

[conowen]大钟的专栏

☰ 目录视图

☰ 摘要视图

RSS 订阅

个人资料



conowen

☰

恒

访问：765917次

积分：7643

等级：BLOG 5

排名：第1309名

原创：58篇

转载：13篇

译文：0篇

评论：427条

文章搜索

博客专栏



大钟的ios开发之旅  
文章：4篇  
阅读：12386



大钟的Android\_NDK开发  
文章：5篇  
阅读：99352



Android学习笔记  
文章：24篇  
阅读：283518

- 文章分类
- android编译 (8)

Android学习笔记 (24)

JAVA学习 (1)

C与C++ (2)

Linux相关 (4)

Android相关 (8)

专家和你聊Web，速来报名

微信开发学习路线高级篇上线

免费公开课平台正式上线啦

恭喜July新书上市

Android的NDK开发(5)———Android JNI层实现文件的read、write与seek操作

分类：Android的NDK开发

2012-05-01 16:46

17005人阅读

评论(3)

收藏

举报

android

jni

file

null

exception

```
/*
 * author: conowen@大钟
 * E-mail: conowen@hotmail.com
 * site:http://www.idealpwr.com/
 * 深圳市动力思维科技发展有限公司
 * http://blog.csdn.net/conowen
 * 注：本文为原创，仅作为学习交流使用，转载请标明作者及出处。
 */
```

1、

在Android的java层实现文件的读写操作是非常简单的，可以参看之前写的博文：<http://blog.csdn.net/conowen/article/details/7296121>

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

```
[cpp]
01. 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

```
[java]
01. String filename = "/sdcard/test.txt";
```

flags：文件打开的方式

flag之间可以作“与”运算，如

```
[java]
01. 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

- 瑞芯微RK (3)
- 嵌入式相关 (1)
- 计算机相关 (2)
- Android多媒体&流媒体开发 (6)
- Android的NDK开发 (5)
- 所想所感 (1)
- 所想所感 ios (1)

- 文章存档
- 2015年09月 (1)
- 2014年11月 (4)
- 2014年04月 (2)
- 2012年08月 (6)
- 2012年07月 (4)
- 展开

- 阅读排行
- Android学习笔记(21)——  
【整理】Android-Recover (49160)
- Android的NDK开发(1)—— (45524)
- Android学习笔记(13)—— (33334)
- Android学习笔记(12)—— (26445)
- RKAndroidTool工具的各项 (22305)
- Android图形系统之Surface (21337)
- 【整理】Libav、FFmpeg (21279)
- Android的文件系统结构 (21223)
- Android的NDK开发(3)—— (21020)
- Android的NDK开发(2)—— (20140)

- 评论排行
- Android学习笔记(21)—— (74)
- 关于havenapetr-FFmpeg (63)
- Android多媒体开发 (3) (50)
- Android多媒体开发 (4) (46)
- Android多媒体开发 (5) (41)
- Android学习笔记(20)—— (15)
- Android学习笔记(12)—— (14)
- Android多媒体开发 (2) (14)
- Android学习笔记(1)—— (9)
- Android的NDK开发(2)—— (8)

- 推荐文章
- \*HTML抽屉效果的实现与展示
- \*2015年校招求职之旅
- \* iOS9使用提示框的正确实现方式
- \* 你不知道的JavaScript-Item18 JScrip的Bug与内存管理
- Android Studio中配置及使用OpenCV示例
- \* 最老程序员创业开发实训4--IOS平台下MVC架构

- 最新评论
- Android多媒体开发 (5) —— guonanyun: 楼主,可以提供一下完整demo的下载地址吗?
- Android的NDK开发(4)—— wangjg0317: 写得很好,谢谢楼

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

```
[cpp] 1
01. 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

```
[cpp] 1
01. 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

```
[cpp] 1
```

主，楼主能否写个http相关的例子？

Android多媒体开发（5）———u014453626: @zhf198909:你的问题我也出现过，经过反复播放视频，发现博主在onCreate中就已经放入M...

