# 常见布局

## 线性布局：LinearLayout

线性布局有水平（horizontal）和垂直（vertical）两个方向，android:orientation="horizontal"。

Android组件的宽高是在创建组件时就分配好的，所以使用match\_parent时注意不要把其他组件顶出去。

设置组件右对齐：android:layout\_gravity="right"

当竖直布局时，只能左右对齐和水平居中，顶部底部对齐竖直居中无效

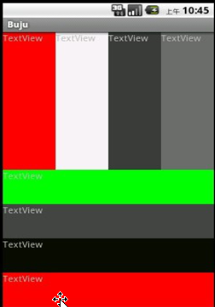
当水平布局时，只能顶部底部对齐和竖直居中

* 权重：权重设置的是按比例分配剩余的空间，android:layout\_weight="1"

先为各个组件分配好预先定义的空间。然后再把剩余的空间按比例分配给各个组件

如果是垂直方向，权重分配的就是垂直方向的空间。反之，权重分配的是水平方向的空间。一般，使用权重时，如果是水平方向，组件的宽定义为0dp。反之，组件的高为0dp。这样使用权重分配的空间更加精准

### 练习：



linearlayout\_demo.xml

|  |
| --- |
| <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  tools:context=*".MainActivity"*  android:orientation=*"vertical"* >    <LinearLayout  android:layout\_weight=*"1"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"0dp"*  android:orientation=*"horizontal"*  >    <TextView  android:layout\_weight=*"1"*  android:layout\_width=*"0dp"*  android:layout\_height=*"match\_parent"*  android:background=*"#ff0000"*/>    <TextView  android:layout\_weight=*"1"*  android:layout\_width=*"0dp"*  android:layout\_height=*"match\_parent"*  android:background=*"#ffffff"*/>    <TextView  android:layout\_weight=*"1"*  android:layout\_width=*"0dp"*  android:layout\_height=*"match\_parent"*  android:background=*"#00000000"*/>    <TextView  android:layout\_weight=*"1"*  android:layout\_width=*"0dp"*  android:layout\_height=*"match\_parent"*  android:background=*"@android:color/darker\_gray"*/>    </LinearLayout>    <LinearLayout  android:layout\_weight=*"1"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"0dp"*  android:orientation=*"vertical"*  >  <TextView  android:layout\_weight=*"1"*  android:layout\_height=*"0dp"*  android:layout\_width=*"match\_parent"*  android:background=*"#00ff00"*/>  <TextView  android:layout\_weight=*"1"*  android:layout\_height=*"0dp"*  android:layout\_width=*"match\_parent"*  android:background=*"@android:color/darker\_gray"*/>  <TextView  android:layout\_weight=*"1"*  android:layout\_height=*"0dp"*  android:layout\_width=*"match\_parent"*  android:background=*"#000000"*/>  <TextView  android:layout\_weight=*"1"*  android:layout\_height=*"0dp"*  android:layout\_width=*"match\_parent"*  android:background=*"#ffcc0000"*/>  </LinearLayout>    </LinearLayout> |

## 相对布局：RelativeLayout

组件默认左对齐、顶部对齐

* 在指定组件的哪一边：
  + android:layout\_toRightOf="@id/tv1"：在指定组件的右边
  + android:layout\_toLeftOf="@id/tv1"：在指定组件的左边
  + android:layout\_above="@id/tv1"：在指定组件的上面
  + android:layout\_below="@id/tv1"：在指定组件的下面
* 与指定组件哪一边对齐：
  + android:layout\_alignRight="@id/tv1"：与指定组件右边对齐
  + android:layout\_alignLeft="@id/center"：与指定组件左边对齐
  + android:layout\_alignTop="@id/center"：在指定组件的上面对齐
  + android:layout\_alignBottom="@id/center"：在指定组建的下面对齐
* 与父元素哪一边对齐
  + android:layout\_alignParentRight="true"：与父元素的右边对齐
  + android:layout\_alignParentLeft="true"：与父元素的左边对齐
  + android:layout\_alignParentBottom="true"：与父元素顶部对齐
  + android:layout\_alignParentTop="true"：与父元素底部对齐
* 居中：
  + android:layout\_centerVertical="true"：相对父元素垂直居中
  + android:layout\_centerHorizontal="true"：相对父元素水平居中
  + android:layout\_centerInParent="true"：相对父元素水平且垂直居中

### 练习：



## 帧布局：FrameLayout

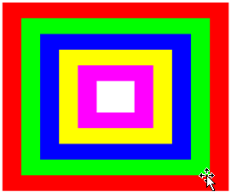
默认组件都是左对齐和顶部对齐，每个组件相当于一个div，可以重叠

可以更改对齐方式：android:layout\_gravity="bottom"

