2、 操作系统在计算机系统中的地位是什么？

地位：操作系統是计算机硬件之上的第一层软件，它本身也是系统软件。它提供了计算机硬件与其他系统软件和应用之间的接口——系统软件与应用程序通过调用操作系统来使用硬件提供的功能和服务。

作用：操作系统是计算机开机后先于其他软件运行的系统软件，它不仅提供了用户对计算机中资源的集中管理、使用和维护，也提供了计算机中软件运行的组织和协调

操作系统为用户提供的接口有哪几种？分别有什么特点？

操作系统为用户提供两个接口：命令接口和系统调用。

命令接口：系统为用户提供的。利用操作命令来组织和控制作业的执行或管理计算机系统。

系统调用：编程人员使用系统调用。请求操作系统提供服务，例如申请和释放外设等类资源、控制程序的执行速度等。

1.系统调用和API的区别：

系统调用是**内核**为用户提供的一系列接口，系统调用把应用程序的请求推给内核函数处理。把处理结果返回给应用

API是一些预定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码。API 函数通常为应用程序员调用实际的系统调用。

两者都可以被开发人员使用来访问内核的底层功能，而程序员更喜欢根据 API 来编程。其主要原因有：

（1）程序的可移植性。程序员根据 API 设计程序，以希望程序能在任何支持同一 API 的系统上编译并执行；但是直接使用系统调用，无法做到直接移植。

（2）对程序员而言，实际系统调用比 API 更为注重细节且更加难用。

尽管如此，在 API 的函数和内核中的相关系统调用之间常常还是存在紧密联系的。事实上，许多 POSIX 和 Windows 的 API 还是类似于 UNIX、Linux 和 Windows 操作系统提供的系统调用。

2.系统调用的工作过程：

运行时支持系统提供了系统调用接口，以链接到操作系统的系统调用，系统调用接口截取了API函数的调用，并调用操作系统中的所需系统调用。通常，每个系统调用都有一个相关数字，而系统调用接口会根据这些数字建立一个索引列表，系统调用接口就可调用系统内核中的所需系统调用，并返回系统调用状态和任何返回值。

3. 操作系统结构优缺点

①整体式结构

优点：接口简单直接，系统效率高。

缺点：可读性较差，也不具备可维护性。一旦某一个过程出了问题，凡是与之存在调用关系的过程都要修改，所以给调试和维护人员带来许多麻烦。

②层次式结构成

优点：（1）把功能实现的无序性改有序性，可显著提高设计的准确性。

（2）把模块问的复杂依赖关系改为单向依赖关系，即高层软件依赖于低层软件。

缺点：分层法的主要困难涉及对层的详细定义，这是因为一层只能使用其下的较低层。

③微内核结构：工作原理：最基本的功能保留在内核，而将那些不需要在核心态执行的功能移到用户态执行，从而降低了内核的设计复杂性。而那些移出的操作系统代码根据分层的原则被划分成若干服务程序，它们的执行相互独立，交互则都借助于微内核进行通信。

优点：有效地分离了内核与服务、服务与服务，使得它们之间的接口更加清晰，维护的代价大大降低，各部分可以独立地优化和演进，从而保证了操作系统的可靠性。

缺点：因为需要频繁地在核心态和用户态之间进行切换，操作系统的执行开销偏大。

④模块化结构——面向对象编程技术

优点：1）将内核映象的尺寸保持在最小，并具有最大的灵活性；2）便于检验新的内核代码，而不需重新编译内核并重新引导。

缺点：1）差的内核模块会导致系统崩溃 2）内核必须维护符号表（内核符号表是一个用来存放所有模块可以访问的那些符号以及相应地址的特殊的表）3）内核要维护模块之间的依赖性。

4.指令执行两种模式的特点：

内核模式：在内核模式下,执行代码具有对底层硬件的完全且不受限制的访问.它可以执行任何CPU指令并引用任何内存地址.内核模式通常保留给操作系统最受信任的功能.内核模式下的崩溃是灾难性的， 将停止整个PC.

用户模式：在用户模式下,执行代码无法直接访问硬件或参考内存.以用户模式运行的代码必须委托给系统API才能访问硬件或内存.由于这种隔离所提供的保护,用户模式下的崩溃始终是可恢复的.

