# 操作系统

内存管理

1. 内存连续分配管理机制：

单一连续分配

固定分区-

动态分区-

1. 内存非连续分配管理机制

分页分配管理机制

分段-

缓存机制

先进先出

最近最久未使用

最近最少使用

时钟算法

进程和线程的区别

## 进程间的通信

**共享存储**

在通信的进程之间存在一块可直接访问的共享空间，通过对这片共享空间进行写/读操作实现进程之间的信息交换。在对共享空间进行写/读操作时，需要使用同步互斥工具（如 P操作、V操作），对共享空间的写/读进行控制。共享存储又分为两种：低级方式的共享是基于数据结构的共享；高级方式则是基于存储区的共享。操作系统只负责为通信进程提供可共享使用的存储空间和同步互斥工具，而数据交换则由用户自己安排读/写指令完成。

需要注意的是，用户进程空间一般都是独立的，要想让两个用户进程共享空间必须通过特殊的系统调用实现，而进程内的线程是自然共享进程空间的。

**消息传递**

在消息传递系统中，进程间的数据交换是以格式化的消息(Message)为单位的。若通信的进程之间不存在可直接访问的共享空间，则必须利用操作系统提供的消息传递方法实现进程通信。进程通过系统提供的发送消息和接收消息两个原语进行数据交换。

直接通信方式：发送进程直接把消息发送给接收进程，并将它挂在接收进程的消息缓冲队列上，接收进程从消息缓冲队列中取得消息。

间接通信方式：发送进程把消息发送到某个中间实体中，接收进程从中间实体中取得消息。这种中间实体一般称为信箱，这种通信方式又称为信箱通信方式。该通信方式广泛应用于计算机网络中，相应的通信系统称为电子邮件系统。

**管道通信**

管道通信是消息传递的一种特殊方式。所谓“管道”，是指用于连接一个读进程和一个写进程以实现它们之间通信的一个共享文件，又名pipe文件。向管道（共享文件）提供输入的发送进程（即写进程），以字符流形式将大量的数据送入（写）管道；而接收管道输出的接收进程（即读进程），则从管道中接收（读）数据。为了协调双方的通信，管道机制必须提供以下三方面的协调能力：互斥、同步和确定对方的存在。

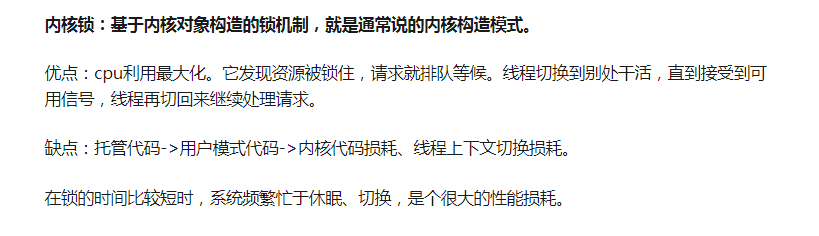
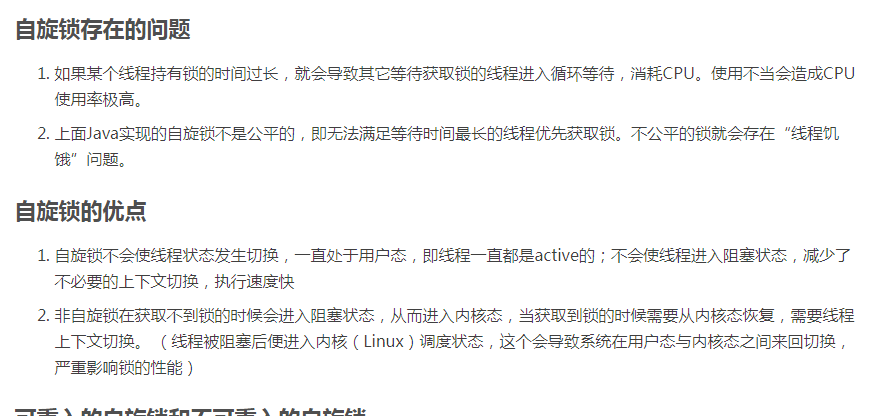
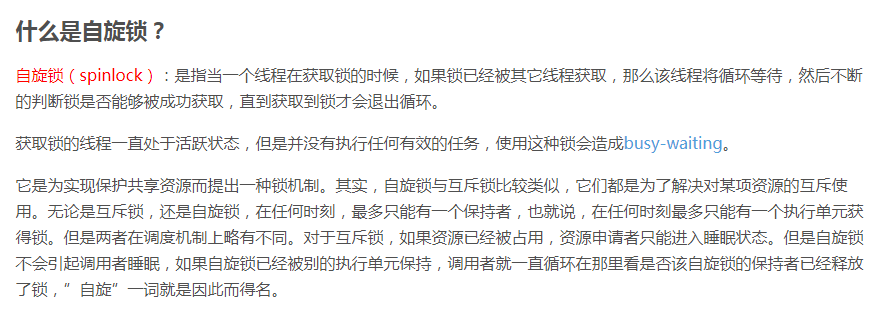
# 集合

