，没有让权等待，进入忙等

记录型信号量

数据结构 资源数目 value 进程链表指针 list

```
typedef struct {
     int value;
     struct process_control_block *list;
}semaphore;
```

wait()和signal()操作:

```
wait(semaphore *S) {
```

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

AND型信号量 将需要的资源一次性分配 防止因没有拿到所需的所有资源而导致死锁

信号量集

$Swait(S_1, t_1, d_1, \ldots, S_n, t_n, d_n)$;

$Ssignal(S_1, d_1, \ldots, S_n, d_n)$;

ti 分配下限值 di 资源需求值

利用信号量实现进程互斥

利用信号量实现前趋关系 利用公共量实现

15. 进程同步问题

生产者-消费者问题

哲学家进餐问题

读者-写者问题

16. 进程通信

共享储存器系统

pipe管道通信 UNIX 互斥 同步 确定对方是否存在

消息传递系统

客户机-服务器系统

1）套接字Socket

2）远程过程调用和远程方法调用

17. 消息传递通信实现方式

1）直接消息传递系统

（1）对称寻址方式

send(receiver，message);　发送一个消息给接收进程

receive(sender，message);　接收 Sender 发来的消息

（2）非对称寻址方式

send(P，message);　发送一个消息给进程 P

receive (id，message)；接收来自任何进程的消息，id 变量可设置为进行通信的发送方进程 id 或名字。

2）信箱通信 间接通信方式

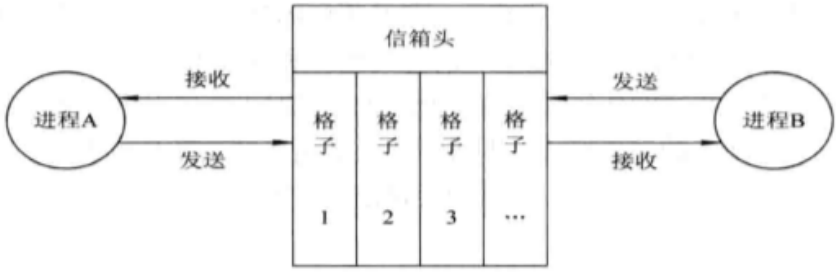


图 2-16　双向信箱示意图

Send(mailbox，message);　　将一个消息发送到指定邮箱
Receive(mailbox，message);　从指定邮箱中接收一个消息

18. 线程Threads

调度和分配的基本单位

同一进程中，线程的切换不会导致进程的切换，但从一个进程中的线程切换到另一个进程中的进程，必然会导致进程的切换

并发性 多个线程之间也可以并发

拥有资源 线程可以拥有资源，并作为系统中拥有资源的一个基本单位；线程并不拥有系统资源，仅有不可少的保证独立运行的资源。

独立性 每个进程有独立的地址空间和其他资源，除了共享全局变量，不允许其他访问；线程则相反

系统开销 进程创建和撤销，需要系统分配或回收进程控制块以及其他资源；进程切换也存在进程上下文的切换；在一些OS中线程的切换，同步和通信无需系统内核干涉

支持多处理机系统 单线程进程 多线程进程

线程状态 执行 就绪 阻塞

线程控制块TCB 线程标识符 一组寄存器 线程运行状态 优先级 线程专有储存区 信号屏蔽 堆栈指针

19. 线程的实现

三、处理机调度与死锁

1. 高级调度 作业调度

低级调度 进程调度

中级调度 提高内存利用率和系统吞吐量

2. 处理机调度算法的目标

1）资源利用率

$$\text{CPU 的利用率} = \frac{\text{CPU 有效工作时间}}{\text{CPU 有效工作时间} + \text{CPU 空闲等待时间}}$$

2）公平性 各个进程合理的CPU时间

3）平衡性 系统资源使用平衡

4）策略强制执行

批处理系统目标

1）平均周转时间短

2）系统吞吐量高

3）处理及利用率高

分时系统的目标

1）响应时间快

2）均衡性 系统响应时间的快慢和用户所请求的服务的复杂性相适应

实时系统的目标

1）截止时间的保证

2）可预测性

3. 作业与作业调度

作业Job 作业步Job Step 作业控制块JCB

收容，运行，完成 后备状态 运行状态 完成状态

先来先服务FCFS调度算法 配合其他算法使用

短作业优先SJF算法

必须预知作业运行时间

对长作业不利

FCFS算法人机无法交互

未考虑作业额紧迫程度

优先级调度算法PSA

高响应比优先调度算法HRRN

$$R_P = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{响应时间}}{\text{要求服务时间}}$$

