

宝贵建议请发送至：[wangzhenyang@itcast.cn](mailto:wangzhenyang@itcast.cn)



黑马程序员

itheima.com

- 编程，始于黑马

# Android 课程同步笔记

Beta 0.01 版

By 阳哥

# Android 手机卫士-09 程序锁

## 1. 程序锁 (★★★★)

### 1.1 程序锁简介

在手机卫士的功能列表界面打开高级工具菜单，在高级工具中添加程序锁功能（如下图所示）。



点击进入程序锁功能主界面（如下图所示）。



在上图界面未加锁中，点击“锁”图片，那么该条目中的程序就会自动添加到已加锁列表，同时该条目被移除掉，在

移除的时候会有一个向右滑动的动画效果。同样的，在已加锁界面点击会出现类似的效果。

## 1.2 程序锁布局实现



该界面的布局其实比较简单，整体是一个 vertical 的 LinearLayout，里面其实嵌套了 3 个 LinearLayout，A、B 和 C，C 没有显示出来是因为 B 和 C 是互斥的两个 LinearLayout，当当前是未加锁状态时，B 的 `android:visibility="visible"`，C 的 `android:visibility="gone"`，当当前是已加锁状态时，B 和 C 的 visibility 属性则分别为隐藏和显示。

布局文件名 `applock_activity.xml`，布局清单如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#4444ff"
        android:gravity="center"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/tv_unlock"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/tab_left_pressed"
            android:gravity="center"
            android:text="未加锁"
            android:textColor="#ffffff" />

        <TextView
            android:id="@+id/tv_lock"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/tab_right_default"
            android:gravity="center"
            android:text="已加锁"
            android:textColor="#ffffff" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/ll_unlock"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/tv_unlock_count"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
```

```
        android:text="未加锁软件"
        android:textColor="#000000" />

<ListView
    android:fastScrollEnabled="true"
    android:id="@+id/lv_unloack_list"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
</ListView>
</LinearLayout>

<LinearLayout
    android:id="@+id/ll_lock"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:visibility="gone"
    >

    <TextView
        android:id="@+id/tv_lock_count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="已加锁软件"
        android:textColor="#000000" />

    <ListView
        android:id="@+id/lv_loack_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>
</LinearLayout>
</LinearLayout>
```

A 布局中的“未加锁”和“已加锁”是两个 TextView，点击的时候更换背景图片就看到了被选中的状态，这些操作是在代码中实现的。

## 1.3 程序锁业务逻辑知识点

**Tips**：程序锁涉及到的知识点清单如下，这些知识点的使用方法在代码中都会出现。

◆ 1) 删除 ListView 中条目时添加动画效果

//添加动画效果

```
TranslateAnimation animation = new TranslateAnimation(Animation.RELATIVE_TO_SELF,
0, Animation.RELATIVE_TO_SELF, 1.0f, Animation.RELATIVE_TO_SELF, 0,
Animation.RELATIVE_TO_SELF, 0);
animation.setDuration(500);
view.startAnimation(animation);
```

◆ 2) SQLiteOpenHelper 的使用

◆ 3) ListView 的进一步使用

## 1.4 程序锁业务逻辑代码的实现

### 1.4.1 com.itheima.mobileSafe.activity.AppLockActivity 代码清单

```
public class AppLockActivity extends Activity implements OnClickListener {
    //已加锁
    private TextView tv_lock;
    //未加锁
    private TextView tv_unlock;
    //已加锁线性布局
    private LinearLayout ll_lock;
    //未加锁线性布局
    private LinearLayout ll_unlock;
    //已加锁 ListView
    private ListView lv_lock_list;
    //未加锁 ListView
    private ListView lv_unlock_list;
    //存储应用信息
    private List<AppInfo> appInfos;
    //存储未加锁应用信息
    private List<AppInfo> unloackappInfos;
    //存储已加锁应用信息
    private List<AppInfo> lockedappInfos;
    private ApplockAdapter adapter;
    //显示未加锁软件个数
```

