

Android6.0 Bitmap存储以及Parcel传输源码分析

2016年05月24日 14:55:54 _houzhi 阅读数：7381 标签： android Bitmap 源码分析 更多

如果想要对Android Bitmap进行更多的操作，理解好Bitmap的实现将会有非常大的帮助，另外Android在6.0中增加了asm存储图片。这篇文章就通过源码来分析Android6.0中的Bitmap。主要分析Java层与native层的Bitmap，以及Bitmap的储存和Parcel传输。源码基于6.0，所以会有一些新的特性。

Bitmap存储方式以及包含的属性

计算机里面图片都是作为数组来存储的，而在Android中Bitmap也是一样。在Java层的Bitmap数组保存为mBuffer。而在native层，Bitmap有四种保存方式，在Bitmap.h文件中有个枚举类

```
1 enum class PixelStorageType {
2     Invalid,
3     External,
4     Java,
5     Ashmem,
6 };
```

Invalid表示图片已经失效了，一般图片free掉之后就会是这种状态。External是外部存储。Java是表示这个Bitmap对应着Java的Bitmap，此时Bitmap会保存着Java层Bitmap的存储数组的弱引用。而Ashmem则是对应着匿名共享内存，表示图片是存储在匿名共享内存当中。后三种类型在Bitmap中对应着一个union类型：

```
1 union {
2     struct {
3         void* address;
4         void* context;
5         FreeFunc freeFunc;
6     } external;
7     struct {
8         void* address;
9         int fd;
10        size_t size;
11    } ashmem;
12    struct {
13        JavaVM* jvm;
14        jweak jweakRef;
15        jbyteArray jstrongRef;
16    } java;
17 } mPixelStorage;
```

另外因为图片是直接保存在一片内存区域，那么它也可以保存在匿名共享内存当中，这就是Fresco在5.0之前干的事情，而将图片放到匿名共享内存当中，不会自动GC，应用会更加流畅，因为不在Java堆，也不用关心Java堆大小的限制而导致OOM。

另外还包含几种属性：

width, height: 图片宽度和高度

mDensity: 设备密度

colorType: 图片颜色类型，RGB或者gray等，图片通道数量

rowBytes: 用来表示图片像素的字节数

alphaType: 图像透明度类型，是否有透明度或者没有透明度

isMutable: 是否易变的

这些属性在进行Parcel传输的时候，都会通过Parcel传递，另外也是为了方便图片操作。

Java层与native层Bitmap

Bitmap的主要实现是在native层，Java层的Bitmap相当于是native层的接口。

Java层Bitmap

Bitmap实际上分为Java层和native层的，Java层包含了一个mBuffer数组用来存储像素，但总的来说Java层只是一个方便Java层应用访问的接口，最终还是通过native层来保存图片内容。在Java层中，我们常用的接口可能是createBitmap，getPixel，setPixel等，但实际上这些函数最终都是调用native层接口实现的，下面是Java层Bitmap的创建函数：

```
1 private static Bitmap createBitmap(DisplayMetrics display, int width, int height,
2     Config config, boolean hasAlpha) {
3     if (width <= 0 || height <= 0) {
4         throw new IllegalArgumentException("width and height must be > 0");
5     }
6     Bitmap bm = nativeCreate(null, 0, width, width, height, config.nativeInt, true); // 这!!!
7     if (display != null) {
8         bm.mDensity = display.densityDpi;
9     }
10    bm.setHasAlpha(hasAlpha);
11    if (config == Config.ARGB_8888 && !hasAlpha) {
12        nativeErase(bm.mFinalizer.mNativeBitmap, 0xff000000);
13    }
14    // No need to initialize the bitmap to zeroes with other configs;
15    // it is backed by a VM byte array which is by definition preinitialized
16    // to all zeroes.
17    return bm;
18 }
```

Bitmap还有很多native方法，具体可以看[Bitmap native 方法](#)。我们重点看createBitmap。

另外在Java层与native层对应的标记是mNativeBitmap变量，它保存的是native层Bitmap的指针地址。这样在native层通过reinterpret_cast即可得到具体的对象。关于这个，可以看Binder机制的实现[Android源码代理模式—Binder](#)。

native层

既然Bitmap的具体实现都是在native，那么看一下native层的Bitmap，native层的Bitmap在frameworks/base/core/jni/android/graphics/Bitmap.cpp中，对应的jni注册部分也在该文件下。看一下native层Bitmap的创建nativeCreate对应的Bitmap_creator函数：

```
1 static jobject Bitmap_creator(JNIEnv* env, jobject, jintArray jColors,
2                               jint offset, jint stride, jint width, jint height,
3                               jint configHandle, jboolean isMutable) {
4     SkColorType colorType = GraphicsJNI::legacyBitmapConfigToColorType(configHandle);
5     if (NULL != jColors) {
6         size_t n = env->GetArrayLength(jColors);
7         if (n < SkAbs32(stride) * (size_t)height) {
8             doThrowAIOOBE(env);
9             return NULL;
10        }
11    }
12
13    // ARGB_4444 is a deprecated format, convert automatically to 8888
14    if (colorType == kARGB_4444_SkColorType) {
15        colorType = kN32_SkColorType;
16    }
17
18    SkBitmap bitmap;
19    bitmap.setInfo(SkImageInfo::Make(width, height, colorType, kPremul_SkAlphaType));
20
21    Bitmap* nativeBitmap = GraphicsJNI::allocateJavaPixelRef(env, &bitmap, NULL);
22    if (!nativeBitmap) {
23        return NULL;
24    }
25
26    if (jColors != NULL) {
27        GraphicsJNI::SetPixels(env, jColors, offset, stride,
28                                0, 0, width, height, bitmap);
29    }
30
31    return GraphicsJNI::createBitmap(env, nativeBitmap,
32                                     getPremulBitmapCreateFlags(isMutable));
33 }
```

看看Bitmap的创建函数，从一创建开始，Bitmap就是先出现在native层的，Android中2D绘图是由skia框架实现的，在上述代码中就对应着SkBitmap。

