LinearLayout

[Общая информация](http://developer.alexanderklimov.ru/android/layout/linearlayout.php#common)  
[Примеры](http://developer.alexanderklimov.ru/android/layout/linearlayout.php#samples)  
[Градиентный фон](http://developer.alexanderklimov.ru/android/layout/linearlayout.php#gradient)  
[Меняем фон программно](http://developer.alexanderklimov.ru/android/layout/linearlayout.php#background)  
[Программная анимация компоновки](http://developer.alexanderklimov.ru/android/layout/linearlayout.php#codeanimation)  
[Отключаем выравнивание по базовой линии](http://developer.alexanderklimov.ru/android/layout/linearlayout.php#bottomline)  
[Разделители](http://developer.alexanderklimov.ru/android/layout/linearlayout.php#divider)

Общая информация

В студии макет **LinearLayout** представлен двумя вариантами - **Horizontal** и **Vertical**. Макет **LinearLayout** выравнивает все дочерние объекты в одном направлении — вертикально или горизонтально. Направление задается при помощи атрибута ориентации **android:orientation**:

* android:orientation="horizontal"
* android:orientation="vertical"

Все дочерние элементы помещаются в стек один за другим, так что вертикальный список компонентов будет иметь только один дочерний элемент в ряду независимо от того, насколько широким он является. Горизонтальное расположение списка будет размещать элементы в одну строку с высотой, равной высоте самого высокого дочернего элемента списка.

<?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="match\_parent"

android:layout\_weight="1"

android:orientation="horizontal">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:layout\_weight="1"

android:background="#aa0000"

android:gravity="center\_horizontal"

android:text="red"/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:background="#00aa00"

android:gravity="center\_horizontal"

android:text="green"/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:background="#0000aa"

android:gravity="center\_horizontal"

android:text="blue"/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:background="#aaaa00"

android:gravity="center\_horizontal"

android:text="yellow"/>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="fill\_parent"

android:layout\_weight="1"

android:orientation="vertical">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="row one"

android:textSize="15dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="row two"

android:textSize="15dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="row three"

android:textSize="15dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="row four"

android:textSize="15dp"/>

</LinearLayout>

</LinearLayout>

В этом примере используются два способа размещения элементов **TextView**: по горизонтали и по вертикали.



У разметки **LinearLayout** есть интересный атрибут **android:layout\_weight**, который назначает индивидуальный вес для дочернего элемента. Этот атрибут определяет "важность" представления и позволяет этому элементу расширяться, чтобы заполнить любое оставшееся пространство в родительском представлении. Заданный по умолчанию вес является нулевым.

Например, если есть три текстовых поля, и двум из них объявлен вес со значением 1, в то время как другому не даётся никакого веса (0), третье текстовое поле без веса не будет расширяться и займёт область, определяемую размером текста, отображаемого этим полем. Другие два расширятся одинаково, чтобы заполнить остаток пространства, не занятого третьим полем. Если третьему полю присвоить вес 2 (вместо 0), это поле будет объявлено как "более важное", чем два других, так что третье поле получит 50% общего пространства, в то время как первые два получат по 25% общего пространства.

Также можно указать атрибут **android:weightSum**. Если атрибуту присвоить значение 100, то можно указывать вес дочерних элементов в удобном виде, как в процентах. Такой способ широко используется веб-мастерами при вёрстке.

Создадим простейшую разметку таким образом, чтобы дочерний элемент занимал половину родительского контейнера:

