Android JNI层实现文件的read、write与seek操作

1、在JNI层实现文件的读写操作的话，就要使用到Linux的读写函数了。

2、打开文件

int open(const char *pathname,int flags, int mode);

返回值：为一个文件句柄（fd），供read、write等操作。
参数： pathname： 打开的文件所在路径字符串。如
String filename = "/sdcard/test.txt";
flags： 文件打开的方式

flag之间可以作“与”运算，如
open(filename, O_CREAT | O_RDWR, mode);

常用flags：
O_RDONLY 以只读方式打开文件
O_WRONLY 以只写方式打开文件
O_RDWR 以可读写方式打开文件。上述三种旗标是互斥的，也就是不可同时使用，但可与下列的旗标利用OR(|)运算符组合。
O_CREAT 若欲打开的文件不存在则自动建立该文件。
O_TRUNC 若文件存在并且以可写的方式打开时，此标志位会令文件长度重新清为0，也就是说文件内容清空。
O_APPEND 当读写文件时会从文件尾开始移动，也就是所写入的数据会以附加的方式加入到文件后面。
O_NONBLOCK 以不可阻断的方式打开文件，也就是无论有无数据读取或等待，都会立即返回进程之中。
O_SYNC 以同步的方式打开文件。
O_NOFOLLOW 如果参数pathname所指的文件为一符号连接，则会令打开文件失败。
O_DIRECTORY 如果参数pathname所指的文件并非为一目录，则会令打开文件失败。

mode： 文件存储权限
S_IRWXU00700 权限，代表该文件所有者具有可读、可写及可执行的权限。
S_IRUSR 或S_IREAD，00400权限，代表该文件所有者具有可读取的权限。
S_IWUSR 或S_IWRITE，00200 权限，代表该文件所有者具有可写入的权限。
S_IXUSR 或S_IEXEC，00100 权限，代表该文件所有者具有可执行的权限。
S_IRWXG 00070权限，代表该文件用户组具有可读、可写及可执行的权限。
S_IRGRP 00040 权限，代表该文件用户组具有可读的权限。

S_IWGRP 00020权限，代表该文件用户组具有可写入的权限。
S_IXGRP 00010 权限，代表该文件用户组具有可执行的权限。
S_IRWXO 00007权限，代表其他用户具有可读、可写及可执行的权限。
S_IROTH 00004 权限，代表其他用户具有可读的权限
S_IWOTH 00002权限，代表其他用户具有可写入的权限。
S_IXOTH 00001 权限，代表其他用户具有可执行的权限。

3、文件的读（read）操作

int read(int fd, unsigned char *buf, int size);
返回值：返回实际读取到的字节数，如果返回0，表示已到达文件尾或是无可读取的数据，此外文件读写位置会随读取到的字节移动。

参数：
fd：表示文件句柄，是由open函数得到
buf：read()函数会把fd 所指的文件传送count个字节到buf指针所指的内存中
size：要读取的字节数

4、写入操作

int write (int fd, const unsigned char *buf, int size);
返回值 ： 如果成功write()，就会返回实际写入的字节数。当有错误发生时则返回-1

参数：
fd：同上
buf：将要写入到文件里面的内容。
size：要写入的字节数

5、跳转操作

int64_t seek(int fd, int64_t pos, int whence)
返回值：成功时则返回目前的读写位置，也就是距离文件开头多少个字节，若有错误则返回-1。

参数：
fd：同上
pos：跳转的相对量，可正可负，表示相对位置的前后关系
whence：跳转的方向，whence取值如下所示

int SEEK_SET　=　　0;//将读写位置指向文件头后再增加offset个位移量。
int SEEK_CUR　=　　1;//以目前的读写位置往后增加offset个位移量。
int EEK_END　=　　2;//将读写位置指向文件尾后再增加offset个位移量。
注：当size参数=0;whence = SEEK_END;时返回值即为文件大小。

6、关闭操作

int close(int fd)