而对于Java存储类型的Bitmap的创建是由GraphicsJNI的allocateJavaPixelRef完成的，allocateJavaPixelRef是从Java层分配像素数组，看看allocateJavaPixelRef的源码

```
1 android::Bitmap* GraphicsJNI::allocateJavaPixelRef(JNIEnv* env, SkBitmap* bitmap,
2                                                     SkColorTable* ctable) {
3     const SkImageInfo& info = bitmap->info();
4     if (info.fColorType == kUnknown_SkColorType) {
5         doThrowIAE(env, "unknown bitmap configuration");
6         return NULL;
7     }
8
9     size_t size;
10    if (!computeAllocationSize(*bitmap, &size)) {
11        return NULL;
```

```

12     }
13
14     // we must respect the rowBytes value already set on the bitmap instead of
15     // attempting to compute our own.
16     const size_t rowBytes = bitmap->rowBytes();
17     // 在这里分配
18     jbyteArray arrayObj = (jbyteArray) env->CallObjectMethod(gVMRuntime,
19                                                             gVMRuntime_newNonMovableArray,
20                                                             gByte_class, size); //在这创建Java层Array
21     if (env->ExceptionCheck() != 0) {
22         return NULL;
23     }
24     SkASSERT(arrayObj);
25     jbyte* addr = (jbyte*) env->CallLongMethod(gVMRuntime, gVMRuntime_addressOf, arrayObj); //获取地址
26     if (env->ExceptionCheck() != 0) {
27         return NULL;
28     }
29     SkASSERT(addr);
30     android::Bitmap* wrapper = new android::Bitmap(env, arrayObj, (void*) addr,
31                                                     info, rowBytes, ctable); //创建native层对象, 在Bitmap构造函数中mPixelStorage中存储了jweak引用。
32     wrapper->getSkBitmap(bitmap); // 在这里会将mPixelStorage的弱引用转换为强引用
33     // since we're already allocated, we lockPixels right away
34     // HeapAllocator behaves this way too
35     bitmap->lockPixels();
36
37     return wrapper;
38 }

```

可以看到, native层是通过JNI方法, 在Java层创建一个数组对象的, 这个数组是对应在Java层的Bitmap对象的buffer数组, 所以图像还是保存在Java堆的。而在native层这里它是通过weak指针来引用的, 在需要的时候会转换为strong指针, 用完之后又去掉strong指针, 这样这个数组对象还是能够被Java堆自动回收。可以看一下native层的Bitmap构造函数:

```

1 Bitmap::Bitmap(JNIEnv* env, jbyteArray storageObj, void* address,
2               const SkImageInfo& info, size_t rowBytes, SkColorTable* ctable)
3     : mPixelStorageType(PixelStorageType::Java) {
4     env->GetJavaVM(&mPixelStorage.java.jvm);
5     mPixelStorage.java.jweakRef = env->NewWeakGlobalRef(storageObj); //创建对Java层对象的弱引用
6     mPixelStorage.java.jstrongRef = nullptr;
7     mPixelRef.reset(new WrappedPixelRef(this, address, info, rowBytes, ctable));
8     // Note: this will trigger a call to onStrongRefDestroyed(), but
9     // we want the pixel ref to have a ref count of 0 at this point
10    mPixelRef->unref();
11 }

```

里面jstrongRef一开始是赋值为null的, 但是在bitmap的getSkBitmap方法会使用weakRef给他赋值:

```

1 void Bitmap::getSkBitmap(SkBitmap* outBitmap) {
2     assertValid();
3     android::AutoMutex _lock(mLock);
4     // Safe because mPixelRef is a WrappedPixelRef type, otherwise rowBytes()

```

```

5    // would require locking the pixels first.
6    outBitmap->setInfo(mPixelRef->info(), mPixelRef->rowBytes());
7    outBitmap->setPixelRef(refPixelRefLocked())->unref(); //refPixelRefLocked
8    outBitmap->setHasHardwareMipMap(hasHardwareMipMap());
9 }
10 void Bitmap::pinPixelsLocked() { //refPixelRefLocked会调用这个方法
11     switch (mPixelStorageType) {
12     case PixelStorageType::Invalid:
13         LOG_ALWAYS_FATAL("Cannot pin invalid pixels!");
14         break;
15     case PixelStorageType::External:
16     case PixelStorageType::Ashmem:
17         // Nothing to do
18         break;
19     case PixelStorageType::Java: {
20         JNIEnv* env = jniEnv();
21         if (!mPixelStorage.java.jstrongRef) {
22             mPixelStorage.java.jstrongRef = reinterpret_cast<jbyteArray>(
23                 env->NewGlobalRef(mPixelStorage.java.jweakRef));//赋值
24             if (!mPixelStorage.java.jstrongRef) {
25                 LOG_ALWAYS_FATAL("Failed to acquire strong reference to pixels");
26             }
27         }
28         break;
29     }
30 }
31 }

```

在native层随时添加删除一个强引用，这样有利于更好地配合Java堆的垃圾回收。图片的数组可能会是非常耗内存的。

在创建了native层的Bitmap后，再用GraphicsJNI的createBitmap创建Java层的Bitmap对象：

```

1  jobject GraphicsJNI::createBitmap(JNIEnv* env, android::Bitmap* bitmap,
2      int bitmapCreateFlags, jbyteArray ninePatchChunk, jobject ninePatchInsets,
3      int density) {
4      bool isMutable = bitmapCreateFlags & kBitmapCreateFlag_Mutable;
5      bool isPremultiplied = bitmapCreateFlags & kBitmapCreateFlag_Premultiplied;
6      // The caller needs to have already set the alpha type properly, so the
7      // native SkBitmap stays in sync with the Java Bitmap.
8      assert_premultiplied(bitmap->info(), isPremultiplied);
9
10     jobject obj = env->NewObject(gBitmap_class, gBitmap_constructorMethodID,
11         reinterpret_cast<jlong>(bitmap), bitmap->javaByteArray(),
12         bitmap->width(), bitmap->height(), density, isMutable, isPremultiplied,
13         ninePatchChunk, ninePatchInsets);//创建Java层Bitmap对象
14     hasException(env); // For the side effect of logging.
15     return obj;
16 }

