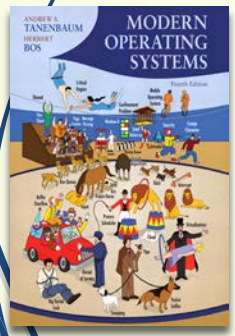
第2-1章　进程概念基础



主讲教师：全红艳

计算机科学与技术学院

本次课程内容

1. 进程的概念
2. 进程的调度
3. 进程的创建与终止
4. 进程的阻塞与唤醒

本章课程教学内容及目的

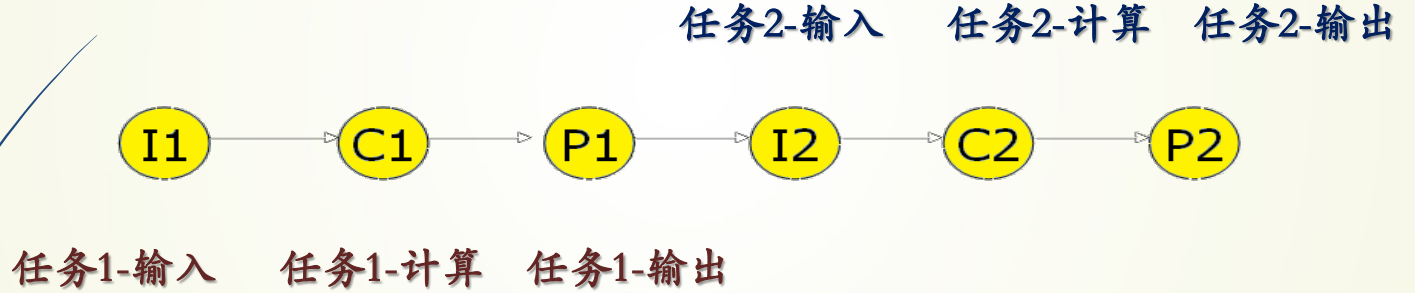
- 讲授操作系统的进程的概念，使学生从资源角度了解为什么要引入进程
- 描述进行的几个特征：数据结构、调度、状态，为进一步掌握进程管理奠定基础
- 讲解进程的创建与终止过程，使学生对进程有深入了解
- 讲解进程的阻塞与唤醒，有利于学生对操作系统原理深入理解

1. 进程的概念

- 从以下三个方面掌握进程的概念：
  - 进程定义 Process Definition
  - 进程状态描述 Process State
  - 进程控制块 Process Control Block (PCB)

