Android系统匿名共享内存Ashmem（Anonymous Shared Memory）简要介绍和学习计划

2011年08月05日 00:11:20 罗升阳 阅读数：85119 标签： android descriptor layout server string 更多

在Android系统中，提供了独特的匿名共享内存子系统Ashmem（Anonymous Shared Memory），它以驱动程序的形式实现在内核空间中。它有两个特点，一是能够辅助内存管理系统来有效地管理不再使用的内存块，二是它通过Binder进程间通信机制来实现进程间的内存共享。本文中，我们将通过实例来简要介绍Android系统的匿名共享内存的使用方法，使得我们对Android系统的匿名共享内存机制有一个感性的认识，为进一步学习它的源代码实现打下基础。

Android系统的匿名共享内存子系统的主体是以驱动程序的形式实现在内核空间的，同时，在系统运行时库层和应用程序框架层提供了访问接口，其中，在系统运行时库层提供了C/C++调用接口，而在应用程序框架层提供了Java调用接口。这里，我们将直接通过应用程序框架层提供的Java调用接口来说明匿名共享内存子系统Ashmem的使用方法，毕竟我们在Android开发应用程序时，是基于Java语言的，而实际上，应用程序框架层的Java调用接口是通过JNI方法来调用系统运行时库层的C/C++调用接口，最后进入到内核空间的Ashmem驱动程序去的。

我们在这里举的例子是一个名为Ashmem的应用程序，它包含了一个Server端和一个Client端实现，其中，Server端是以Service的形式实现的，在这里Service里面，创建一个匿名共享内存文件，而Client是一个Activity，这个Activity通过Binder进程间通信机制获得前面这个Service创建的匿名共享内存文件的句柄，从而实现共享。在Android应用程序框架层，提供了一个MemoryFile接口来封装了匿名共享内存文件的创建和使用，它实现在frameworks/base/core/java/android/os/MemoryFile.java文件中。下面，我们就来看看Server端是如何通过MemoryFile类来创建匿名共享内存文件的以及Client是如何获得这个匿名共享内存文件的句柄的。

在MemoryFile类中，提供了两种创建匿名共享内存的方法，我们通过MemoryFile类的构造函数来看看这两种使用方法：

```java
public class MemoryFile
{
    ......

    /**
     * Allocates a new ashmem region. The region is initially not purgable.
     *
     * @param name optional name for the file (can be null).
     * @param length of the memory file in bytes.
     * @throws IOException if the memory file could not be created.
     */
    public MemoryFile(String name, int length) throws IOException {
        mLength = length;
        mFD = native_open(name, length);
        mAddress = native_mmap(mFD, length, PROT_READ | PROT_WRITE);
        mOwnsRegion = true;
    }

    /**
     * Creates a reference to an existing memory file. Changes to the original file
     * will be available through this reference.
     * Calls to {@link #allowPurging(boolean)} on the returned MemoryFile will fail.
     *
     * @param fd File descriptor for an existing memory file, as returned by
     *        {@link #getFileDescriptor()}. This file descriptor will be closed
     *        by {@link #close()}.
     * @param length Length of the memory file in bytes.
     * @param mode File mode. Currently only "r" for read-only access is supported.
     * @throws NullPointerException if <code>fd</code> is null.
     * @throws IOException If <code>fd</code> does not refer to an existing memory file,
     *         or if the file mode of the existing memory file is more restrictive
     *         than <code>mode</code>.
     *
     * @hide
     */
    public MemoryFile(FileDescriptor fd, int length, String mode) throws IOException {
        if (fd == null) {
            throw new NullPointerException("File descriptor is null.");
        }
        if (!isMemoryFile(fd)) {
            throw new IllegalArgumentException("Not a memory file.");
        }
        mLength = length;
        mFD = fd;
        mAddress = native_mmap(mFD, length, modeToProt(mode));
        mOwnsRegion = false;
    }

    ......
}
```

从注释中，我们可以看出这两个构造函数的使用方法，这里就不再详述了。两个构造函数的主要区别是第一个参数，第一种构造方法是以指定的字符串调用JNI方法native_open来创建一个匿名共享内存文件，从而得到一个文件描述符，接着就以这个文件描述符为参数调用JNI方法natvie_mmap把这个匿名共享内存文件映射在进程空间中，然后就可以通过这个映射后得到的地址空间来直接访问内存数据了；第二种构造方法是以指定的文件描述符来直接调用JNI方法natvie_mmap把这个匿名共享内存文件映射在进程空间中，然后进行访问，而这个文件描述符就必须要是一个匿名共享内存文件的文件描述符，这是通过一个内部函数isMemoryFile来验证的，而这个内部函数isMemoryFile也是通过JNI方法调用来进一步验证的。前面所提到的这些JNI方法调用，最终都是通过系统运行时库层进入到内核空间的Ashmem驱动程序中去，不过这里我们不关心这些JNI方法、系统运行库层调用以及Ashmem驱动程序的具体实现，在接下来的两篇文章中，我们将会着重介绍，这里我们只关注MemoryFile这个类的使用方法。

前面我们说到，我们在这里举的例子包含了一个Server端和一个Client端实现，其中， Server端就是通过前面一个构造函数来创建一个匿名共享内存文件，接着，Client端过Binder进程间通信机制来向Server请求这个匿名共享内存的文件描述符，有了这个文件描述符之后，就可以通过后面一个构造函数来共享这个内存文件了。

因为涉及到Binder进程间通信，我们首先定义好Binder进程间通信接口。Binder进程间通信机制的相关介绍，请参考前面一篇文章Android进程间通信（IPC）机制Binder简要介绍和学习计划，这里就不详细介绍了，直接进入主题。

首先在源代码工程的packages/experimental目录下创建一个应用程序工程目录Ashmem。关于如何获得Android源代码工程，请参考在Ubuntu上下载、编译和安装Android最新源代码一文；关于如何在Android源代码工程中创建应用程序工程，请参考在Ubuntu上为Android系统内置Java应用程序测试Application Frameworks层的硬件服务一文。这里，工程名称就是Ashmem了，它定义了一个路径为shy.luo.ashmem的package，这个例子的源代码主要就是实现在这里了。下面，将会逐一介绍这个package里面的文件。