```

在创建过程中，将刚刚创建的Java层Array和native层的bitmap指针也都会传给Java层Bitmap的构造函数。

另外对于External存储类型的Bitmap，它的创建如下：

```
1 Bitmap::Bitmap(void* address, void* context, FreeFunc freeFunc,
2               const SkImageInfo& info, size_t rowBytes, SkColorTable* ctable)
3     : mPixelStorageType(PixelStorageType::External) {
4     mPixelStorage.external.address = address;
5     mPixelStorage.external.context = context;
6     mPixelStorage.external.freeFunc = freeFunc;
7     mPixelRef.reset(new WrappedPixelRef(this, address, info, rowBytes, ctable));
8     // Note: this will trigger a call to onStrongRefDestroyed(), but
9     // we want the pixel ref to have a ref count of 0 at this point
10    mPixelRef->unref();
11 }
```

而Ashmem则是保存一个fd，以及asm地址和大小：

```
1 Bitmap::Bitmap(void* address, int fd,
2               const SkImageInfo& info, size_t rowBytes, SkColorTable* ctable)
3     : mPixelStorageType(PixelStorageType::Ashmem) {
4     mPixelStorage.ashmem.address = address;
5     mPixelStorage.ashmem.fd = fd;
6     mPixelStorage.ashmem.size = ashmem_get_size_region(fd);
7     mPixelRef.reset(new WrappedPixelRef(this, address, info, rowBytes, ctable));
8     // Note: this will trigger a call to onStrongRefDestroyed(), but
9     // we want the pixel ref to have a ref count of 0 at this point
10    mPixelRef->unref();
11 }
```

native层Bitmap会针对不同的存储类型，做不同的处理。

Parcel传递

首先在Java层Bitmap实现了Parcelable接口，所以他是能够通过Parcel来传递的,看看Bitmap的parcelable部分的源码：

```
1 public final class Bitmap implements Parcelable {
2     ...
3     /**
4      * Write the bitmap and its pixels to the parcel. The bitmap can be
5      * rebuilt from the parcel by calling CREATOR.createFromParcel().
6      * @param p    Parcel object to write the bitmap data into
7      */
8     public void writeToParcel(Parcel p, int flags) {
9         checkRecycled("Can't parcel a recycled bitmap");
10        if (!nativeWriteToParcel(mFinalizer.mNativeBitmap, mIsMutable, mDensity, p)) {
11            throw new RuntimeException("native writeToParcel failed");
12        }
13    }
```

```
13     }
14     public static final Parcelable.Creator<Bitmap> CREATOR
15         = new Parcelable.Creator<Bitmap>() {
16
17
18         public Bitmap More ...createFromParcel(Parcel p) {
19             Bitmap bm = nativeCreateFromParcel(p);
20             if (bm == null) {
21                 throw new RuntimeException("Failed to unparcel Bitmap");
22             }
23             return bm;
24         }
25         public Bitmap[] More ...newArray(int size) {
26             return new Bitmap[size];
27         }
28     };
29     ...
30 }
```

写入和读取分别调用了nativeWriteToParcel, nativeCreateFromParcel。先看看nativeWriteToParcel对应的native层方法Bitmap_writeToParcel:

```
1  static jboolean Bitmap_writeToParcel(JNIEnv* env, jobject,
2                                     jlong bitmapHandle,
3                                     jboolean isMutable, jint density,
4                                     jobject parcel) {
5
6     // 根据handle创建native层图片, 写入图片相关的一些附加信息, width, height, colorType, density等等。
7     if (parcel == NULL) {
8         SkDebugf("----- writeToParcel null parcel\n");
9         return JNI_FALSE;
10    }
11
12    android::Parcel* p = android::parcelForJavaObject(env, parcel);
13    SkBitmap bitmap;
14
15    android::Bitmap* androidBitmap = reinterpret_cast<Bitmap*>(bitmapHandle);
16    androidBitmap->getSkBitmap(&bitmap);
17
18    p->writeInt32(isMutable);
19    p->writeInt32(bitmap.colorType());
20    p->writeInt32(bitmap.alphaType());
21    p->writeInt32(bitmap.width());
22    p->writeInt32(bitmap.height());
23    p->writeInt32(bitmap.rowBytes());
24    p->writeInt32(density);
25
26    if (bitmap.colorType() == kIndex_8_SkColorType) {
27        SkColorTable* ctable = bitmap.getColorTable();
28        if (ctable != NULL) {
```

```

29         int count = ctable->count();
30         p->writeInt32(count);
31         memcpy(p->writeInplace(count * sizeof(SkPMColor)),
32             ctable->readColors(), count * sizeof(SkPMColor));
33     } else {
34         p->writeInt32(0);    // indicate no ctable
35     }
36 }
37 // 关键看这部分传输代码!!!
38 // Transfer the underlying ashmem region if we have one and it's immutable.
39 android::status_t status;
40 int fd = android::Bitmap->getAshmemFd(); // 获取匿名共享内存, 如果是图片是在匿名共享内存
41 if (fd >= 0 && !isMutable && p->allowFds()) { // 如果成功获取, 并且图片不是mutable, 同时允许fd(mAllowFds默认为True)
42     status = p->writeDupImmutableBlobFileDescriptor(fd); // 最终会直接把文件fd传过去
43     if (status) {
44         doThrowRE(env, "Could not write bitmap blob file descriptor.");
45         return JNI_FALSE;
46     }
47     return JNI_TRUE;
48 }
49 // 如果不能通过fd传递, 则传输Blob数据, 也就是相当于直接把像素数据传递过去。
50 // Copy the bitmap to a new blob.
51 bool mutableCopy = isMutable;
52
53 size_t size = bitmap.getSize();
54 android::Parcel::WritableBlob blob;
55 status = p->writeBlob(size, mutableCopy, &blob);
56 if (status) {
57     doThrowRE(env, "Could not copy bitmap to parcel blob.");
58     return JNI_FALSE;
59 }
60
61 bitmap.lockPixels();
62 const void* pSrc = bitmap.getPixels();
63 if (pSrc == NULL) {
64     memset(blob.data(), 0, size);
65 } else {
66     memcpy(blob.data(), pSrc, size);
67 }
68 bitmap.unlockPixels();
69
70 blob.release();
71 return JNI_TRUE;
72 }

