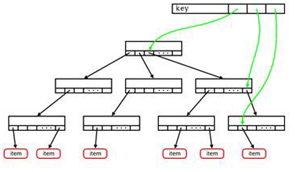
# Radix tree文档

**一、概述**

Linux radix树最广泛的用途是用于内存管理，结构address\_space通过radix树跟踪绑定到地址映射上的核心页，该radix树允许内存管理代码快速查找标识为dirty或writeback的页。Linux radix树的API函数在lib/radix-tree.c中实现。

Linux基数树（radix tree）是将指针与long整数键值相关联的机制，它存储有效率，并且可快速查询，用于指针与整数值的映射（如：IDR机制）、内存管理等。



上图显示了一个有3级结点的radix树，每个数据条目（item）可用3个6位的键值（key）进行索引，键值从左到右分别代表第1~3层结点位置。没有孩子的结点在图中不出现。因此，radix树为稀疏树提供了有效的存储，代替固定尺寸数组提供了键值到指针的快速查找。   
以index=0x5BFB68为例，化为二进制，每6位为一组：10110(22,第一层编号) 111111(63第二次编号) 101101(45第三层编号) 101000(40第四层编号)。

**二、数据结构介绍**

介绍将以linux 4.18.6的内核为例。

/\*

\* @count is the count of every non-NULL element in the ->slots array

\* whether that is an exceptional entry, a retry entry, a user pointer,

\* a sibling entry or a pointer to the next level of the tree.

\* @exceptional is the count of every element in ->slots which is

\* either radix\_tree\_exceptional\_entry() or is a sibling entry for an

\* exceptional entry.

\*/

struct radix\_tree\_node {

unsigned char shift; /\* 当前slot在树的第几层，6bit或者4bit一层 \*//\* Bits remaining in each slot \*/

unsigned char offset; /\* Slot offset in parent \*/

unsigned char count; /\* 当前的slot一共有几个有效的节点 \*//\* Total entry count \*/

unsigned char exceptional; /\*用于shmem/tmpfs保存swap entries \*/ /\* Exceptional entry count \*/

struct radix\_tree\_node \*parent; /\* Used when ascending tree \*/

struct radix\_tree\_root \*root; /\* The tree we belong to \*/

union {

struct list\_head private\_list; /\* For tree user \*/

struct rcu\_head rcu\_head; /\* Used when freeing node \*/

};

void \_\_rcu \*slots[RADIX\_TREE\_MAP\_SIZE];

unsigned long tags[RADIX\_TREE\_MAX\_TAGS][RADIX\_TREE\_TAG\_LONGS];/\*tag最多支持3中类型的tag，二维的第二个参数为每个类型tag需要的long个数，如每个node需要表示64个子节点，系统为32bit，则RADIX\_TREE\_TAG\_LONGS=64 /32=2,

如果系统为64bit，则该宏为1\*/

};