/\*------------华丽的分割线---------------\*/

这段是给像我一样对smb没有什么概念的童鞋们看的，自己理解的也只是皮毛如有错误请理解，如果对smb熟悉的可以飘过这一段。

smb有时候也叫做samba，和ftp一样也是一种共享文件的协议，但是比ftp强的是它不仅只有下载文件的权限，对文件的下增删改都是可以的。Windows上的贡献文件夹就是基于smb的协议的。

在IOS下由于设备的硬盘都不是很大，所以现在有很多的硬件厂商都对做wifi共享文件有兴趣，早期的协议很多都是采用ftp的,为什么我会知道这些因为以前公司是做外包的很多硬件厂商拿过来的硬件都是基于ftp的。但是后来发现用ftp会有诸多不变，有的厂商就把协议修改成了今天讨论的smb协议了。

不知道是不是我Google的方式不对，还是因为确实smb在IOS方面的资料很少。

在网上找了很久都没有找到一个能用的demo，这个demo都是在网上找了好几demo合在一起的。比如说demo里面的libsmbclient.a文件就是在网上找的samba协议的源代码重新编译的但是基本上都是在基于<https://github.com/kolyvan/kxsmb/>工程下修改的。注意该demo只是实现了smb的下载，并没有实现上传删除这些功能的，如果实现删除上传这些功能我会在后面介绍。

下面先说下在mac上如果使用smb访问文件，主要是作用在测试smb程序的时候，你可以这样先操作测试下你的配置是否正确，来确认问题的原因。

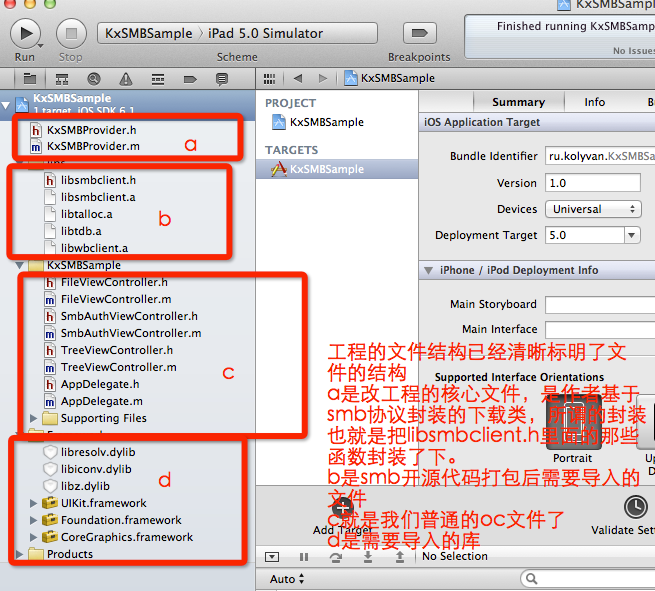
而且还可以做为和其它Windows电脑贡献文件的一种方式，OK扯蛋完毕下面正式介绍下smb的使用。



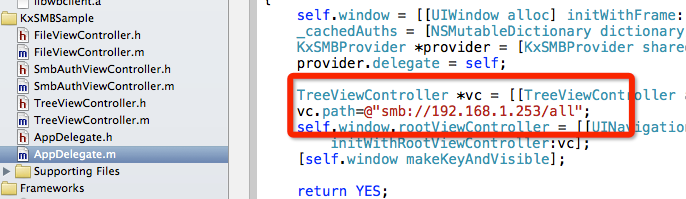


/\*-------------华丽的分割线-------------\*/

1---》首先介绍下工程的文件结构



如果要测试的童鞋请记得修改smb服务器地址，并且登陆action的回调地址也是在AppDelegate这个类下面实现的。



说明1



说明2

2---》使用说明

由于demo是已经实现了获取文件夹目录列表并且下载文件，所以就不对下载说明了。

上面说到了KxSMBProvider类是作者自己基于libsmbclient.h封装的类，建议大家都看看这2个核心文件。后面提到的上传删除也都是模仿KxSMBProvider类的调用顺序方式去调用libsmbclient.h文件的函数方法。

我自己本来是封装了一个类的，但是由于写这个类的时候比较匆忙加了一些不必要的方法和属性搞的比较混乱，想修改下但由于现在只有一台电脑也没有Windows电脑做共享文件夹了，基于防止误导他人原则所以我只是把主要流程几起代码贴出来。自己动下手组织一下就可以的了。