```

从源码可以知道, 如果是匿名共享内存存储, 那么writeToParcel会通过匿名共享内存的方式将匿名共享文件传递过去, 看看writeDupFileDescriptor方法:

```

1 status_t Parcel::writeDupFileDescriptor(int fd)
2 {

```



```

3     int dupFd = dup(fd);
4     if (dupFd < 0) {
5         return -errno;
6     }
7     status_t err = writeFileDescriptor(dupFd, true /*takeOwnership*/);
8     if (err) {
9         close(dupFd);
10    }
11    return err;
12 }

```

如果是保存的数组数据，那么会直接将像素数据转换为Blob来传递。这是在6.0的源码中是如此的，在5.0的源码中，还没有增加这些东西，5.0的源码中只有普通的将像素存储区域memcpy来传。Android在3.0中增加了inBitmap，在4.4增加了不同大小的图片使用inBitmap。

而nativeCreateFromParcel对应了native层的Bitmap_createFromParcel，在6.0的源码里面源码如下(去掉了DEBUG_PARCEL):

```

1 static jobject Bitmap_createFromParcel(JNIEnv* env, jobject, jobject parcel) {
2     .....
3     // 一开始读取图片相关的一些信息，比如说width, height, density, colorType等等，并存于SkImageInfo中。并且对ColorType的相关处理，这些占用的内存都很小，关键看像素的传递
4
5     SkColorTable* ctable = NULL;
6     if (colorType == kIndex_8_SkColorType) {
7         int count = p->readInt32();
8         if (count < 0 || count > 256) {
9             // The data is corrupt, since SkColorTable enforces a value between 0 and 256,
10            // inclusive.
11            return NULL;
12        }
13        if (count > 0) {
14            size_t size = count * sizeof(SkPMColor);
15            const SkPMColor* src = (const SkPMColor*)p->readInplace(size);
16            if (src == NULL) {
17                return NULL;
18            }
19            ctable = new SkColorTable(src, count);
20        }
21    }
22
23    // Read the bitmap blob.
24    size_t size = bitmap->getSize();
25    android::Parcel::ReadableBlob blob;
26    android::status_t status = p->readBlob(size, &blob); //这里对应writeDupFileDescriptor
27    if (status) {
28        SkSafeUnref(ctable);
29        doThrowRE(env, "Could not read bitmap blob.");
30        return NULL;
31    }
32    // 关键看这部分传输代码!!!!
33    // Map the bitmap in place from the ashmem region if possible otherwise copy.

```

```
34     Bitmap* nativeBitmap;
35     if (blob.fd() >= 0 && (blob.isMutable() || !isMutable) && (size >= ASHMEM_BITMAP_MIN_SIZE)) {
36
37         // Dup the file descriptor so we can keep a reference to it after the Parcel
38         // is disposed.
39         int dupFd = dup(blob.fd());
40         if (dupFd < 0) {
41             blob.release();
42             SkSafeUnref(ctable);
43             doThrowRE(env, "Could not allocate dup blob fd.");
44             return NULL;
45         }
46
47         // Map the pixels in place and take ownership of the ashmem region.
48         nativeBitmap = GraphicsJNI::mapAshmemPixelRef(env, bitmap.get(),
49             ctable, dupFd, const_cast<void*>(blob.data()), !isMutable);
50         SkSafeUnref(ctable);
51         if (!nativeBitmap) {
52             close(dupFd);
53             blob.release();
54             doThrowRE(env, "Could not allocate ashmem pixel ref.");
55             return NULL;
56         }
57
58         // Clear the blob handle, don't release it.
59         blob.clear();
60     } else {
61
62         // Copy the pixels into a new buffer.
63         nativeBitmap = GraphicsJNI::allocateJavaPixelRef(env, bitmap.get(), ctable);
64         SkSafeUnref(ctable);
65         if (!nativeBitmap) {
66             blob.release();
67             doThrowRE(env, "Could not allocate java pixel ref.");
68             return NULL;
69         }
70         bitmap->lockPixels();
71         memcpy(bitmap->getPixels(), blob.data(), size);
72         bitmap->unlockPixels();
73
74         // Release the blob handle.
75         blob.release();
76     }
77
78     return GraphicsJNI::createBitmap(env, nativeBitmap,
79         getPremulBitmapCreateFlags(isMutable), NULL, NULL, density);
80 }
```

这个是与writeToParcel相互对应的，如果是asm则直接读取文件fd，如果是数据，则传对应数据。

总结

上面就是Bitmap在Java层与native的表现，Bitmap的操作基本都是在native层，Java层与native层通过一个handle相互对应。在6.0Bitmap总共有四种存储形式，也增加了asm的存储。在进行Parcel传输的时候，针对asm，Parcel传输的fd，这样能够减少很多内存的消耗。在Android6.0内部，很多图片也开始存储在asm里面了。不过在Java层还没有提供将图片保存在匿名共享内存里面。

白发千万不要染也别拔，这个方法在家做，自然变黑

盛世·熹歆



想对作者说点什么

用Parcelable接口传递Bitmap

Android中Intent传递对象有两个方法，一个是让对象实现Serializable接口，另一个是让对象实现Parcelable接口，S...

8286

来自：wike的专栏

Android4.0 Bitmap Parcel传输源码分析

很久之前就看到有网友遇到用Parcel传Bitmap的时候，会遇到因为图片太大而报错，都在讨论传输Bitmap的时候的...

1819

来自：破馨

Android Bitmap那些事

本文转载自:http://tlovezh.com/2016/05/31/Android-Bitmap%E9%82%A3%E4%BA%9B%E4%BA%8B/, 在平时的...

936

来自：MaizerBLog



开发一个app大概需要多少钱呢

百度广告

Android Bitmap变迁与原理解析（4.x-8.x）

App开发不可避免的要和图片打交道，由于其占用内存非常大，管理不当很容易导致内存不足，最后OOM，图片的...