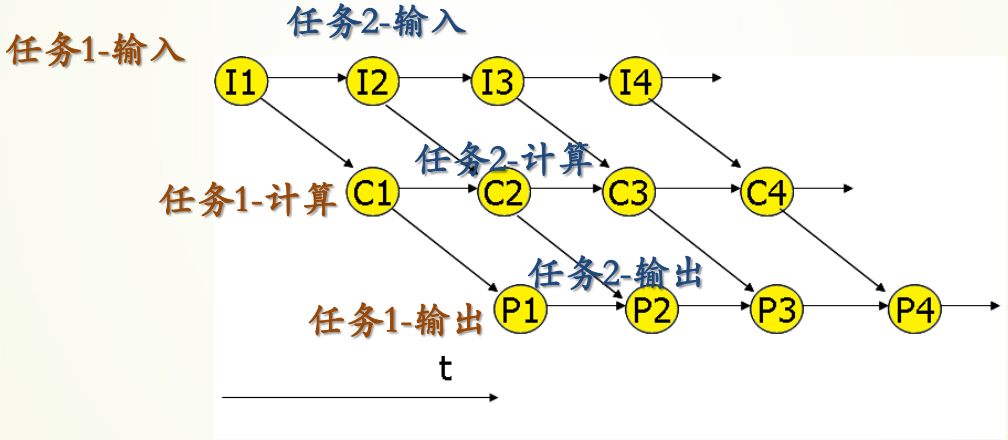
（1）进程概念的引入

- 早期计算机系统，一次只允许执行一个程序，其控制计算机全部资源
- 顺序性：多个程序按固定顺序执行



进程概念的引入

- 多道程序并发执行，充分利用系统资源
- 任务的并发，提高资源资源的利用率



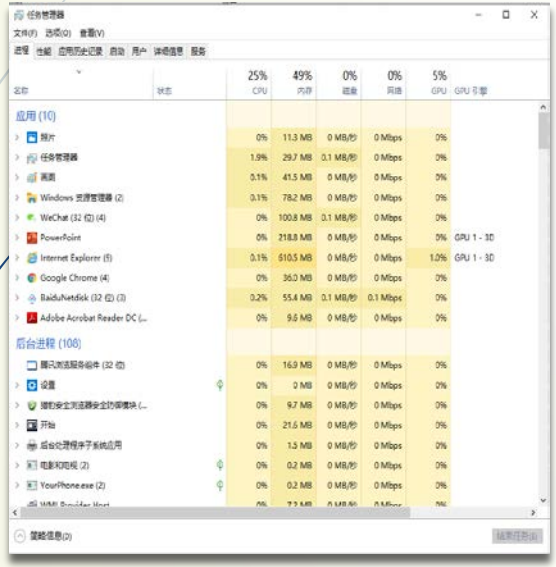
导致进程概念的产生

操作系统中进程问题

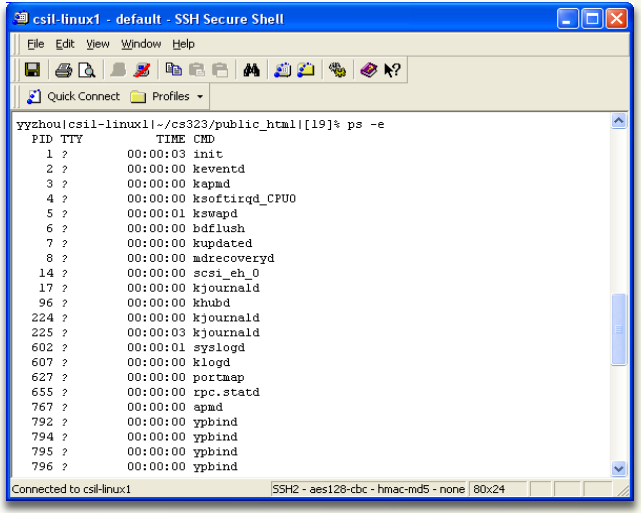
- 多道程序并发执行产生的问题：
  - CPU（处理器）如何分派给进程使用？
  - 资源如何分给各个进程？避免死锁
  - 进程之间通讯如何处理？

在操作系统中查看进程

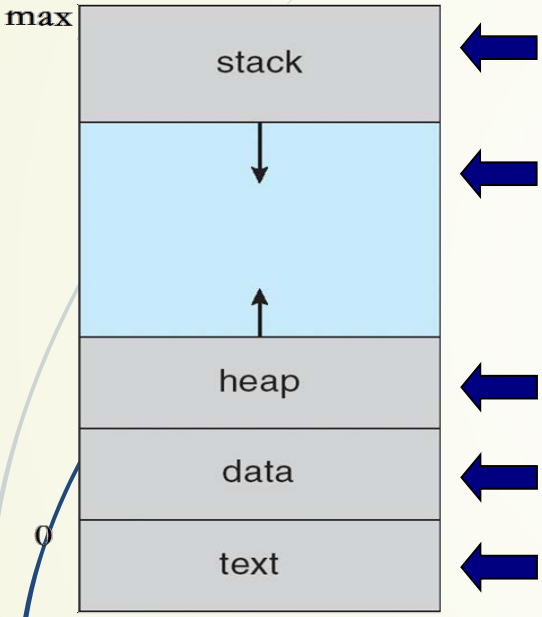
Windows任务管理器



Unix例子: ps



进程在内存中的形式



1. Stack containing temporary data(Function parameters, return addresses, local variables)
2. Current activity including program counter, processor registers
3. Heap containing memory dynamically allocated during run time
4. Data section containing global variables
5. The program code, called text section

进程的基础知识

- 进程更能真实地描述任务的并发执行.
- 进程由程序和数据两部分组成.
- 进程是动态的：进程有生命周期，有诞生、有消亡，是短暂的过程.
- 一个程序可以对应多个进程.
- 进程具有创建其它进程的功能

