```
ypedef struct dictEntry {
    // key
    void *key;
    // value
    union {
        void *val;
        uint64_t u64;
        int64_t s64;
        double d;
    } v;
    // 指向下一个节点的指针，形成链表
    struct dictEntry *next;
} dictEntry;
```

Redis的哈希表dictht，实现原理和java中的HashMap是类似的，核心都是：数组 + 链表

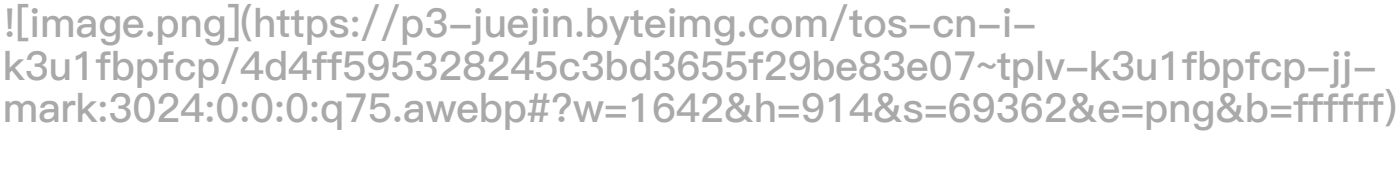
另外，还需要解决以下两个问题：

1. 如何确定key，放到数组的哪个下标下面
2. 不同的key，也可能放到相同的数组下标，如何解决

计算key的数组下标

主要是两个步骤：

1. 计算key的hashcode：依据哈希函数，计算key的哈希值hashcode
2. 计算数组下标：hashcode & sizemask(数组大小 - 1)，位运算更高效

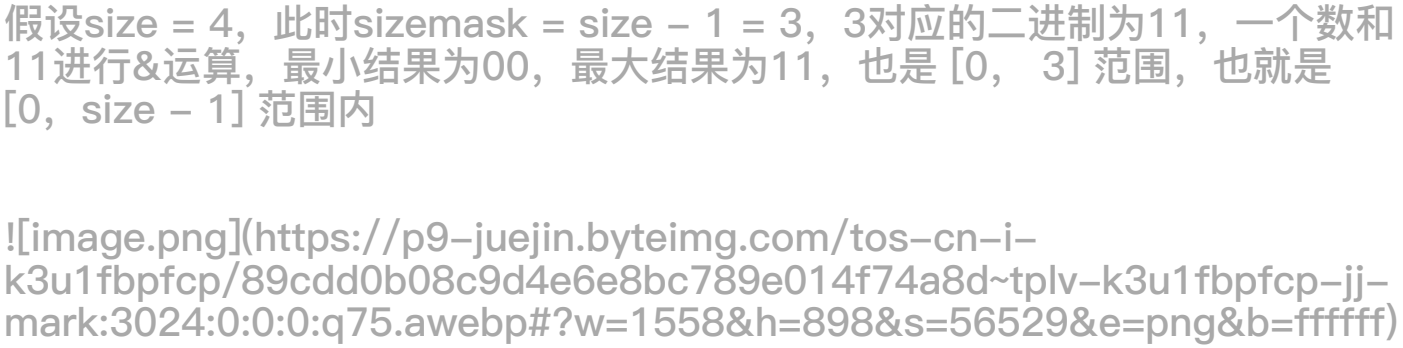


额外扩展：为什么hashcode % size的结果和hashcode & sizemask是一样的

hashcode % size，取模操作，结果肯定在 [0，size – 1] 范围内

重点看下hashcode & sizemask

假设size = 4，此时sizemask = size – 1 = 3，3对应的二进制为11，一个数和11进行&运算，最小结果为00，最大结果为11，也是 [0， 3] 范围，也就是 [0，size – 1] 范围内

