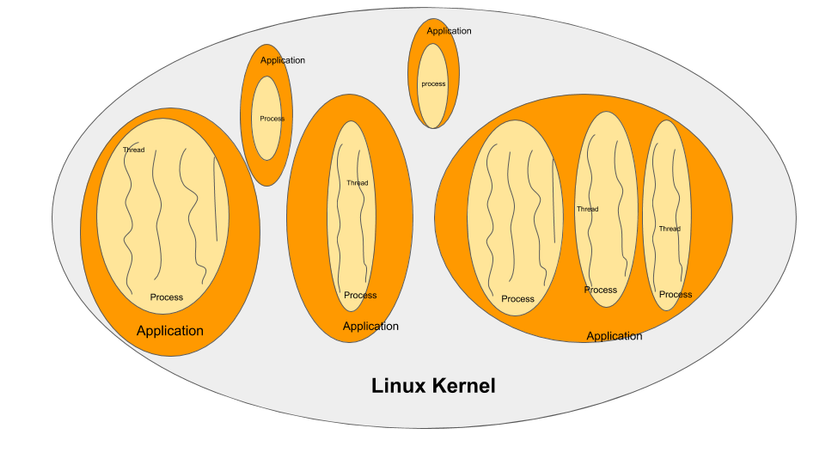
**Process trong Android**

Android có 4 components chính là ***Activity***, ***Service***, ***BroadcastReceiver***, ***ContentProvider***, khi một component của ứng dụng được khởi chạy mà trước đó ứng dụng không có bất kì component nào đang chạy, Hệ thống android khởi động một tiến trình (process) Linux cho ứng dụng với một luồng duy nhất.

Mặc định tất cả các component của cùng một ứng dụng sẽ được chạy trên một process và thread duy nhất (nó còn được gọi là Main Thread).

Các bạn có thể hiểu đơn giản rằng, Trong điện thoại có rất nhiều ứng dụng, mỗi ứng dụng lại có rất nhiều các thành phần, mỗi thành phần lại chạy trên các luồng khác nhau, và các luồng này thì được quản lý bằng các process khác nhau. Các bạn có thể hình dung qua hình sau:



* Mỗi Process cung cấp tài nguyên cần thiết để thực thi chương trình. Mỗi Process có một không gian địa chỉ ảo, có các mã thực thi, có các lệnh xử lý các đối tượng hệ thống, một ngữ cảnh bảo mật, kích thước làm việc tối thiểu và tối đa, ... và phải có ít nhất một Thread thực thi.
* Một Thread là một thực thể trong một Process, là đối tượng được lên kế hoạc để thực thi. Trong Process thường sẽ có nhiều Thread và tất cả các Thread này sẽ chia sẽ không gian địa chỉ ảo và tài nguyên hệ thống trong Process. Ngoài ra mỗi Thread lại có công việc riêng của nó đó là: duy trì xử lý ngoại lệ, ưu tiên lập lịch trình, lưu dữ cục bộ luồng,... Môi trường Thread có thể bao gồm: phần đăng kí Thread với máy chủ, nhân stack, ... quan trọng là có một môi trường (Thread cũng có ngữ cảnh riêng của nó).
* Tóm lại thì:

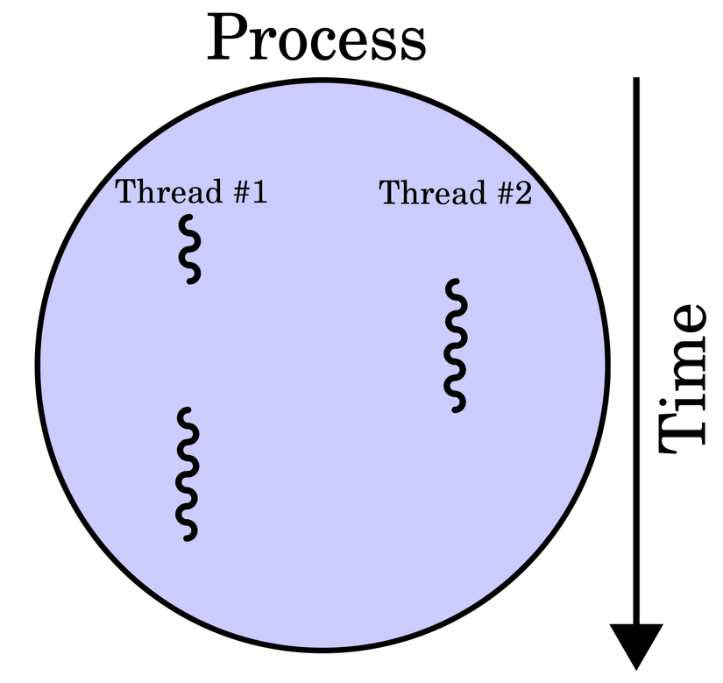
***Process = Program + State of all Threads executing in Program***