作业等待时间相同，要求服务时间越短，其优先权越高，类似SJF算法

服务时间相同，作业优先权又决定于等待时间，类似FCFS算法

长作业随着等待时间增加，优先级上升，得以处理

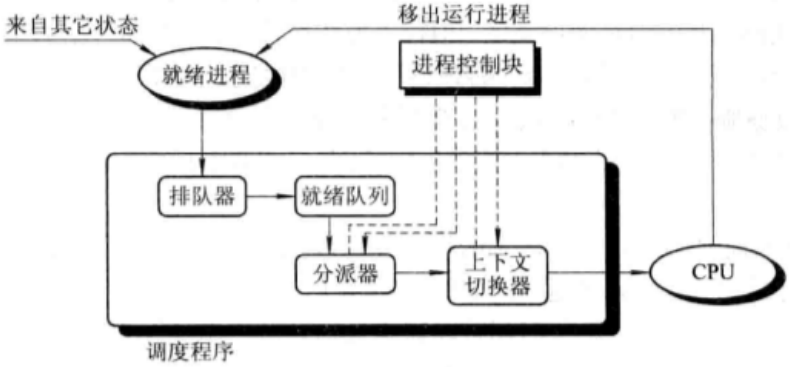
4. 进程调度

保存处理机现场信息

按照某种算法选取进程

将处理器分配给进程

调度机制 排队器 分派器 上下文切换器



进程调度方式

非抢占方式

正在执行的进程运行完毕或无法继续运行

正在执行的进程因提出IO请求而暂停

在通信或同步中执行原语操作，例如Block

抢占方式 对于批处理机 可以防止长进程长时间占用处理机 公平 实现人机交互

优先权原则

短进程优先原则

时间片原则

5. **轮转调度算法** 基于时间片的轮转RR

   FCFS排序 按照时间片 每个运行（如果均能在一个时间片内完成，退化为FCFS算法）

   优先级调度算法

   非抢占式 抢占式

   静态优先级 动态优先级

   多队反馈列调度算法 多个就绪队列实现不同调度要求

   多个就绪队列

   每个队列都使用FCFS算法 如果没有完成 降队列

   按照队列优先级调度

   保证调度算法

   公平分享调度算法

6. **实时调度**

   必要信息 就绪时间 开始截止时间完成时间 处理时间 资源要求 优先级

   非抢占轮转调度 非抢占优先调度

   基于时钟中断的抢断优先级调度 立即抢占的优先级调度

   最早截止时间优先EDF 截止时间为优先级

   非抢占调度方式用于非周期实时任务

   抢占调度方式用于周期实时任务

   最低松弛度优先LLF

   可抢占调度

7. **死锁**

   可重用性资源

   每个单元只能分配给一个进程使用，不能多个进程共享

   进程在使用可重用性资源的时候顺序： 请求资源 使用资源 释放资源

   可重用性资源单元数目相对固定

   可消耗性资源

   单元数目不断变化

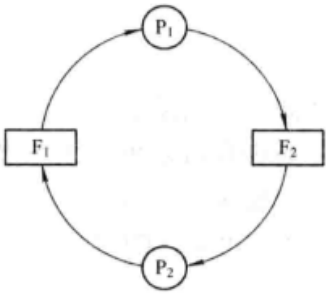
   进程运行中可以被创造

   可以申请多个并不返回

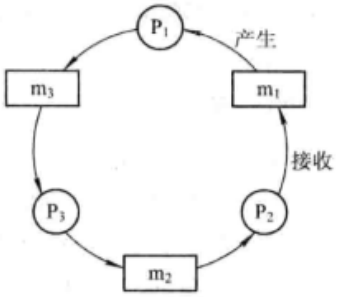
   可抢占性资源 CPU主存 不会引起死锁

   不可抢占性资源 等待进程结束自然释放

   竞争不可抢占资源引起死锁 共享文件