5.进程和线程的区别和联系

①在引入了线程的系统中，线程是作为执行和调度的基本单位，而进程作为资源分配的基本单位，同一进程中的线程切换不会引起进程切换

②在引入了线程的系统中，不仅进程之间可以并发执行，而且一个进程的多个线程之间也可以并发执行，因此可以更有效地使用系统资源，具有更好的并发性。

③进程是拥有资源的独立单位，而线程只拥有少量的资源，但它可以访问其隶属进程的资源，线程创建、终止、切换时的系统开销比进程小

④由于同一进程的线程共享内存和文件，因此在同一个地址空间里，它们之间的通信无需调用内核，从而提高了通信的效率

6.写时复制

父进程创建子进程时，最初父子进程共享内存空间。等到子进程修改数据时才真正分配内存空间，这是对程序性能的优化，可以延迟甚至是避免内存拷贝，当然目的就是避免不必要的内存拷贝。

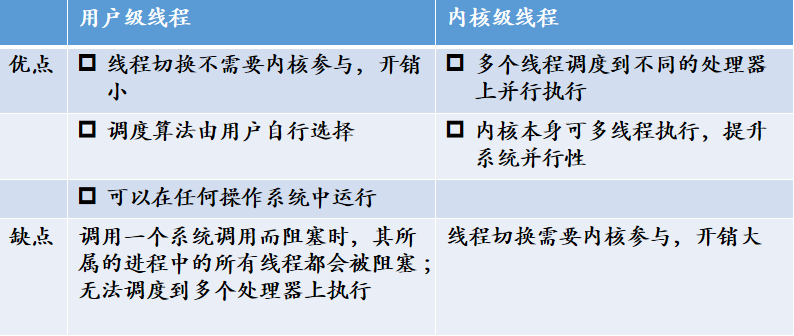
12.用户级线程和内核级线程 定义和优缺点比较

1.内核级线程

线程管理的所有工作都是由内核完成的。用户通过操作系统给应用程序提供的应用程序编程接口API来进行线程管理。

2.用户级线程

线程管理的所有工作都是由应用程序完成的，内核意识不到线程的存在，内核以进程为单位进行调度。操作系统提供给用户一个线程库对线程进行操作。



6.设备数据传输控制方法——原理、优缺点

①轮询方式：轮询方式也称为程序直接控制方式，该方式采用用户程序直接控制主机与外部设备之间进行输入/输出操作。CPU必须不停地循环测试I/O设备的状态端口，当发现设备处于准备好(Ready)状态时，CPU就可以与I/O设备进行数据存取操作。

优点：

简单、易控制，控制器的外围接口控制逻辑少。

缺点：

CPU只能与外设进行串行工作、效率低、速度慢。

适合于慢速设备工作方式

② 中断控制方式的基本使得CPU向I/O控制器发出I/O命令思想是：引入中断处理机构，将轮询方式中的CPU最大限度地解放出来，后能够调度其他进程执行，无需空转轮询，提升CPU与外设之间的并发执行能力。

优点：

极大地提升了CPU的利用效率。初步实现了CPU与外设的并行工作。

缺点：

仍旧会消耗大量的CPU时间。容易造成数据丢失的现象。

③DMA方式是直接内存访问，是一种不经过CPU而直接从内存存取数据的数据传输模式。中断模式下硬盘和内存之间的数据传输是由CPU来控制的；而在DMA模式下，CPU只须向DMA控制器下达指令，让DMA控制器来处理数据的传送，数据传送完毕再把信息反馈给CPU，这样就很大程度上减轻了CPU资源占。

优点：

传输速率快，数据不易丢失，用于传输大批量的数据，总体效果较好

缺点：

用于内存连续存放数据的传输；对于不连续存放的数据，需要启动多次DMA过程

适用于高速设备

④通道本质上是一个简单的处理器，专门负责输入、输出控制，具有执行I/O指令的能力，并通过执行通道I/O程序来控制I/O操作。

7. 缓冲区的引入

（1）缓解设备之间速度差异的矛盾。

（2）缓冲区可以缓解设备之间传输数据大小不一致的矛盾

（3）支持应用程序I/O的复制语义

8.什么是设备独立性？

“独立”两字体现在于：应用程序独立于具体使用的物理设备。为了实现设备独立性而引入了逻辑设备和物理设备这两个概念。

10.进程控制块的地位和作用

进程控制块是进程存在的唯一标志，是存在系统进程管理中一种重要的数据结构

PCB存放进程的管理和控制信息，操作系统通过进程控制块PCB感知到进程的存在.