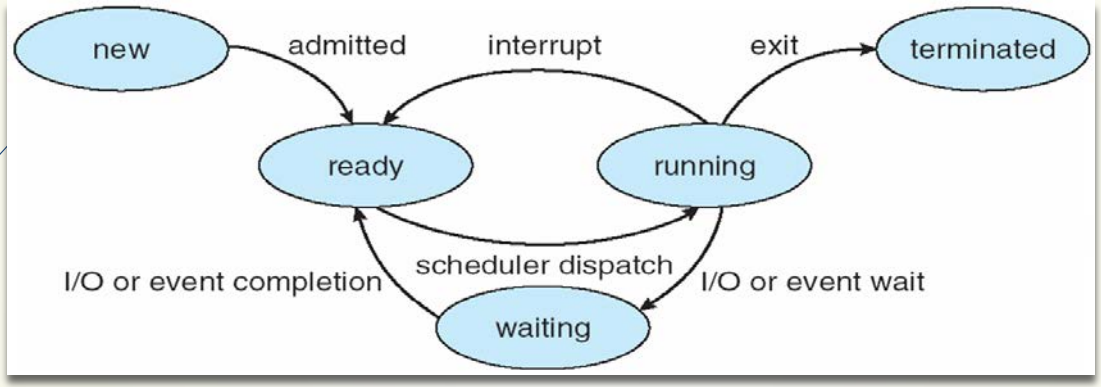
进程特征

- **结构特征**：进程实体由程序段、数据段和PCB构成；
- **动态性**：进程是程序的一次执行过程；
- **并发性**：多个进程同存于内存中，且在一段时间内同时运行，是进程的重要特征；
- **独立性**：进程是独立运行、独立分配资源的基本单位；
- **异步性**：进程按照独立的、不可预知的速度向前推进。
- **交互性**：进程之间需要交互

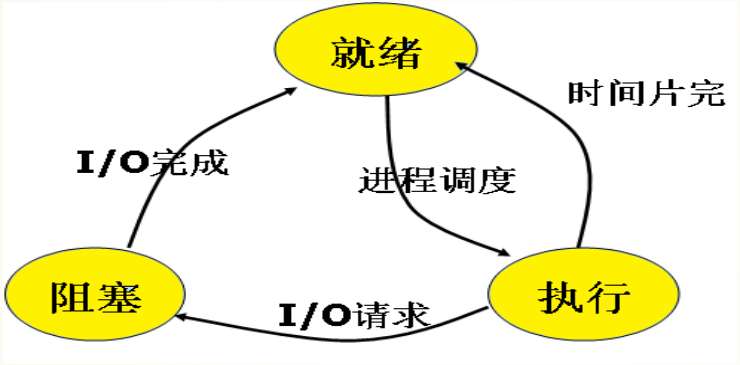
（2） 进程状态

- 进程是有生命期的，生命期内经历各个状态的变化：
  - 新建状态（new） --- 正准备创建进程
  - 运行状态（Running） --- 任务指令正在执行
  - 等待状态（waiting）--- 进程正在等待某个事件发生（被阻塞了Blocked）
  - 就绪状态（ ready）--- 进程正在等待被分配处理器
  - 终止状态（terminated ）--- 进程完成执行过程

进程状态转换图


new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting

进程状态转换图



进程的三种基本状态及其转换

（3）进程控制块（PCB，Process Control Block)

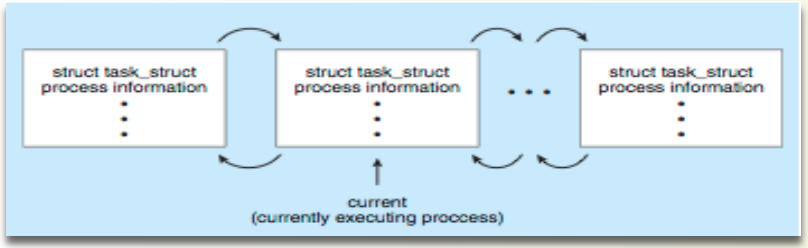
- PCB包含与进程相关的信息：
  - 进程的状态
  - 程序计数器
  - CPU寄存器
  - CPU调度信息
  - 内存管理信息
  - 资源计数信息
  - I/O状态信息

进程表达

- 利用C的结构体表达 task_struct

- task_struct成员：

```
pid t pid;                    /* 标识符 */
long state;                   /* 进程状态*/
unsigned int timeslice        /*调度信息*/
struct taskstruct *parent;    /* 父进程信息*/
struct  listhead children;    /*子进程信息 */
struct  filesstruct *files;   /*进程打开文件信息 */
struct  mmstruct *mm;         /* 当前程序的地址空间*/
```