```
private TextView tv_unlock_count;
//显示已加锁软件个数
private TextView tv_lock_count;
//自定义的软件锁 dao
private AppLockDao dao;
private ApplockAdapter lockAdapter;
private ApplockAdapter unlockAdapter;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.applock_activity);
    //初始化控件、集合、dao
    tv_lock = (TextView) findViewById(R.id.tv_lock);
    tv_unlock = (TextView) findViewById(R.id.tv_unlock);
    ll_lock = (LinearLayout) findViewById(R.id.ll_lock);
    ll_unlock = (LinearLayout) findViewById(R.id.ll_unlock);
    //已加锁、未加锁设置点击事件
    tv_lock.setOnClickListener(this);
    tv_unlock.setOnClickListener(this);
    lv_lock_list = (ListView) findViewById(R.id.lv_loack_list);
    lv_unlock_list = (ListView) findViewById(R.id.lv_unloack_list);
    //通过自定义的工具类获取应用信息
    appInfos = AppInfoProvider.getAllAppInfos(this);
    //给 ListView 设置适配器
    lv_unlock_list.setAdapter(adapter);
    tv_lock_count = (TextView) findViewById(R.id.tv_lock_count);
    tv_unlock_count = (TextView) findViewById(R.id.tv_unlock_count);
    unloackappInfos = new ArrayList<AppInfo>();
    lockedappInfos = new ArrayList<AppInfo>();
    //
    dao = new AppLockDao(this);
    for(AppInfo info : appInfos){
        boolean find = dao.find(info.getPackName());
        if (find) {
            lockedappInfos.add(info);
        }else {
            unloackappInfos.add(info);
        }
    }
    unlockAdapter = new ApplockAdapter(true);
    lockAdapter = new ApplockAdapter(false);
    lv_unlock_list.setAdapter(unlockAdapter);
```

```
lv_lock_list.setAdapter(lockAdapter);
}

private class ApplockAdapter extends BaseAdapter {
    private boolean isFlag = true; // 未加锁
    public ApplockAdapter(boolean isFlag){
        this.isFlag = isFlag;
    }
    @Override
    public int getCount() {
        if (isFlag) {
            tv_unlock_count.setText("未加锁软件("+unloackappInfos.size()+")");
            return unloackappInfos.size();
        } else {
            tv_lock_count.setText("已加锁软件("+lockedappInfos.size()+")");
            return lockedappInfos.size();
        }
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        final View view;
        ViewHolder holder;
        if (convertView != null) {
            view = convertView;
            holder = (ViewHolder) view.getTag();
        } else {
            view = View.inflate(AppLockActivity.this, R.layout.list_applock_item,
null);
            holder = new ViewHolder();
            holder.iv_icon = (ImageView) view.findViewById(R.id.iv_icon);
            holder.tv_name = (TextView) view.findViewById(R.id.tv_name);
            holder.iv_status = (ImageView) view.findViewById(R.id.iv_status);
        }
    }
}
```



```

        view.setTag(holder);
    }
    final AppInfo appInfo;
    //判断当前需要显示的是已加锁还是未加锁
    if (isFlag) {
        appInfo = unloackappInfos.get(position);
        holder.iv_status.setImageResource(R.drawable.Lock);
    }else {
        appInfo = lockedappInfos.get(position);
        holder.iv_status.setImageResource(R.drawable.unlock);
    }
    holder.tv_name.setText(appInfo.getName());
    holder.iv_icon.setImageDrawable(appInfo.getIcon());
    holder.iv_status.setOnClickListener(new OnClickListener() {
        boolean canClick = true;
        @Override
        public void onClick(View v) {
            if (!canClick) {
                return ;
            }
            if (isFlag) { //点击未加锁的
                canClick = false;
                //添加动画效果
                TranslateAnimation animation = new
TranslateAnimation(Animation.RELATIVE_TO_SELF, 0, Animation.RELATIVE_TO_SELF,
1.0f, Animation.RELATIVE_TO_SELF, 0, Animation.RELATIVE_TO_SELF, 0);
                animation.setDuration(500);
                view.startAnimation(animation);
                dao.add(appInfo.getPackName());
                unloackappInfos.remove(appInfo);
                lockedappInfos.add(appInfo);
                new Handler().postDelayed(new Runnable() {

                    @Override
                    public void run() {
                        lockAdapter.notifyDataSetChanged();
                        unlockAdapter.notifyDataSetChanged();
                        canClick = true;
                    }
                }, 500);
            }else { //点击加锁的
                canClick = false;
                TranslateAnimation animation = new

```