也可以这样: android:layout\_gravity="bottom|center"，底部对齐和居中

不能相对于其他组件布局

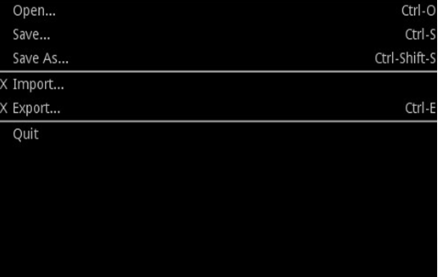
### 练习：



## 表格布局：TableLayout

* 每个<TableRow/>节点是一行，它的每个子节点是一列
* 表格布局中的节点可以不设置宽高，因为设置了也无效
* 根节点<TableLayout/>的子节点宽为匹配父元素，高为包裹内容
* <TableRow/>节点的子节点宽为包裹内容，高为包裹内容
* 以上默认属性无法修改
* 根节点中可以设置以下属性，表示让第1列拉伸填满屏幕宽度的剩余空间
* android:stretchColumns="1"
* <TableRow/>节点的子节点可以设置以下属性
  + android:layout\_column="1"：给这个列，设置为第一列
  + android:layout\_span="2"：让这一列占用两列的空间

### 练习：



## 绝对布局：AbsoluteLayout

直接指定组件的x、y坐标：

android:layout\_x="144dp"

android:layout\_y="154dp"

# 、LogCat

* 日志信息总共分为5个等级
  + verbose：冗余，最低等级
  + debug：调试
  + info：正常等级的信息，普通信息
  + warn：警告
  + error：错误
* 定义过滤器方便查看
  + 一般过滤条件为tag的名称
  + System.out.print输出的日志级别是info，tag是System.out

## Android提供的日志输出api。可以为输出的日志定义tag的名称，和日志详细信息

* + Log.v(TAG, "加油吧，童鞋们");
  + Log.d(TAG, "加油吧，童鞋们");
  + Log.i(TAG, "加油吧，童鞋们");
  + Log.w(TAG, "加油吧，童鞋们");
  + Log.e(TAG, "加油吧，童鞋们");

# 、Android的存储

* 内部存储空间
* RAM内存：运行内存，相当于电脑的内存
* ROM内存：存储内存，相当于电脑的硬盘
  + data/data/应用包名
* 外部存储空间
  + SD卡：相当于电脑的移动硬盘
    - 2.2之前，sd卡路径：sdcard
    - 4.3之前，sd卡路径：mnt/sdcard
    - 4.3开始，sd卡路径：storage/sdcard
  + 为了兼容低版本，Android把旧版的的路径改为4.3之后版本路径的快捷方式。

就是2.2和4.3版本的sd卡路径，指向的是4.3sd卡的路径

# 、在内部存储空间中读写文件

## 小案例：用户输入账号密码，勾选“记住账号密码”，点击登录按钮，登录的同时持久化保存账号和密码

### 完成布局

activity\_main.xml

|  |
| --- |
| <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  tools:context=*".MainActivity"*  android:orientation=*"vertical"* >  <EditText  android:id=*"@+id/et\_username"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:hint=*"用户名"* />  <EditText  android:id=*"@+id/et\_password"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:inputType=*"textPassword"*  android:hint=*"密码"* />    <RelativeLayout  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*>    <CheckBox  android:id=*"@+id/cb"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_centerVertical=*"true"*  android:text=*"记住用户名和密码"*/>    <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentRight=*"true"*  android:onClick=*"login"*  android:text=*"登录"*/>  </RelativeLayout>  </LinearLayout> |

### 业务逻辑完成：MainActivity.java

# 吐司对话框：

|  |
| --- |
| /\*  \* 创建吐司对话框  \* > 需要一个上下文对象Context  \* > 对话框的文本信息  \* > 持续的时间：  \* 只能是Toast.LENGTH\_LONG和Toast.LENGTH\_SHORT，5秒和3秒  \* 或者1和0  \*/  Toast t = Toast.makeText(**this**, "登录成功", Toast.LENGTH\_SHORT);  //显示吐司对话框  t.show(); |