这里要用到的Binder进程间通信接口定义在src/shy/luo/ashmem/IMemoryService.java文件中：

```java
package shy.luo.ashmem;

import android.util.Log;
import android.os.IInterface;
import android.os.Binder;
import android.os.IBinder;
import android.os.Parcel;
import android.os.ParcelFileDescriptor;
import android.os.RemoteException;

public interface IMemoryService extends IInterface {
    public static abstract class Stub extends Binder implements IMemoryService {
        private static final String DESCRIPTOR = "shy.luo.ashmem.IMemoryService";

        public Stub() {
            attachInterface(this, DESCRIPTOR);
        }

        public static IMemoryService asInterface(IBinder obj) {
            if (obj == null) {
                return null;
            }

            IInterface iin = (IInterface)obj.queryLocalInterface(DESCRIPTOR);
            if (iin != null && iin instanceof IMemoryService) {
                return (IMemoryService)iin;
            }

            return new IMemoryService.Stub.Proxy(obj);
        }

        public IBinder asBinder() {
            return this;
        }

        @Override
        public boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws android.os.Remot
            switch (code) {
            case INTERFACE_TRANSACTION: {
                reply.writeString(DESCRIPTOR);
                return true;
            }
            case TRANSACTION_getFileDescriptor: {
                data.enforceInterface(DESCRIPTOR);

                ParcelFileDescriptor result = this.getFileDescriptor();

                reply.writeNoException();

                if (result != null) {
                    reply.writeInt(1);
                    result.writeToParcel(reply, 0);
                } else {
                    reply.writeInt(0);
                }

                return true;
            }
            case TRANSACTION_setValue: {
                data.enforceInterface(DESCRIPTOR);

                int val = data.readInt();
                setValue(val);

                reply.writeNoException();

                return true;
            }
            }

            return super.onTransact(code, data, reply, flags);
        }

        private static class Proxy implements IMemoryService {
            private IBinder mRemote;

            Proxy(IBinder remote) {
                mRemote = remote;
            }

            public IBinder asBinder() {
                return mRemote;
            }

            public String getInterfaceDescriptor() {
                return DESCRIPTOR;
            }

            public ParcelFileDescriptor getFileDescriptor() throws RemoteException {
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();

                ParcelFileDescriptor result;

                try {
                    data.writeInterfaceToken(DESCRIPTOR);

                    mRemote.transact(Stub.TRANSACTION_getFileDescriptor, data, reply, 0);

                    reply.readException();
                    if (0 != reply.readInt()) {
                        result = ParcelFileDescriptor.CREATOR.createFromParcel(reply);
                    } else {
                        result = null;
                    }
                } finally {
                    reply.recycle();
                    data.recycle();
                }

                return result;
            }

            public void setValue(int val) throws RemoteException {
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();

                try {
                    data.writeInterfaceToken(DESCRIPTOR);
                    data.writeInt(val);

                    mRemote.transact(Stub.TRANSACTION_setValue, data, reply, 0);

                    reply.readException();
                } finally {
                    reply.recycle();
                    data.recycle();
                }
            }
        }

        static final int TRANSACTION_getFileDescriptor = IBinder.FIRST_CALL_TRANSACTION + 0;
        static final int TRANSACTION_setValue = IBinder.FIRST_CALL_TRANSACTION + 1;

    }

    public ParcelFileDescriptor getFileDescriptor() throws RemoteException;
    public void setValue(int val) throws RemoteException;
}
```

这里主要是定义了IMemoryService接口，它里面有两个调用接口：

```java
    public ParcelFileDescriptor getFileDescriptor() throws RemoteException;
    public void setValue(int val) throws RemoteException;
```

同时，还分别定义了用于Server端实现的IMemoryService.Stub基类和用于Client端使用的代理IMemoryService.Stub.Proxy类。关于Binder进程间通信机制在应用程序框架层的Java接口定义，请参考前面Android系统进程间通信Binder机制在应用程序框架层的Java接口源代码分析一文。