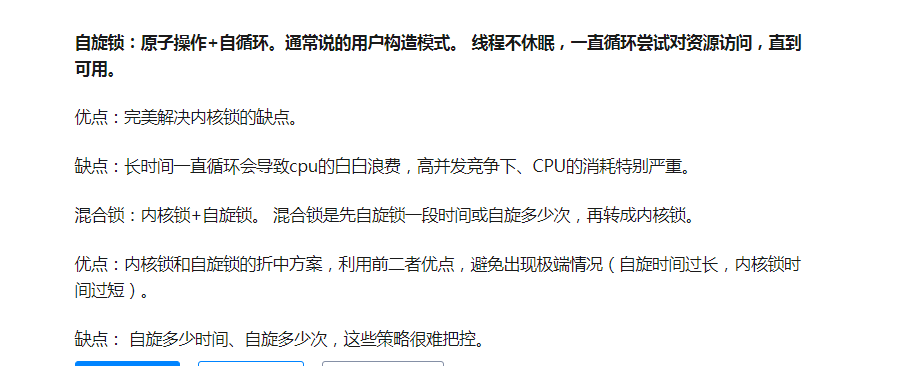
# 多线程

## Threadlocal

https://blog.csdn.net/wuyangyang555/article/details/82774693

## 自旋锁



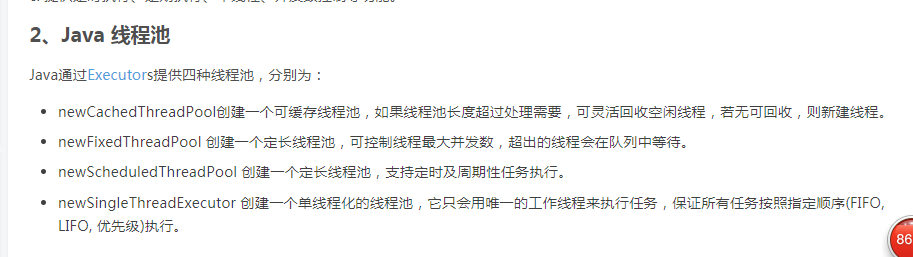


## TreeMap

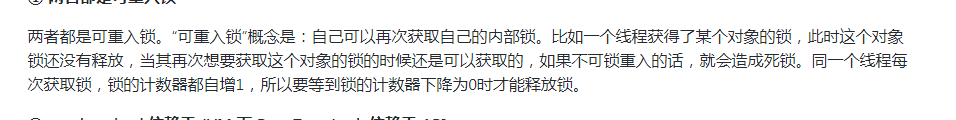
<https://yikun.github.io/2015/04/06/Java-TreeMap%E5%B7%A5%E4%BD%9C%E5%8E%9F%E7%90%86%E5%8F%8A%E5%AE%9E%E7%8E%B0/>