338

来自：happylishang的专栏

Android Bitmap保存到SQLite

public class ImageDB extends SQLiteOpenHelper { /**数据库版本号*/ private final static int DATABASE_VERSI...

7690

来自：life02的专栏

android开发：activity之间如何传递bitmap数据

问题出现的场景：从activityA跳转到activityB,这activityB中获得bitmap数据，通过intent保存bitmap数据，但是返回...

1365

来自：shakdy的博客

Android传递Bitmap的几种简单方式

转载请注明出处：http://blog.csdn.net/zhangphil Android传递Bitmap的几种简单方式 一，通过Intent的Bundle。比...

8850

来自：sinat_30474567的博客

Android4.4 Surface从java到native的创建过程

研究Surface的创建流程需要一个精简的流程，不然跳转太多，脑子都乱了。先讲一个大致的过程：ViewRootImpl在...

158

来自：ywlyg的专栏



白发千万不要染也别拔，这个方法在家做，自然变黑

盛世·熹歆

<div>将bitmap对象保存到本地，返回保存图片路径</div> <div>private static final String SD_PATH = "/sdcard/dskqxt/pic/"; private static final String IN_PATH...</div>		👁 2.3万
来自： snow_lyGirl		
<div>文章热词</div> <div>机器学习 机器学习课程 机器学习教程 深度学习视频教程 深度学习学习</div>		
<div>相关热词</div> <div>android存储bitmap c++ 实现存储bitmap图像类 android bitmap 复制bitmap android6.0 uri android6.0和7.0 python教程+chm python教程百家号</div>		
<div>C++保存Bitmap图片</div> <div>关于Bitmap图片4字节对齐，Bitmap图片保存到磁盘时为什么需要4字节对齐,可以参考https://blog.csdn.net/xiaosua...</div>		👁 664
来自： anjisi的博客		
<div><div><div><div>fanfan</div><div>关注</div><div>109篇文章</div></div></div><div><div><div>一口仨馍</div><div>关注</div><div>57篇文章</div></div></div><div><div><div>aspook</div><div>关注</div><div>150篇文章</div></div></div><div><div><div>freshui</div><div>关注</div><div>52篇文章</div></div></div></div>		
<div>Android bitmap对象序列化</div> <div>由于BitMap没有实现序列化接口，所以当序列化到本地或者在activity之间序列化传递时，会报异常。 在这里介绍个...</div>		👁 899
来自： Richard.Dai的专栏		
<div>Android基础 -- Activity之间传递数据（bitmap和map对象）</div> <div>做项目的时候需要用到在2个activity之间传递一些数据，之前做的都是</div>		👁 4.6万
来自： xueerfei的专栏		
<div>Android Bitmap深入介绍（一）---基础</div> <div>在Android应用开发中，我们经常需要跟图片打交道，而图片一个很麻烦的问题是占用内存非常大，经常导致OOM...</div>		👁 2.4万
来自： 破磬		
<div><div><div>gk-tzy解决方案</div><div>百度广告</div></div></div>		
<div>Intent/Bundle传递Bitmap的时候失败甚至崩溃</div> <div>在Android开发中，我们有时候需要传递一个Bitmap给一个Activity。我们最直接的做法便是通过Intent/Bundle...</div>		👁 3166
来自： 阿飞的博客		
<div>关于GDI+在透明Bitmap上绘制时边缘有黑色重影的解决（WPF中）</div> <div>最近在做软件工程课程的作业，其中有一个地方需要用到GDI+绘制图像并将句柄传给托盘当作托盘图标，可是因为...</div>		👁 248
来自： q886yes的博客		
<div>VC++ SaveBitmapToFile</div> <div>void SaveBitmapToFile(BYTE* pBitmapBits, LONG IWidth, LONG IHeight,WORD wBitsPerPixel, LPCTSTR lpsz...</div>		👁 884
来自： sdafrma4的专栏		
<div>量化GDI+：快速Bitmap读写像素</div> <div>写在前面的话： 本文针对GDI+下Bitmap操作（Get/SetPixel）进行测试，而非寻求最快速的位图处理方式。如果你...</div>		👁 1928
来自： songshu5555的专栏		
<div>VC/MFC实现：位图CBitmap对象保存成为bmp，bmp转jpg，截屏保存jpg（GDI+）</div> <div>位图对象保存为bmp： 加载位图及路径 strExtension = "bmp"; m_strFile = filedlg.GetPathName() + '.' + strExtensio...</div>		👁 1053
来自： WHEgqing的专栏		