Có nghĩa là ***Process bao gồm chương trình cùng với tất cả trạng thái thực thi của các Thread trong chương trình đó***.

**Thread và MultiThread trong Android**

Vậy Thread là gì? Thread được định nghĩa là một luồng dùng để thực thi một chương trình. Java Virtual Machine cho phép một chương trình có thể có nhiều Thread thực thi đồng thời. Mỗi Thread đều có độ ưu tiên của nó.

Trong Android chúng ta có rất nhiều process,với mỗi một process chúng ta có thể tạo ra nhất nhiều thread trong đó, như hình sau:



Khi một ứng dụng được khởi chạy, hệ thống sẽ tự động khởi tạo một Thread cho ứng dụng cùng với một Process, gọi là ***Main Thread***. Thread này vô cùng quan trọng vì nó phụ trách việc nhận tương tác từ phía giao diện người dùng, bao gồm cả việc vẽ các sự kiện và update lại các giao diện người dùng. Như vậy Main Thread đôi khi còn được gọi là ***UI Thread*** .

Có khi nào ***Main Thread*** lại không được gọi là ***UI Thread*** không? Đó là khi một chương trình có nhiều hơn một Thread phụ trách việc xử lý giao diện.

Còn một trường hợp nữa là ***Worker Thread***, chính là Thread mà bạn tạo thêm cho chương trình để nó thực thi một công việc nào đó không liên quan đến giao diện, Thread này cũng được gọi là ***Background Thread***.

Khi ứng dụng của bạn thực hiện các tác vụ kéo dài để thực hiện các lệnh từ người dùng, việc chỉ sử dụng một luồng duy nhất trong Process mà hệ thống tạo ra sẽ gây rất chậm cho ứng dụng nhất là những tác vụ tuơng tác với internet, hoặc kết nối với database. Việc block UI trong một khoảng thời gian dài (trung bình là 5s) sẽ gây ra hiện tượng ANR (application not responding). Người dùng có thể quyết định tắt ứng dụng của các bạn.

Vì vậy chúng ta cần tuân thủ các rules khi sử dụng UI Thread như sau :

1. Không được block UI Thead
2. Không được truy cập vào Android UI Tookit từ thread không phải UI Thread. (Không được update UI ngoài Main Thread)

Để tạo mới Thread ta có hai cách. Cách thứ nhất là kế thừa (extends) từ class Thread:

private class MyThread extends Thread{

@Override

public void run() {

//TODO

}

}

...

new MyThread().start();

Cách thứ 2 là thực thi (implements) interface Runnable:

private class MyRunnable implements Runnable{

@Override

public void run() {

//TODO

}

}

...

new Thread(new MyRunnable()).start();

Ví dụ :

private void startCounting() {  
 Thread thread = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 for (int i = 0; i <= 10; i++) {  
 final int value = i;  
 // Không chết chương trình nhưng mà rất tù  
// mTvCount.post(new Runnable() {  
// @Override  
// public void run() {  
// mTvCount.setText(String.valueOf(value));  
// }  
// });  
  
// mTvCount.setText(String.valueOf(value)); chắc chắn chết mẹ chương trình  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
 thread.start();  
 }  
}

Ở trong Thread này nó đã trực tiếp tác động đến UI bằng câu lệnh *mTvCount.setText(value)* và không thỏa mãn rule thứ 2 nên code sẽ không chạy được. Từ ***Worker Thread*** ta không thể update UI. Từ đó Android đã cung cấp cho Worker Thread một số phương thức sau để làm điều đó.

