linux内核常用的数据结构有如下几个：

链表

队列

映射

二叉树

这里主要描述环形双向链表和红黑树。

# 一、链表

链表是linux内核中最简单、最普通的数据结构；链表是一种存放和操作可变数量元素的数据结构。

单向链表：每个元素都包含一个指向下一个元素的指针，只能从头向后连接的链表

双向链表：每个元素除了包含一个指向下一个元素的指针，还包含一个指向前一个元素的指针，同时可以向前和向后连接的链表

环形链表：在上面的基础上，收尾相连形成一个环，分为环形单向链表和环形双向链表。Linux内核的标准链表主要采用环形双向链表。

细节见另外一篇文章《linux内核双向链表学习》。

# 二、队列

任何操作系统内核都少不了一种编程模型：生产者和消费者。该模式中，生产者创造数据，消费者读取消息和处理包。实现该模型的最简单的方式就是使用队列，linux内核通用队列实现称为kfifo。

1、创建队列

int kfifo\_alloc(struct kfifo \*fifo,unsigned int size, gfp\_t gfp\_mask);

该函数创建并初始化一个大小为size的kfifo，内核使用gfp\_mask标识分配队列。

void kfifo\_init(struct kfifo \*fifo, void \*buffer, unsigned int size)

该函数创建并初始化一个kfifo对象，将使用有buffer指向的size字节大小的内存。

两个函数size必须是2的次幂。

2、推入队列数据

unsigned int kfifo\_in(struct kfifo \*fifo, const void \*from, unsigned int len)

该函数把from指针指向的len字节数据拷贝到fifo所指的队列中。如果成功，则返回推入数据的字节大小，如果队列中的空闲字节小于len，则该函数最多拷贝队列可用空间那么多数据。

3、摘取队列数据

unsigned int kfifo\_out(struct kfifo \*fifo, void \*to, unsigned int len)

该函数从fifo所指向的队列中拷贝出长度为len字节的数据到to所指的缓冲中。如果成功，该函数则返回拷贝的数据长度。如果队列中数据小于len，则该函数拷贝出的数据必然小于需要的数据大小。

unsigned int kfifo\_out\_peek(struct kfifo \*fifo, void \*to, unsigned int len, unsigned int offset)

函数和kfifo\_out类似，但出口偏移不增加，而且摘取的数据仍然可被下次kfifo\_out摘取。

4、获取队列长度

unsinged int kfifo\_size(struct kfifo \*fifo)

获取存储kfifo队列空间总体大小

unsinged int kfifo\_len(struct kfifo \*fifo)

获取存储kfifo队列中数据大小

unsinged int kfifo\_avail(struct kfifo \*fifo)

获取存储kfifo队列剩余空间大小

int kfifo\_is\_empty(struct kfifo \*fifo)

int kfifo\_is\_full(struct kfifo \*fifo)

非0代表空或满，0则相反

5、重置和撤销队列

void kfifo\_reset(struct kfifo \*fifo)

清除内容

void kfifo\_free(struct kfifo \*fifo)

释放kfifo\_alloc分配的队列

# 三、映射

一个映射，也常称关联数组，其实就是一个由唯一键组成的集合，而每个键必然关联一个特定的值。这种键到值的关联关系称为映射。至少支持三个操作：

Add(key, value)

Remove(key)

Value = Lookup(key)

虽然散列表是一种映射，但并非所有的映射都需要通过散列表实现。除了使用散列表，映射也可以通过自平衡二叉搜索树存储数据。虽然键到值的映射属于一个通用说法，但是更多时候特指使用二叉树而非散列表实现的关联数组。

# 四、二叉树

树结构是一个能提供分层的树型数据结构的特定数据结构。

一个二叉搜索树是一个节点有序的二叉树，其顺序通常遵循下列法则：

A. 根的左分支节点值小于根节点值

B. 右分支节点值都大于根节点

C. 所有的子树都是二叉搜索树

一个平衡二叉树是一个所有叶子节点深度差不超过1的二叉树。处于树底层的节点称为叶子节点，一个节点的深度指从其根节点起，到达它一共需经过的父节点数目。红黑树是一种自平衡二叉树，linux主要的平衡二叉树数据结构就是红黑树，它遵循下列法则：

A．所有节点要么着红色，要么着黑色

B．根节点是黑色

B．叶子节点都是黑色

C．叶子节点不包含数据

D．所有非叶子节点都有两个子节点

E．如果一个节点是红色，则它的子节点都是黑色

F．从任一节点到其每个叶子节点的所有简单路径都包含相同数目的黑色节点

Linux实现的红黑树称为rbtree，在文件lib/rbtree.c中定义，关于详细介绍可以参考一下博客地址：

http://www.cnblogs.com/skywang12345/p/3245399.html