Android多媒体开发（5）———u014453626: @zhf198909:你的问题我也出现过，经过反复播放视频，发现博主在onCreate中就已经放入M...

【整理】SISD、MIMD、SIMD、mayfla: 非常棒，学习了

Android学习日记(21)———利sanbo\_xyz: java.lang.ClassNotFoundException: org.gjt.mm.mysql...

Android学习日记(13)———利wlccomeon: 使用的挺方便的，现在忘了，来看看。

Android显示系统之View与Surfacewzw88486969: 后面的这个例子不太好,在主线程中画图,然后在另一个线程中刷屏

Android多媒体开发（4）———JUNSON009: 可以共享一下libmad工程吗，git clone下载不了

Android多媒体开发（5）———allencheung2010: 楼主，你好。探讨一个问题在播放实时流的时候，会有出现停顿，我觉得是audiotrack.write(...

```
01. 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

```
[cpp] C ?
01. int SEEK_SET = 0; //将读写位置指向文件头后再增加offset个位移量。
02. int SEEK_CUR = 1; //以目前的读写位置往后增加offset个位移量。
03. int SEEK_END = 2; //将读写位置指向文件尾后再增加offset个位移量。
```

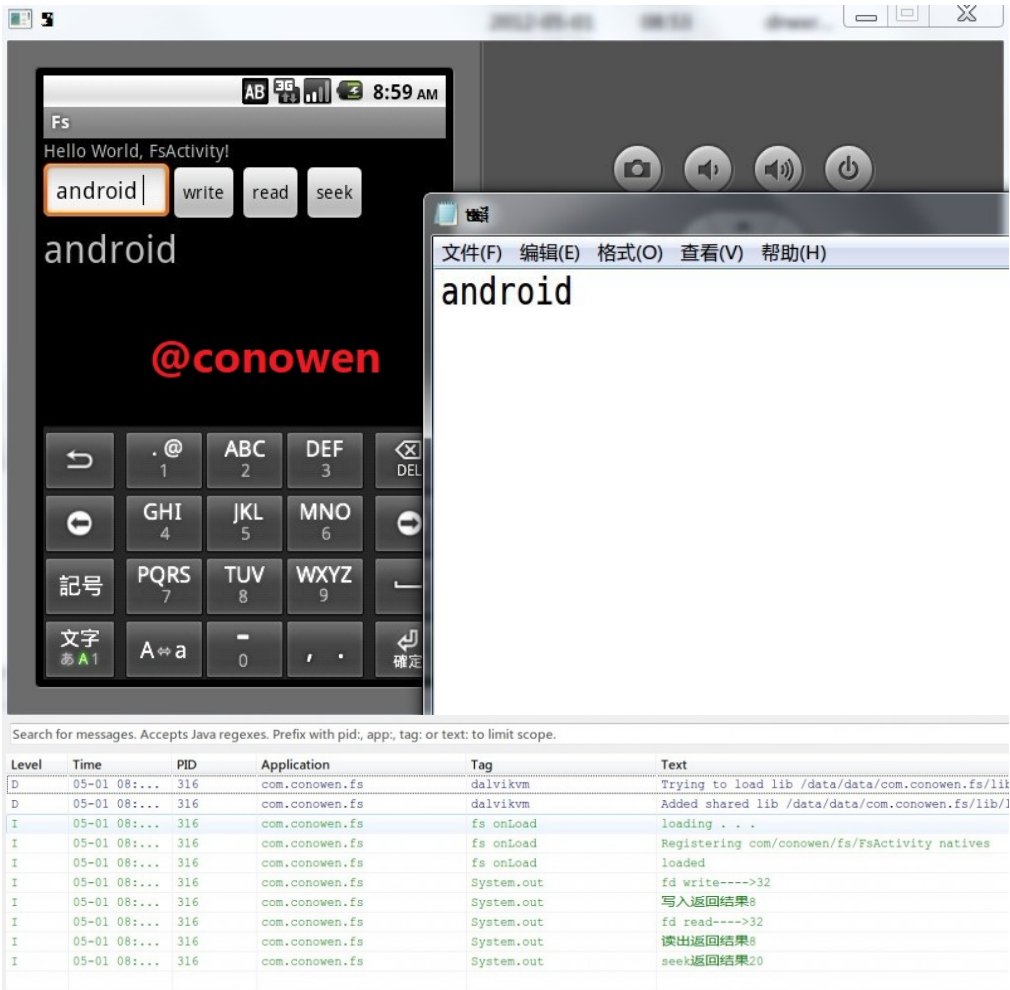
注：当size参数=0;whence = SEEK\_END;时返回值即为文件大小。

## 6、关闭操作