7.1、JNI代码：（有JNI_onLoad函数）

```
//fs.c
#include <unistd.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <stdlib.h>
#include <fcntl.h>

 int file_open(const char *filename, int flags){
    int fd;
    fd = open(filename, flags, 0666);
    if (fd == -1)
        return -1;
    return fd;
}

 int file_read(int fd, unsigned char *buf, int size){
    return read(fd, buf, size);
}

 int file_write(int fd, const unsigned char *buf, int size){
    return write(fd, buf, size);
}

 int64_t file_seek(int fd, int64_t pos, int whence){
    if (whence == 0x10000) {
        struct stat st;
        int ret = fstat(fd, &st);
        return ret < 0 ? -1 : st.st_size;
    }
    return lseek(fd, pos, whence);
}

 int file_close(int fd){
    return close(fd);
}



//jni.c
#define TAG "fs_jni"
```

```
#include <android/log.h>
#include "jniUtils.h"

static const char* const kClassPathName = "com/conowen/fs/FsActivity";

jint Java_com_conowen_fs_FsActivity_NativeFileOpen( JNIEnv* env, jobject
thiz,jstring filename,jint flags ){
        const char *filename_char = (*env)->GetStringUTFChars(env,filename,
NULL);
       return file_open(filename_char, flags);
}

jint Java_com_conowen_fs_FsActivity_NativeFileRead(JNIEnv* env, jobject
thiz,int fd,jbyteArray buf,jint size){
       unsigned char *buf_char = (char*)((*env)->GetByteArrayElements(env,buf,
NULL));
       return file_read(fd, buf_char,  size);
}
jint Java_com_conowen_fs_FsActivity_NativeFileWrite(JNIEnv* env, jobject
thiz,int fd,jbyteArray buf,jint size){
       unsigned char *buf_char = (char*)((*env)->GetByteArrayElements(env,buf,
NULL));
       return file_write(fd, buf_char,  size);
}
jlong Java_com_conowen_fs_FsActivity_NativeFileSeek(JNIEnv* env, jobject
thiz,int fd,jlong Offset,jint whence){
       return file_seek(fd, Offset,  whence);
}
jint Java_com_conowen_fs_FsActivity_NativeFileClose(JNIEnv* env, jobject
thiz,int fd){
       return file_close(fd);
}

/*****************************JNI
registration.***********************************/
static JNINativeMethod gMethods[] = {
   {"NativeFileOpen",       "(Ljava/lang/String;I)I",          (void
*)Java_com_conowen_fs_FsActivity_NativeFileOpen},
   {"NativeFileRead",        "(I[BI)I",                                  (void
*)Java_com_conowen_fs_FsActivity_NativeFileRead},
   {"NativeFileWrite",      "(I[BI)I",                        (void
*)Java_com_conowen_fs_FsActivity_NativeFileWrite},
   {"NativeFileSeek",       "(IJI)J",                          (void
*)Java_com_conowen_fs_FsActivity_NativeFileSeek},
   {"NativeFileClose",      "(I)I",                           (void
*)Java_com_conowen_fs_FsActivity_NativeFileClose},
```

```
};

int register_com_conowen_fs_FsActivity(JNIEnv *env) {
        return jniRegisterNativeMethods(env, kClassPathName, gMethods,
sizeof(gMethods) / sizeof(gMethods[0]));

}



//jniUtils.h
#ifndef _JNI_UTILS_H_
#define _JNI_UTILS_H_
#include <stdlib.h>
#include <jni.h>
#ifdef __cplusplus
extern "C"
{
#endif
int jniThrowException(JNIEnv* env, const char* className, const char* msg);
JNIEnv* getJNIEnv();
int jniRegisterNativeMethods(JNIEnv* env,
                        const char* className,
                        const JNINativeMethod* gMethods,
                        int numMethods);
#ifdef __cplusplus
}
#endif
#endif /* _JNI_UTILS_H_ */



//onLoad.cpp
#define TAG "fs_onLoad"
#include <android/log.h>
#include "jniUtils.h"
extern "C" {
extern int register_com_conowen_fs_FsActivity(JNIEnv *env);
}

static JavaVM *sVm;
/*
 * Throw an exception with the specified class and an optional message.
 */
int jniThrowException(JNIEnv* env, const char* className, const char* msg) {
    jclass exceptionClass = env->FindClass(className);
```

```
    if (exceptionClass == NULL) {
        __android_log_print(ANDROID_LOG_ERROR,
                            TAG,
                            "Unable to find exception class %s",
                            className);
        return -1;
    }

    if (env->ThrowNew(exceptionClass, msg) != JNI_OK) {
        __android_log_print(ANDROID_LOG_ERROR,
                            TAG,
                            "Failed throwing '%s' '%s'",
                            className, msg);
    }
    return 0;
}

JNIEnv* getJNIEnv() {
    JNIEnv* env = NULL;
    if (sVm->GetEnv((void**) &env, JNI_VERSION_1_4) != JNI_OK) {
        __android_log_print(ANDROID_LOG_ERROR,
                                            TAG,
                                            "Failed to obtain JNIEnv");
        return NULL;
    }
    return env;
}

/*
 * Register native JNI-callable methods.
 *
 * "className" looks like "java/lang/String".
 */
int jniRegisterNativeMethods(JNIEnv* env,
                            const char* className,
                            const JNINativeMethod* gMethods,
                            int numMethods){
    jclass clazz;

    __android_log_print(ANDROID_LOG_INFO, TAG, "Registering %s natives\n",
className);
    clazz = env->FindClass(className);
    if (clazz == NULL) {
        __android_log_print(ANDROID_LOG_ERROR, TAG, "Native registration
unable to find class '%s'\n", className);
        return -1;
```

```
    }
    if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
        __android_log_print(ANDROID_LOG_ERROR, TAG, "RegisterNatives failed
for '%s'\n", className);
        return -1;
    }
    return 0;
}
//Dalvik虚拟机加载C库时，第一件事是调用JNI_OnLoad()函数
jint JNI_OnLoad(JavaVM* vm, void* reserved) {
    JNIEnv* env = NULL;
    jint result = JNI_ERR;
        sVm = vm;

    if (vm->GetEnv((void**) &env, JNI_VERSION_1_4) != JNI_OK) {
        __android_log_print(ANDROID_LOG_ERROR, TAG, "GetEnv failed!");
        return result;
    }
    __android_log_print(ANDROID_LOG_INFO, TAG, "loading . . .");

    if(register_com_conowen_fs_FsActivity(env) != JNI_OK) {
        __android_log_print(ANDROID_LOG_ERROR, TAG, "can't load
register_com_conowen_fs_FsActivity");
        goto end;
        }
    __android_log_print(ANDROID_LOG_INFO, TAG, "loaded");

    result = JNI_VERSION_1_4;

end:
    return result;
}
```

