**总体说明**

三种基于跳跃表的KV容器：基于内存的跳跃表（skiplist）、基于磁盘日志的跳跃表（safesl）、基于共享内存的跳跃表（smsl）。三种跳跃表都是单线程实现的，因此safesl和smsl可能会在日志记录或共享内存扩容时影响性能导致存储速度减慢。详细使用方法请参阅readme.txt。

每种跳跃表都提供了DEMO，其中共享内存的跳跃表DEMO必须在Linux上编译运行。DEMO打开后，可以输入命令进行测试，如：

set 10 100

get 10

del 10

具体DEMO的使用命令可参阅代码。

每种跳跃表的key还需要有相应转换为string的方法，以便于记录日志或在屏幕上打印状态。

**基于内存的跳跃表**

包括skiplist.hpp后就可以使用基于内存的跳跃表了。该跳跃表完全将数据保存在内存中，因此进程被杀或者电脑断电数据将丢失。

这种跳跃表的存储和查找都是最快的。

你可以使用几乎任何类型作为key和value，但key和value的类型必须有拷贝构造函数和重载等于号。

**基于磁盘日志的跳跃表**

包括safesl.hpp就可以使用了。该跳跃表会对存储记录保存日志，日志会先写入缓存区，不会立即存入磁盘，需要及时调用land\_log()方法将日志写入磁盘，如果不调用land\_log()，也会在日志达到一定数量时写入磁盘，但未写入磁盘的日志对应的内容会在断电后丢失。

意外断电后，可以通过日志恢复内容，也可以手动调用dump\_to\_file()方法将内容落地，落地后，如果再通过日志恢复，可以同时将落地文件作和日志为参数传入，这样系统就会删除已落地文件对应的日志内容，从而减少日志文件的体积。

Key和value可以是任何类型，包括string，vector等，但key和value的类必须要手动提供转换为二进制流的函数。例如，key是string，则需要提供函数将string转换为Binary类（定义在safesl.h）。Key和value需要重载拷贝构造和等于号。

**基于共享内存的跳跃表**

包括smsl.hpp即可使用。只支持Linux，不支持Windows。内容保存到共享内存上，因此进程意外退出后可以恢复。进程正常退出时也可以设置flag，告诉进程是否应该释放共享内存，如不释放，下次进程启动时还可以恢复。

Key和value的类型必须是只在栈中分配内存的类型，例如int，double，只包括栈中分配内存的类型的struct类型也可以支持。不支持string，vector等类型。Key和value需要重载拷贝构造和等于号。