<https://blog.csdn.net/yankai0219/article/details/8186998>

我希望我可以写出一系列文章，每一章都是先讲数据结构，然后再讲这种类型的数据结构是如何在Nginx中实现的。但是由于个人对Nginx的了解，有所欠缺，因此到了Nginx中的数据结构时，就不能很好的举例说明。只能简单罗列。

本文主要讲述了Nginx中Hash表初始化的不同，其他内容还没有详细阅读。这篇文章就当做抛砖引玉了。

Nginx中Hash相关文章

     1.Nginx中Hash表的设计及实现

<http://www.linuxidc.com/Linux/2012-08/67040.htm>

<http://blog.csdn.net/gsnumen/article/details/7817396>

<http://blog.csdn.net/livelylittlefish/article/details/6636229>

     2.简单应用

<http://blog.csdn.net/lifeibo/article/details/5897126>

<http://code.google.com/p/nginxsrp/wiki/NginxCodeReview#ngx_hash>

基本数据结构介绍

Hash表结构中桶的结构

|  |
| --- |
| typedef struct {               //hash结构     ngx\_hash\_elt\_t  \*\*buckets; //hash桶(有size个桶)     ngx\_uint\_t        size;    //hash桶个数  } ngx\_hash\_t; |

Hash表的桶中结点的结构

|  |
| --- |
| typedef struct {               //hash元素结构     void             \*value;   //value，即某个key对应的值，即<key,value>中的value     u\_short           len;     //name长度     u\_char            name[1]; //某个要hash的数据(在nginx中表现为字符串)，即<key,value>中的key } ngx\_hash\_elt\_t; |

重要操作：

     ngx\_int\_t  **ngx\_hash\_init**(ngx\_hash\_init\_t \*hinit, ngx\_hash\_key\_t \*names, ngx\_uint\_t nelts) ;

     作用：

     1）根据关键字的数量nelts及一个bucket的大小bucket\_size，动态的计算出所需的hash表中所需要的hash桶的个数。（在代码中，在**for** (size = start; size < hinit->max\_size; size++)，没有进入goto found时）

     2）然后将names中数据写入hash表中（found:中工作）

          首先得到每个桶所使用的实际大小  ，test[hash\_value]数组来记录这个实际大小。通过遍历整个name，从而可以计算出每个桶中实际大小

          其次，通过test[hash\_value]数组得到整个hash表中使用的大小，hash表中使用的大小记录在变量len中。

          之后，创建，作用未知。

          再次，分配空间。elts = ngx\_palloc(hinit->pool, len + ngx\_cacheline\_size);  elts分配了整个hash表中所需要的准确的内存的大小。elts所分配的这块内存就是hash表所使用的内存。 通过test[hash\_value]所记录的每个桶所使用的内存大小，来获得每个桶的地址。每个桶的地址保存在数组buckets[hash\_value]中。

          最后，就是将ngx\_hash\_key\_t \* names中内容存入hash表中。方法非常巧妙.将test数组清零，通过test[key]获得元素在每个桶中的位置，bucket[key]+test[key]获得元素在elts这块内存中的地址。elt就是记录元素在elts这块内存中的地址。对elt的操作，就是将元素放入hash表中。

          当然，必不可少的还有，设置每个桶的结束元素为NULL。

     在这个函数里面test数组的作用非常多，简直是一个数组通吃整个函数。首先：test数组用于计算每个桶中所需要内存的实际大小。其次：将test数组清零后，通过test数据获得每个元素在每个桶所处的内存区域中的实际位置。

     说明：1）之前hash表初始化函数中，我经常用到的是固定大小的hash表，直接进行初始化即可。

          2）在ngx\_hash\_init中，根据要放入hash表中元素的个数，动态调整hash表的大小，并将元素插入hash表中。