4.虚拟机的概念

指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。

虚拟机监视器，虚拟化技术的核心

7.虚拟机的实现原理是什么

每一台虚拟机都是在 Hypervisor 的基础上建立起来的。Hypervisor 是一种运行在物理服务器和操作系统之间的中间软件层，可允许多个操作系统和应用共享一套基础物理硬件。Hypervisor 是所有虚拟化技术的核心。但执行Hypervisor 时，它会给每一台虚拟机分配适量的内存、CPU、网络和磁盘，并加载所有虚拟机的客户操作系统。

11.虚拟机的作用

①VM可以在同一台计算机上完成软件针对不同操作系统的测试过程。

②java虚拟级屏蔽了与具体操作平台相关的信息。

1. 在CPU调度中，openEuler为什么没有采用最短进程优先（Shortest Job First, SJF）算法？

参考答案：

基于以下两个原因，openEuler没有采用SJF算法：(1) SJF算法会导致运行时间长的进程响应时间越来越长，从而产生进程饥饿现象；(2) 因为很难预估每个进程的运行时间，SJF算法在实现上存在难点。

2、 什么原因引起了死锁？产生了死锁的四个条件是什么？

引起死锁的原因是资源竞争，死锁的发生必须具体以下四个条件：

（1）互斥条件。指进程的共享资源必须保持使用的互斥性，即任何一个时刻只能分配给一个进程使用，互斥条件是形成死锁最根本的原因，因为如果资源不要求排它性地使用，那么一定不会造成请求资源而无法满足的局面。

（2）占有且等待条件。一个进程占有了某些资源之后又要申请新的资源而得不到满足时，处于等待资源的状态，且不释放已经占用的资源。

（3）不可剥夺条件。任何进程不能抢夺另一个进程所占用的资源，即已经被占用的资源只能由占用进程自己来释放。

（4）环路条件。存在一组进程P1,P2,…,Pn，其中每个进程分别等待另一个进程所占用的资源，形成环路等待条件。



7、有三个并发进程input,copy,output，input进程负责从输入设备读入信息并把信息放入到缓冲区buffer1中，copy进程负责把buffer1中的内容加工后复制到缓冲区buffer2中；output进程负责从buffer2中取出信息并送到打印机中输出，请使用WAIT，SIGNAL操作写出上述三个进程的同步算法。

EmptyB1、EmptyB2表示buffer1与buffer2缓冲区中的空闲单元的数目，

其初值为 N1与N2

Full\_B1、Full\_B2表示buffer1与buffer2满缓冲单元的数目，其初值为 0

output进程：

Repeat

Wait(Full\_B2);

从buffer2中取出一个数据Y；

打印Y；

Signal(EmptyB2);

Until false;

copy进程：

Repeat

Wait(EmptyB2);

Wait(Full-B1);

从buffer1中取出一个数据；

将该数据拷贝到buffer2中；

Signal(Full\_B2);

Signal(EmptyB1);

Until false;

Input进程：

Repeat

读入信息X；

Wait(EmptyB1);

X存入Buffer1中；

Signal(Full\_B1);

Until false;

12. Belady现象原因和解决（好好了解下）

实页数增加 —> 能贮存的页数增加 —> 哪些页？—> 后面来的页

先进先出的替换算法，完全不考虑使用频率，即使增加了实页数，多贮存的部分接下来常访问可能性也不一定大（看运气），也就并不一定能增加命中率。

根据场景选择合适的算法

13.文件系统为什么要区分逻辑结构、物理结构

逻辑结构：指一个文件在用户面前所呈现的形式。又称文件组织。

物理结构：指文件在外存上的存储组织形式。这不仅和存储介质的存储性能有关，还与所采用的外存分配方式有关。

逻辑结构主要关注数据元素之间的关系，而物理结构则关注数据的存储方式。

文件的逻辑结构是用户可见的结构，逻辑文件从结构上分成二种形式：

(1)无结构的流式文件，是指对文件内信息不再划分单位，它是依次的一串字符流构成的文件。

(2)有结构的记录式文件, 是用户把文件内的信息按逻辑上独立的含义划分信息单位，每个单位称为一个逻辑记录（简称记录）。

1）堆结构

堆结构是最简单的文件组织形式 。文件中的记录可以长短不一，每条记录可以拥有不同的属性。这些记录的组织方式是按照生成的先后顺序排列而成。也就是说，记录的排列顺序与记录内容是无关的。