```
[cpp] C ?
01. int close(int fd)
```

## 7、简单示例

效果图：



### 7.1、JNI代码：（有JNI\_onLoad函数）

```
[cpp] C ?
01. //fs.c
```

```
02. #include <unistd.h>
03. #include <sys/stat.h>
04. #include <sys/time.h>
05. #include <stdlib.h>
06. #include <fcntl.h>
07.
08.
09. int file_open(const char *filename, int flags)
10. {
11.     int fd;
12.
13.     fd = open(filename, flags, 0666);
14.     if (fd == -1)
15.         return -1;
16.
17.     return fd;
18. }
19.
20. int file_read(int fd, unsigned char *buf, int size)
21. {
22.
23.     return read(fd, buf, size);
24. }
25.
26. int file_write(int fd, const unsigned char *buf, int size)
27. {
28.
29.     return write(fd, buf, size);
30. }
31.
32.
33. int64_t file_seek(int fd, int64_t pos, int whence)
34. {
35.
36.     if (whence == 0x10000) {
37.         struct stat st;
38.         int ret = fstat(fd, &st);
39.         return ret < 0 ? -1 : st.st_size;
40.     }
41.     return lseek(fd, pos, whence);
42. }
43.
44. int file_close(int fd)
45. {
46.
47.     return close(fd);
48. }
```

```
[cpp] C {}
01. //jni.c
02. #define TAG "fs_jni"
03.
04. #include <android/log.h>
05. #include "jniUtils.h"
06.
07.
08.
09. static const char* const kClassName = "com/conowen/fs/FsActivity";
10.
11.
12. jint
13. Java_com_conowen_fs_FsActivity_NativeFileOpen( JNIEnv* env, jobject thiz, jstring filename, jint
14. {
15.
16.     const char *filename_char = (*env)->GetStringUTFChars(env, filename, NULL);
17.
18.     return file_open(filename_char, flags);
19. }
20. jint
21. Java_com_conowen_fs_FsActivity_NativeFileRead(JNIEnv* env, jobject thiz, jint fd, jbyteArray buf,
22. {
23.
24.     unsigned char *buf_char = (char*)((env)->GetByteArrayElements(env, buf, NULL));
```

```

25.     return file_read(fd, buf_char, size);
26. }
27.
28. jint
29. Java_com_conowen_fs_FsActivity_NativeFileWrite(JNIEnv* env, jobject thiz,int fd,jbyteArray buf
{
30.
31.     unsigned char *buf_char = (char*)((env)->GetByteArrayElements(env,buf, NULL));
32.
33.     return file_write(fd, buf_char, size);
34. }
35.
36. jlong
37. Java_com_conowen_fs_FsActivity_NativeFileSeek(JNIEnv* env, jobject thiz,int fd,jlong Offset,jl
{
38.
39.     return file_seek(fd, Offset, whence);
40. }
41.
42. jint
43. Java_com_conowen_fs_FsActivity_NativeFileClose(JNIEnv* env, jobject thiz,int fd){
44.
45.     return file_close(fd);
46. }
47.
48. /*****JNI registration.*****/
49. static JNINativeMethod gMethods[] = {
50.     {"NativeFileOpen",      "    (void *)Java_com_conowen_fs_FsActivity_NativeFileOpen},
51.     {"NativeFileRead",     "    (void *)Java_com_conowen_fs_FsActivity_NativeFileRead},
52.     {"NativeFileWrite",    "    (void *)Java_com_conowen_fs_FsActivity_NativeFileWrite},
53.     {"NativeFileSeek",     "    (void *)Java_com_conowen_fs_FsActivity_NativeFileSeek},
54.     {"NativeFileClose",    "    (void *)Java_com_conowen_fs_FsActivity_NativeFileClose},
55. };
56.
57. int register_com_conowen_fs_FsActivity(JNIEnv *env) {
58.     return jniRegisterNativeMethods(env, kClassName, gMethods, sizeof(gMethods) / sizeof(g
59. }
60. }

```