<div>为什么说白发不能拔也不建议染，白头发该怎么处理？</div> <div><div></div><div>盛世·熯燚</div></div>		
<div>拍照+剪裁+bitmap</div> <div>启动相机拍摄 Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE); //下面这句指定调用相机拍照后...</div>	<div>2885</div>	
<div>Android系统源码分析-bitmap的加载</div> <div>引导在Android的开发中,有图片是非常常见的了,但是对于图片的加载 处理遇到问题也是经常出现的,对于开发者而言...</div>	<div>1293</div>	来自： 罗享的博客
<div>jni操作bitmap</div> <div>JNIEXPORT jint JNICALL JNI_SetImage(JNIEnv * env, jobject obj, jobject bitmap) { char *pData=0; ...</div>	<div>1109</div>	
<div>【Java CV与Android】JavaCV实现IplImage与Bitmap的相互转换</div> <div>代码简单，如下：/** * IplImage转化为Bitmap * @param iplImage * @return */ public Bitmap...</div>	<div>4645</div>	来自： zgjl2012的专栏
<div>android byte[]数组，bitmap，drawable之间的相互转换</div> <div>Byte[]转Bitmap BitmapFactory.decodeByteArray(data, 0, data.length); 复制代码 Bitmap转Byte[] ...</div>	<div>1.3万</div>	来自： Yelbosh的专栏
<div>白发千万不要染也别拔，这个方法在家做，自然变黑</div> <div><div></div><div>盛世·熯燚</div></div>		
<div>Parcel数据传输过程，简要分析Binder流程</div> <div>Android Binder框架跨进程传输的，基本都是通过Parcel来封装的数据[注1]，这个是十分底层的实现了。对于Frame...</div>	<div>3462</div>	来自： freshui的专栏
<div>3分钟教你图解Bitmap编码传输</div> <div>在Android的图片传输的流程为： Created with Raphaël 2.1.2Bitmap二进制流Byte数组Base64编码String进行传输B...</div>	<div>194</div>	来自： 江子涛Tesla的专栏
<div>Android中Parcelable接口跨进程传递复杂数据的应用</div> <div>在前面的一篇文章基于Android应用开发的跨进程通信实现(IPC)>介绍了通过跨进程实现基本数值类型(String)的传递...</div>	<div>1545</div>	来自： qq_27485253的博客
<div>Android环境下通过SOCKET传递Parcel包并解出数据的例子</div> <div>之前做过了在android下通过socket发送数据的实验，也做过了parcel包的制作和解包的实验（这两个实验的源程序...</div>	<div>781</div>	来自： 沧海一声笑的专栏
<div>Android进程间通信系列-----进程间的数据传递载体Parcel</div> <div>Parcel是一种数据的载体，用于承载希望通过IBinder发送的相关信息（包括数据和对象引用）。也就是说Parcel是...</div>	<div>813</div>	来自： 用我一生换你十年天真无邪
<div>揭秘：头上长白发竟是身体缺了它？饭后吃点它，白发轻松变黑发！</div> <div><div></div><div>探可黑发·熯燚</div></div>		
<div>使用Bitmap保存图片文件到指定路径</div> <div>使用Bitmap保存图片文件到指定路径</div>	<div>1256</div>	来自： An_nA的博客

<div>图片流，bitmap保存</div> <div><div>/** * 保存方法 */ public void saveBitmap(Bitmap bitmap) { Long time = System.currentTimeMillis(); ...</div><div>来自： 安卓学习乐园</div></div>	<div>👁 2278</div>
<div>Android Bitmap保存 以及 屏幕截取工具类</div> <div>Bitmap保存 以及 屏幕截取工具类</div>	<div><div>👁 802</div><div>来自： 总Li`的专栏</div></div>
<div>android中保存Bitmap图片到指定文件夹中的方法</div> <div><div>/** 保存方法 */ public void saveBitmap() { Log.e(TAG, "保存图片"); File f = new File("/sdcard/nameca...</div><div>来自： J 灬安之若死</div></div>	<div>👁 671</div>
<div>Bitmap实现圆，保存Bitmap到文件</div> <div><div>/** * 转换图片成圆形 * @param bitmap 传入Bitmap对象 * @return */ public static Bitmap t...</div><div>来自： fydsw1314的专栏</div></div>	<div>👁 786</div>
<div>为什么说白发不能拔也不建议染，白发该怎么处理？</div> <div>崇贺·熾燚</div>	
<div>Android上在两个Activity之间传递Bitmap对象</div> <div>通过异步Task网络下载图片，实现图片保存为内部对象在多个Activity之间共享使用，以及Java对象串行化之后在Ac... 来自： 关注微信公众号【OpenC...</div>	<div>👁 8651</div>
<div>android开发之bitmap使用</div> <div>bitmap是android中重要的图像处理工具类，通过bitmap可以对图像进行剪切、旋转、缩放等操作，同时还可以指定... 来自： 江南一点雨的专栏</div>	<div>👁 4148</div>
<div>深入了解Bitmap源码解析及经验总结</div> <div>Bitmap的分析与使用在Android应用里，最耗费内存的就是图片资源。而且在Android系统中，读取位图Bitmap时， ... 来自： 这个时代，作为程序员可...</div>	<div>👁 2074</div>
<div>Android Activity之间传递图片(Bitmap)的方法</div> <div>、这篇文章介绍了Android Activity之间传递图片(Bitmap)的方法。 来自： SirSnow的博客</div>	<div>👁 929</div>
<div>MFC CBitmapButton+文字写入</div> <div>在class view的工程目录下，添加类CImgButton，继承自CBitmapButton。选择CImgButton类的property，重载Dra... 来自： CodeHoMo的专栏</div>	<div>👁 435</div>
<div>北京戒烟特困难，不吸烟就难受？三天戒烟方法，只需这个.....</div> <div>喜瀛洲健康·熾燚</div>	
<div>android parcel传输复杂数据结构</div> <div>先传送结构体大小，再传数据。注意结构体内存对齐。 <i>/*send*/ Parcel data, reply; char *p = (char *)&obj) data.writ...</i> 来自： LBO4031的专栏</div>	<div>👁 710</div>
<div>Parcalable接口使用（ android传递结构体数据的方法）</div> <div>对于Android来说传递复杂类型，主要是将自己的类转换为基础的字节数组，Activity之间传递数据是通过Intent实现... 来自： sun_x_t的专栏</div>	<div>👁 1619</div>
<div>Android6.0源码分析之Settings（二）</div> <div>本博文主要讲述6.0列表的加载 首先外层布局 setContentView(mIsShowingDashboard ? R.layout.settings_main...</div> <div>来自： fanfan</div>	<div>👁 4751</div>

