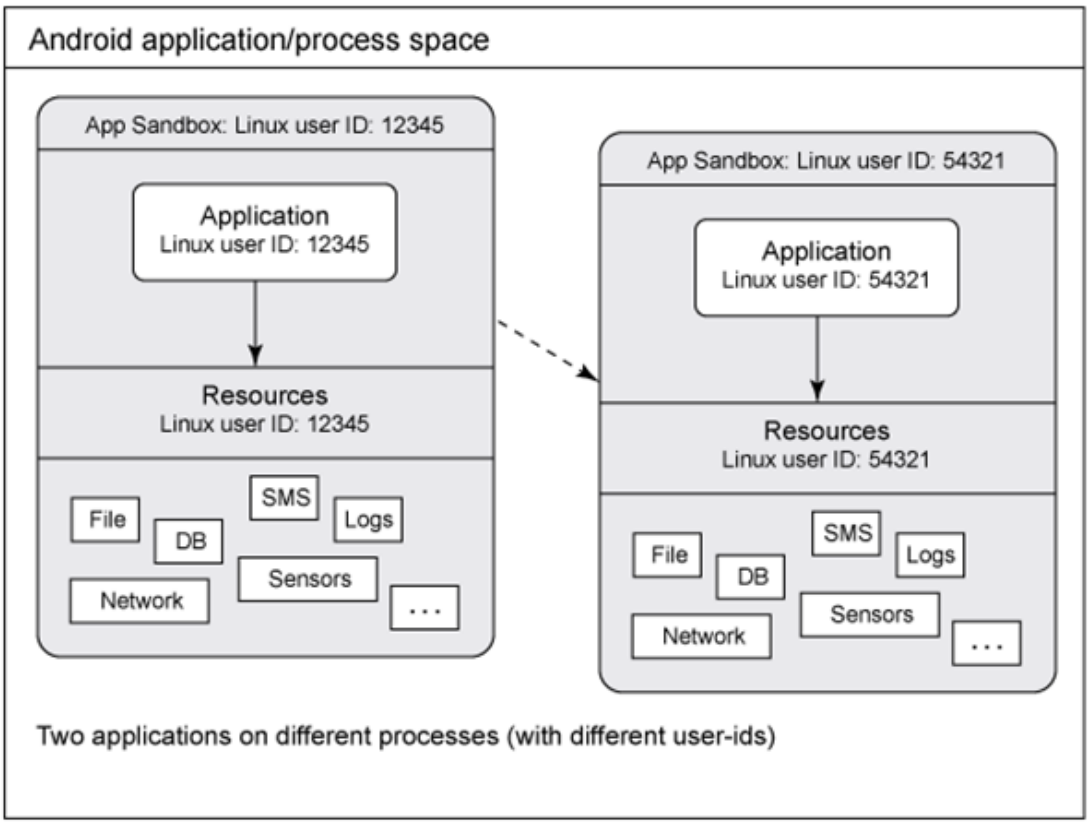
Android 进程和线程

1.1 知识解析

在安装 Android 应用程序的时候，Android 会为每个程序分配一个 Linux 用户 ID，并设置相应的权限，这样其它应用程序就不能访问此应用程序所拥有的数据和资源了。

下图中，两个 Android 应用程序，各自运行在其自己的基本沙箱或进程上。它们有不同的 Linux user ID。



当一个程序第一次启动时，Android 会同时启动一个对应的主线程（Main Thread），主线程主要负责处理与 UI 相关的事件，如：用户的按键事件，用户接触屏幕的事件以及

屏幕绘图事件，并把相关的事件分发到对应的组件进行处理。所以主线程通常又被叫做 UI 线程。

在开发 Android 应用时必须遵守单线程模型的原则： Android UI 操作并不是线程安全的并且这些操作必须在 UI 线程中执行。

如果在非 UI 线程中直接操作 UI，会抛出 android.view.ViewRoot$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views

由于 UI 线程负责事件的监听和绘图，因此，必须保证 UI 线程能够随时响应用户的需求，UI 线程里的操作应该像中断事件那样短小，费时的操作（如网络连接）需要另开线程，否则，如果 UI 线程超过 5s 没有响应用户请求，会弹出对话框提醒用户终止应用程序。

如果在新创建的线程中需要对 UI 进行设定，就可能违反单线程模型，因此 android 采用一种复杂的 Message Queue 机制保证线程间通信。

Message Queue 是一个消息队列，用来存放通过 Handler 发布的消息。Android 在第一次启动程序时会默认会为 UI thread 创建一个关联的消息队列以及 Looper，可以通过 Looper.myQueue()得到当前线程的消息队列，用来管理程序的一些上层组件，如 Activities、BroadcastReceivers 等等。你可以在自己的子线程中创建 Handler 与 UI thread 通信。

Android 通过 Looper、Handler 来实现消息循环机制，Android 消息循环是针对线程的（每个线程都可以有自己的消息队列和消息 Looper）。