```

[cpp] C
01. //jniUtils.h
02. #ifndef _JNI_UTILS_H_
03. #define _JNI_UTILS_H_
04.
05. #include <stdlib.h>
06. #include <jni.h>
07.
08. #ifdef __cplusplus
09. extern "C"
10. {
11. #endif
12.
13. int jniThrowException(JNIEnv* env, const char* className, const char* msg);
14.
15. JNIEnv* getJNIEnv();
16.
17. int jniRegisterNativeMethods(JNIEnv* env,
18.     const char* className,
19.     const JNINativeMethod* gMethods,
20.     int numMethods);
21.
22. #ifdef __cplusplus
23. }
24. #endif
25.
26. #endif /* _JNI_UTILS_H_ */

```

```

[cpp]
01. //onLoad.cpp
02. #define TAG "fs_onLoad"
03.
04. #include <android/log.h>
05. #include "jniUtils.h"
06.
07. extern "C" {
08.
09. extern int register_com_conowen_fs_FsActivity(JNIEnv *env);
10.
11. }
12.
13. static JavaVM *sVm;
14.
15. /*
16.  * Throw an exception with the specified class and an optional message.
17.  */
18. int jniThrowException(JNIEnv* env, const char* className, const char* msg) {
19.     jclass exceptionClass = env->FindClass(className);
20.     if (exceptionClass == NULL) {
21.         __android_log_print(ANDROID_LOG_ERROR,
22.             TAG,
23.             "Unable to find exception class %s",
24.             className);
25.         return -1;
26.     }
27.
28.     if (env->ThrowNew(exceptionClass, msg) != JNI_OK) {
29.         __android_log_print(ANDROID_LOG_ERROR,
30.             TAG,
31.             "Failed throwing '%s' '%s'",
32.             className, msg);
33.     }
34.     return 0;
35. }
36.
37. JNIEnv* getJNIEnv() {
38.     JNIEnv* env = NULL;
39.     if (sVm->GetEnv((void**) &env, JNI_VERSION_1_4) != JNI_OK) {
40.         __android_log_print(ANDROID_LOG_ERROR,
41.             TAG,
42.             "Failed to obtain JNIEnv");
43.         return NULL;
44.     }
45.     return env;
46. }
47.
48. /*
49.  * Register native JNI-callable methods.
50.  *
51.  * "className" looks like "java/lang/String".
52.  */
53. int jniRegisterNativeMethods(JNIEnv* env,
54.                             const char* className,
55.                             const JNINativeMethod* gMethods,
56.                             int numMethods)
57. {
58.     jclass clazz;
59.
60.     __android_log_print(ANDROID_LOG_INFO, TAG, "Registering %s natives\n", className);
61.     clazz = env->FindClass(className);
62.     if (clazz == NULL) {
63.         __android_log_print(ANDROID_LOG_ERROR, TAG, "Native registration unable to find class");
64.         return -1;
65.     }
66.     if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
67.         __android_log_print(ANDROID_LOG_ERROR, TAG, "RegisterNatives failed for '%s'\n", classl
68.         return -1;
69.     }
70.     return 0;
71. }
72. //Dalvik虚拟机加载C库时，第一件事是调用JNI_OnLoad()函数
73. jint JNI_OnLoad(JavaVM* vm, void* reserved) {
74.     JNIEnv* env = NULL;
75.     jint result = JNI_ERR;
76.     sVm = vm;
77.

```

