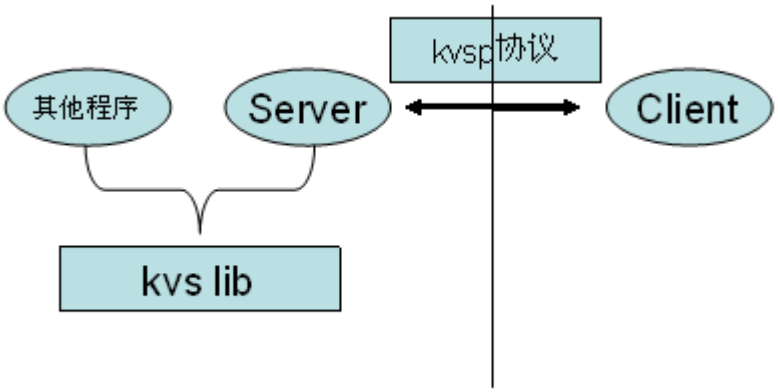
# KVSystem使用文档

## 一、介绍

KVS 是单机 key-value 存储系统，目前提供 C 形式的库和 Client&Server 程序。作为一个系统，使用者有：调用 PUT/GET 接口的程序员、对系统进行监测的系统管理员。该系统暂时没有为系统管理员提供相关接口。开发环境: Ubuntu10.04(32 位)，理论上，系统运行时内存使用 550M。

系统框架：




## 二、安装

下载：

```
svn  checkout  http://smcached.googlecode.com/svn/trunk/  kvs
```

进入 kvs 安装目录：

```
~/kvs$ sudo make all
```

系统依次自动安装 libkvs.so(KVS 的 C 语言共享库)、libkvsp.so(KVS 用于 CS 的协议)、用于通信的 Server。

## 三、使用例子

在安装好系统之后，可以通过两种方式使用：

1.通过 C 语言的 KVS 共享库在本机创建 KVS 实例(大文件)来使用。通过填充 KVS_SET 进行实例的初始化，通过 put get delete 接口进行操作。编译方式：

```
~/kvs/sample$ gcc test_lib.c -lkvs -lm -o test_lib
```

一个简单的例子见 sample/test_lib.c。

2.自己编写 Client 程序，通过协议与 Server 通信。编译方式：

```
~/kvs/sample$ gcc client.c -lkvsp -o client
```

一个简单的例子见 sample/client.c。

# 四、接口详细说明

**C 语言的 KVS 共享库接口(kvs.h):**

```
/* kvs.h */
#define    KVS_IMAGE_SIZE                (548684*1024)

/*一些返回状态信息*/
#define    KVS_INIT_SUCCESS          0x00000000
.....
#define    KVS_DELETE_NG_NOT_EXIST          0xffff00f0

/*初始化方式*/
#define    KVS_CREATE                0
#define    KVS_LOAD                  1
#define    KVS_PATHNAME_LEN          1024      /*FIXME:*/

typedef struct
{
    int         init_type;     /*指定初始化 KVS 实例的方式，KVS_CREATE 为创建，KVS_LOAD 为加载*/
    int         size_in_g;     /*模拟的硬盘的最大容量，以 G 为单位。*/
    char        disk_file[KVS_PATHNAME_LEN];
    char        disk_log[KVS_PATHNAME_LEN];
    char        sync_log[KVS_PATHNAME_LEN];
    char        index_log[KVS_PATHNAME_LEN];
} KVS_SET;

/*以下为提供的接口*/

/*  系统初始化与退出函数，传入 KVS_SET 结构即可调用。其中 char** ret_p 返回 image 的首地址，是为了以后
进行周期性的写入文件做准备。*/
int    kv_init(KVS_SET*, char** ret_p);
int    kv_exit();

/*  普通的 get/put/delete 操作，  其中 kv_get 操作返回 value size */
int    kv_get(char* key, int key_size, char* buf, int buf_size);
int    kv_put(char* key, int key_size, char* value, int value_size);
int    kv_delete(char* key, int key_size);
```

```
/* 因为上层调用 GET 操作时，不知道 value_size，无法精确地申请恰当大小的内存。
 * 该接口中，内存的申请发生在接口内部，而释放则有调用者负责。
 * 需要 kvs 库和应用层使用统一的内存管理器，暂时为默认的系统 malloc()/free()。
 */
int    kv_get_2(char* key, int key_size, char** buf, int* buf_size);
```

```
/* PUT 大文件中，有时会需要一段段的传入 value。
 * 可减小 Server 端的 Application buffer 的大小。
 * 传入的 value 必须要以 16K 大小为单位。
 */
int    kv_put_index(char* key, int key_size, int value_size);
int    kv_put_seg(int disk_location, char* value, int value_size_in_16k);
```

**KVS 的通信协议接口(kvsp.h):**

通信协议正在改进中，使用方式请参考(sample/client.c)

```
/*定义 command 的三种操作 */
#define GET                     0X55
#define PUT                     0XAA
#define DELETE                  0XFF


/*command packet */
typedef struct
{
    int type;
    int key_length;
    int value_length;
    int filling;
}COMPACK;


/* 与 server 端连接，返回 sockfd
 */
int conn_serv(struct sockaddr_in *servaddr);


/* 发送命令， command 包括 GET PUT DELETE
 */
int data_trans(int command, int sockfd, char *key, int key_len, char **value, int *value_len);


/* 接受数据，详见 sample/client.c
 */
int recv_stat(int sockfd, char *buffer);
int recv_data(int sockfd, char *buffer, int data_size);
```