```

TranslateAnimation(Animation.RELATIVE_TO_SELF, 0, Animation.RELATIVE_TO_SELF,
-1.0f, Animation.RELATIVE_TO_SELF, 0, Animation.RELATIVE_TO_SELF, 0);
        animation.setDuration(500);
        view.startAnimation(animation);
        dao.delete(appInfo.getPackageName());
        lockedappInfos.remove(appInfo);
        unlockappInfos.add(appInfo);
        new Handler().postDelayed(new Runnable() {

            @Override
            public void run() {
                lockAdapter.notifyDataSetChanged();
                unlockAdapter.notifyDataSetChanged();
                canClick = true;
            }
        }, 500);
    }

    });

    return view;
}

}

static class ViewHolder {
    ImageView iv_icon;
    TextView tv_name;
    ImageView iv_status;
}
//切换两个 LinearLayout
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.tv_unlock:
            tv_unlock.setBackgroundResource(R.drawable.tab_left_pressed);
            tv_lock.setBackgroundResource(R.drawable.tab_right_default);
            ll_unlock.setVisibility(View.VISIBLE);
            ll_lock.setVisibility(View.GONE);
            break;
        case R.id.tv_lock:
            tv_lock.setBackgroundResource(R.drawable.tab_right_pressed);

```

```
        tv_unlock.setBackgroundResource(R.drawable.tab_left_default);
        ll_lock.setVisibility(View.VISIBLE);
        ll_unlock.setVisibility(View.GONE);
        break;
    }
}
```

## 1.4.2 com.itheima.mobileSafe.db.dao.AppLockDao 代码清单

在 1.4.1 代码清单中，我们将加锁的程序存储在 SQLiteDatabase 中，因此我们在 dao 层目录结构下创建了

AppLockDao 类，该类提供了程序锁信息的 CRUD 方法，其代码清单如下：

```
public class AppLockDao {
    private AppLockOpenHelper openHelper;
    private Context context;

    public AppLockDao(Context context) {
        this.context = context;
        openHelper = new AppLockOpenHelper(context);
    }

    /**
     *
     * @param number
     * @param model
     */
    public void add(String number) {
        Uri uri = Uri.parse("content://com.itheima.mobileSafe.dbchange");
        context.getContentResolver().notifyChange(uri, null);
        SQLiteDatabase database = openHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("packname", number);
        database.insert("applock", null, values);
        database.close();
    }

    public void delete(String number) {
        SQLiteDatabase database = openHelper.getWritableDatabase();
        database.delete("applock", "packname=?", new String[] { number });
    }
}
```

```
        database.close();
    }

    public void update(String number, String model) {
        SQLiteDatabase database = openHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        database.update("applock", values, "packname=?", new String[] { number });
        database.close();
    }

    public boolean find(String number) {
        SQLiteDatabase database = openHelper.getReadableDatabase();
        Cursor query = database.query("applock", null, "packname=?", new String[]
{ number }, null, null, null);
        if (query.moveToNext()) {
            database.close();
            return true;
        } else {
            database.close();
            return false;
        }
    }

    public List<String> findAll() {
        SQLiteDatabase database = openHelper.getReadableDatabase();
        List<String> list = new ArrayList<String>();
        Cursor cursor = database.query("applock", new String[] { "packname" }, null,
null, null, null, null);
        while (cursor.moveToNext()) {
            String packname = cursor.getString(0);
            list.add(packname);
        }
        cursor.close();
        database.close();
        return list;
    }

    public int getTotal() {
        int total = 0;
        SQLiteDatabase database = openHelper.getReadableDatabase();
        Cursor cursor = database.rawQuery("select count(*) from applock ", null);
        if (cursor.moveToNext()) {
            total = cursor.getInt(0);
        }
    }
}
```

```
    }  
    cursor.close();  
    database.close();  
    return total;  
}  
}
```

### 1.4.3 com.itheima.mobileSafe.db.AppLockOpenHelper 代码清单

```
public class AppLockOpenHelper extends SQLiteOpenHelper {  
  
    public AppLockOpenHelper(Context context) {  
        super(context, "applock.db", null, 1);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        String sql = "create table applock(_id integer primary key  
autoincrement,packname varchar(20))";  
        db.execSQL(sql);  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
    }  
}
```

## 2. 不在最近任务列表显示 Activity (★★)

长按手机 Home 键，会列出最近任务列表中所有的 Activity（如下图所示），如果我们不想让我们的某个 Activity 出现在任务列表中，那么这时候就需要给我们的 Activity 额外配置一些信息了。

在 AndroidManifest.xml 的 Activity 节点中添加如下属性：