```

78.     if (vm->GetEnv((void**) &env, JNI_VERSION_1_4) != JNI_OK) {
79.         __android_log_print(ANDROID_LOG_ERROR, TAG, "GetEnv failed!");
80.         return result;
81.     }
82.
83.     __android_log_print(ANDROID_LOG_INFO, TAG, "loading . . .");
84.
85.
86.     if(register_com_conowen_fs_FsActivity(env) != JNI_OK) {
87.         __android_log_print(ANDROID_LOG_ERROR, TAG, "can't load register_com_conowen_fs_FsActi
88.         goto end;
89.     }
90.
91.     __android_log_print(ANDROID_LOG_INFO, TAG, "loaded");
92.
93.     result = JNI_VERSION_1_4;
94.
95. end:
96.     return result;
97. }

```

## 7.2、Android.mk文件

```

[cpp]
01. LOCAL_PATH := $(call my-dir)
02.
03. include $(CLEAR_VARS)
04.
05. LOCAL_MODULE := fs
06. LOCAL_SRC_FILES := fs.c jni.c onLoad.cpp
07. LOCAL_LDLIBS += -llog
08.
09. include $(BUILD_SHARED_LIBRARY)

```

## 7.3、java层代码

```

[java]
01. /* author:conowen
02.  * data:2012.5.1
03.  * e-mail:conowen@hotmail.com
04.  */
05. package com.conowen.fs;
06.
07. import java.io.UnsupportedEncodingException;
08.
09. import android.app.Activity;
10. import android.os.Bundle;
11. import android.view.View;
12. import android.view.View.OnClickListener;
13. import android.widget.Button;
14. import android.widget.EditText;
15. import android.widget.TextView;
16.
17. public class FsActivity extends Activity {
18.     String filename = "/sdcard/test.txt";
19.     EditText writestrET;
20.     Button writeBT;
21.     Button readBT;
22.     Button seekBT;
23.     TextView readTV;
24.     String writeStr;
25.     byte[] buf_write;
26.     byte[] buf_read;
27.     int fd;
28.
29.     int O_ACCMODE = 0003;
30.     int O_RDONLY = 00;
31.     int O_WRONLY = 01;
32.     int O_RDWR = 02;
33.     int O_CREAT = 0100; /* not fcntl */
34.     int O_EXCL = 0200; /* not fcntl */

```