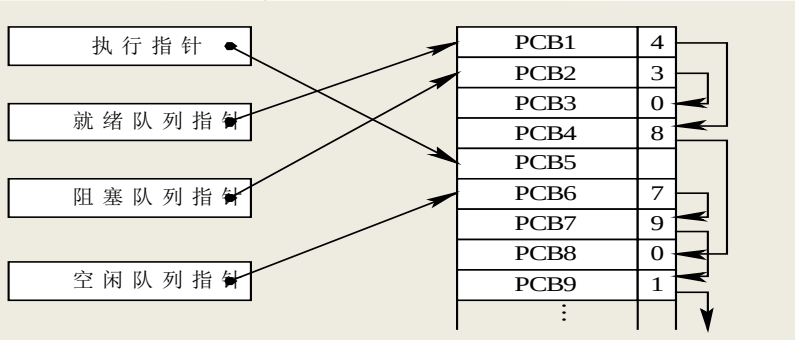


多个task_struct

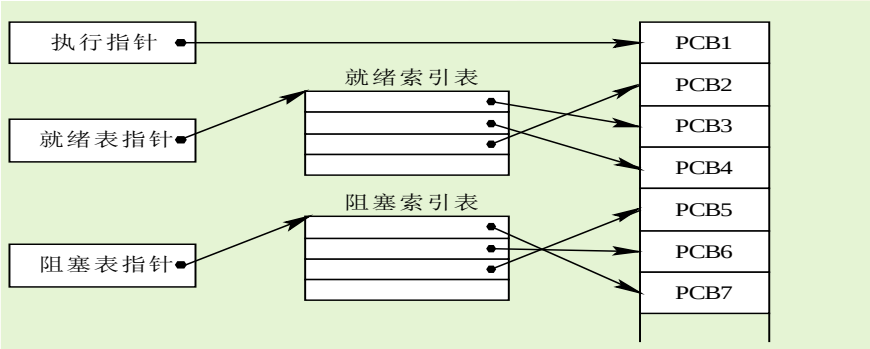
进程控制块PCB的管理（多个进程）

- 系统把所有PCB组织在一起，并放在内存固定区域，就构成了PCB表。
  - **链表：** 同一状态的进程其PCB成一链表，不同状态对应不同状态的链表
    各状态的进程形成不同的链表：就绪链表、阻塞链表
  - **索引表：** 同一状态的进程归入一个index表（由index指向PCB），不同状态对应不同索引
    各状态的进行形成不同的索引表：就绪索引表、阻塞索引表
  - **PCB表的规模**（多少个PCB）取决于系统中可同时存在的最多进程数，称为系统的并发度

PCB组织方式



PCB链接队列示意图



按索引方式组织PCB

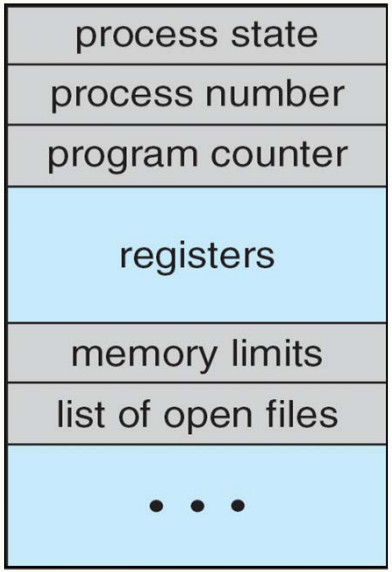
PCB成员信息

进程标识符

处理机状态

进程调度信息

进程控制信息



PCB主要成员信息

- **进程标识符**
  - 用于唯一地标识进程。
  - 一个进程通常有两种标识符：内部标识符（操作系统提供的进程的序号）和外部标识符（由创建者提供）
- **处理机状态**
  - 各种寄存器中的内容
  - 通用寄存器、指令计数器、程序状态字PSW、用户栈指针
  - 含有要访问下一条指令的地址、状态信息、栈指针等
- **进程调度信息**
  - 进程状态： 新建、执行等
  - 进程优先级：描述进程使用处理机优先级数
  - 其它与进程调度算法有关
  - 事件：进程状态转变发生的事件，阻塞原因
- **进程控制信息**
  - 程序和数据的地址
  - 进程同步和通信信息：消息队列指针、信号量等；
  - 链接指针：下一个进程PCB的首地址。