2.1---》全局说明所以的操作返回0那么表示操作失败

2.2首先先导入下面2个类

#import "libsmbclient.h"

#import "KxSMBProvider.h"

2.3然后定义以下2个变量

/\*连接桥\*/

SMBCCTX \*\_context;

/\*上传文件中会用到，表示上传了当前多少数据\*/

long lSize;

/\*上传文件中会用到，表示当前上传的数据源\*/

NSData \*upLoadData;

这是下面会用到的\_path变量的路径列子 //\_path = @"smb://192.168.0.111/项目共享文件夹/iCard/common/test/test.c";

2.4打开连接和关闭连接

/\*创建认证\*/

-(void)createContext{

if (!\_context) {

\_context = [KxSMBProvider openSmbContext];

}

}

/\*关闭连接\*/

-(void)closeFile{

if (\_context) {

[KxSMBProvider closeSmbContext:\_context];

\_context = NULL;

}

}

2.1.4操作流程总结

我把整个文件操作的流程理解为对数据库的操作，这样就很容易的可以理解整个操作步走了。

所以SMBCCTX的作用可以想象成为一个数据库的连接。

SMBCFILE的作用那就是具体的命令操作了。

所以我们的第一步就是创建一个数据库连接[self createContext];最后一个步骤必定是关闭一个数据库连接[self closeFile];

3---》（删除，上传， 修改文件权限 ，创建文件夹）方法

3.1修改文件权限属性

0777是权限最大的命令，有时候删除文件的时候删除失败可能就是权限不够，所以一般删除的时候都可以判断下文件命令权限是多少，然后判断是否需要修改权限值。怎么判断权限属性值请跳回到说明1

[self changFileModal:0777];

/\*修改文件的属性权限\*/

-(void)changFileModal:(int)iModal{

[self createContext];

int i =smbc\_chmod(\_path.UTF8String, iModal);

NSLog(@"i---->>>>action=====%d",i);

/\*命令是否失败\*/

if (i==0) {

[delegate changFileModalErro:index];

}else{

[delegate changFileModalFinish:index];

}

[self closeFile];

}

3.2删除文件

/\*删除文件\*/

-(void)deleteAction{

[self createContext];

smbc\_unlink\_fn unlinkFn = smbc\_getFunctionUnlink(\_context);

if (unlinkFn(\_context,\_path.UTF8String) != 0) {

[delegate deleteErro:index];

}else{

[delegate deleteFinish:index];

}

[self closeFile];

}

3.3创建文件夹

/\*创建文件夹\*/

-(void)createFolder{

[self createContext];

int fd = smbc\_mkdir(\_path.UTF8String, 0777);

if (fd != 0) {

[delegate createFoldeErro:index];

}else{

[delegate createFoldeFinish:index strName:strName];

}

[self closeFile];

}

3.4上传文件

一般的文件都上传的比较大，所以文件的上传都是一段段流上传的。

3.4.1 下面先分享下OC对文件的一些操作

/\*上传文件的时候先获取下文件的基本信息，比如文件大小文件是否存在\*/

NSFileManager \*fm = [NSFileManager defaultManager];

NSError \*error = nil;

NSDictionary\* dictFile = [[fm attributesOfItemAtPath:strLocalFile error:&error] retain];

if (error)

{

NSLog(@"getfilesize error: %@", error);

}

long nFileSize = [dictFile fileSize]; //得到文件大小

NSFileHandle \*fileHandle = [NSFileHandle fileHandleForReadingAtPath:strLocalFile];

/\*从多少字节处读取文件\*/

[fileHandle seekToFileOffset:lCurSize];

/\*读取文件大小\*/

NSData \*data = [fileHandle readDataOfLength:iReadSize];

3.4.1 通过SMB上传文件

首先上传操作前需要先打开连接桥[self createContext]; 最后该文件全部上传完成或者上传文件失败的时候需要关闭连接桥 [self closeFile];

-(void)upLoadAction{

int fd;

/\*判断是否是上传第一波数据，如果是那么把权限开到最大

如果服务器中该路径有相同文件存在那么也将会被清空

\*/

if (lSize==0) {

fd = smbc\_open(\_path.UTF8String, O\_RDWR | O\_CREAT | O\_TRUNC |O\_APPEND, 0777);

}else{

fd = smbc\_open(\_path.UTF8String, O\_RDWR |O\_APPEND, 0777);

}

/\*获取文件操作权限是否失败\*/

if (fd < 0) {

smbc\_close(fd);