## 线程池

https://liuzho.github.io/2017/04/17/%E7%BA%BF%E7%A8%8B%E6%B1%A0%EF%BC%8C%E8%BF%99%E4%B8%80%E7%AF%87%E6%88%96%E8%AE%B8%E5%B0%B1%E5%A4%9F%E4%BA%86/

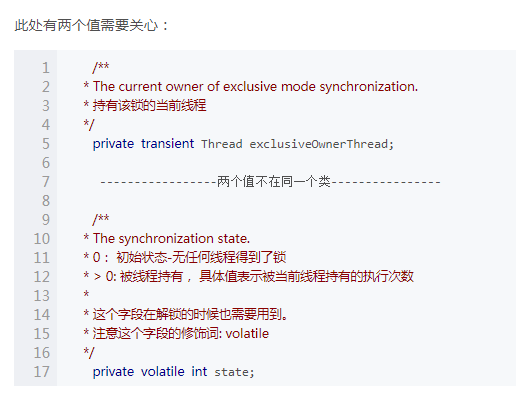


## 可重入锁有什么用



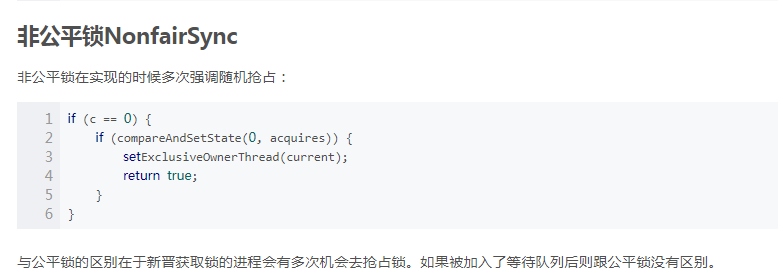
子类改写了父类的 synchronized 方法，然后调用父类中的方法，此时如果内置锁不是可重入的，那么这段代码将产生死锁。





公平锁和非公平锁





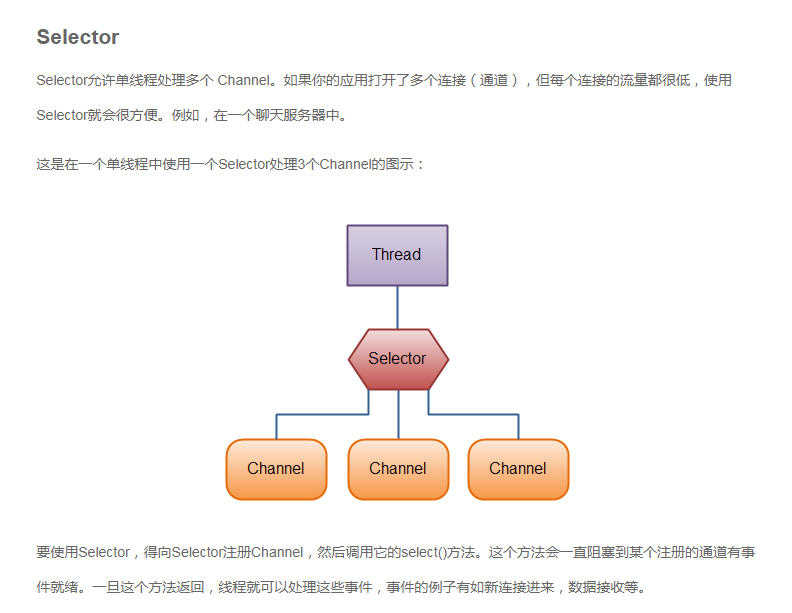
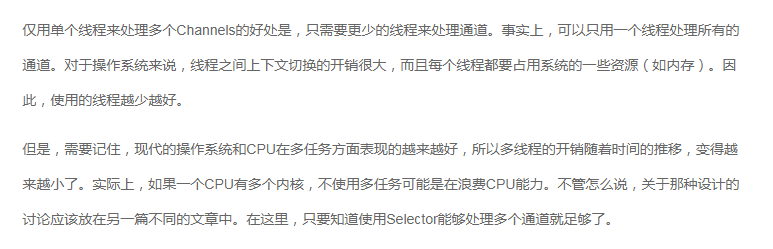
# 设计模式