2. 进程的调度

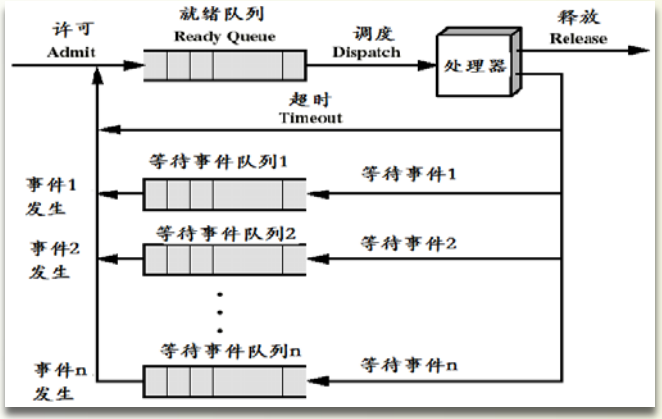
目标：

- 使CPU利用率最高
- CPU在各个进程之间快速切换（ ***Quickly*** switch ）

（1）进程调度的任务及队列

- 选一个进程，下一个CPU周期将占有CPU
- 需要维护进程的调度队列
  - 作业队列（Job queue）： 系统中所有进程的集合
  - 就绪队列（Ready queue ）：内存进程集合，等待执行
  - 设备队列（Device queues）：等待I/O 设备的进程

进程生命期内在各个队列中不断移动

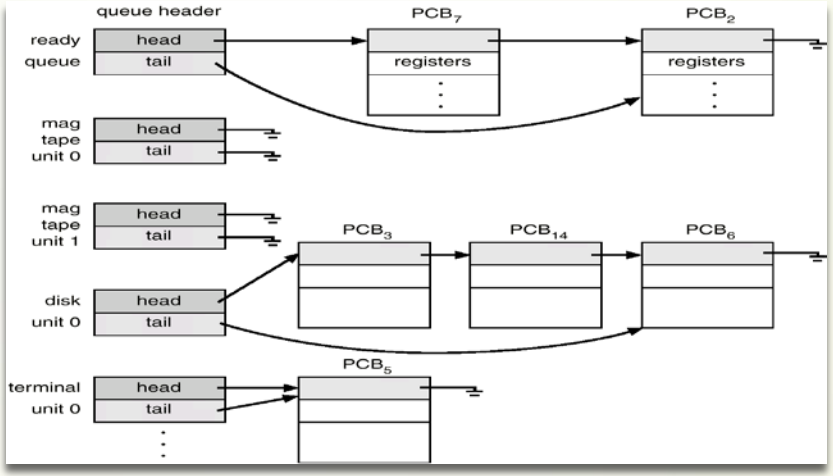


等待各种事件的队列

设备调度队列实例

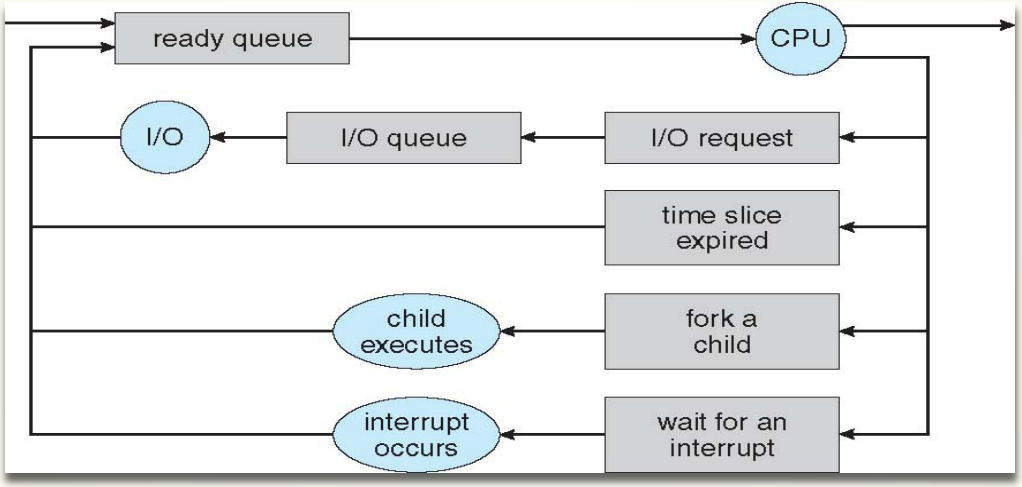
- 作业队列
- 就绪队列：linked list
- 设备队列

例如：设备队列……



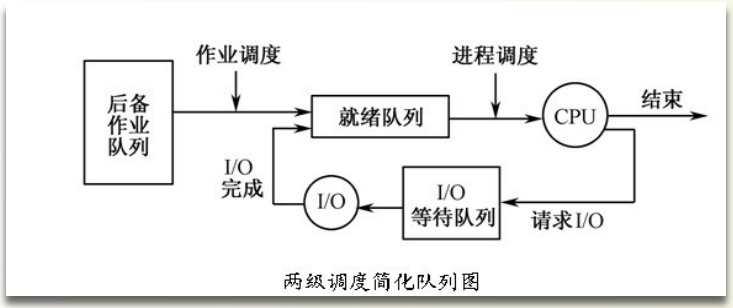
（2）进程调度的流程

调度示意图表示队列、资源、调度过程流程



进程调度类型

- 长期调度程序（调度器）或高级调度（Long-term scheduler ）
  - 从作业队列选择进程放入就绪队列
  - 作业调度或宏观调度，周期长-seconds, minutes
- 短期调度程序（调度器）或低级调度（Short-term scheduler ）
  - 从就绪队列选择进程即将使用CPU
  - 可能是系统中唯一的调度程序
  - 周期短 -milliseconds
- 中期调度（Medium-term scheduling）