堆结构的文件有利于记录的追加，能够较好地用于穷举查找。但是如果需要按照关键字查找某条记录时，只能搜索文件的全部内容，效率会很差。因此这种组织方式对于大部分文件都是不适用的。

2）顺序结构的文件

顺序结构是一种最常见的文件组织形式。记录在文件中按照某个规则进行排序，通常按照某个关键字（如按照数字或字母顺序）排序。如果需要频繁地对文件中的记录进行查找、增加或删除操作，则可以采用这样的结构来组织文件。

3）散列结构的文件

如果考虑到搜索时间复杂度，采用散列结构来组织文件是一种较好的方法。

散列文件使用散列函数将记录的关键字值经过计算转化为记录的逻辑号（也可理解为记录的编号）。

物理结构

1）连续文件

将一个文件中逻辑上连续的信息存放到存储介质的依次相邻的物理块上便形成顺序结构，这类文件叫连续文件，又称顺序文件。

优点：

1）连续文件既支持顺序读写也支持随机读写。

2）大量的顺序读写效率高（连续文件续写使得磁头移动距离少）

缺陷：

1）建立文件前需要能预先确定文件长度。通常而言，输出文件的大小难以事前预估，无法使用连续文件的物理结构模式。

2）修改、插入和追加文件记录有困难；

3）会造成少量外部碎片。

2）链接文件

一个文件的信息存放在若干不连续的物理块中，各块之间通过指针链接，前一个物理块中有一个指针域，指向下一个物理块，最后的物理块的指针域存放符号“∧”，表示无后继块。这样的文件称为链接文件。

3）索引文件

一个文件的信息存放在若干不连续物理块中，系统为每个文件建立一个专用数据结构----索引表，表中每一栏目指出文件信息所在的逻辑块号和与之对应的物理块号。索引表的物理地址则由文件说明信息项（文件目录）给出。

5、某系统A,B,C,D这4类资源供5个进程共享，进程对资源的需求和分配情况如表所示。现在系统还剩资源A类1个，B类5个，C类2个和D类0个，按照银行家算法回答下面问题：

|  |  |  |
| --- | --- | --- |
| 进程 | 已占有资源数目 | 最大需求数 |
| A B C D | A B C D |
| P1 | 0 0 1 2 | 0 0 1 2 |
| P2 | 1 0 0 0 | 1 7 5 0 |
| P3 | 1 3 5 4 | 2 3 5 6 |
| P4 | 0 6 3 2 | 0 6 5 2 |
| P5 | 0 0 1 4 | 1. 6 5 6 |

1）现在系统是否处于安全状态？

2）如果现在进程P2提出需要(0,4,2,0)个资源的请求，系统能否去满足它的请求？

解：现在可用资源向量为：(1,5,2,0)，计算出每个进程仍需资源的矩阵如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 进程 | 已占有资源数目 | 最大需求数 | 尚需资源数 |
| A B C D | A B C D | A B C D |
| P1 | 0 0 1 2 | 0 0 1 2 | 0 0 0 0 |
| P2 | 1 0 0 0 | 1 7 5 0 | 0 7 5 0 |
| P3 | 1 3 5 4 | 2 3 5 6 | 1 0 0  2 |
| P4 | 0 6 3 2 | 0 6 5 2 | 0 0 2 0 |
| P5 | 1. 0 1 4 | 0 6 5 6 | 0 6 4 2 |

在那个地址,进程P1已经拥有足够资源,在执行完后可归还C和D资源,同时,进程P3能够取得足够的资源完成执行,其后,P2、P4、P5能够依次取得足够的资源来完成,因此,用银行家算法分派资源时,系统出于平安状态。安全序列为:P1,P3,P2,P4,P5