有了Binder进程间通信接口之后，接下来就是要在Server端实现一个本地服务了。这里，Server端实现的本地服务名为MemoryService，实现在src/shy/luo/ashmem/MemoryService.java文件中：

```java
package shy.luo.ashmem;

import java.io.FileDescriptor;
import java.io.IOException;

import android.os.Parcel;
import android.os.MemoryFile;
import android.os.ParcelFileDescriptor;
import android.util.Log;

public class MemoryService extends IMemoryService.Stub {
    private final static String LOG_TAG = "shy.luo.ashmem.MemoryService";
```

```
package shy.luo.ashmem;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;
import android.os.ServiceManager;

public class MyService extends Service {
    private final static String LOG_TAG = "shy.luo.ashmem.Service";

    private MemoryService memoryService = null;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
Log.i(LOG_TAG, "Create Memory Service...");
memoryService = new MemoryService();

        try {
            ServiceManager.addService("shy.luo.ashmem.service", memoryService);
            Log.i(LOG_TAG, "Success to add memory service.");
        } catch (RuntimeException ex) {
            Log.i(LOG_TAG, "Failed to add Memory Service.");
            ex.printStackTrace();
        }

    }

    @Override
    public void onStart(Intent intent, int startId) {
        Log.i(LOG_TAG, "Start Memory Service.");
    }

    @Override
    public void onDestroy() {
Log.i(LOG_TAG, "Stop Memory Service.");
    }
}
```

```
    memoryService = new MemoryService();

    try {
        ServiceManager.addService("shy.luo.ashmem.service", memoryService);
        Log.i(LOG_TAG, "Success to add memory service.");
    } catch (RuntimeException ex) {
        Log.i(LOG_TAG, "Failed to add Memory Service.");
        ex.printStackTrace();
    }
```

```
package shy.luo.ashmem;

import java.io.FileDescriptor;
import java.io.IOException;

import shy.luo.ashmem.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.MemoryFile;
import android.os.ParcelFileDescriptor;
import android.os.ServiceManager;
import android.os.RemoteException;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class Client extends Activity implements OnClickListener {
    private final static String LOG_TAG = "shy.luo.ashmem.Client";

IMemoryService memoryService = null;
MemoryFile memoryFile = null;

    private EditText valueText = null;
    private Button readButton = null;
    private Button writeButton = null;
    private Button clearButton = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

    IMemoryService ms = getMemoryService();
            if(ms == null) {
                startService(new Intent("shy.luo.ashmem.service"));
            } else {
    Log.i(LOG_TAG, "Memory Service has started.");
            }

        valueText = (EditText)findViewById(R.id.edit_value);
        readButton = (Button)findViewById(R.id.button_read);
        writeButton = (Button)findViewById(R.id.button_write);
        clearButton = (Button)findViewById(R.id.button_clear);

    readButton.setOnClickListener(this);
        writeButton.setOnClickListener(this);
        clearButton.setOnClickListener(this);

        Log.i(LOG_TAG, "Client Activity Created.");
    }

    @Override
    public void onResume() {
        super.onResume();

    Log.i(LOG_TAG, "Client Activity Resumed.");
    }

    @Override
    public void onPause() {
        super.onPause();

    Log.i(LOG_TAG, "Client Activity Paused.");
    }


    @Override
    public void onClick(View v) {
        if(v.equals(readButton)) {
                int val = 0;

            MemoryFile mf = getMemoryFile();
            if(mf != null) {
                try {
                    byte[] buffer = new byte[4];
                mf.readBytes(buffer, 0, 0, 4);
                    val = (buffer[0] << 24) | ((buffer[1] & 0xff) << 16) | ((buffer[2] & 0xff) << 8) | (buffer[3] & 0xff);
                } catch(IOException ex) {
                Log.i(LOG_TAG, "Failed to read byte from memory file.");
            ex.printStackTrace();
                }
            }

        String text = String.valueOf(val);
        valueText.setText(text);
        } else if(v.equals(writeButton)) {
        String text = valueText.getText().toString();
                int val = Integer.parseInt(text);

        IMemoryService ms = getMemoryService();
            if(ms != null) {
                try {
                ms.setValue(val);
                } catch(RemoteException ex) {
            Log.i(LOG_TAG, "Failed to set value to memory service.");
            ex.printStackTrace();
                }
            }
        } else if(v.equals(clearButton)) {
        String text = "0";
        valueText.setText(text);
        }
    }

    private IMemoryService getMemoryService() {
        if(memoryService != null) {
            return memoryService;
        }
    memoryService = IMemoryService.Stub.asInterface(
                        ServiceManager.getService("shy.luo.ashmem.service"));
    Log.i(LOG_TAG, memoryService != null ? "Success to get memory service." : "Failed to get memory service.");
        return memoryService;
    }

    private MemoryFile getMemoryFile() {
        if(memoryFile != null) {
            return memoryFile;
        }
    IMemoryService ms = getMemoryService();
        if(ms != null) {
            try {
            ParcelFileDescriptor pfd = ms.getFileDescriptor();
                if(pfd == null) {
            Log.i(LOG_TAG, "Failed to get memory file descriptor.");
                    return null;
                }

                try {
            FileDescriptor fd = pfd.getFileDescriptor();
                    if(fd == null) {
            Log.i(LOG_TAG, "Failed to get memory file descriptor.");
                        return null;
                    }
                    memoryFile = new MemoryFile(fd, 0, "");
                } catch(IOException ex) {
            Log.i(LOG_TAG, "Failed to create memory file.");
            ex.printStackTrace();
                }
            } catch(RemoteException ex) {
            Log.i(LOG_TAG, "Failed to get file descriptor from memory service.");
            ex.printStackTrace();
            }
        }
        return memoryFile;
    }
}
```

```java
			ex.printStackTrace();
		}
	}

	return memoryFile;
}
}
```

Client端的界面主要包含了三个按钮Read、Write和Clear，以及一个用于显示内容的文本框。

这个Activity在onCreate时，会通过startService接口来启动我们前面定义的Server进程。调用startService时，需要指定要启动的服务的名称，这里就是"shy.luo.ashmem.server"了，后面我们会在程序的描述文件AndroidManifest.xml看到前面的Server类是如何和名称"shy.luo.ashmem.server"关联起来的。关于调用startService函数来启动自定义服务的过程，可以参考Android系统在新进程中启动自定义服务过程（startService）的原理分析一文。

内部函数getMemoryService用来获取IMemoryService。如果是第一次调用该函数，则会通过ServiceManager的getService接口来获得这个IMemoryService接口，然后保存在类成员变量memoryService中，以后再调用这个函数时，就可以直接返回memoryService了。

内部函数getMemoryFile用来从MemoryService中获得匿名共享内存文件的描述符。同样，如果是第一次调用该函数，则会通过IMemoryService的getFileDescriptor接口来获得MemoryService中的匿名共享内存文件的描述符，然后用这个文件描述符来创建一个MemoryFile实例，并保存在类成员变量memoryFile中，以后再调用这个函数时，就可以直接返回memoryFile了。

有了memoryService和memoryFile后，我们就可以在Client端访问Server端创建的匿名共享内存了。点击Read按钮时，就通过memoryFile的readBytes接口把共享内存中的整数读出来，并显示在文本框中；点击Write按钮时，就通过memoryService这个代理类的setVal接口来调用MemoryService的本地实现类的setVal服务，从而把文本框中的数值写到Server端创建的匿名共享内存中去；点击Clear按钮时，就会清空文本框的内容。这样，我们就可以通过Read和Write按钮来验证我们是否在Client和Server两个进程中实现内存共享了。