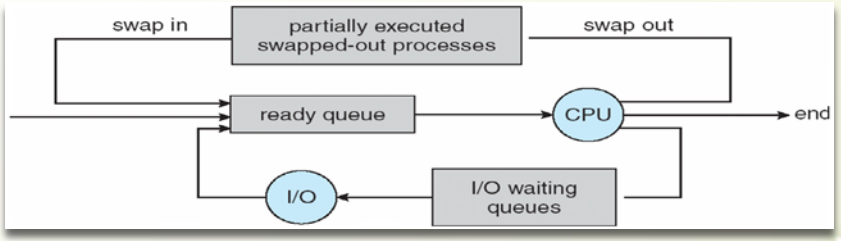
两级调度简化队列图

短期调度

- 短期调度即CPU调度，从内存中的就绪进程队列中选择一个进程来执行。
- 短期调度分为：
  - 抢占调度(Preemptive Mode)：允许暂停某个进程的执行重新分配处理机
    - 抢占的原则有：
      - 优先权原则
      - 时间片原则
  - 非抢占调度

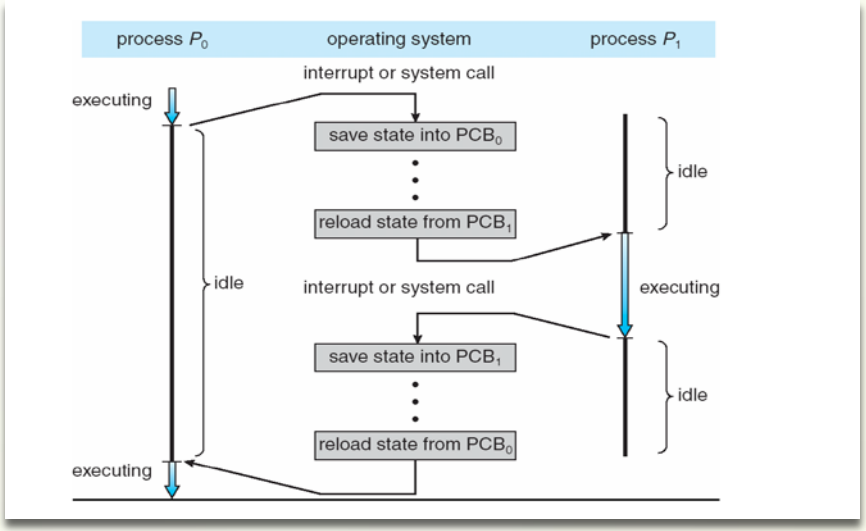
中期调度

1. 又称中级调度，或中程调度(Medium-Term Scheduling)。
2. 为了提高内存利用率和系统吞吐量
3. 使那些暂时不能运行的进程不再占用宝贵的内存资源，将它们调至外存上去等待，进程状态改为就绪驻外存状态或挂起状态
4. 重新调入内存：上述进程重新具备运行条件、且内存有空闲时，由中级调度来重新调入内存，并修改其状态为就绪状态，挂在就绪队列上等待进程调度
5. 上述过程被称为交换技术（swapping）



（3）进程之间的切换

- CPU从一个进程切换到另一个进程，称为上下文切换
- 上文状态保存（将硬件状态保存到PCB），调出下文状态（从PCB初始化到硬件中）
- 进程的状态信息保存在 PCB中
- 处理器从一个进程切换到另一个进程



进程切换的系统开销

- 上下文切换需要较多时间开销，切换过程中操作系统只能做无用工作，切换时间与硬件有关系
- 需要切换的上下文包括：
  - 各种寄存器，例如，通用寄存器、程序计数器PC、程序状态字寄存器PS等
  - 程序段在经过编译之后形成的机器指令代码集（或称正文段）、数据集
  - 各种堆栈值
  - PCB结构