```
android:excludeFromRecents="true"
```



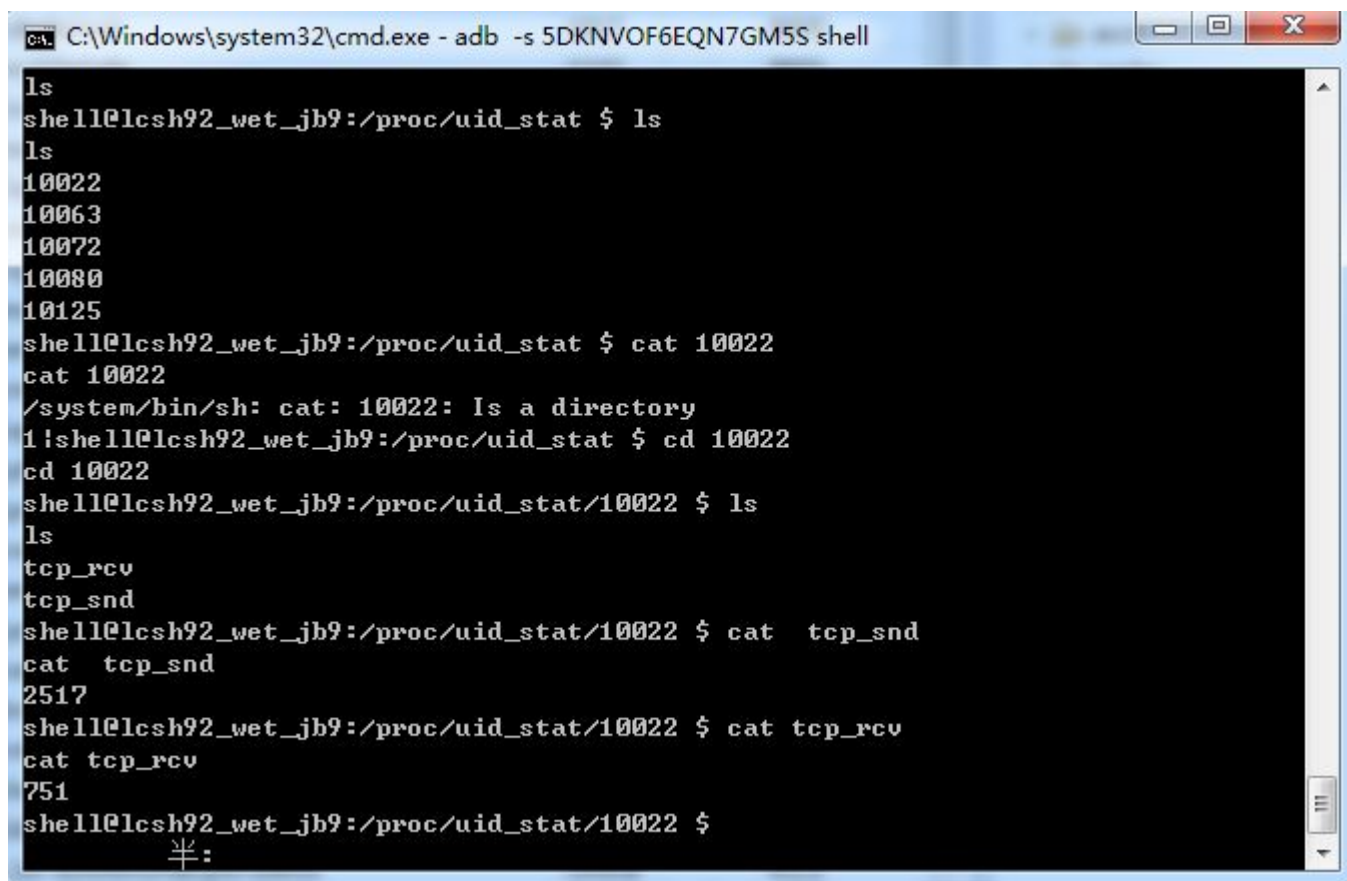
**Tips** : 上面的功能在 Android4.0 版本以后才添加的，在老版本中无此功能。

### 3. 流量统计简介 (★★)

Android 系统将各个进程使用的流量信息存储在 `/proc/uid_stat` 文件中，通过访问这个文件我们可以从中获取各个进程 (uid) 的流量使用情况，uid 是操作系统给每个应用分配的。

uid\_stat 目录下存放的是多个文件夹，文件夹名称就是 uid 名，进入 uid 目录，可以看到两个文件 `tcp_rcv` 和 `tcp_snd`。这两个文件分别存储了流量接收和发送的字节数。

如下图所示，是本人手机中 uid\_stat 文件目录内容。