现在，我们再来看看Client界面的配置文件，它定义在res/layout/main.xml文件中：

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center">
        <TextView
                        android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/value">
        </TextView>
        <EditText
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:id="@+id/edit_value"
                android:hint="@string/hint">
        </EditText>
    </LinearLayout>
     <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center">
        <Button
                android:id="@+id/button_read"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/read">
        </Button>
        <Button
                android:id="@+id/button_write"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/write">
        </Button>
        <Button
                android:id="@+id/button_clear"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/clear">
        </Button>
    </LinearLayout>
</LinearLayout>
```

相关的字符串定义在res/values/strings.xml文件中：

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="app_name">Ashmem</string>
    <string name="value">Value</string>
    <string name="hint">Please input a value...</string>
    <string name="read">Read</string>
    <string name="write">Write</string>
    <string name="clear">Clear</string>
</resources>
```

这样，界面的相关配置文件就介绍完了。

我们还要再来看程序描述文件AndroidManifest.xml的相关配置，它位于Ashmem目录下：

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="shy.luo.ashmem"
      android:sharedUserId="android.uid.system"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Client"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
                <service
                        android:enabled="true"
                        android:name=".Server"
                        android:process=".Server" >
                        <intent-filter>
                <action android:name="shy.luo.ashmem.server"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
                </service>
    </application>
</manifest>
```

这里我们可以看到，下面的配置项把服务名称"shy.luo.ashmem.server"和本地服务类Server关联了起来：

```xml
    <service
        android:enabled="true"
        android:name=".Server"
        android:process=".Server" >
        <intent-filter>
            <action android:name="shy.luo.ashmem.server"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
    </service>
```

这样，我们就可以通过startService(new Intent("shy.luo.ashmem.server"))来启动这个Server了。不过，在Android中，启动服务是需要权限的，所以，下面这一行配置获取了启动服务需要的相应权限：

```
android:sharedUserId="android.uid.system"
```

最后，我们来看工程的编译脚本文件Android.mk，它位于Ashmem目录下：

```makefile
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE_TAGS := optional

LOCAL_SRC_FILES += $(call all-subdir-java-files)

LOCAL_PACKAGE_NAME := Ashmem

LOCAL_CERTIFICATE := platform

include $(BUILD_PACKAGE)
```

这里又有一个关键的地方：

```
LOCAL_CERTIFICATE := platform
```

因为我们需要在程序中启动Service，所以要配置这一行，并且要把源代码工程放在Android源代码平台中进行编译。

这样，整个例子的源代码实现就介绍完了，接下来就要编译了。有关如何单独编译Android源代码工程的模块，以及如何打包system.img，请参考如何单独编译Android源代码中的模块一文。

执行以下命令进行编译和打包：

```
USER-NAME@MACHINE-NAME:~/Android$ mmm packages/experimental/Ashmem
USER-NAME@MACHINE-NAME:~/Android$ make snod
```

这样，打包好的Android系统镜像文件system.img就包含我们前面创建的Ashmem应用程序了。

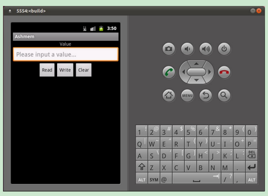
再接下来，就是运行模拟器来运行我们的例子了。关于如何在Android源代码工程中运行模拟器，请参考在Ubuntu上下载、编译和安装Android最新源代码一文。

执行以下命令启动模拟器：