（2）P2 进程提出资源请求 (0,4,2,0),采用银行家算法，假设资源获得分配，则相关矩阵如下表：现在可用资源向量为：(1,1,0,0)

|  |  |  |  |
| --- | --- | --- | --- |
| 进程 | 已占有资源数目 | 最大需求数 | 尚需资源数 |
| A B C D | A B C D | A B C D |
| P1 | 0 0 1 2 | 0 0 1 2 | 0 0 0 0 |
| P2 | 1 4 2 0 | 1 7 5 0 | 0 3 3 0 |
| P3 | 1 3 5 4 | 2 3 5 6 | 1 0 0  2 |
| P4 | 0 6 3 2 | 0 6 5 2 | 0 0 2 0 |
| P5 | 1. 0 1 4 | 0 6 5 6 | 0 6 4 2 |

P1进程可执行完毕，然后释放资源，则可用资源向量为：(1,1,0,0)+(0,0,1,2)=(1,1,1,2)

P3进程可执行完毕，然后释放资源，则可用资源向量为：(1,1,1,2)+(1,3,5,4)=(2,4,6,6)

P2进程可执行完毕，然后释放资源，则可用资源向量为：(2,4,6,6)+(1,4,2,0)=(3,8,8,6)

P4进程可执行完毕，然后释放资源，则可用资源向量为：(3,8,8,6)+(0,6,3,2)=(3,14,11,8)

P5进程可执行完毕，然后释放资源，则可用资源向量为：(3,14,11,8)+(0,0,1,4)=(3,14,12,12)

此时系统是安全状态，所以P2进程的请求是可以分配的。

8、对于下列三个进程，采用不可强占的调度方式，当第一个进程进入系统后开始调度，它们的提交时间及执行时间由下表给出。分别计算采用先来先服务，最短作业优先和最高响应比调度算法时它们的调度顺序和平均周转时间。

|  |  |  |
| --- | --- | --- |
| 进程号 | 到达队列时间（时） | 需运行时间（小时） |
| 1 | 10：00 | 2：00 |
| 2 | 10：10 | 1：00 |
| 3 | 11：50 | 0：50 |

解：（1）先来先服务调度算法：

执行顺序：1----2-----3

平均周转时间：（120+170+120）/3 = 137分钟

（2）最短作业优先

执行顺序：1----3-----2

平均周转时间：（120+220+60）/3 = 133分钟

（3）最高响应比

进程1最先执行，当12时执行完毕，此时

进程2的响应比：（60+110）/60=28.3

进程3的响应比：（50+10）/50=1.2

因此进程2先执行

执行顺序：1----2-----3

1. 某操作系统的磁盘文件空间共有500块，若用字长为32位的位示图管理盘块空间，试问：

(1)位示图需要多少个字？

(2)如果b（盘块号）,i，j从1开始计数，第i字第j位对应的块号是多少？

(3)如果b（盘块号）,i，j 从0开始计数，第i字第j位对应的块号是多少？

（1）位示图需要的字数计算：INT（500/32）=16个字。

（2）块号b=(i-1)\*32+j

（3）块号b=i\*32+j

简答题4道

综合题3道：

银行家算法、进程调度算法、LRU页面置换算法

编程题：同步互斥

进程的四个特性 动态性、并发性、独立性、异步性

引起进程切换的事件有：中断 陷阱 系统调用

操作系统的目标包括：方便性 高效性 易于维护 开放性

机制与策略

区别对待机制（mechanism）和策略（policy）是操作系统设计的一种非常有效的思路。大部分的编程问题都可以被切割成两个部分：

* “需要提供什么功能（机制）（what）”
* “怎样实现这些功能（策略）（how）”

基本状态：运行态 阻塞态 就绪态

处理器的执行模式分为系统态和用户态

为了提供对互斥的支持，系统必须满足以下条件： 互斥 有限等待 空闲让进

使用硬件实现互斥中断禁用

专用机器指令

信号量实现互斥

管程

进程之间的高级通信机制可以大体分为以下几类：

共享内存方式

管道通信机制

消息传递通信 send原语和receive原语

银行家算法不需要破坏死锁形成的四个条件，所以比预防死锁的限制要少，但是在使用中也有以下一些限制：

* 1. 必须事先声明每个进程的资源最大需求量。
  2. 进程之间必须是无关的，即进程之间的执行顺序没有任何的同步要求；
  3. 系统中可供分配的资源数目必须是固定的
  4. 进程在占有资源的，不能退出。