单例模式 （4种）懒汉线程不安全，懒汉线程安全，饿汉线程安全，双重校验锁

工厂模式 简单工厂，工厂模式，抽象工厂模式

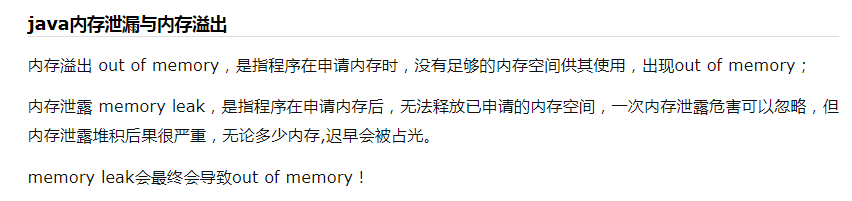
# NIO

### 为什么使用Selector?



# JVM

类加载机制



# 数据库

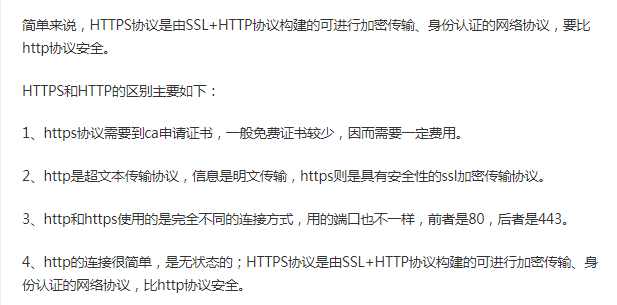
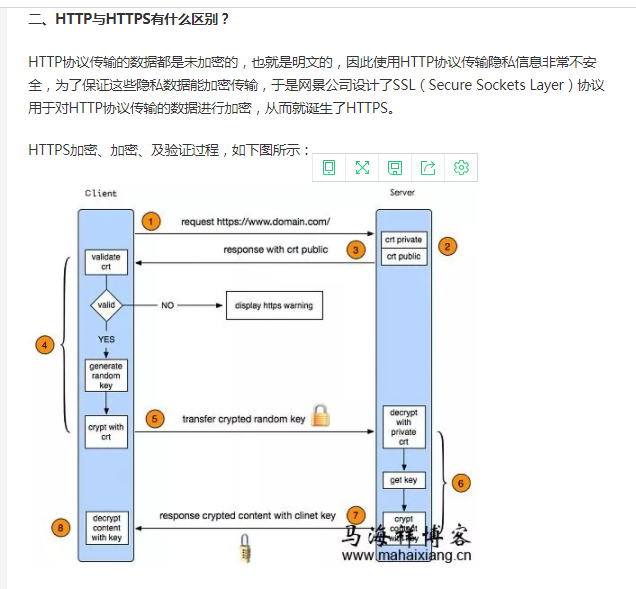
引擎