NSLog(@"fd---->>>%d",fd);

[delegate upLoadStatusErro];

}else{

/\*上传流的的大小\*/

NSUInteger size = [upLoadData length] / sizeof(char);

char \*buff = (char\*) [upLoadData bytes];

NSLog(@"---->>>%d",size);

/\*写入流\*/

int err = smbc\_write(fd, buff, size);

/\*此处不是很确定是否需要关闭，但是个人认为是需要关闭fb的\*/

smbc\_close(fd);

lSize += size;

/\*上传成功还是失败\*/

if (err < 0) {

NSLog(@"err---->>>%d",err);

[delegate upLoadStatusErro];

}else{

[delegate upLoadStatus:index size:lSize];

}

}

}

3.5 KxSMBProvider文件中常用的方法，基本上我们常用的一下需求通过下面的组合使用就可以达到我们的需求了。上面解释的那些方法也都是在这些里面组合使用的。 KxSMBProvider文件比较大，所以我就把这个列出来了。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Callable functions for files. \*

\* Each callable has a function signature typedef, a declaration \*

\* for the getter, and a declaration for the setter. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef SMBCFILE \* (\*smbc\_open\_fn)(SMBCCTX \*c,

const char \*fname,

int flags,

mode\_t mode);

smbc\_open\_fn smbc\_getFunctionOpen(SMBCCTX \*c);

void smbc\_setFunctionOpen(SMBCCTX \*c, smbc\_open\_fn fn);

typedef SMBCFILE \* (\*smbc\_creat\_fn)(SMBCCTX \*c,

const char \*path,

mode\_t mode);

smbc\_creat\_fn smbc\_getFunctionCreat(SMBCCTX \*c);

void smbc\_setFunctionCreat(SMBCCTX \*c, smbc\_creat\_fn);

typedef ssize\_t (\*smbc\_read\_fn)(SMBCCTX \*c,

SMBCFILE \*file,

void \*buf,

size\_t count);

smbc\_read\_fn smbc\_getFunctionRead(SMBCCTX \*c);

void smbc\_setFunctionRead(SMBCCTX \*c, smbc\_read\_fn fn);

typedef ssize\_t (\*smbc\_write\_fn)(SMBCCTX \*c,

SMBCFILE \*file,

const void \*buf,

size\_t count);

smbc\_write\_fn smbc\_getFunctionWrite(SMBCCTX \*c);

void smbc\_setFunctionWrite(SMBCCTX \*c, smbc\_write\_fn fn);

typedef int (\*smbc\_unlink\_fn)(SMBCCTX \*c,

const char \*fname);

smbc\_unlink\_fn smbc\_getFunctionUnlink(SMBCCTX \*c);

void smbc\_setFunctionUnlink(SMBCCTX \*c, smbc\_unlink\_fn fn);

typedef int (\*smbc\_rename\_fn)(SMBCCTX \*ocontext,

const char \*oname,

SMBCCTX \*ncontext,

const char \*nname);

smbc\_rename\_fn smbc\_getFunctionRename(SMBCCTX \*c);

void smbc\_setFunctionRename(SMBCCTX \*c, smbc\_rename\_fn fn);

typedef off\_t (\*smbc\_lseek\_fn)(SMBCCTX \*c,

SMBCFILE \* file,

off\_t offset,

int whence);

smbc\_lseek\_fn smbc\_getFunctionLseek(SMBCCTX \*c);

void smbc\_setFunctionLseek(SMBCCTX \*c, smbc\_lseek\_fn fn);

typedef int (\*smbc\_stat\_fn)(SMBCCTX \*c,

const char \*fname,

struct stat \*st);

smbc\_stat\_fn smbc\_getFunctionStat(SMBCCTX \*c);

void smbc\_setFunctionStat(SMBCCTX \*c, smbc\_stat\_fn fn);

typedef int (\*smbc\_fstat\_fn)(SMBCCTX \*c,

SMBCFILE \*file,

struct stat \*st);

smbc\_fstat\_fn smbc\_getFunctionFstat(SMBCCTX \*c);

void smbc\_setFunctionFstat(SMBCCTX \*c, smbc\_fstat\_fn fn);

typedef int (\*smbc\_statvfs\_fn)(SMBCCTX \*c,

char \*path,

struct statvfs \*st);

smbc\_statvfs\_fn smbc\_getFunctionStatVFS(SMBCCTX \*c);