竞争可消耗资源引起死锁 进程通信



进程顺序不当引起死锁

定义：如果一组进程中的每一个进程都在等待仅有该组进程中的其他进程才能引发的事件，那么这组进程是死锁的

形成的必要条件：

互斥条件

请求和保持条件

不可抢占条件

循环等待条件

8. **死锁处理方法**

预防死锁

破坏“请求和保持”条件

第一种协议 一次性申请所有需要的资源 资源浪费 进程饥饿

第二种协议 只获得运行初期所需要的资源，释放资源，然后再申请

破坏“不可抢占”条件 必须释放 重新申请 增加开销 降低吞吐

破坏“循环等待”条件 进行编号

避免死锁

安全状态和不安全状态

银行家算法

数据结构 可利用资源 最大需求矩阵 分配矩阵 需求矩阵

死锁的检测

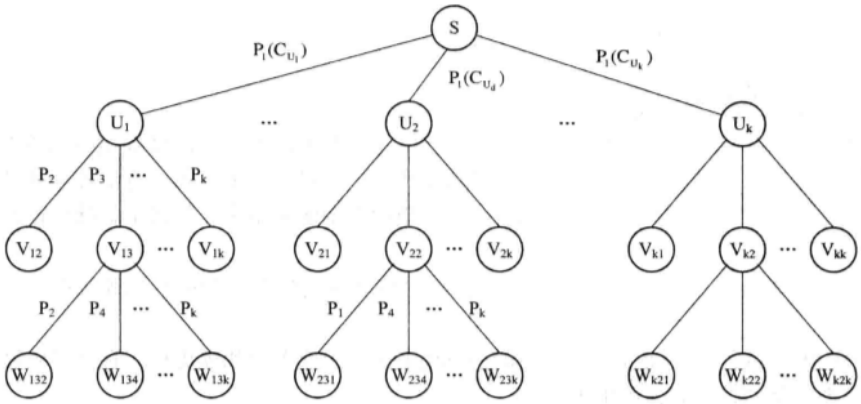
资源分配图

死锁的充分条件：当且仅当S状态的资源分配图是不可完全简化的 死锁定理

死锁的解除

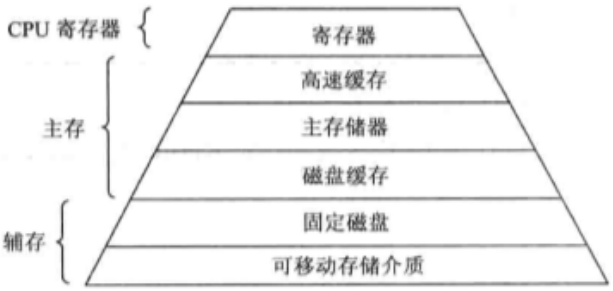
终止所有死锁进程

逐个终止进程 付出代价最小的死锁解除算法



四、储存器管理

1. 层次结构



2. 程序过程

(1) 编译，由编译程序(Compiler)对用户源程序进行编译，形成若干个目标模块(Object Module)；

(2) 链接，由链接程序(Linker)将编译后形成的一组目标模块以及它们所需要的库函数链接在一起，形成一个完整的装入模块(Load Module)；

(3) 装入，由装入程序(Loader)将装入模块装入内存。

装入：

绝对装入方式 绝对地址的目标代码

可重定位装入方式 静态重定位

动态运行时的装入方式 逻辑地址

链接：

静态链接方式： 对相对地址进行修改 变换外部调用符号

装入时动态链接： 便于修改和更新 便于实现对目标模块的共享

运行时动态链接： 程序执行时再进行

3. 连续分配储存管理方式

固定分区分配 划分分区 内存分配

动态分区分配

空闲分区表 空闲分区链

分配内存 回收内存

首次适应FF算法 低址空间不断被划分

循环首次适应NF算法 缺乏大的空间分区

最佳适应BF算法 满足需求的最小空间 留下过多碎片

最坏适应WF算法 满足需求的最大空间

基于索引的动态分区分配算法（略）

快速适应算法 伙伴系统 哈希算法

动态可重定位分区分配

紧凑 小空间拼凑成大空间

动态重定位 使用重定位寄存器