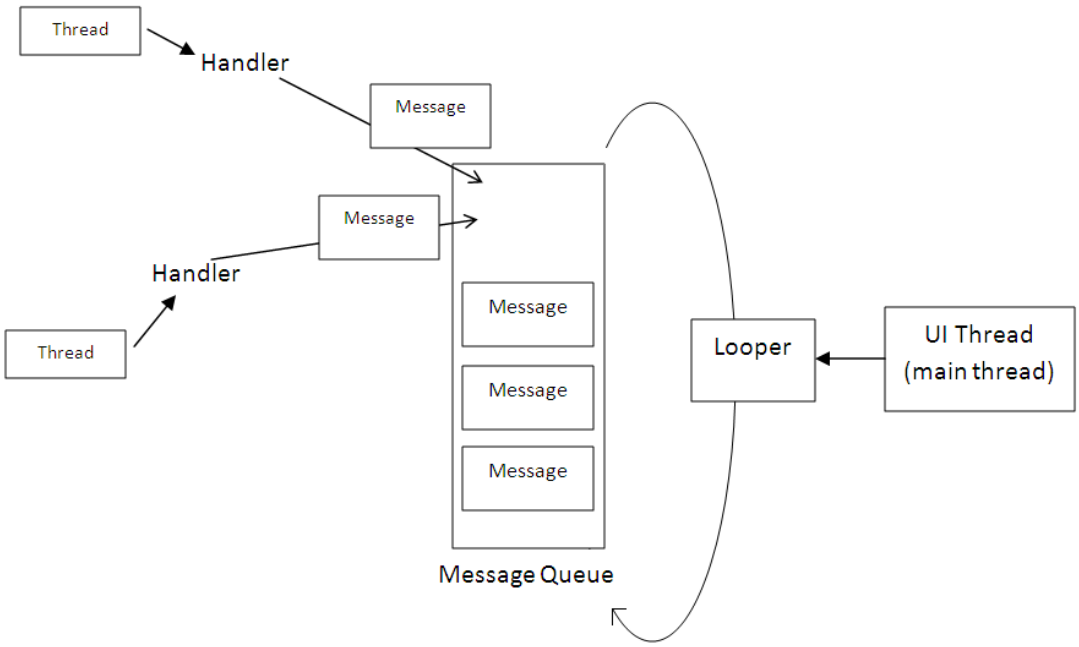
Android 系统中，Looper 负责管理线程的消息队列和消息循环。我们可以通过 Looper.myLooper()得到当前线程的 Looper 对象，通过 Looper.getMainLooper()可以获得当前进程的主线程的 Looper 对象。

一个线程可以存在（当然也可以不存在）一个消息队列和一个消息循环（Looper）。

Looper.prepare();…Looper.loop();

构造 Handler 的时候可以指定一个 Looper 对象，如果不指定则利用当前线程的 Looper 创建。

下图展示了 Looper 和 Handler 合作处理异步信息的方法。



Message

Message 类可以用于封装要传递的信息。其中包含了消息 ID，消息处理对象以及处理的数据等，由 MessageQueue 统一管理，由 Handler 处理（包括发送和处理）。

Message 属性：

- int what
- Object obj
- int arg1
- int arg2

Message 方法

- setData(Bundle data)
- Bundle getData()

一般使用调用 Message 静态方法的方式来获得 Message 对象：

Message obtain(Handler h, int what, int arg1, int arg2, Object obj) 其中，

Handler 用于指定这个 Message 由哪个 Handler 来处理（handle）

```
Message obtain(Handler h, int what, Object obj)
Message obtain(Handler h, int what)
Message obtain(Handler h)
Message obtain(Handler h, Runnable callback)
Message obtain()
Message obtain(Handler h, int what, int arg1, int arg2)
```

Handler

Looper（循环器），不断地从 MessageQueue 中抽取 Message 执行。因此，一个 MessageQueue 需要一个 Looper。

Handler 发出的 Message 并不是直接就交给 Handler 处理的，而是会放入 MessageQueue，由 Looper 根据先后顺序取出来再交给 Handler 处理。

方法	描述
void handleMessage(Message message)	通过这个方法处理消息
boolean sendEmptyMessage(int what)	发送只有一个 what 值的消息
boolean sendMessage(Message message)	发送消息到 MessageQueue，通过 Looper 取出后交给 Handler，Handler 即可用 handleMessage 处理

boolean hasMessage(int what)	判断是否有 what 值的消息
boolean post(Runnable r)	将一个线程添加到消息队列

Looper

创建的工作线程默认是没有消息循环和消息队列的，如果想让该线程具有消息队列和消息循环，需要在线程中首先调用 Looper.prepare()来创建消息队列，然后调用 Looper.loop()进入消息循环。 如下所示：

```
class LooperThread extends Thread {
      public Handler mHandler;

      public void run() {
          Looper.prepare();

          mHandler = new Handler() {
              public void handleMessage(Message msg) {
                  // process incoming messages here
              }
          };

          Looper.loop();
      }
  }
```

1.2 功能演示

1.4 职业素质

当一个程序第一次启动的时候，Android 会启动一个 LINUX 进程和一个主线程。默认的情况下，所有该程序的组件都将在该进程和线程中运行。 同时，Android 会为每个应用程序分配一个单独的 LINUX 用户。Android 会尽量保留一个正在运行进程，只在内存资源出现不足时，Android 会尝试停止一些进程从而释放足够的资源给其他新的进程使用，也能保证用户正在访问的当前进程有足够的资源去及时地响应用户的事件。