索引

优化

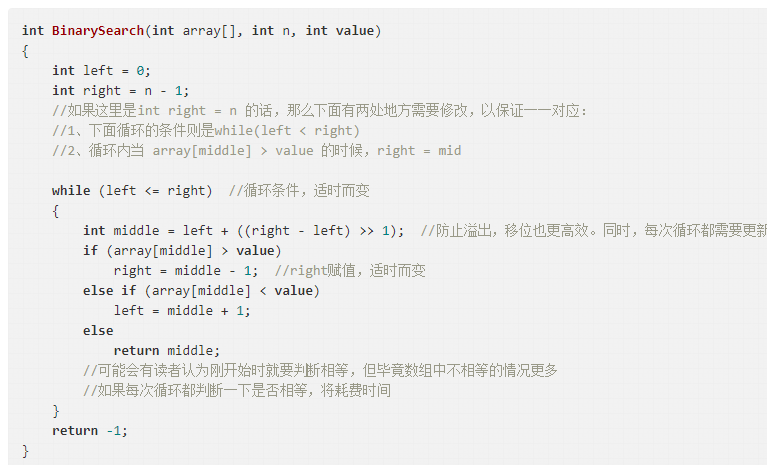
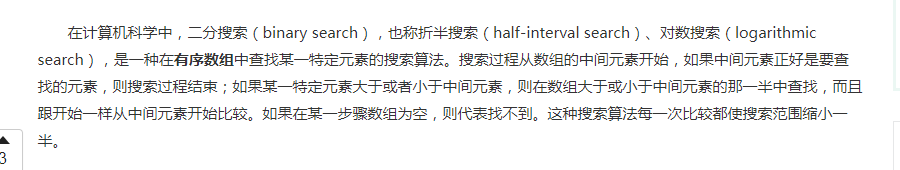
<https://coolshell.cn/articles/1846.html>