1. \*Activity.runOnUiThread(Runnable)\*

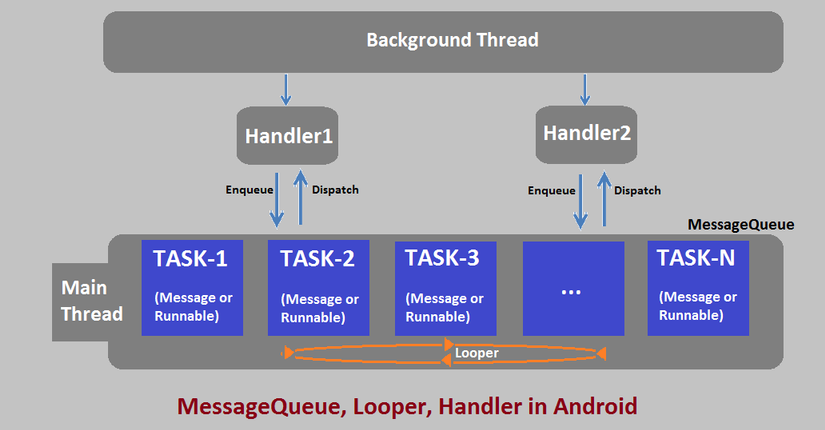
2. \*View.post(Runnable)\*

3. \*View.postDelayed(Runnable, long)\*

Nhưng chúng ta sẽ không dùng cách này vì nếu phương thức trên không có tham số là một View hay Activity truyền vào thì nó không thể update UI được, hoặc một số bài toán với dữ liệu lớn thì cũng không thể cập nhật giao diện bằng cách này được. Vậy giải pháp là gì? Chúng ta có hai cách đó là dùng ***Handler***, hoặc là ***AsyncTask***.

**Handler trong Android**

Handler là gì? Là một đối tượng Android cung cấp dùng để liên kết, trao đổi giữa các Thread với nhau, thường là Main Thread (UI Thread) với các Worker Thread (Background Thread). Handler có nhiệm vụ gửi và thực thi các Message hoặc Runnable tới Message Queue của Thread sinh ra nó (Handler). Handler luôn được gắn kết với một Thread (Thread sinh ra nó) cũng với Message Queue (của Thread đó).



Handler này sinh ra là thằng giao tiếp giữa Worker Thread và UI Thread

Thằng Handler này nó sẽ đóng gói các dữ liệu bằng cái phương thức sendMessage

Để hứng được cái message này thì dùng phương thức handleMessage

Ví dụ :

package com.example.task8bt;  
  
import android.os.Handler;  
import android.os.Message;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
 private TextView mTvCount;  
 private Button mBtStart;  
 private Handler mHandler;  
 private static final int *MESSAGE\_COUNT* = 101;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 initView();  
 initHandler();  
 }  
  
 private void initHandler() {  
 mHandler = new Handler(){  
 @Override  
 public void handleMessage(Message msg) {  
 switch (msg.what) {  
 case *MESSAGE\_COUNT*:  
 mTvCount.setText(String.*valueOf*(msg.arg1));  
 break;  
 default:  
 break;  
 }  
 }  
 };  
 }  
  
 */\*\*  
 \* Lấy ra View theo ID và đặt sự kiện cho View  
 \*/* private void initView() {  
 mTvCount = findViewById(R.id.*tv\_count*);  
 mBtStart = findViewById(R.id.*btn\_start*);  
 mBtStart.setOnClickListener(this);  
 }  
  
 */\*\*  
 \* Thực hiện hành động khi người dùng click vào view  
 \*  
 \** ***@param*** *v : View  
 \*/* @Override  
 public void onClick(View v) {  
// Dùng swicth case để chia trường hợp bắt sự kiện cho từng View  
 switch (v.getId()) {  
 case R.id.*btn\_start*:  
 startCounting();  
 break;  
 default:  
 break;  
 }  
 }  
  
 private void startCounting() {  
 Thread thread = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 for (int i = 0; i <= 10; i++) {  
 final int value = i;  
 // Không chết chương trình nhưng mà rất tù  
// mTvCount.post(new Runnable() {  
// @Override  
// public void run() {  
// mTvCount.setText(String.valueOf(value));  
// }  
// });  
  
// mTvCount.setText(String.valueOf(value)); chắc chắn chết mẹ chương trình  
 Message msg = new Message();  
 msg.what = *MESSAGE\_COUNT*;  
 msg.arg1 = value;  
 mHandler.sendMessage(msg);  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
 thread.start();  
 }  
}

**AsyncTask trong Android**