<div>持久化保存Parcelable实践</div> <div>最近突发奇想，希望持久化保存Activity的信息，做应用恢复使用，而在Activity中最基本的Intent，Bundle都是基于P...</div>		👁 2434
<div>安装方式B--使用ClouderaManager的Parcels包进行安装</div> <div>安装方式B--使用ClouderaManager的Parcels包进行安装安装方式B使用从ClouderaManager依赖库下载的包进行安...</div>		👁 4492
<div><div></div><div><div>Make Travel Moments Big</div><div>广告 Hilton Hotels and Resorts</div></div></div> <div>Visit Site</div>		①
<div>Android保存Bitmap到本地图库</div> <div>原文->http://stormzhang.github.io/android/2014/07/24/android-save-image-to-gallery/ 最近有些用户反映保存图片...</div>		👁 2.1万
<div>安卓保存bitmap到本地</div> <div>在用picasso的时候，想把获取的的bitmap保存到本地： Picasso.with(getActivity()).load(imgUrls.get(position)).into(...</div>		👁 1895
<div>android bitmap保存为文件及读取</div> <div>android bitmap保存为文件及读取</div>		👁 6292
<div>Android—将Bitmap图片保存到SD卡目录下或者指定目录</div> <div>直接上代码就不废话啦 一：保存到SD卡下 File file = new File(Environment.getExternalStorageDirectory(), System....</div>		👁 2.3万
<div>将Bitmap保存为文件File的方法</div> <div>/** * 保存文件 * @param bm * @param fileName * @throws IOException */ ...</div>		👁 3.3万
<div>不可思议的OOM</div> <div>作者：陶菜菜 链接：http://www.jianshu.com/p/e574f0ffdb42 来源：简书 著作权归作者所有。商业转载请联系作者...</div>		👁 2813
<div><div></div><div><div>VMware软件定义存储：原理剖析和设计指南</div><div>云计算与虚拟化技术丛书</div></div></div>		
<div>IntelliJ IDEA 最新注册码（截止到2019年5月4日）</div> <div>温馨提示：本教程的 GitHub 地址为「intellij-idea-tutorial」，欢迎感兴趣的童鞋Star、Fork，纠错。IntelliJ IDEA 注...</div>		👁 131521
<div>MyBatis——mapper.xml提升指南</div> <div>MyBatis常见细节问题 一、前言 MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映...</div>		👁 9977

12/5/2018			Android6.0 Bitmap存储以及Parcel传输源码分析 - 破磬 - CSDN博客		
如何设计一个灵活的 MySQL 数据表，应对灵活多变的需求			 708		
我曾设计过一个活动报名数据表，每次发布的活动都不一样，需要的字段也不同。按照平常的业务设计理念，需要...			来自： GitChat		
VSCode设置中文语言显示			 43412		
Vscode是一款开源的跨平台编辑器。默认情况下，vscode使用的语言为英文(us)，如何将其显示语言修改成中文...			来自： 飞扬的博客		
【《Unity Shader入门精要》 提炼总结】(十四)第十四章·开启深度写入的半透明效果&ShaderLab的混合命...			 27		
本文由@唐三十胖子出品，转载请注明出处。 文章链接： https://blog.csdn.net/iceSony/article/details/8467382...			来自： 唐三十胖子的博客		
最新迅雷“应版权方要求，文件无法下载”的解决办法			 195150		
迅雷下载有的电影电视剧的时候会出现：应版权方要求，文件无法下载，或者显示迅雷任务包含违规内容 无法继续...			来自： 徐奕的专栏		
Webstorm 最新激活码 多种破解方式(持续更新...)			 62030		
方法：License server 注册 安装完成，打开Webstorm，在弹出的License Activation窗口中选择“License server”，...			来自： 老妖儿的博客		
SQL提升（二）			 10030		
Sql表操作提升 一、前言 Sql是最重要的关系数据库操作语言，现在基本上任何与数据库相关的操作都离不开s...			来自： 愤怒的懒洋洋的博客		
Postman 使用方法详解			 165899		
一、Postman背景介绍 用户在开发或者调试网络程序或者是网页B/S模式的程序的时候是需要一些方法来跟踪网页...			来自： fxbn123的博客		
2018最新Web前端经典面试题及答案			 333146		
本篇收录了一些面试中经常会遇到的经典面试题以及自己面试过程中遇到的一些问题，并且都给出了我在网上收集...			来自： wdlhao的博客		
（二）MyBatis核心组件（配图详解&代码实现）			 544		
MyBatis的核心组件分为4个部分 SqlSessionFactoryBuilder（构造器）：根据xml或java代码生成SqlSessionFactory...			来自： 青衣煮茶		
史上最简单的 SpringCloud 教程 终章			 1205866		
转载请标明出处： http://blog.csdn.net/forezp/article/details/70148833 本文出自方志朋的博客 错过了这一篇，你可...			来自： 方志朋的专栏		
pyCharm最新2018激活码			 1183953		
本教程对jetbrains全系列可用例：IDEA、WebStorm、phpstorm、clion等 因公司的需求，需要做一个爬取最近上映...			来自： 昌昌		
精选！15个必备的VSCode插件			 187754		
Visual Studio Code 是由微软开发的一款免费、跨平台的文本编辑器。由于其卓越的性能和丰富的功能，它很快就...			来自： Dean		
整理了10个干净、好用的BT、磁力链搜索网站给大家			 68790		
现在越来越流行在线看视频了，但是对于我得收藏癖爱好者，还是希望可以有比较好的资源网站的，尤其是种子、...			来自： YXAPP的技术分享		
webstorm 2018 激活破解方法大全			 653800		
webstorm 作为最近最火的前端开发工具,也确实对得起那个价格,但是秉着勤俭节约的传统美德,我们肯定是能省则省...			来自： 唐大帅的编程之路		