# 网络



# 算法

二分搜索的过程



二叉树DFS深度 -前序遍历

BFS -层次

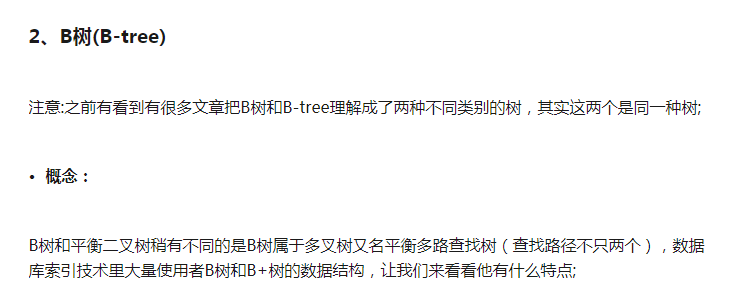
中序

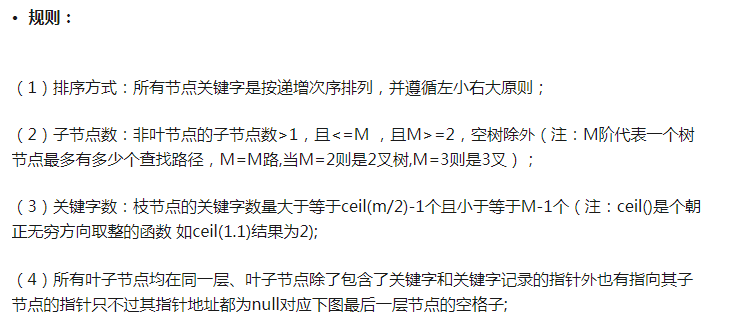
## 后序

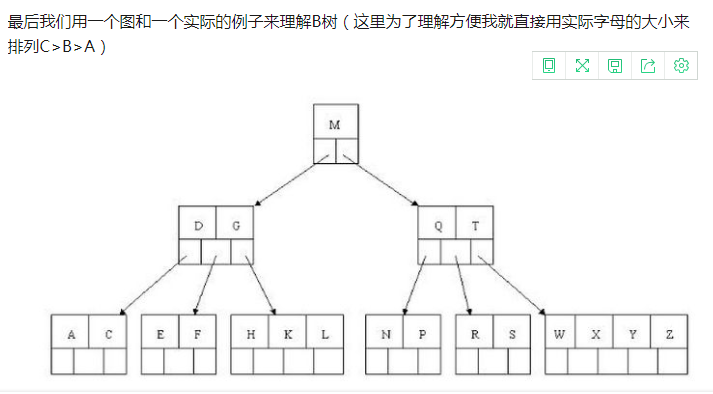
## Prim和

链表-环

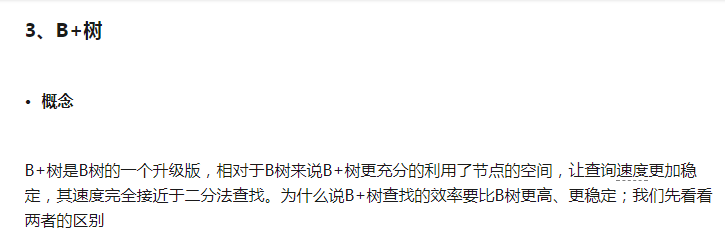
B树和B+树

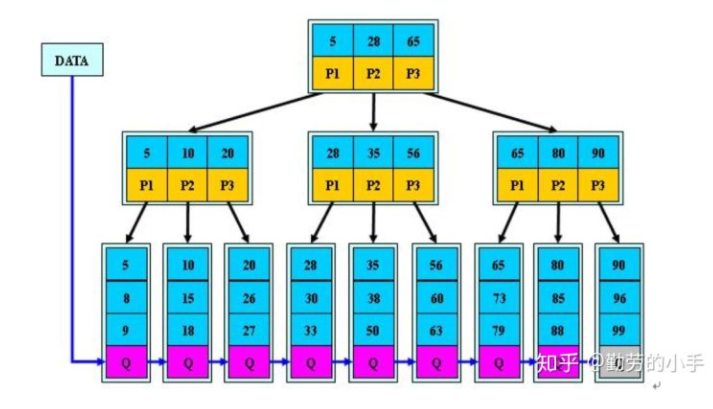
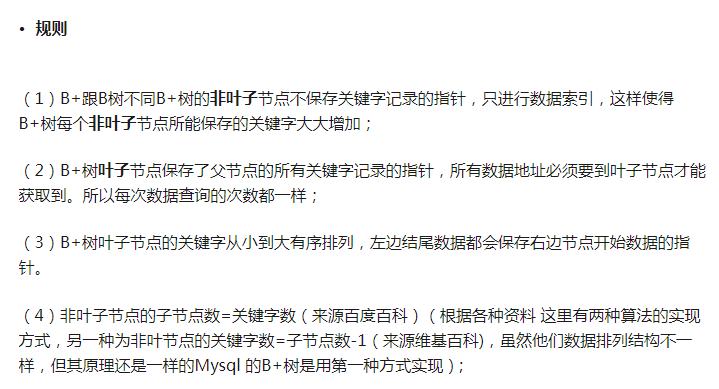


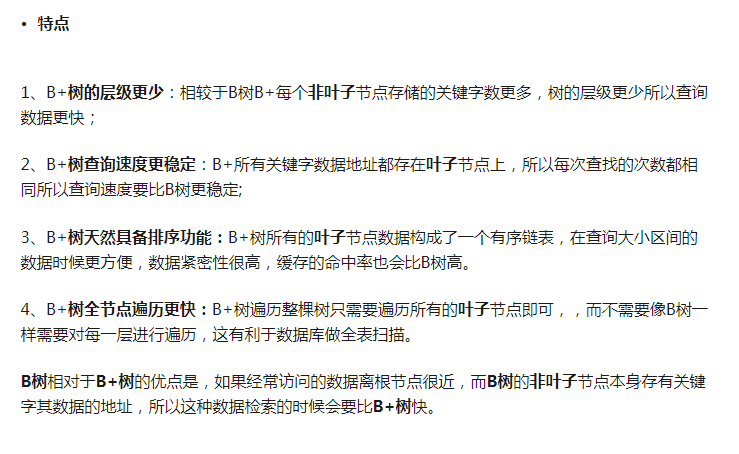




B+







红黑树