```
C:\Windows\system32\cmd.exe - adb -s 5DKNVOF6EQN7GM5S shell
ls
shell@lclsh92_wet_jb9:/proc/uid_stat $ ls
ls
10022
10063
10072
10080
10125
shell@lclsh92_wet_jb9:/proc/uid_stat $ cat 10022
cat 10022
/system/bin/sh: cat: 10022: Is a directory
1!shell@lclsh92_wet_jb9:/proc/uid_stat $ cd 10022
cd 10022
shell@lclsh92_wet_jb9:/proc/uid_stat/10022 $ ls
ls
tcp_rcv
tcp_snd
shell@lclsh92_wet_jb9:/proc/uid_stat/10022 $ cat tcp_snd
cat tcp_snd
2517
shell@lclsh92_wet_jb9:/proc/uid_stat/10022 $ cat tcp_rcv
cat tcp_rcv
751
shell@lclsh92_wet_jb9:/proc/uid_stat/10022 $
半:
```

**Tips** : uid\_stat 文件目录在手机每次重启的时候都会清空，而且只有当手机连接网络并产生流量的情况下才会创建该文件目录，因此我们在模拟器上可能看不到该目录。可以连接我们的真机，连接前先在 cmd 中用 adb devices 查看设备名称，然后在 cmd 中 adb -s 设备名（拷贝过来就行，真机的一般比较长） shell。

uid 是操作系统分配的并且不是唯一的，可能多个应用程序公用一个 uid（不过这种情况基本没有发现过）。如果进行流量统计需要知道应用的包名和 uid 的对应关系。获取 uid 代码片段：

```
// 获取 PackageManager 对象
PackageManager pm = context.getPackageManager();
// 获取所有安装包信息
List<PackageInfo> packages = pm.getInstalledPackages(0);
// 创建 AppInfo 集合用于存储应用信息
List<AppInfo> appInfos = new ArrayList<AppInfo>();
for (PackageInfo info : packages) {
    int uid = info.applicationInfo.uid;
    System.out.println(info.applicationInfo.packageName+"对应的 uid="+uid);
}
```

## 4. 抽屉效果 (★)

在手机卫士中只简单介绍流量统计的原理，而不再实现具体的代码。同时在流量统计功能中我们将引入一个比较传统的抽屉效果的使用。如下所以，当鼠标点击锁图标并往上滑动时可以像抽屉一样抽出一部分视图，这就是抽屉效果。



抽屉效果在流量统计功能中实现，该类名为 `com.itheima.mobileSafe.activity.TrafficManagerActivity`，因为我们并没有业务逻辑，因此这里只给出能实现上图效果的布局代码。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="55dp"
```



```
        android:background="#8866ff00"
        android:gravity="center"
        android:text="流量统计"
        android:textColor="#000000"
        android:textSize="20sp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <View
        android:layout_width="match_parent"
        android:layout_height="250dp"
        />
    <SlidingDrawer
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:content="@+id/myContent"
        android:handle="@+id/myHandle"
        >
        <LinearLayout
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:orientation="vertical"
            >
            <ImageView
                android:id="@+id/myHandle"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/lock" />
        </LinearLayout>
        <LinearLayout
            android:gravity="center"
            android:background="#440000"
            android:id="@+id/myContent"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >

            <TextView
                android:text="我是抽屉"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    </LinearLayout>
</SlidingDrawer>
</LinearLayout>
</LinearLayout>
```

**Tips**：抽屉的核心类 `SlidingDrawer`，该类必须有 `android:content="@+id/myContent"` 和

`android:handle="@+id/myHandle"` 属性，其 id 对应 `SlidingDrawer` 的子节点。

**至此，本文档完！**

2015 年 2 月 16 日 星期一 11:08:12

河南省济源市梨林镇