切片结构体

切片的创建

声明但未初始化：数组地址为0，也就是未指定存放的空间

空切片(如 `var s = []int{}`)：完整地进行了初始化，容量为0，指针指向一段特定的内存区域

指定长度和容量(如 `s2 := make([]int, 0, 10)`)：先分配内存，初始化array，len=0，cap=10

同时指定(如 `s3 := make([]int, 10)`)：先分配内存，初始化array，len=10，cap=10

用数组初始化(如 `s4 := []int{1,2,3,4}`)：编译时就初始化一段内存，将数据存放进去，声明时array指向这段内存

切片的截取

`s[a:b]`
其中a和b可以缺省，缺省时a默认为0，b默认为len-1

截取的底层逻辑：
1、创建新切片
2、原切片的地址进行偏移后赋值给新切片

因此，新切片与原切片共享数组，修改元素时会互相影响

切片的赋值

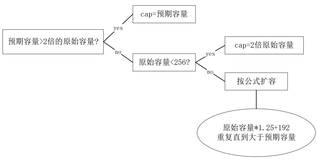
同切片截取，分配新slice结构，重设len和cap，共用底层数组

切片的追加

情况一、追加个数小于剩余容量，不会触发扩容，且底层存储位置不变

情况二、追加个数大于剩余容貌，会触发扩容，数组移动到新开辟的空间

扩容机制



切片的拷贝

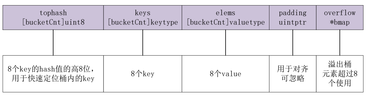
生成新slice结构体，共享底层数组

如果要深拷贝，先用make创建一个新切片，再用copy复制

map的底层原理

使用的是hash函数，数据存放在桶中，每个桶存放8个key-value
桶的数量是2的整数次幂，因为计算机中与运算效率高，而与运算要求2的整数次幂

桶bmap结构体



map底层数据结构-hmap结构体

count int	flags uint8	B uint8	noverflow uint16	hash0 uint32	buckets unsafe.Pointer	oldbuckets unsafe.Pointer	nevacuate uintptr	extra *mapextra
map中键值对的数目，调用len函数返回该值		记录桶的数目，桶的数目为2^B	使用的溢出桶的数量	哈希种子	指向桶数组的指针	记录扩容阶段原桶数组的位置，指向桶数组的指针	记录渐进式扩容的进度，小于该编号的bmap已经扩容到新桶中	可选字段，用于记录溢出桶的信息

map类型的变量本质上是一个指向该结构体的指针，因此它是引用类型的

在64位的计算机中，一个key经过hash后会得到一个64位的hash值，该hash值的低B位即为key所在桶的编号，高8位即为key在桶内的位置

扩容机制

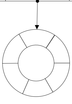
负载因子**loadFactor，即平均每个桶中的元素个数
一个桶中最多能容纳的键值对为loadFactor*2^B个

当负载因此超过6.5，即平均每个桶中的元素超过6.5就会触发扩容

渐进式扩容：因为扩容时移动数据是很费时的，因此采用分批次迁移
1、先分配足够多的新桶
2、执行读写时，分批将旧桶中的数据迁移到新桶
3、结构体中的oldbuckets和nevacuate用于记录扩容进度

channel底层结构-hchan结构体

qcount uint	dataqsiz uint	buf unsafe.Pointer	elemsize uint16	closed uint32	elemtype *_type	sendx uint	recvx uint	recvq waitq	sendq waitq	lock mutex
循环数组中的元素数量，长度	循环数组的大小，容量	指向底层循环数组的指针（环形缓冲区）	元素的大小	是否关闭的标志 0未关闭 1已关闭	元素类型	下次写的位置	下次读的位置	读等待队列	写等待队列	互斥锁，保证读写channel时的并发安全



sendq表示阻塞在当前channel上的发送者goroutine
recvq表示阻塞在当前channel上的接收者goroutine
具体结构为双向链表，链表中记录goroutine抽象出来的结构体sudog

channel的创建实际上是创建了一个指向该结构体的指针，因此是引用类型

分配内存流程：
情况一、无缓冲channel，直接给hchan结构体分配内存并返回指针

情况二、有缓冲channel，且元素不包含指针类型，则一次性为hchan结构体和底层循环数组分配连续内存，并返回指针（需要连续内存空间）

情况三、有缓冲channel，且元素包含指针类型，则分别分配hchan结构体内存和底层循环数组的内存，并返回指针 （可以利用内存碎片）

一些限制：
1、元素大小不能超过64k
2、元素的对齐大小不能超过8字节
3、计算出来的所需内存不能超过限制

发送过程

情况一、同步发送：读等待队列中存在接收者goroutine，直接将发送者的数据发送给第一个等待接收的goroutine

情况二、异步发送：读等待队列为空，且循环数组未满，把数据加入队尾

情况三、阻塞发送：无缓冲或循环数组已满，则将当前goroutine加入写等待队列，并将其挂起等待唤醒

这三种情况下都会在开始时加锁，结束时解锁以保证并发安全

接收过程

情况一、同步接收：写等待队列中存在发送者goroutine，将第一个发送者的数据拷贝给接收者，并将其唤醒

情况二、异步接收：写等待队列为空，且循环数组非空，把循环数组的队首元素拷贝给接收者

情况三、阻塞接收：写等待队列为空，且循环数组也为空，将当前goroutine加入读等待队列，并挂起等待唤醒

这三种情况下都会在开始时加锁，结束时解锁以保证并发安全

关闭channel

1、边界检查
2、从recvq释放所有readers
3、从sendq释放所有writers
4、唤醒所有readers和writers