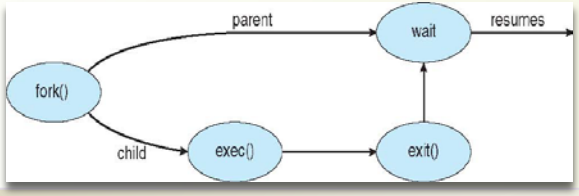
3. 进程的创建与终止

对于进程的常见操作包括：

- 创建进程
- 终止进程
- 阻塞与唤醒
- 挂起与激活

（1）创建进程(Process Creation)

- 父进程创建子进程，子进程进一步创建其它进程，形成进程树
- 进程采用标识符（pid）进行管理和标识



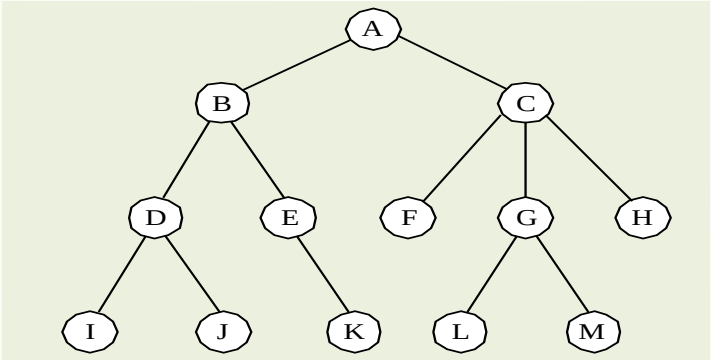
- **资源共享**问题：
  - 父子进程**共享所有的资源**
  - 子进程共享**父进程的部分资源**
  - 父子进程之间**无共享资源**

- **执行顺序**问题：
  - 父子进程同时进行
  - 父进程等待子进程，直到其结束后，再继续进行

- **地址空间**问题：
  - 子进程复制父进程的地址空间
  - 子进程进一步启动自己的任务

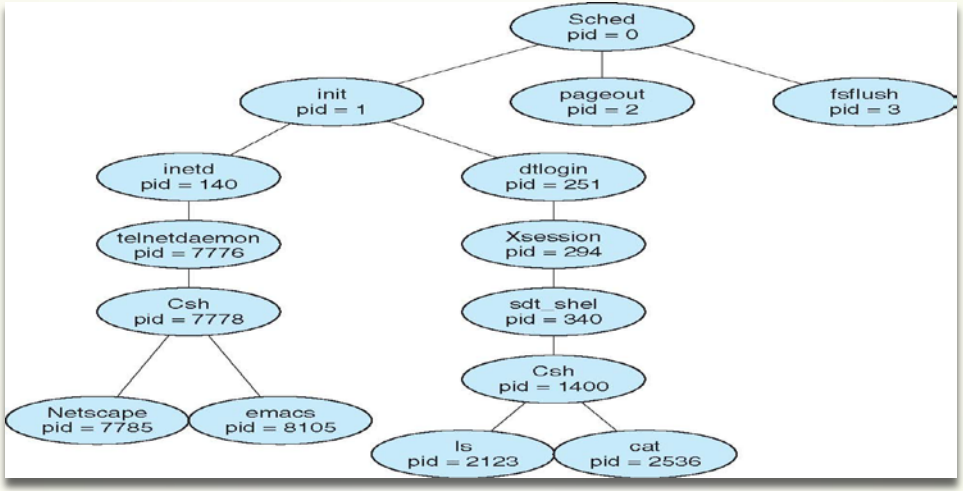
创建进程

- 进程图(Process Graph)的产生
- 父进程创建子进程，子进程进一步创建它的子进程，不断蔓延，产生进程树



进程树

Solaris系统中创建的进程树实例



进程树实例

引起进程创建的原因

(1) 一个新的用户登录

用户交互地登录（logon）

(2) 新的作业调度任务

新的一批工作启动

(3) 启动一个操作系统提供的服务

服务需要创建进程，例如: printing

(4) 应用请求:

用户程序自己创建进程, 利用 CreateProcess创建

创建进程的过程

- 申请空白PCB。
- 为新进程分配资源。
- 初始化进程控制块。
- 将新进程插入就绪队列，如果进程就绪队列能够接纳新进程， 便将新进程插入就绪队列。

创建进程的过程

- 申请空白PCB。
- 为新进程分配资源。
- 初始化进程控制块。
- 将新进程插入就绪队列，如果进程就绪队列能够接纳新进程， 便将新进程插入就绪队列。

创建进程实例

C Program Forking Separate Process

```
#include <sys/types.h>
#include <studio.h>
#include <unistd.h>
int main()
{
        pid_t  pid;
      /* fork another process */
      pid = fork();
      if (pid < 0) {
           /* error occurred */
      fprintf(stderr, "Fork Failed");
      return 1;
      }
else if (pid == 0)
          {     /* child process */
          execlp("/bin/ls", "ls", NULL);
     }
   else {          /* parent process */
         wait (NULL);
                  /* parent will wait for the
   child */
              printf ("Child Complete");
           }
return 0;
     }
```