<?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:background="#0000FF"

android:orientation="vertical"

android:weightSum="100">

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="0dp"

android:layout\_weight="50"

android:background="#FFFF00"

android:gravity="center">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="10dp"

android:gravity="center"

android:text="и треснул мир напополам"

android:textColor="#000000"

android:textSize="20sp"

android:textStyle="bold"/>

</LinearLayout>

</LinearLayout>



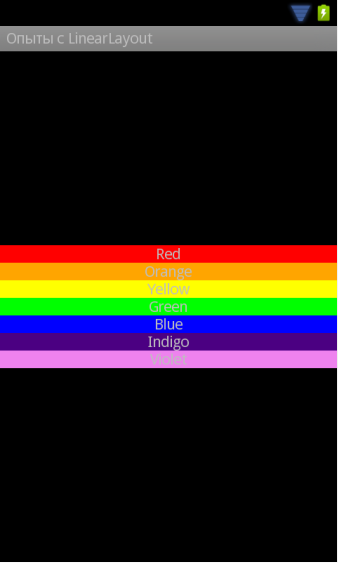
Примеры

Рассмотрим возможности **LinearLayout** на примерах.

Создадим 7 текстовых меток и присвоим им цвета радуги. Расположим их друг за другом. Получим следующий результат



Отцентрируем текст в **TextView** при помощи свойства **Gravity**, установив значение **Center**. Аналогично поступим и с **LinearLayout**, чтобы выровнять по центру текстовые метки.



Цветные полоски получились слишком узкими. Расширим их за счет увеличения размера шрифта (**TextSize**) у текстовых меток.



Стало чуть лучше, но все равно пропадает много свободного пространства. Совсем не хочется видеть чёрный цвет сверху и снизу. Здесь нам придёт на помощь свойство **Layout weight**. Так как число 7 не совсем удобное для деления, то поступим следующим образом. Пяти элементам присвоим вес 0.14, а двум - 0.15, что в сумме даст 1.0. Теперь элементы равномерно заполнят весь экран.



Если мы хотим сделать плавное увеличение высоты полоски, то вес нужно распределить таким образом: 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.22.



Чем хорош этот способ? Мы не привязываемся к точным размерам, а позволяем системе самой расчитывать равномерное распределение элементов по экрану. Если в Eclipse вы выберите режим Landscape, то экран будет выводиться в альбомном режиме и при этом элементы по-прежнему будет равномерно распределены.



Градиентный фон

Если вам нужен градиентный фон для **LinearLayout**, то создайте в папке **res/drawable** xml-файл, например, **gradient.xml**:

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle" >

<gradient

android:angle="90"

android:endColor="#00FF00"

android:startColor="#FF0000" />

</shape>

Далее остаётся только прописать файл в свойстве **Background**:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

**android:background="@drawable/gradient"**

android:orientation="vertical" >



Меняем фон программно

Чтобы программно сменить фоновый цвет у **LinearLayout**, нужно вызвать метод **setBackgroundColor()**. Пример изменения фона можно найти в статье [Android::Класс android.graphics.Color](http://developer.alexanderklimov.ru/android/catshop/color.php).

Программная анимация компоновки

Хочу показать один пример программной анимации. Не знаю, имеет ли пример практическую ценность, но для общего развития не помешает. Добавьте в шаблон **LinearLayout** несколько кнопок, текстовых полей и других элементов на ваше усмотрение. Далее пишем код для обработчика щелчка кнопки и вспомогательный класс для анимации:

public void onClick(View view) {

LinearLayout linearLayout = (LinearLayout) findViewById(R.id.linearLayout);

linearLayout.startAnimation(new ViewAnimation());

}

public class ViewAnimation extends Animation {

int centerX, centerY;

@Override

public void initialize(int width, int height, int parentWidth,

int parentHeight) {

super.initialize(width, height, parentWidth, parentHeight);

setDuration(5000);

setFillAfter(true);

setInterpolator(new LinearInterpolator());

centerX = width / 2;

centerY = height / 2;

}

@Override

protected void applyTransformation(float interpolatedTime,

Transformation transformation) {

final Matrix matrix = transformation.getMatrix();

matrix.setScale(interpolatedTime, interpolatedTime);

}

}

Когда вы щелкните на кнопке, то **LinearLayout** будет плавно увеличиваться в размерах. Данный приём можно использовать не только к компоновке, но и к любому объекту **View**, например, к [аналоговым часам](http://developer.alexanderklimov.ru/android/views/analogclock_digitalclock.php#animation)