```
USER-NAME@MACHINE-NAME:~/Android$ emulator
```

模拟器启动起，就可以在Home Screen上看到Ashmem应用程序图标了：

点击Ashmem图标，启动Ashmem应用程序，界面如下：



这样，我们就可以验证程序的功能了，看看是否实现了在两个进程中通过使用Android系统的匿名共享内存机制来共享内存数据的功能。

通过这个例子的学习，相信读者对Android系统匿名共享内存子系统Ashmem有了一个大概的认识，但是，这种认识还是停留在表面上。我们在文章开始时就提到，Android系统匿名共享内存子系统Ashmem两个特点，一是能够辅助内存管理系统来有效地管理不再使用的内存块，二是它通过Binder进程间通信机制来实现进程间的内存共享。第二个特点我们在上面这个例子中看到了，但是似乎还不够深入，我们知道，在Linux系统中，文件描述符其实就是一个整数，它是用来索引进程保存在内核空间的打开文件数据结构的，而且，这个文件描述符只是在进程内有效，也就是说，在不同的进程中，相同的文件描述符的值，代表的可能是不同的打开文件，既然是这样，把Server进程中的文件描述符传给Client进程，似乎就没有用了，但是不用担心，在传输过程中，Binder驱动程序会帮我们处理好一切，保证Client进程拿到的文件描述符是在本进程中有效的，并且它指向就是Server进程创建的匿名共享内存文件。至于第一个特点，我们也准备在后续学习Android系统匿名共享内存子系统Ashmem时，再详细介绍。

因此，为了深入了解Android系统匿名共享内存子系统Ashmem，在接下来的两篇文章中，围绕上面提到的两个特点，分别学习：

1. Android系统匿名共享内存子系统Ashmem是如何够辅助内存管理系统来有效地管理不再使用的内存块的？

2. Android系统匿名共享内存子系统Ashmem是如何通过Binder进程间通信机制来实现进程间的内存共享的？

学习完这两篇文章后，相信大家对 Android系统匿名共享内存子系统Ashmem就会有一个更深刻的认识了，敬请关注。

老罗的新浪微博：http://weibo.com/shengyangluo，欢迎关注！

- code dream： 你这里的service并没有单独运行在一个线程里吧？ (2年前 #18楼)
- Aaronwang062441： 希望能开源 (2年前 #17楼)
- DainZhang： 很透彻 拜读 (3年前 #16楼)
- ljl7288185： 为什么不在Service创建的时候生成MemoryFile对象实例，然后Client只需调用Service提供的的setValue()和"getVal()"，关于怎样操作MemoryFile的东西客户端不用知道吧 (3年前 #15楼) 查看回复(2)
- Kevin.Suen： hi 老罗， 很高兴看到你的blog, 我是做android for smart TV的，我们使用的android最低版本是4.4，我只是熟悉audio这一部分，想通过你这本书来学习一下其它部分，不过看起来改变相对2.3挺大的，有什么好的建议吗？ 谢谢 (3年前 #14楼) 查看回复(2)
- joyinmay： 不好意思，请问一下，IMemoryService.java是通过 interface IMemoryService { ParcelFileDescriptor getFileDescriptor(); void setValue(int val); } 这个IMemoryService.aidl文件生成的吗？ (4年前 #13楼)
- standback： 在client拿到fileDescription 构造 出自己的memoryFile后。应该可以直接 用这个memoryFile来对共享内存进行写操作了，为什么还要调用远程的setvalue呢。。我只是想确认一下这样也是可以的。 (4年前 #12楼) 查看回复(1)
- pengguowen： client 经常使用binder通信从server进程获取到FileDescriptor ，这个FileDescriptor 包含着由server进程调用native_open函数，打开/dev/ashmem 设备节点所产生的linux文件描述符，client使用FileDescriptor 调用MemoryFile(FileDescriptor fd, int length, String mode) 构造函数来初始化memoryFile ，而构造函数里面调用native_mmap函数将server打开/dev/ashmem所返回的linux文件描述符进行mmap操作，这有问题吧， 我哪里理解错了？ (5年前 #11楼) 查看回复(3)
- 余龙飞： hi，请教一个问题 pmem申请内存失败，但是ashmem分配成功，这个一般是什么原因？ 是不是ram不足了？ (5年前 #10楼) 查看回复(1)
- BadPattern： 在实际应用的开发过程中，貌似匿名共享内存用的不多，我是从来没用过,也因为根本不知道有这东西....... 有个问题想问老罗： 我想了想，在实际项目中，Binder已足够应付大多数进程间通信的需求，是不是如果需要传递大数据(如将一个内存中的Bitmap图片资源传递给另外一个进程)的时候，匿名共享内存机制就很适合？ 如果我遇到传递图片资源给另外一个进程的场景，一般都是写入本地文件，传个文件路径过去。。。 (5年前 #9楼) 查看回复(2)
- do多多do： 老罗大牛： 按照你的说法，Android里面的共享内存只能通过Binder的方式来实现吗。难道没有一个目录具有读写权限的吗？ (5年前 #8楼) 查看回复(1)
- 寻则时易： Mr Luo, Thanks. "这个Server继承了Android系统应用程序框架层提供的Service类，当它被启动时，运行在一个独立的进程中" 这里表示不能理解 ，启动Service 就会创建新新的进程？？ (6年前 #7楼) 查看回复(1)

上一页 1 2 下一页

- **Android匿名共享内存和MemoryFile** 6876
  今天要讲的内容相对冷门，日常开发使用率较低。但是，有些场景下，它也许可以给我们一个新的思路，帮我们解决一些棘... 来自： viclee
- **Android匿名共享内存笔记** 1786
  1、ashmem_open() static int ashmem_open(struct inode *inode, struct file *file) { struct ashmem_area... 来自： guoqifa29的专栏
- **Android Binder 分析——匿名共享内存（Ashmem）** 2745
  前面分析了 binder 中用来打包、传递数据的 Parcel，一般用来传递 IPC 中的小型参数和返回值。binder 目前每个进程 mm... 来自： omnispace的博客
- **安卓ashmem(匿名共享内存映射)学习native篇** 939
  本文是基于这篇文章之上的： http://blog.csdn.net/u010657219/article/details/41243473 先看看之前三个进程的关系： 由... 来自： unbroken
- **Android系统匿名共享内存（Anonymous Shared Memory）C++调用接口分析** 4.2万
  在Android系统中，针对移动设备内存空间有限的特点，提供了一种在进程间共享数据的机制：匿名共享内存，它能够辅助... 来自： 老罗的Android...
- **Android系统匿名共享内存Ashmem** 2052
  转自：http://blog.csdn.net/luoshengyang/article/details/6651971 在Android系统中，提供了独特的匿名共享内存子系统As... 来自： Hknock的专栏
- **Android匿名共享内存系列(好文)** 473
  点击学习匿名共享内存 来自： unbroken
- **Anonymous Shared Memory android 匿名共享内存** 299
  原址 两个特点： 能够辅助内存管理系统来有效的管理不再使用的内存块 通过Binder进程间通信机制来实现进程间的内存共... 来自： unbroken
- **Android 匿名共享内存Java接口分析** 3995
  在Android匿名共享内存驱动源码分析中介绍了匿名共享内存的驱动实现过程，本文在Android匿名共享内存驱动基础上，介... 来自： 深入剖析Androi...

文章热词 NLP学习学习 深度学习学习 软件设计学习 Ngnix学习 邮件服务器学习

相关热词 android系统介绍 c#共享内存 android系统jar包 android系统c# android应用开发介绍 python课程学习计划 python语言学习计划

- **Android高性能文件类MemoryFile** 461
  很多网友抱怨Android处理底层I/O性能不是很理想，如果不想使用NDK则可以通过MemoryFile类实现高性能的文件读写操作... 来自： 无线.无极线