<div><div><div>Beyond Compare 4注册码 Beyond Compare 4注册激活码下载(附破解教)</div><div>下载来源出处：Beyond Compare 4注册码 Beyond Compare 4注册码是一款针对“Beyond Compare 4”而制作的激...</div></div></div>	<div><div>27737</div><div></div></div>
<div><div><div>Swagger教程二</div><div>Swagger搭建Restful接口教程二 一、前言 上一章节我们说的是swagger-ui也就是swagger1,接下来我们说的是...</div></div></div> <div>来自：愤怒的懒洋洋的博客</div>	<div><div>22573</div><div></div></div>
<div><div><div>门罗币 xmr 超级详细的CPU xmr挖矿教程</div><div>门罗币 xmr 最详细的CPU 挖矿教程 基础 CUP 挖矿教程 如何挖矿？ Step1:获得一个钱包地址 钱包分为两个部分讲,...</div></div></div> <div>来自：qq_39863517的博客</div>	<div><div>31065</div><div></div></div>
<div><div><div>2017智慧树 伟大的红楼梦测试答案题库</div><div>伟大的红楼梦智慧树答案伟大的《红楼梦》章节答案 伟大的《红楼梦》智慧树答案，智慧树测验答案课后习题答案 ...</div></div></div> <div>来自：tangnanman0029的博客</div>	<div><div>100144</div><div></div></div>
<div><div><div>迅雷因版权方无法下载解决方法，附带破解不限速方法</div><div>迅雷下载有的电影电视剧的时候会出现：应版权方要求，文件无法下载，或者显示迅雷任务包含违规内容 无法继续...</div></div></div> <div>来自：boy火巨的博客</div>	<div><div>30962</div><div></div></div>
<div><div><div>支付宝赚赏金的4种玩法（引流+变现日入200+）</div><div>支付宝赚赏金</div></div></div> <div>来自：Martin 的博客</div>	<div><div>32024</div><div></div></div>
<div><div><div>Navicat for MySQL 安装和破解（完美）</div><div>Navicat工具使用方便，但需要破解。不然你有钱就买吧，没关系的，哈哈哈1、安装Navicat软件 安装成功之后进行...</div></div></div> <div>来自：一次次尝试</div>	<div><div>128519</div><div></div></div>
<div><div><div>最新最详细的黑苹果10.13.4安装教程</div><div>最新最全求详细的13.4的黑苹果安装教程，带安装工具和镜像！安装10.13的过程中，在论坛High Sierra版能查到的...</div></div></div> <div>来自：qq_28735663的博客</div>	<div><div>149449</div><div></div></div>
<div><div><div>Swagger使用指南</div><div>1：认识SwaggerSwagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。...</div></div></div> <div>来自：sanyaoxu_3的博客</div>	<div><div>61558</div><div></div></div>
<div><div><div>什么是B-树？</div><div>什么是B-树呢？B-树全名 Balance Tree，读做B树(中间的-，只是分隔作用，不要读做B减树哦)。# B树的特征 ...</div></div></div> <div>来自：卖克的专栏</div>	<div><div>15498</div><div></div></div>
<div><div><div>Python的Pillow库进行图像文件处理（配图详解）</div><div>本文详解的讲解了使用Pillow库进行图片的简单处理，使用PyCharm开发Python的详细过程和各种第三方库的安装...</div></div></div> <div>来自：青衣煮茶</div>	<div><div>759</div><div></div></div>
<div><div><div>排序算法（三）起泡排序验证性实验</div><div>请创建一个一维整型数组用来存储待排序关键词，关键词从数组下标为1的位置开始存储，下标为0的位置不存储关...</div></div></div> <div>来自：青衣煮茶</div>	<div><div>481</div><div></div></div>
<div><div><div>C#Winform窗口移动</div><div>在我们将Winform自带的边框隐藏之后，我们需要自己编写窗口的移动。 思路就是1.获得点击左键时当前鼠标的坐...</div></div></div> <div>来自：Maybe_ch的博客</div>	<div><div>21828</div><div></div></div>
<div><div><div>MySql数据库优化必须注意的四个细节（方法）</div><div>MySQL 数据库性能的优化是 MySQL 数据库发展的必经之路， MySQL 数据库性能的优化也是 MySQL 数据库前...</div></div></div> <div>来自：青衣煮茶</div>	<div><div>463</div><div></div></div>

- 开源改变了小米什么？

小米与开源的渊源还要从 Android 的开源说起。作者 | 唐小引发自小米开源技术峰会现场 出品 | CSDN (ID: CSDN...

来自: CSDN资讯
- SQL提升（一）

Sql不常见关键字提升 一、前言 Sql是最重要的关系数据库操作语言，现在基本上任何与数据库相关的操作都离不开...

来自: 愤怒的懒洋洋的博客
- IntelliJ IDEA（2018）安装详解

第一步：进入官网下载IntelliJ IDEA https://www.jetbrains.com/idea/download/#section=windows，选择适合版本下...

来自: 闲暇时光
- （二）Maven的坐标和依赖&利用Maven实现邮件发送

本文中 将《Maven实战》中对坐标和依赖的定义展示给初学Maven的程序猿们，并加上书中实例展示，具体详细请...

来自: 青衣煮茶
- Proxyee-down的下载与安装教程

Proxyee-down是monkeyWie在Github上的一个开源项目，向作者致敬。最新版的Proxyee-down为3.12（2018.10...

来自: shadandeajian的博客
- 史上最全Java面试题（带全部答案）

今天要谈的主题是关于求职，求职是在每个技术人员的生涯中都要经历多次。对于我们大部分人而言，在进入自己...

来自: 林老师带你学编程
- 军事理论课答案（西安交大版）

1.1 1 【单选题】我国陆地领土面积排名世界第几？（C） A、 1 B、 2 C、 3 D、 4 2 【单选题】以下哪个国家不属于...

来自: ling_wang的博客
- 各大磁力种子搜索引擎对比

现在磁力种子搜索引擎质量参差不齐，现在就重点整理几个常用的种子搜索站，做个对比分析 1.屌丝搜-最懂屌丝的...

来自: lizhengnanhua的专栏
- 100个小学生猜字谜大全及答案

100个小学生猜字谜大全及答案 1.字谜：山上还有山。猜一字，答案是:出 2.字谜：十张口，一颗心。猜一字，答案...

来自: 欢迎光临 包国工作室
- DirectX修复工具增强版

最后更新：2018-11-23 DirectX修复工具最新版：DirectX Repair V3.7 增强版 NEW! 版本号：V3.7.0.26539 大小: 1...

来自: VBcom的专栏



_houzhi

关注

原创	粉丝	喜欢	评论
108	50	12	21

等级: 博客 5

访问: 20万+

积分: 3123

排名: 1万+

勋章: 



最新文章

- Hugo源码分析
- Android MessageQueue源码分析
- 为什么Activity生命周期函数是运行在UI线程
- AppWidget源码分析（2）---updateAppWidget过程分析.md
- AppWidget源码分析(1)---接口类

博主专栏

- 

Android源码分析设计模式

阅读量：19963 6 篇
- 

深入理解Android

阅读量：83443 17 篇

个人分类

- 安卓源码解析 17篇
- 设计模式 5篇
- 开源项目源码解析 4篇
- Android 53篇
- java高级特性 3篇

展开

归档

- 2016年11月 1篇