Отключаем выравнивание по базовой линии

Допустим, у вас есть следующая разметка:

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="70dip"

android:background="#ffffff" >

<Button

android:layout\_width="0dip"

android:layout\_height="70dip"

android:layout\_weight="1"

android:gravity="center"

android:text="Котик"

android:textSize="15sp" />

<Button

android:layout\_width="0dip"

android:layout\_height="70dip"

android:layout\_weight="1"

android:gravity="center"

android:text="Кот, кошка и котята"

android:textSize="15sp" />

<Button

android:layout\_width="0dip"

android:layout\_height="70dip"

android:layout\_weight="1"

android:gravity="center"

android:text="Кошечка"

android:textSize="15sp" />

</LinearLayout>

Если посмотрим, что получилось, то увидим, что средняя кнопка опустилась вниз.



Строго говоря, разметка составлена не совсем правильно, используя жёстко установленные величины. Но будем считать, что такой код достался от другого программиста и заказчик не хочет его кардинально менять. Причина такого поведения кнопки в том, что по умолчанию Android пытается выравнивать элементы по некой базовой линии. А средняя кнопка имеет слишком длинный текст и она вот таким причудливым образом сместилась вниз. Можно попробовать использовать свойства **gravity**, но можно поступить проще. Добавьте атрибут**android:baselineAligned="false"** к **LinearLayout** и все три кнопки будут аккуратно находиться на одной линии. Имейте в виду, может пригодится.

Разделители

Начиная с API 11, у **LinearLayout** появился новый атрибут **android:divider**, позволяющий задать графический разделитель между кнопками. Создадим в папке **res/drawable** файл**separator.xml**:

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android" >

<size android:width="1dp" />

<solid android:color="#90909090" />

</shape>

Разметка для активности:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:adjustViewBounds="true"

android:divider="@drawable/separator"

android:dividerPadding="3dp"

android:orientation="horizontal"

android:showDividers="middle" >

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="?android:attr/buttonBarButtonStyle"

android:text="Yes"

android:layout\_weight="1"

android:id="@+id/button1"

android:textColor="#00b0e4" />

<Button

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

style="?android:attr/buttonBarButtonStyle"

android:text="No"

android:layout\_weight="1"

android:id="@+id/button2"

android:textColor="#00b0e4" />

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="?android:attr/buttonBarButtonStyle"

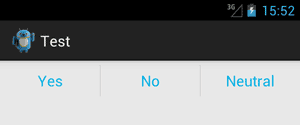
android:text="Neutral"

android:layout\_weight="1"

android:id="@+id/button3"

android:textColor="#00b0e4" />

</LinearLayout>



Разделители могут оказаться полезными. В статье [Grid Spacing on Android](http://cyrilmottier.com/2014/11/17/grid-spacing-on-android/) показан хороший пример на эту тему.

Допустим, мы хотим вывести в ряд три кнопки под каким-то компонентом, например, логотипом компании. Разметка может быть следующей.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="@dimen/spacing\_medium">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="128dp"

android:background="@color/light\_gray"

android:gravity="center"

android:text="@string/application\_logo"

android:textAppearance="@android:style/TextAppearance.Material.Display1" />

<LinearLayout

android:id="@+id/buttons\_container"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<Button

android:id="@+id/btn\_first"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:background="@drawable/purple"

android:text="@string/button\_1" />

<Button

android:id="@+id/btn\_second"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:background="@drawable/indigo"

android:text="@string/button\_2" />

<Button

android:id="@+id/btn\_third"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

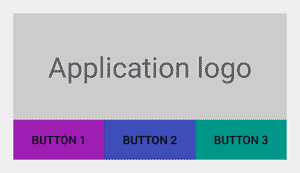
android:background="@drawable/teal"

android:text="@string/button\_3" />

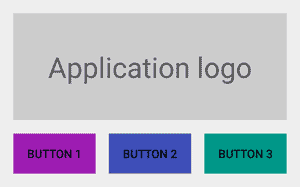
</LinearLayout>

</LinearLayout>

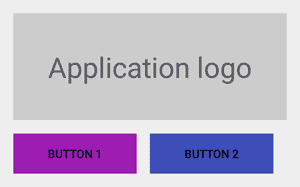
Вместо **@dimen/spacing\_medium** можете подставить 8dp, а цвета придумать свои, если будете проверять пример самостоятельно.