Mirage520 关注 902篇文章 | 姥姥教我学编程 关注 229篇文章 | WangMark 关注 303篇文章

- **Android匿名共享(ashmem)内存使用分析 - Andy_0755 - CSDN博客** 11-22
  ashmem是android封装提供的一种共享内存实现方式。驱动会在/dev/下面注册ashmem字符设备,供上层做文件操作(ashmem驱动后面分析)。 ...
- **Android匿名共享内存系列(好文) - unbroken - CSDN博客** 11-10
  l在上一篇Android 匿名共享内存C接口分析中介绍了Android系统的匿名共享内存C语言访问接口,本文在前文的基础上继续介绍Android系统的匿名共享内...
- **Android8.0.0-r4的内存共享机制** 355
  一、匿名共享内存的共享原理 在Android系统中,每一块匿名共享内存都是使用一个文件描述符来描述的，而这个文件描... 来自： nwpushuai的博客
- **共享内存:mmap函数实现** 4004
  内存映射的应用: 以页面为单位,将一个普通文件映射到内存中,通常在需要对文件进行频繁读写时使用,这样用内存读写取代I/... 来自： 一步一脚印
- **Android-MemoryFile介绍** 1万
  Tricking Android MemoryFile 之前在做一个内存优化的时候，使用到了MemoryFile，由此发现了MemoryFile的一些特性以... 来自： wzgiceman
- **进程间通信 SharedMemory 最后例子没看明白** 907
  http://www.devbean.net/2013/11/qt-study-road-2-ipc/ 上一章我们了解了有关进程的基本知识。我们将进程理解为相互独立... 来自： Bryan要加油
- **Postgresql 启动报错 FATAL: could not map anonymous shared memory** 1971
  FATAL: could not map anonymous shared memory: Cannot allocate memory HINT: This error usually mea...
- **进程间通信之共享内存（Share Memory）** 3402
  为了实现多个进程之间的通信，我们可以用系统页面文件来最为共享内存文件。 1.首先我们创建第一个进程，用系统函数C... 来自： nk_wang的专栏
- **MemoryFile匿名共享内存简单案例** 319
  MemoryFile:匿名共享内存用匿名共享内存来实现进程间内存共享Binder不能传递太大的数据。最大通常限制为1M。通过文... 来自： xzc的博客
- **Android系统匿名共享内存Ashmem（Anonymous Shared Memory）驱动程序源代码分析** 633
  在上一文章Android系统匿名共享内存Ashmem（Anonymous Shared Memory）简要介绍和学习计划中，我们简要介绍了A... 来自： Mirage520的专栏
- **深入理解Linux进程间通信（IPC）-- 共享内存shared memory mmap** 1895
  6. 共享内存（上）简介： 共享内存可以说是最有用的进程间通信方式，也是最快的IPC形式。两个不同进程A、B共享内存... 来自： WangMark的专栏
- **Android匿名共享(ashmem)内存使用分析** 90
  最近遇到个问题，需要使用共享内存。之前看audioTrack与audioFlinger部份时就看到共存内存的分配与同步，但是...
- **Android 匿名共享内存驱动源码分析** 3812
  Android系统的匿名共享内存Ashmem驱动程序利用了Linux的共享内存子系统导出的接口来实现，本文通过源码分析方式详... 来自： 深入剖析Androi...
- **Android 匿名共享内存C接口分析** 403
  原址 在Android 匿名共享内存驱动源码分析中详细分析了匿名共享内存在Linux内核空间的实现，虽然内核空间实现了匿名... 来自： unbroken
- **共享内存（Shared Memory）介绍** 1.3万
  共享内存是常用的进程间通信，两个进程可以直接共享访问同一块内存区域。... 来自： wangrl的博客
- **Android系统匿名共享内存（Anonymous Shared Memory）Java调用接口分析** 2158
  一、Ashmem驱动程序 ~/Android/kernel/goldfish ----include ----linux ... 来自： jltxgcy的专栏
- **共享内存方式(SharedMemory)** 755
  共享内存方式(SharedMemory) 内存映射一般是做进程间通讯的，线程间通信有太多方法了。 a.设定一块共享内存区域,得... 来自： tony2278的专栏
- **android 使用SMD(共享内存)作為跨processes溝通媒介** 523
  前言: 多核間的通訊 在智慧型手機SOC平台中，為了保證個別功能在平台上的流暢運行，都會依照功能使用專門的處理器來...
- **cuda shared memory 静态分配和动态分配** 2800
  静态分配加上前缀 shared__shared__ int _ss[1024];动态分配当我们在编程时，不清楚shared memory 数组开多大，就要... 来自： 朝气蓬勃
- **Linux 高级编程 - 共享内存 Shared Memory** 1675
  Linux 高级编程 - 共享内存 Shared Memory 来自： cdeveloper 的专栏
- **caffe 源码学习（三） SyncedMemory 与 shared_ptr** 3966
  在blob.hpp中我们会看到protected成员变量：protected: shared_ptr data_; shared_ptr diff_; shared_ptr shape_d... 来自： 牛闯的博客
- **(转)Android系统匿名共享内存Ashmem（Anonymous Shared Memory）在进程间共享的原理分析...** 2
  在前面一篇文章Android系统匿名共享内存Ashmem（Anonymous Shared Memory）驱动程序源代码分析中，我们系统地... 来自： s3834569的博客
- **Android系统匿名共享内存Ashmem（Anonymous Shared Memory）在进程间共享的原理分析** 343
  在前面一篇文章Android系统匿名共享内存Ashmem（Anonymous Shared Memory）驱动程序源代码分析中，我们系统地... 来自： 白马笑西风的专栏
- **【并行计算-CUDA开发】关于共享内存（shared memory）和存储体(bank)的事实和疑惑** 1767
  关于共享内存（shared memory）和存储体(bank)的事实和疑惑 主要是在研究访问共享内存会产生bank conflict时，自己产... 来自： ZhangPY的专栏
- **共享内存： Shared Memory** 1025
  Linux进程间通信方法有： 同一主机的父子进程间：管道 同一主机上的进程间：FIFO（命名管道）、消息队列、信号量、... 来自： Huntinux
- **cuda编程：关于共享内存（shared memory）和存储体(bank)的事实和疑惑** 4511
  主要是在研究访问共享内存会产生bank conflict时，自己产生的疑惑。对于这 来自： linger(心怀梦想...
- **Android应用层使用共享内存机制进行进程间交换数据** 7028
  Android应用层使用共享内存机制进行进程间交换数据Android提供了一个高效的共享内存机制。如果应用中涉及到在多个进... 来自： haohaoxuexi20...
- **Android中匿名共享内存demo** 623
  package com.example.serverprocess.memory; import android.content.Context; import android.os.MemoryF... 来自： 笨鸟的专栏
- **共享内存实现进程间大数据的交换** 544
  引言 进程间的数据交换和共享是一种非常重要和实用的技术。大、中型软件的开发设计多是由众多程序设计人员的合... 来自： zhangzeshuai的...
- **Android内存共享 Java层内存共享 Native层内存共享** 436
  最近研究Android6.0流媒体传输，用到了Android里面的内存共享（Ashmem匿名共享），研究了两天怎么用的，现在分享... 来自： tan_zhenqi的专栏
- **linux 进程占用内存查询** 1.8万
  作者: 黄永兵/译 出处:51CTO.com 阅读提示：本文是为那些经常疑惑的人准备的，"为什么一个简单的KDE文本编辑器要... 来自： bugouyonggan...