```

35.     int O_NOCTTY    = 0400; /* not fcntl */
36.     int O_TRUNC     = 01000; /* not fcntl */
37.     int O_APPEND    = 02000;
38.     int O_NONBLOCK  = 04000;
39.     int O_NDELAY    = O_NONBLOCK;
40.     int O_SYNC      = 010000;
41.     int O_FSYNC     = O_SYNC;
42.     int O_ASYNC     = 020000;
43.
44.     int SEEK_SET     = 0; //将读写位置指向文件头后再增加offset个位移量。
45.     int SEEK_CUR     = 1; //以目前的读写位置往后增加offset个位移量。
46.     int SEEK_END     = 2; //将读写位置指向文件尾后再增加offset个位移量。
47.
48.     /** Called when the activity is first created. */
49.     @Override
50.     public void onCreate(Bundle savedInstanceState) {
51.         super.onCreate(savedInstanceState);
52.         setContentView(R.layout.main);
53.         writestrET = (EditText) findViewById(R.id.writeET);
54.         writeBT = (Button) findViewById(R.id.writeBT);
55.         readBT = (Button) findViewById(R.id.readBT);
56.         seekBT = (Button) findViewById(R.id.seekBT);
57.         readTV = (TextView) findViewById(R.id.readTV);
58.         writeBT.setOnClickListener(new OnClickListener() {
59.
60.             @Override
61.             public void onClick(View v) {
62.                 // TODO Auto-generated method stub
63.                 fd = NativeFileOpen(filename, O_CREAT | O_RDWR);
64.                 System.out.println("fd_write---->" + fd);
65.                 writeStr = writestrET.getText().toString();
66.                 buf_write = writeStr.getBytes();
67.                 int ret_write = NativeFileWrite(fd, buf_write, buf_write.length);
68.                 System.out.println("写入返回结果" + ret_write);
69.                 NativeFileClose(fd);
70.
71.             }
72.         });
73.         readBT.setOnClickListener(new OnClickListener() {
74.
75.             @Override
76.             public void onClick(View v) {
77.                 // TODO Auto-generated method stub
78.                 fd = NativeFileOpen(filename, O_CREAT | O_RDWR);
79.                 System.out.println("fd_read---->" + fd);
80.                 buf_read = new byte[buf_write.length];
81.                 int ret_read = NativeFileRead(fd, buf_read, buf_write.length);
82.
83.                 System.out.println("读出返回结果" + ret_read);
84.                 try {
85.                     readTV.setText( new String(buf_read, "GB2312") + "");
86.                 } catch (UnsupportedEncodingException e) {
87.                     // TODO Auto-generated catch block
88.                     e.printStackTrace();
89.                 }
90.                 NativeFileClose(fd);
91.             }
92.         });
93.         seekBT.setOnClickListener(new OnClickListener() {
94.
95.             @Override
96.             public void onClick(View v) {
97.                 // TODO Auto-generated method stub
98.                 fd = NativeFileOpen(filename, O_CREAT | O_RDWR);
99.                 long Offset=20;
100.                 long ret_seek =NativeFileSeek(fd, Offset, SEEK_CUR);
101.
102.                 System.out.println("seek返回结果" + ret_seek);
103.
104.                 NativeFileClose(fd);
105.                 /*
106.                  1) 欲将读写位置移到文件开头时:
107.                     lseek (int fildes,0,SEEK_SET) ;
108.                  2) 欲将读写位置移到文件尾时:
109.                     lseek (int fildes, 0,SEEK_END) ;
110.                  3) 想要取得目前文件位置时:
111.                     lseek (int fildes, 0,SEEK_CUR) ;
112.                  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。
113.                  */

```



```
113.         }
114.     });
115.
116. }
117.
118. public native int NativeFileOpen(String filename, int flags);
119.
120. public native int NativeFileRead(int fd, byte[] buf, int sizes);
121.
122. public native int NativeFileWrite(int fd, byte[] buf, int sizes);
123.
124. public native long NativeFileSeek(int fd, long Offset, int whence);
125. //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)。
126.
127. public native int NativeFileClose(int fd);
128.
129. static {
130.     System.loadLibrary("fs");
131. }
132. }
```

最后记得在manifest加上SD卡操作权限

```
[html]
01. <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
02. <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```

版权声明：本文为博主原创文章，未经博主允许不得转载。

- 上一篇 Android的NDK开发(4)———JNI数据结构之JNINativeMethod
- 下一篇 Android多媒体开发（3）———使用Android NDK编译havenapetr-FFmpeg-7c27aa2

顶

7

踩

0

主题推荐 android color 开发 ndk jni 操作

猜你在找

- Android底层技术: Java层系统服务(Android Service)
- Android漫游记2——ELF可执行文件格式
- Android零基础实战开发 实战项目: 类FaceBook
- Android 44源码编译以及遇到的小问题
- Cocos2d-Lua手游开发基础篇
- 如何做好移动安全梆梆加固后的APK破解提取dex
- Android底层技术: HAL驱动开发
- 一个使用第三方静态库a的jni实例以及 ndk 使用第
- NinePatch图片制作从入门到精通
- Android AIDL来实现进程间通讯

 ExaVault


Best Business-Class FTP

Rated 5 Stars on Trustpilot ★★★★★

Start Your Free Trial

查看评论

3楼 changcsw 2014-12-15 21:01发表



非常不错，多谢分享

2楼 conowen 2012-08-27 16:47发表