Видно, что напрашивается дизайн в виде сетки. Отсутствие пространства между кнопками может создать неудобства у пользователя. Добавим их. У контейнера **@id/buttons\_container**добавим **android:layout\_marginTop="@dimen/spacing\_medium"**, а у первой и второй кнопки добавим **android:layout\_marginRight="@dimen/spacing\_medium"** (напомню, можно использовать 8dp)



Всё отлично работает до того момента, если нам понадобится программно убрать с экрана третью кнопку. Сделать это можно через **View.GONE**. И что мы увидим?



Теперь вторая кнопка не выровнена по правому краю. Некрасиво. Очень плохим решением станет программный пересчёт всех величин, чтобы выровнять компоненты. Как вариант, использовать другой тип разметки, например, **GridLayout**. Но у него есть свои проблемы с отступами и вам будет тяжело добиться нужного результата.

Проблема красиво решается с помощью упомянутых выше разделителей. Создадим в папке **res/drawable** файл **spacer\_medium.xml**:

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<size

android:width="@dimen/spacing\_medium"

android:height="@dimen/spacing\_medium" />

<solid android:color="@android:color/transparent" />

</shape>

Применим разделители

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:divider="@drawable/spacer\_medium"

android:orientation="vertical"

android:padding="@dimen/spacing\_medium"

android:showDividers="middle">

<!-- TextView -->

<LinearLayout

android:id="@+id/buttons\_container"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:divider="@drawable/spacer\_medium"

android:orientation="horizontal"

android:showDividers="middle">

<!-- Buttons -->

</LinearLayout>

</LinearLayout>

Теперь кнопки всегда будут аккуратно выравнены по краям, независимо от их количества - две или три.

Программное создание разметки LinearLayout

В некоторых случаях может понадобиться создать **LinearLayout** программным способом. Сначала создаётся объект **LayoutParams**, на его основе создаётся **LinearLayout**, а позже в него добавляются дочерние компоненты.

LayoutParams params = newLinearLayout.LayoutParams(LayoutParams.MATCH\_PARENT,

LayoutParams.WRAP\_CONTENT);

// create a layout

LinearLayout layout = newLinearLayout(this);

layout.setOrientation(LinearLayout.VERTICAL);

// create a textview

TextView textView = newTextView(this);

textView.setText("This is a TextView");

textView.setLayoutParams(params);

// create a button

Button button = newButton(this);

button.setText("This is a Button");

button.setLayoutParams(params);

// adds the textview

layout.addView(textView);

// adds the button

layout.addView(button);

// create a layout param for the layout

LinearLayout.LayoutParams layoutParam = newLinearLayout.LayoutParams(LayoutParams.MATCH\_PARENT,

LayoutParams.WRAP\_CONTENT);

this.addContentView(layout, layoutParam);

Также программно можно управлять настройками **LinearLayout** через тот же объект **LayoutParams**. Разместите кнопку с обработчиком щелчка.

<?xml version="1.0" encoding="utf-8"?>

<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"

android:orientation="vertical"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context="ru.alexanderklimov.as21.MainActivity">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge"/>

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onClick"

android:text="Meow"/>

</LinearLayout>

Код для кнопки.

public void onClick(View view) {

((TextView)findViewById(

R.id.textView)).setText("Меняем отступ программно");

LinearLayout.LayoutParams params = (LinearLayout.

LayoutParams) view.getLayoutParams();

params.leftMargin += 5;

}

Каждый щелчок на кнопке будет увеличивать отступ на пять единиц и кнопка будет смещаться в сторону. Интересно, что если убрать **TextView**, то кнопка перестаёт двигаться. Причина мне неизвестна.