- 下载 **安卓Ashmem使用实列** 11-18
  安卓很多模块都用到了ashmem，为了更好的学习各个模块，了解ashmem的原理是必不可少的！
- **android -- MemoryFile 内存文件（IO性能优化）** 1.1万
  前些天还自已写了一个内存文件用于http数据交互使用，没想到今天翻代码，尽然找到类似的现有实现，真是汗！ 呵呵，那... 来自： andyhuabing的...
- **Android Binder 分析——匿名共享内存(好文）** 1294
  原址 文章目录 1. 原理概述2. java 层接口3. native 层接口4. kernel 驱动5. 例子 前面分析了 binder 中用来打包、传递数据... 来自： unbroken
- **nginx共享内存：共享内存的实现** 2206
  nginx中， 作者为我们提供了方便共享内存的使用的接口，关于共享内存的使用在我之前的文章中有介绍。这次我们来研究... 来自： Shellcode.Cool!
- **使用 Shared Memory 协议创建有效的连接字符串** 536
  从运行在同一台计算机上的客户端到 Microsoft SQL Server 的连接使用 shared memory 协议。共享内存没有可配置的属性... 来自： tianlianchao198...
- **cuda shared(共享内存)—一个容易错误的点** 483
  可能出现的代码情况： template __global__ void kCopy(int row_size, float **col_valuess, float *row_values) { in... 来自： gonaYet的博客
- **linux共享内存（shared memory）使用介绍** 56
  共享内存是IPC机制中的一种，顾名思义，它允许两个不相关的进程访问同一段内存，这也是一种传递数据的有效方式。 通... 来自： 天下
- **android ashmem 匿名共享内存** 941
  1. ashmem是基于linux里的shmem实现的，而shmem又是基于tmpfs伪文件系统。 2. ashmem驱动实现了/dev/ashmem设备...
- **Android系统Bluetooh移植简要说明** 382
  一、Bluez编译配置支持 在BoardConfig.mk中添加： BOARD_HAVE_BLUETOOTH := true。 二、启动hciattach BlueZ核心... 来自： JiangKan的专栏
- **匿名内存** 1078
  1.匿名内存的概念 匿名内存是指那些在文件系统里没有相对应的"储备文件"的那些内存，堆和栈空间的内存都属于匿名内存... 来自： 知了112的专栏
-  **学习OpenCV 3（中文版）**
  这本书提供了C++开源计算机视觉库（OpenCV）3.0版本的工作指南
- **IntelliJ IDEA 最新注册码（截止到2019年5月4日） - 维C果糖的博客** 131521
  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 注册码 ... 来自： 维C果糖的博客
- **MyBatis——mapper.xml提升指南 - 愤怒的懒洋洋的博客** 9977
  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射。My... 来自： 愤怒的懒洋洋的...
- **如何设计一个灵活的 MySQL 数据表，应对灵活多变的需求 - GitChat** 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，需要每一次活... 来自： GitChat
- **VSCode设置中文语言显示 - 飞扬的博客** 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了？ 1）... 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/84673829 ... 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 - 徐奕的专栏** 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 无法继续下载。... 来自： 徐奕的专栏
- **Webstorm 最新激活码 多种破解方式(持续更新...) - 一只喵~** 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"，在输入... 来自： 老妖儿的博客
- **SQL提升（二） - 愤怒的懒洋洋的博客** 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sql。所以... 来自： 愤怒的懒洋洋的...
- **Postman 使用方法详解 - fxbin123的博客** 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请求的，... 来自： fxbin123的博客
- **2018最新Web前端经典面试试题及答案 - wdlhao的博客** 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的答案。... 来自： wdlhao的博客
- **（二）MyBatis核心组件（配图详解＆代码实现） - 青衣煮茶** 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。 SqlS... 来自： 青衣煮茶
- **史上最简单的 SpringCloud 教程 | 终章 - 方志朋的专栏** 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，你可能再也... 来自： 方志朋的专栏
- **pyCharm最新2018激活码 - 昌昌** 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的电影... 来自： 昌昌

罗升阳
博客专家
关注

原创	粉丝	喜欢	评论
188	3万+	298	8529

等级： 博客 8　访问： 1267万+
积分： 4万+　排名： 103
勋章：

进击的程序员

《Android系统源代码情景分析》一书正在进击的程序员网(http://0xcc0xcd.com)中连载，点击进入

最新文章

- 《Android系统源代码情景分析》连载回忆录：灵感之源
- Android WebView硬件加速渲染网页UI的过程分析
- Android WebView执行GPU命令的过程分析
- Android WebView启动Chromium渲染引擎的过程分析
- Android WebView加载Chromium动态库的过程分析

归档

- 2017年1月 1篇
- 2016年12月 3篇
- 2016年11月 4篇
- 2016年10月 3篇
- 2016年9月 4篇

展开

热门文章

- 那两年炼就的Android内功修养
  阅读量：441898
- Android进程间通信（IPC）机制Binder简要介绍和学习计划
  阅读量：412108
- Android应用程序启动过程源代码分析
  阅读量：388356
- 在Ubuntu上下载、编译和安装Android最新源代码
  阅读量：268334
- Android硬件抽象层（HAL）概要介绍和学习计划
  阅读量：216253

最新评论

- Android应用程序启动过程源代...
  wangwen_999：这个着实牛逼，我照着你这个搞的，不然我要研究很久很久，加速了我的学习，这个过程着实不易，关键是3个...
- 那两年炼就的Android内功修养
  liao_hb：问问老罗这些书，你都是每章每节都看完了？
- 《Android系统源代码情景分析...
  qq_33982241：向大佬学习，可是网站怎么打不开呢？
- 那两年炼就的Android内功修养
  straing：膜拜，隐隐的成为我android路上的指路人了。
- 《Android系统源代码情景分析...
  shike951128：仰望大神，期望有一天能齐肩，期望有一天能超越。

- **精选！15个必备的VSCode插件 - Dean** 👁 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，它很快就受到了大... 来自： Dean
- **整理了10个干净、好用的BT、磁力链搜索网站给大家 - YXAPP的技术分享** 👁 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、磁力链网... 来自： YXAPP的技术分享
- **webstorm 2018 激活破解方法大全 - 唐大帅的编程之路** 👁 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啊。 方... 来自： 唐大帅的编程之路
