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

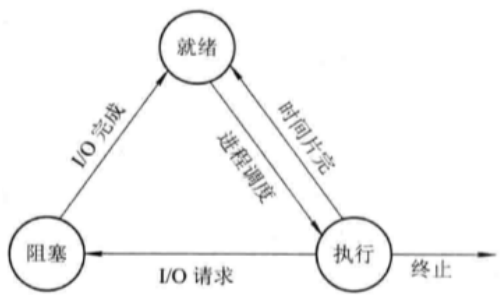

图 2-5 进程的三种基本状态及其转换

引入两种常见状态：创建状态 终止状态

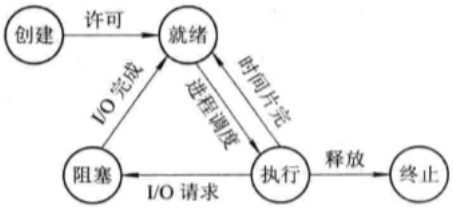

图 2-6　进程的五种基本状态及转换

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

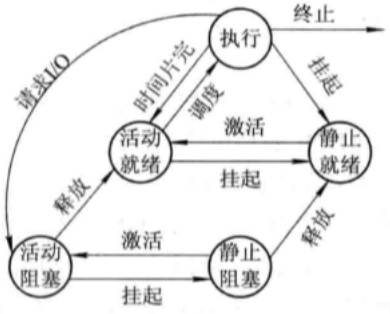

图 2-7　具有挂起状态的进程状态图

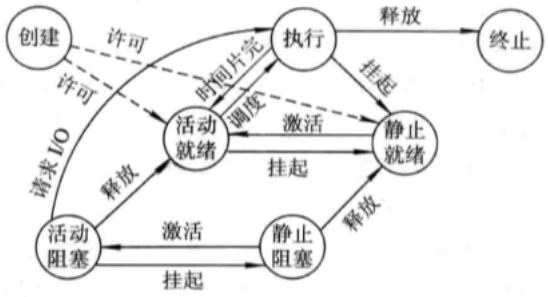

图 2-8　具有创建、终止和挂起状态的进程状态图

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, …, S_n, t_n, d_n);

Ssignal(S_1, d_1, …, 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