7.2、Android.mk文件

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := fs
LOCAL_SRC_FILES := fs.c jni.c onLoad.cpp
LOCAL_LDLIBS  += -llog

include $(BUILD_SHARED_LIBRARY)
```

7.3、java层代码

```
/* author:conowen
 * data:2012.5.1
 * e-mail:conowen@hotmail.com
 */
package com.conowen.fs;

import java.io.UnsupportedEncodingException;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class FsActivity extends Activity {
	String filename = "/sdcard/test.txt";
	EditText writestrET;
	Button writeBT;
	Button readBT;
	Button seekBT;
	TextView readTV;
	String writeStr;
	byte[] buf_write;
	byte[] buf_read;
	int fd;

	int O_ACCMODE  =    0003;
	int O_RDONLY   =      00;
	int O_WRONLY   =      01;
	int O_RDWR     =      02;
	int O_CREAT    =    0100; /* not fcntl */
	int O_EXCL     =    0200; /* not fcntl */
	int O_NOCTTY   =   0400; /* not fcntl */
	int O_TRUNC    =   01000; /* not fcntl */
	int O_APPEND   =   02000;
	int O_NONBLOCK =   04000;
	int O_NDELAY   = O_NONBLOCK;
	int O_SYNC     =  010000;
	int O_FSYNC    =  O_SYNC;
	int O_ASYNC    =  020000;

	int SEEK_SET   =            0;//将读写位置指向文件头后再增加offset个位移
```

量。
int SEEK_CUR = 1;//以目前的读写位置往后增加offset个位移量。
int EEK_END = 2;//将读写位置指向文件尾后再增加offset个位移量。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.main);
	writestrET = (EditText) findViewById(R.id.writeET);
	writeBT = (Button) findViewById(R.id.writeBT);
	readBT = (Button) findViewById(R.id.readBT);
	seekBT = (Button) findViewById(R.id.seekBT);
	readTV = (TextView) findViewById(R.id.readTV);
	writeBT.setOnClickListener(new OnClickListener() {

		@Override
		public void onClick(View v) {
			// TODO Auto-generated method stub
			fd = NativeFileOpen(filename, O_CREAT | O_RDWR);
			System.out.println("fd_write---->" + fd);
			writeStr = writestrET.getText().toString();
			buf_write = writeStr.getBytes();
			int ret_write = NativeFileWrite(fd, buf_write,
buf_write.length);
			System.out.println("写入返回结果" + ret_write);
			NativeFileClose(fd);

		}
	});
	readBT.setOnClickListener(new OnClickListener() {

		@Override
		public void onClick(View v) {
			// TODO Auto-generated method stub
			fd = NativeFileOpen(filename, O_CREAT | O_RDWR);
			System.out.println("fd_read---->" + fd);
			buf_read = new byte[buf_write.length];
			int ret_read = NativeFileRead(fd, buf_read,
buf_write.length);

			System.out.println("读出返回结果" + ret_read);
			try {
				readTV.setText( new String(buf_read, "GB2312") + "");
			} catch (UnsupportedEncodingException e) {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                NativeFileClose(fd);
            }
        });
        seekBT.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                fd = NativeFileOpen(filename, O_CREAT  | O_RDWR);
                long Offset=20;
                long ret_seek =NativeFileSeek(fd, Offset, SEEK_CUR);

                System.out.println("seek返回结果" + ret_seek);

                NativeFileClose(fd);
                /*      1) 欲将读写位置移到文件开头时:
                            lseek（int fildes,0,SEEK_SET）；
                        2) 欲将读写位置移到文件尾时:
                            lseek（int fildes，0,SEEK_END）；
                        3) 想要取得目前文件位置时:
                            lseek（int fildes，0,SEEK_CUR）；
                返回值：当调用成功时则返回目前的读写位置，也就是距离文件开头多少个字节。若有错误则返回-1，errno 会存放错误代码。
                 * */
            }
        });

    }

    public native int NativeFileOpen(String filename, int flags);

    public native int NativeFileRead(int fd, byte[] buf, int sizes);

    public native int NativeFileWrite(int fd, byte[] buf, int sizes);

    public native long NativeFileSeek(int fd, long Offset, int whence);
    //Offset：偏移量，每一读写操作所需要移动的距离，单位是字节的数量，可正可负（向前移，向后移）。

    public native int NativeFileClose(int fd);

    static {
        System.loadLibrary("fs");
```

```
    }
}
```

最后记得在manifest.xml里面加上SD卡操作权限

```
   <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
   <uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```