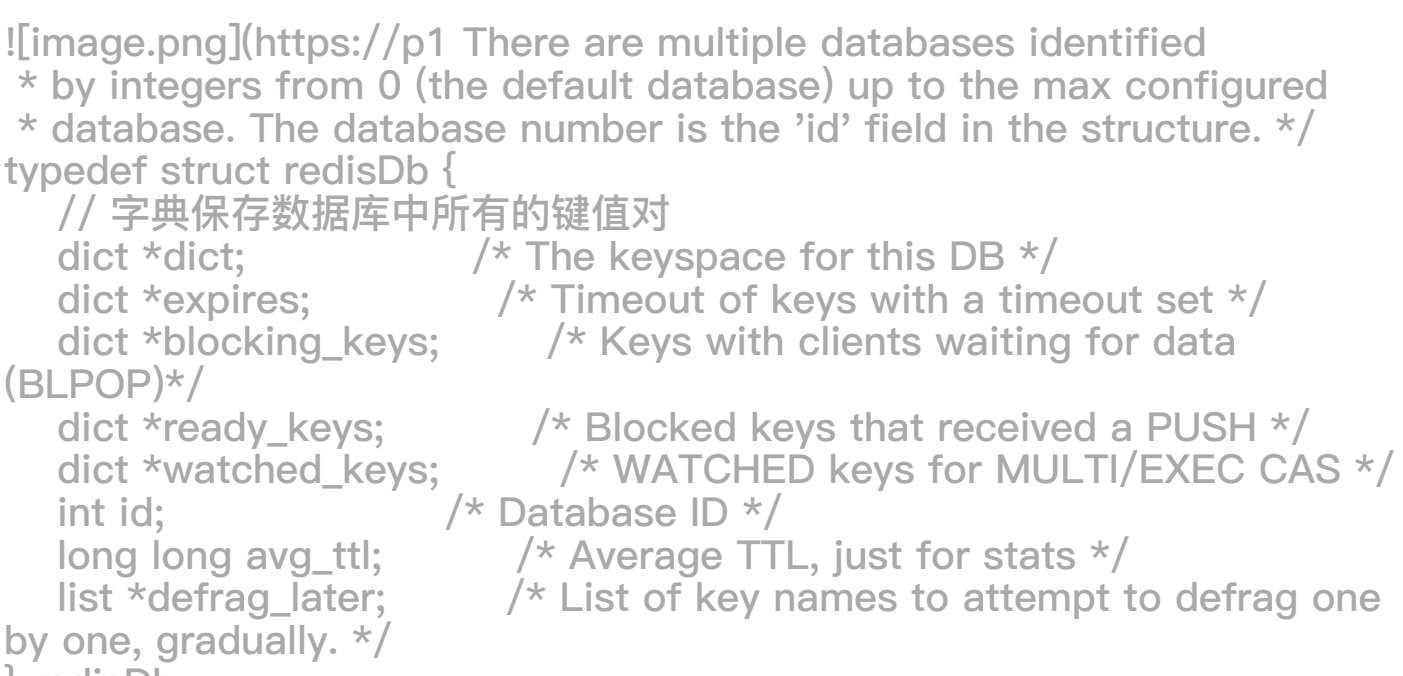


解决下标冲突

多个不同的key，也可能计算出相同的数组下标，比如：

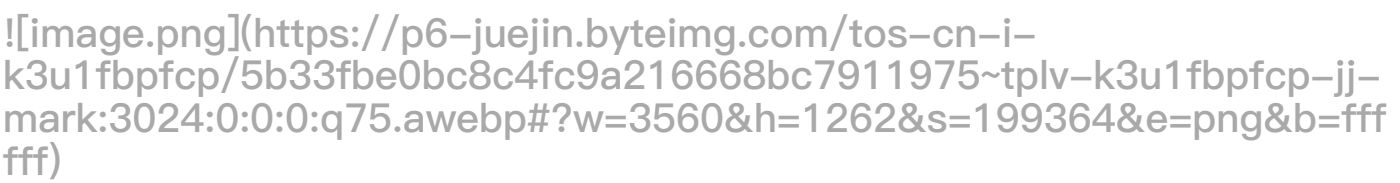
1. 不同的key，计算出的hashcode相同。也就是哈希冲突
2. 不同的key，计算出的hashcode不同，但 hashcode & sizemask 的结果相同

Redis使用链表来解决这类问题，数组下标内如果有多个不同的key的话，形成单向链表



```
There are multiple databases identified
 * by integers from 0 (the default database) up to the max configured
 * database. The database number is the 'id' field in the structure. */
typedef struct redisDb {
    // 字典保存数据库中所有的键值对
    dict *dict;                 /* The keyspace for this DB */
    dict *expires;              /* Timeout of keys with a timeout set */
    dict *blocking_keys;        /* Keys with clients waiting for data
(BLPOP)*/
    dict *ready_keys;           /* Blocked keys that received a PUSH */
    dict *watched_keys;         /* WATCHED keys for MULTI/EXEC CAS */
    int id;                     /* Database ID */
    long long avg_ttl;          /* Average TTL, just for stats */
    list *defrag_later;         /* List of key names to attempt to defrag one
by one, gradually. */
} redisDb;
```

```




总结
==

* 哈希表使用数组 + 链表实现，使用链表解决哈希冲突
* 哈希表负载因子 = 节点个数used / 数组大小size，太大节点冲突严重，太小浪费空间，使用rehash来解决
* Redis字典其实就是对哈希表dictht + rehash的封装，持有两个哈希表ht[0]和ht[1]，rehash操作就是把ht[0]数据迁移到ht[1]中
* 渐进式rehash，就是分多次、渐进式的完成rehash
* Redis可以有多个数据库，默认是16个
* 每个数据库中包含一个字典dict字段，用来存储数据库中所有的键值对
 原文链接: https://juejin.cn/post/7386250013632954407
```