- **Beyond Compare 4注册码|Beyond Compare 4注册激活码下载(附破解教) - weixin_39528277的博客** 👁 27737
  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"而制作的激活工具...
- **Swagger教程二 - 愤怒的懒洋洋的博客** 👁 22573
  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swagge... 来自： 愤怒的懒洋洋的...
- **门罗币 xmr 超级详细的CPU xmr挖矿教程 - qq_39863517的博客** 👁 31065
  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,一个是... 来自： qq_39863517...
- **2017智慧树 伟大的红楼梦测试答案题库 - tangnanman0029的博客** 👁 100144
  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 1 【多... 来自： tangnanman00...
- **迅雷因版权方无法下载解决方法，附带破解不限速方法 - boy火巨的博客** 👁 30962
  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 无法继续下载。... 来自： boy火巨的博客
- **支付宝赚赏金的4种玩法（引流+变现日入200+） - Martin 的博客** 👁 32024
  支付宝赚赏金 来自： Martin 的博客
- **Navicat for MySQL 安装和破解（完美） - 一次次尝试** 👁 128519
  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 安装成功之后进行破解。 ... 来自： 一次次尝试
- **最新最详细的黑苹果10.13.4安装教程 - qq_28735663的博客** 👁 149449
  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版能查到的资料很... 来自： qq_28735663...
- **Swagger使用指南 - sanyaoxu_3的博客** 👁 61558
  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。总体目... 来自： sanyaoxu_3的...
- **什么是B-树？ - 卖克的专栏** 👁 15498
  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 B树首... 来自： 卖克的专栏
- **Python的Pillow库进行图像文件处理（配图详解） - 青衣煮茶** 👁 759
  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与使用。 ... 来自： 青衣煮茶
- **排序算法（三）起泡排序验证性实验 - 青衣煮茶** 👁 481
  请创建一个一维整型数组用来存储待排序关键码，关键码从数组下标为1的位置开始存储，下标为0的位置不存储关键码。... 来自： 青衣煮茶
- **C#Winform窗口移动 - Maybe_ch的博客** 👁 21828
  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标 2.获得... 来自： Maybe_ch的博客
- **MySql数据库优化必须注意的四个细节（方法） - 青衣煮茶** 👁 463
  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 数据库前进的见... 来自： 青衣煮茶
- **开源改变了小米什么？ - CSDN资讯** 👁 1432
  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：CSDNNews）... 来自： CSDN资讯
- **SQL提升（一） - 愤怒的懒洋洋的博客** 👁 10105
  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sql。所... 来自： 愤怒的懒洋洋的...
- **IntelliJ IDEA（2018）安装详解 - 闲暇时光** 👁 39781
  第一步：进入官网下载IntelliJ IDEA https://www.jetbrains.com/idea/download/#section=windows，选择适合版本下载；第二... 来自： 闲暇时光
- **（二）Maven的坐标和依赖＆利用Maven实现邮件发送 - 青衣煮茶** 👁 548
  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，具体详细请购买《M... 来自： 青衣煮茶
- **Proxyee-down的下载与安装教程 - shadandeajian的博客** 👁 120127
  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更新），... 来自： shadandeajian...
- **史上最全Java面试题（带全部答案） - 林老师带你学编程** 👁 96103
  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，在进入自己心仪的公... 来自： 林老师带你学编程
- **军事理论课答案（西安交大版） - ling_wang的博客** 👁 876644
  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金砖五... 来自： ling_wang的博客
- **各大磁力种子搜索引擎对比 - lizhengnanhua的专栏** 👁 404520
  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-最懂屌丝的搜索引... 来自： lizhengnanhua...
- **100个小学生猜字谜大全及答案 - 欢迎光临 包国工作室** 👁 180647
  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，答案是:思 3.... 来自： 欢迎光临 包国...
- **DirectX修复工具增强版 - VBcom的专栏** 👁 1669389
  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: 107MB/... 来自： VBcom的专栏
- **Android匿名共享内存笔记 - guoqifa29的专栏 - CSDN博客** 10-17
  Android匿名共享内存的实现主要在ashmem驱动中,把驱动搞懂就搞懂了这个机制。1、ashmem_open()static int ashmem_open(struct inode *inode, stru...
- **Android系统匿名共享内存Ashmem - u011748810的专栏 - CSDN博客** 11-11
  android系统的匿名共享内存系统的主体是以驱动程序的形式实现在内核空间的,同时,在系统运行时库和应用程序框架层提供了访问接口,其中,在系统运行时...
- **Android 匿名共享内存Java接口分析 - 深入剖析Android..._CSDN博客** 11-23
  在Android 匿名共享内存驱动源码分析中介绍了匿名共享内存的驱动实现过程,本文在Android匿名共享内存驱动基础上,介绍Android匿名共享内存对外Andr...
- **Android 匿名共享内存C++接口分析 - unbroken - CSDN博客** 11-21
  在上一篇Android 匿名共享内存C接口分析中介绍了Android系统的匿名共享内存C语言访问接口,本文在前文的基础上继续介绍Android系统的匿名共享内存...
- **Android匿名共享内存(Ashmem)原理 - happylishang的专栏 - CSDN博客** 11-14
- **Android系统匿名共享内存Ashmem - Hknock的专栏 - CSDN博客** 11-15
  在Android系统中,提供了独特的匿名共享内存子系统Ashmem(Anonymous Shared Memory),它以驱动程序的形式实现在内核空间中。它有两个特点,一是能...
- **Android Binder 分析——匿名共享内存(好文) - CSDN博客** 9-22
  native 层的 android 搞了2个: MemroyHeapBase: 这个类代表一块共享内存区域。 MemroyBase: 这个是上面一块共享内存区域中的一段,就是说 native 层...