# 、使用路径api读取内部存储空间的路径

ContextWrapper的方法：

* getFilesDir()：得到的file对象的路径是data/data/com.itheima.rwinrom2/files

存放在这个路径下的文件，只要你不删，它就一直在

* getCacheDir()：得到的file对象的路径是data/data/com.itheima.rwinrom2/cache

存放在这个路径下的文件，当内存不足时，有可能被删除

系统管理应用界面的清除缓存，会清除cache文件夹下的东西，清除数据，会清除整个包名目录下的东西

Environment类的方法

* getDataDirectory():返回一个File，其路径是内部存储（rom）的真是路径。其实就是data目录

# 、使用路径api读取外部存储空间的路径

部分手机品牌会更改sd卡的路径

Environment类的方法

getExternalStorageDirectory()：返回一个File对象，其路径是sd卡的真实路径

# 、检测sd卡状态

Environment类的方法

getExternalStorageState()：返回的是一个字符串

sd卡的状态有：也是字符串

* MEDIA\_UNKNOWN: 不能识别sd卡
* MEDIA\_REMOVED: 没有sd卡
* MEDIA\_UNMOUNTED: sd卡存在但是没有挂载
* MEDIA\_CHECKING: sd卡正在准备
* MEDIA\_MOUNTED：sd卡已经挂载，可用

if(Environment.getExternalStorageState().equals(Environment.MEDIA\_MOUNTED)) {}

# 、在外部存储空间中读写文件

## 小案例：登录

* 页面布局：activity\_main.xml
* 业务逻辑：MainActivity.java
* 写sd卡需要获取权限：

|  |
| --- |
| <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/> |

在4.0之前，读sd卡是不要权限的

但是4.0之后，读sd卡就可以有权限。

是因为，在手机的开发者模式中，多了一个选项，叫做保护sd卡。

如果勾上这个选项，那么读取sd卡就需要权限

|  |
| --- |
| <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/> |

# 、获取sd卡剩余空间

存储设备会被分为若干个区块，每个区块有固定的大小

区块大小 \* 区块数量 等于 存储设备的总大小

区块大小 \* 可用区块 等于 存储设备的剩余大小

## 案例：获取sd卡剩余空间

# 、文件读写权限

* 指的是谁能访问这个文件
* 在Android中，每一个应用，都是一个独立的用户。可以把多个用户设置进一个用户组
* 使用10个字母表示：drwxrwxrwx
* 第一个字母：
  + d：表示文件夹
  + -：表示文件
* 第一组rwx：表示的是文件拥有者（owner）对文件的权限
  + r：read，读
  + w：write，写
  + x：execute，执行
* 第二组rwx：表示的是跟文件拥有者属于同一用户组的用户（grouper）对文件的权限
* 第三组rwx：表示的其他用户（other）对文件的权限

## 案例：创建一个文件，权限是其他文件可读可写

使用ContextWrapper的方法openFileOutput(name, mode)，传入文件名，和权限

得到的是一个FileOutputStream，路径默认是：data/data/com.itheima.permission/files，不可更改

权限有：

* Context. MODE\_PRIVATE: 私有的。文件拥有者、同一用户组，可读写
* Context. MODE\_WORLD\_READABLE: 全局可读
* Context. MODE\_WORLD\_WRITEABLE：全局可写

openFileOutput(“in.txt”, MODE\_WORLD\_READABLE | MODE\_WORLD\_WRITEABLE) : 全局可读，并且全部可写

# 、SharedPreferences

* 用SharedPreferences保存零散数据(用户名密码)，是很合适的
* 它是使用xml文件来保存的，类似map的键值对
* 路径在data/data/com.itheima.sharedpreference/share\_prefs

## 存数据

|  |
| --- |
| //获取sharedpreference，使用ContextWrapper的方法。给出文件名，不用后缀。给出权限  SharedPreferences sp = getSharedPreferences("info", *MODE\_PRIVATE*);  //拿到编辑器  Editor ed = sp.edit();  //写数据，给出key，和value。可以写字符串，int…  ed.putString("username", "yonghu");  ed.putString("password", "mima");  //提交  ed.commit(); |

## 取数据

|  |
| --- |
| //得到SharedPreferences  SharedPreferences sp = getSharedPreferences("info", *MODE\_PRIVATE*);  /\*  \* 获取数据，可以获取字符串,int...  \* > 给出key  \* > 默认值：通过key找不到value，就返回默认值  \*/  String username = sp.getString("useranme", "");  String password = sp.getString("password", ""); |

## 存储方式

|  |
| --- |
| <?xml version=*'1.0'* encoding=*'utf-8'* standalone=*'yes'* ?>  <map>  <string name=*"password"*>yonghu</string>  <string name=*"username"*>mima</string>  </map> |

## 案例：登录

code/ login\_sharedpreference

# 、生成xml文件

## 案例：备份短信到sd卡

* 注意，写sd卡需要权限

# 、使用序列化器生成xml文件：XmlSerializer

## 案例：备份短信到sd卡

# 使用pull解析xml

* 事件类型主要有五种
  + START\_DOCUMENT：xml头的事件类型
  + END\_DOCUMENT：xml尾的事件类型
  + START\_TAG：开始节点的事件类型
  + END\_TAG：结束节点的事件类型
  + TEXT：文本节点的事件类型

## 案例：pull解析xml