void smbc\_setFunctionStatVFS(SMBCCTX \*c, smbc\_statvfs\_fn fn);

typedef int (\*smbc\_fstatvfs\_fn)(SMBCCTX \*c,

SMBCFILE \*file,

struct statvfs \*st);

smbc\_fstatvfs\_fn smbc\_getFunctionFstatVFS(SMBCCTX \*c);

void smbc\_setFunctionFstatVFS(SMBCCTX \*c, smbc\_fstatvfs\_fn fn);

typedef int (\*smbc\_ftruncate\_fn)(SMBCCTX \*c,

SMBCFILE \*f,

off\_t size);

smbc\_ftruncate\_fn smbc\_getFunctionFtruncate(SMBCCTX \*c);

void smbc\_setFunctionFtruncate(SMBCCTX \*c, smbc\_ftruncate\_fn fn);

typedef int (\*smbc\_close\_fn)(SMBCCTX \*c,

SMBCFILE \*file);

smbc\_close\_fn smbc\_getFunctionClose(SMBCCTX \*c);

void smbc\_setFunctionClose(SMBCCTX \*c, smbc\_close\_fn fn);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Callable functions for directories. \*

\* Each callable has a function signature typedef, a declaration \*

\* for the getter, and a declaration for the setter. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef SMBCFILE \* (\*smbc\_opendir\_fn)(SMBCCTX \*c,

const char \*fname);

smbc\_opendir\_fn smbc\_getFunctionOpendir(SMBCCTX \*c);

void smbc\_setFunctionOpendir(SMBCCTX \*c, smbc\_opendir\_fn fn);

typedef int (\*smbc\_closedir\_fn)(SMBCCTX \*c,

SMBCFILE \*dir);

smbc\_closedir\_fn smbc\_getFunctionClosedir(SMBCCTX \*c);

void smbc\_setFunctionClosedir(SMBCCTX \*c, smbc\_closedir\_fn fn);

typedef struct smbc\_dirent \* (\*smbc\_readdir\_fn)(SMBCCTX \*c,

SMBCFILE \*dir);

smbc\_readdir\_fn smbc\_getFunctionReaddir(SMBCCTX \*c);

void smbc\_setFunctionReaddir(SMBCCTX \*c, smbc\_readdir\_fn fn);

typedef int (\*smbc\_getdents\_fn)(SMBCCTX \*c,

SMBCFILE \*dir,

struct smbc\_dirent \*dirp,

int count);

smbc\_getdents\_fn smbc\_getFunctionGetdents(SMBCCTX \*c);

void smbc\_setFunctionGetdents(SMBCCTX \*c, smbc\_getdents\_fn fn);

typedef int (\*smbc\_mkdir\_fn)(SMBCCTX \*c,

const char \*fname,

mode\_t mode);

smbc\_mkdir\_fn smbc\_getFunctionMkdir(SMBCCTX \*c);

void smbc\_setFunctionMkdir(SMBCCTX \*c, smbc\_mkdir\_fn fn);

typedef int (\*smbc\_rmdir\_fn)(SMBCCTX \*c,

const char \*fname);

smbc\_rmdir\_fn smbc\_getFunctionRmdir(SMBCCTX \*c);

void smbc\_setFunctionRmdir(SMBCCTX \*c, smbc\_rmdir\_fn fn);

typedef off\_t (\*smbc\_telldir\_fn)(SMBCCTX \*c,

SMBCFILE \*dir);

smbc\_telldir\_fn smbc\_getFunctionTelldir(SMBCCTX \*c);

void smbc\_setFunctionTelldir(SMBCCTX \*c, smbc\_telldir\_fn fn);

typedef int (\*smbc\_lseekdir\_fn)(SMBCCTX \*c,

SMBCFILE \*dir,

off\_t offset);

smbc\_lseekdir\_fn smbc\_getFunctionLseekdir(SMBCCTX \*c);

void smbc\_setFunctionLseekdir(SMBCCTX \*c, smbc\_lseekdir\_fn fn);

typedef int (\*smbc\_fstatdir\_fn)(SMBCCTX \*c,

SMBCFILE \*dir,

struct stat \*st);

smbc\_fstatdir\_fn smbc\_getFunctionFstatdir(SMBCCTX \*c);

void smbc\_setFunctionFstatdir(SMBCCTX \*c, smbc\_fstatdir\_fn fn);

4---》如果在使用的时候有什么问题可以联系下我，只要有空很乐意和大家一起讨论。