CPU 使用率 吞吐量 周转时间 等待时间 响应时间

短程调度程序也叫进程调度，将决定哪一个就绪进程将获得处理器。进程调度可能发生在3个时机：

1）一个处于运行态的进程结束时，从就绪进程队列中选择一个就绪进程运行。

2）当一个正在运行的进程阻塞时，短程调度程序也将从就绪进程队列中选择一个就绪进程运行。

3）在一个I/O中断发生时，短程调度程序必须做出调度决策。如果中断来自I/O设备，则等待该I/O设备的进程状态被改为就绪。这时是让中断前的进程继续运行，还是立刻调度新的就绪进程运行，或调度其它别的进程运行，取决于系统采用的调度算法。

静态地址重定位 动态重定位

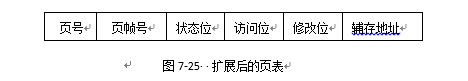
静态链接 装入时动态链接 运行时动态链接

单一连续分配方案 固定大小的分区方案 大小不等的分区方案 动态分区的分配方案

防止地址越界 使用上、下界寄存器 基址-限长存储保护

防止操作越权 访问控制字段

局部性原理是指程序在执行过程中的一个较短时间内，所执行的指令地址或操作数地址分别局限于一定的内存区域中。



页面缓冲算法 空闲页面链表 修改页面链表

解决系统抖动的方法1）工作集策略2）缺页率策略

自底向上的文件结构组成：（1）数据项（2）记录。（3）文件。

文件目录管理的基本功能包括：

(1) 实现“按名存取”。

(2) 快速检索文件目录。

(3)实现文件共享。

(4) 允许文件重名。

文件需要进行合理的保护。可以通过以下方式实现：

* 口令保护
* 加密保护
* 存储控制矩阵
* 存取控制表

虚拟文件系统（VirtualFileSystem,VFS）隐藏了各种硬件的具体细节，把文件系统操作和不同文件系统的具体实现细节分离开来，为所有的设备提供了统一的接口，可支持多达数十种不同的文件系统。

设计设配分配策略中，需要考虑许多如下一些相关因素：

1设备的固有属性

2.系统采用的分配算法

3.设备分配的安全性

4.设备的无关性

设备控制表DCT

系统为每个设备配置一张设备控制表，用于记录本设备的情况，如设备类型，设备标识号、设备状态。设备队列等信息。

2）控制器控制表COCT

系统为每个控制器设备一张用于记录本控制器情况的控制器控制表。

3）系统设备表 SDT

记录系统中全部设备的情况，每个设备占一个表目，包括设备类型、设备标识符、设备控制表，设备驱动程序入口等。在配有通道控制器系统的设备管理中，还要有通道控制表CHCT，用来记录通道的特性、状态及其他管理信息。

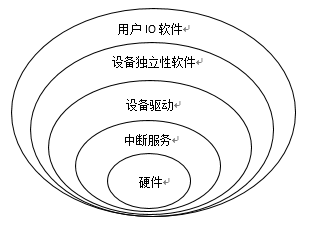
设备分配过程

1．分配设备

2．分配控制器

3．分配通道

I/O软件设计的原则:效率和通用性



柱面、磁头、扇区

磁盘读写时间=寻道时间+延迟时间+传输时间

2. 简述openEuler在CPU调度时是如何把先进先出、轮转调度和优先级调度结合起来的？

参考答案：

(1) openEuler为每个优先级维护一个进程链表，调度时，从优先级最高的链表中选择一个进程占用CPU；如果优先级最高的链表中没有进程，调度程序从次高优先级的链表中选择一个进程占用CPU，以此类推。(2) 同时，openEuler的每个进程都有一个调度策略：a. 如果调度程序选择的进程使用FIFO调度策略，那么该进程会一直霸占CPU直至运行完成或者被更高优先级的进程抢占或者该进程自己发生阻塞；b. 如果被选择的进程使用Round-Robin（RR）调度策略，则其将运行指定的时间片，除非发生阻塞或提前运行结束。如果该进程用完时间片后还不能结束，调度程序会把它添加到进程优先级对应的链表尾部，然后把CPU让给优先级相同的其他进程。

