Redis

Marvin

struct sdshdr struct sdshdr

服务器端处理请求：
事件循环结构：

```
itypedef struct aeEventLoop {
    int maxfd;   /* highest file descriptor currently registered */
    int setsize; /* max number of file descriptors tracked */
    long long timeEventNextId;   //生成事件id
    time_t lastTime;     /* Used to detect system clock skew */
    aeFileEvent *events; /* Registered events 已经注册的文件事件*/
    aeFiredEvent *fired; /* Fired events 已就绪的文件事件*/
    aeTimeEvent *timeEventHead;       //时间事件
    int stop;                         //事件处理器开关
    void *apidata; /* This is used for polling API specific data */
    aeBeforeSleepProc *beforesleep;     //处理事件前要执行函数
} aeEventLoop;

//File event structure
typedef struct aeFileEvent {
    int mask; /* one of AE_(READABLE|WRITABLE) */
    aeFileProc *rfileProc;
    aeFileProc *wfileProc;
    void *clientData;
} aeFileEvent;

/* Time event structure */
typedef struct aeTimeEvent {
    long long id; /* time event identifier. */
    long when_sec; /* seconds */
    long when_ms; /* milliseconds */
    aeTimeProc *timeProc;   //事件处理函数
    aeEventFinalizerProc *finalizerProc;        //事件释放函数
    void *clientData;
    struct aeTimeEvent *next;          //指针
} aeTimeEvent;

/* A fired event */
typedef struct aeFiredEvent {
    int fd; //已就绪文件描述符
    int mask;                        //事件类型掩码
} aeFiredEvent;
```

```
aeMain(redis.c 2721)->aeProcessEvents->
fe->rfileProc(eventLoop,fd,fe->clientData,mask)指向readQueryFromClient->
processInputBuffer->processCommand;
```

基本流程是从客户端读取数据到缓冲区，执行缓冲区内的命令

```
typedef struct redisObject {
    unsigned type:4;                 //用来存储string，list和set
 //位段数据类型
    unsigned notused:2;     /* Not used */
    unsigned encoding:4;
    unsigned lru:22;        /* lru time (relative to server.lruclock) */
    int refcount;
    void *ptr;
} robj;


typedef struct redisDb {
    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;
} redisDb;



typedef void redisCommandProc(redisClient *c);
typedef int *redisGetKeysProc(struct redisCommand *cmd, robj **argv,
int argc, int *numkeys, int flags);

struct redisCommand {
    char *name;
    redisCommandProc *proc;
    int arity;
    char *sflags; /* Flags as string representation, one char per flag. */
    int flags;    /* The actual flags, obtained from the 'sflags' field. */
    /* Use a function to determine keys arguments in a command line. */
    redisGetKeysProc *getkeys_proc;
    /* What keys should be loaded in background when calling this command? */
    int firstkey; /* The first argument that's a key (0 = no keys) */
    int lastkey;  /* The last argument that's a key */
    int keystep;  /* The step between first and last key */
    long long microseconds, calls;
};
```

客户端初始化部分

redisContext *redisContextInit(void)函数初始化：

```
static redisContext *redisContextInit(void) {
    redisContext *c;

    c = calloc(1,sizeof(redisContext));
    if (c == NULL)
        return NULL;

    c->err = 0;
    c->errstr[0] = '\0';
    c->obuf = sdsempty();
    c->reader = redisReaderCreate();
    return c;
}
```

```
typedef struct redisContext {
    int err; /* Error flags, 0 when there is no error */
    char errstr[128]; /* String representation of error when applicable */
    int fd;         //文件描述符，socket
    int flags;
    char *obuf; /* Write buffer */
    redisReader *reader; /* Protocol reader */
} redisContext;


redisReader *redisReaderCreate(void) {       初始化
    redisReader *r;
    r = calloc(sizeof(redisReader),1);
    if (r == NULL)
        return NULL;
    r->err = 0;
    r->errstr[0] = '\0';
    r->fn = &defaultFunctions;
    r->buf = sdsempty();
    r->maxbuf = REDIS_READER_MAX_BUF;
    if (r->buf == NULL) {
        free(r);
        return NULL;
    }
    r->ridx = -1;
    return r;
}

typedef struct redisReader {
    int err; /* Error flags, 0 when there is no error */
    char errstr[128]; /* String representation of error when applicable */
    char *buf; /* Read buffer */
    size_t pos; /* Buffer cursor */
    size_t len; /* Buffer length */
    size_t maxbuf; /* Max length of unused buffer */

    redisReadTask rstack[9];
    int ridx; /* Index of current read task */
    void *reply; /* Temporary reply pointer */

    redisReplyObjectFunctions *fn;
    void *privdata;
}redisReader;


对应的是fn,在hiredis.c中有具体实现
static redisReplyObjectFunctions defaultFunctions = {
    createStringObject,
    createArrayObject,
    createIntegerObject,
    createNilObject,
    freeReplyObject
};

typedef struct redisReplyObjectFunctions {
    void *(*createString)(const redisReadTask*, char*, size_t);
    void *(*createArray)(const redisReadTask*, int);
    void *(*createInteger)(const redisReadTask*, long long);
    void *(*createNil)(const redisReadTask*);
    void (*freeObject)(void*);
} redisReplyObjectFunctions;

typedef struct redisReadTask {
    int type;
    int elements; /* number of elements in multibulk container */
    int idx; /* index in parent (array) object */
    void *obj; /* holds user-generated value for a read task */
    struct redisReadTask *parent; /* parent task */
    void *privdata; /* user-settable arbitrary field */
} redisReadTask;
```