（2）终止进程(Process Termination)

- 进程执行最后语句并要求操作系统删除它（ exit ）
  - 输出数据结果，从子进程到父进程（通过wait ）
  - 进程的资源由操作系统回收

终止进程(Process Termination)

- 父进程可以终止子进程的执行（ abort ）
  - 子进程已访问超过分配的资源范围
  - 子进程再没有要执行的任务
  - 父进程结束时，有些系统不允许子进程继续工作，按层级结束各级子进程

引起进程终止的原因

(1) 停止命令Halt

(2) 用户注销

(3) 用户程序退出

(4) 程序错误（例如没有可用的资源）

(5) 正常完成任务

(6) 超过时间限制

(7) 内存不可用

引起进程终止的事件

- 任务正常结束，进程已经运行完成
  - ✓ 例如，在批处理系统中，在最后一条指令Halt系统调用时，产生一个中断，去通知OS本进程已经完成
  - ✓ 在分时系统中，用户可利用Logs off表示进程运行完， 此时同样可产生一个中断，去通知OS进程已运行完毕

引起进程终止的事件

- 任务异常结束。出现某些错误和故障而迫使进程终止
  - 越界错误，这是指程序所访问的存储区，已越出该进程的区域；
  - 保护错，进程试图去访问一个不允许访问的资源或文件
  - 非法指令，程序试图去执行一条不存在的指令。
  - 特权指令错，用户进程试图去执行一条只允许OS执行的指令；
  - 运行超时，进程的执行时间超过了指定的最大值；
  - 等待超时，进程等待某事件的时间， 超过了规定的最大值；
  - 算术运算错，进程试图去执行一个被禁止的运算，例如，被0除
  - I/O故障，这是指在I/O过程中发生了错误等。

引起进程终止的事件

- 外界干预。进程应外界的请求而终止运行
  - 操作员或操作系统干预。
  - 父进程请求。 由于父进程具有终止自己的任何子孙进程的权利， 因而当父进程提出请求时，系统将终止该进程；
  - 父进程终止。 当父进程终止时，OS也将他的所有子孙进程终止。

进程的终止过程

假设要终止进程为P:

- 根据P的标识符，从PCB队列中检索出P进程PCB，读出进程状态。
- 若P为执行状态，立即终止P，置调度标志为真，用于指示需要重新调度
- 若P有子孙进程，将它们予以终止
- 将P拥有的资源，或者归还给其父进程， 或者归还给系统
- 将 P(它的PCB)从队列(或链表)中移出

4. 进程的阻塞与唤醒

引起进程阻塞和唤醒的事件

- 请求系统服务
- 启动某种操作
- 新数据尚未到达
- 现有进程任务完成，无新工作可做

（1）进程阻塞过程

- 调用阻塞原语block把自己阻塞
- 进入block过程后，应先立即停止执行，把进程控制块中状态由“执行”改为阻塞
- 将PCB插入阻塞队列。应将本进程插入到具有相同事件的阻塞(等待)队列
- 将处理机分配给另一就绪进程，并进行切换，处理机状态保留到PCB中，，再按新进程的PCB信息重置CPU状态

（2）进程唤醒过程

- 当被阻塞进程所期待事件出现时，如I/O完成或其所期待的数据已经到达
- 某进程用完I/O设备调用唤醒原语wakeup，将等待该事件的进程唤醒
- 唤醒过程：把被阻塞进程从阻塞队列移出，将其PCB中的现行状态由阻塞改为就绪，再将该PCB插入到就绪队列中

（3）进程的挂起

- 当出现引起进程挂起的事件时，比如，用户进程请求将自己挂起
- 或者父进程请求将自己的某个子进程挂起， 系统将利用挂起原语suspend( )将指定进程或处于阻塞状态的进程挂起。

（4）进程的激活

- 当发生激活进程的事件时
- 例如，父进程或用户进程请求激活指定进程，若该进程驻留在外存而内存中已有足够的空间时，则可将在外存上处于静止就绪状态的进程换入内存。系统将利用激活原语active( )将指定进程激活。

作业

- **Assignment2-1**