# **你的程序崩溃吗？**

无论多么认真，多么经验丰富的大牛，都不能保证应用程序不会崩溃。唯一能控制的就是崩溃出现的几率。那么怎么处理这个潜在的地雷，提高用户体验呢？可以各抒己见。在这里我做一下总结，方法就是：  
Thread.setDefaultUncaughtExceptionHandler(this);

# 一、“监听”崩溃异常

要想捕捉处理异常，首先要监听异常。说监听其实不用我们去设置监听器，因为系统帮我们做好了，我们需要做的就是设置默认异常处理器。同样系统为我们提供了默认的处理类，可以通过如下代码获取：  
Thread.getDefaultUncaughtExceptionHandler();  
可是我们需要做自己的处理，怎么办呢？自定义自己的异常处理类。实现Thread.UncaughtExceptionHandler接口，实现uncaughtException()方法。代码如下：

public class CrashHandler implements Thread.UncaughtExceptionHandler {

@Override

public void uncaughtException(Thread t,Throwable e) {}

然后通过如下代码设置监听，把异常处理对象传递进去：  
Thread.setDefaultUncaughtExceptionHandler(this);

# 二、处理异常

一般的处理无非，提示用户“这没啥，不就是崩溃么”、保存成日志文件、上传到服务器、重启应用程序。

下面贴出我根据个人习惯封装的工具类，没有使用单例模式，感觉OkHttp的Builder很好玩，就用了Builder。

public class CrashHandler implements UncaughtExceptionHandler {  
 public static final String *TAG* = CrashHandler.class.getSimpleName();  
 private Thread.UncaughtExceptionHandler mDefaultHandler; //系统异常处理器  
  
 private boolean debug; //是否调试模式  
 private Context context; //需要用到的上下文  
 private String tip; //错误时的提示信息  
 private String fileDir;//文件路径  
 private String fileName; //文件名不包括后缀  
 private String suffix;//后缀名  
  
 private CrashHandler(Builder builder) {  
 this.debug = builder.debug;  
 this.context = builder.contex;  
 this.tip = builder.tip;  
 this.fileName = builder.fileName;  
 this.fileDir = builder.fileDir;  
 this.suffix = builder.suffix;  
 }  
  
 */\*\*  
 \* 开始注册处理崩溃异常  
 \*/* public void catching() {  
 this.mDefaultHandler = Thread.*getDefaultUncaughtExceptionHandler*();  
 Thread.*setDefaultUncaughtExceptionHandler*(this);  
 Log.*d*(*TAG*, "Start catching");  
 }  
  
 */\*\*  
 \* 销毁异常捕捉  
 \*/* public void stop() {  
  
 }  
  
 @Override  
 public void uncaughtException(Thread t, Throwable e) {  
 // 若为调试则输出错误信息  
 if (debug) {  
 e.printStackTrace();  
 return;  
 }  
 if (!handleException(e) && mDefaultHandler != null) {  
 mDefaultHandler.uncaughtException(t, e);  
 } else {  
 restartApplication();  
 }  
 System.*exit*(0);// android.os.Process.killProcess(android.os.Process.myPid());  
 }  
  
 */\*\*  
 \* 处理异常  
 \*  
 \** ***@param*** *ex  
 \** ***@return*** *\*/* private boolean handleException(Throwable ex) {  
 if (ex == null) {  
 return false;  
 }  
  
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 Looper.*prepare*();  
 Toast.*makeText*(context, tip, Toast.*LENGTH\_LONG*).show();  
 Looper.*loop*();  
 }  
 }).start();  
  
 write2File(ex); //写入文件  
 upload2Server(ex); //上传到服务器  
  
 return true;  
 }  
  
 */\*\*  
 \* 把错误信息写入文件  
 \*  
 \** ***@param*** *ex  
 \*/* private void write2File(Throwable ex) {  
 // 如果没有存储卡则返回  
 String state = Environment.*getExternalStorageState*();  
 if (!TextUtils.*equals*(Environment.*MEDIA\_MOUNTED*, state)) {  
 return;  
 }  
 // 取得存储目录  
 File directory = getStoreFile();  
 if (!directory.exists()) {  
 directory.mkdir();  
 }  
 // 当前时间  
 long now = System.*currentTimeMillis*();  
 Date date = new Date(now);  
 SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
 String time = format.format(date);  
  
 File file = new File(directory + File.*separator* + fileName + "\_" + time + suffix);  
 FileWriter fw = null;  
 try {  
 fw = new FileWriter(file);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 if (fw == null) return;  
 PrintWriter pw = new PrintWriter(new BufferedWriter(fw));  
 // 写时间  
 pw.println(time);  
 writePhoneInfo(pw);  
 pw.println();  
 ex.printStackTrace(pw);  
 pw.close();  
 }  
  
 */\*\*  
 \* 存储在应用缓存目录下  
 \*  
 \** ***@return*** *\*/* private File getStoreFile() {  
 String path = null;  
 String state = Environment.*getExternalStorageState*();  
 boolean isRemovable = Environment.*isExternalStorageRemovable*();  
 if (TextUtils.*equals*(state, Environment.*MEDIA\_MOUNTED*) || !isRemovable) {  
 path = this.context.getExternalCacheDir().toString();  
 } else {  
 path = this.context.getCacheDir().toString();  
 }  
 return new File(path + File.*separator* + fileDir);  
 }  
  
 */\*\*  
 \* 手机型号等信息  
 \*  
 \** ***@param*** *pw  
 \*/* private void writePhoneInfo(PrintWriter pw) {  
 String packageName = this.context.getPackageName();  
 PackageManager packageManager = this.context.getPackageManager();  
 PackageInfo packageInfo = null;  
 try {  
 packageInfo = packageManager.getPackageInfo(packageName, PackageManager.*GET\_ACTIVITIES*);  
 } catch (PackageManager.NameNotFoundException e) {  
 e.printStackTrace();  
 }  
 pw.println("Application Version: " + packageInfo.versionName + "\_" + packageInfo.versionCode);  
 pw.println("OS Version: " + Build.VERSION.*RELEASE* + "\_" + Build.VERSION.*SDK\_INT*);  
 pw.println("Vendor: " + Build.*MANUFACTURER*);  
 pw.println("Model: " + Build.*MODEL*);  
 pw.println("CPU ABI: " + Build.*CPU\_ABI*);  
 }  
  
 */\*\*  
 \* 把错误信息上传到服务器  
 \*  
 \** ***@param*** *ex  
 \*/* private void upload2Server(Throwable ex) {  
 // insert code here  
 }  
  
 */\*\*  
 \* 重启Application  
 \*/* private void restartApplication() {  
 String packageName = this.context.getPackageName();  
 PackageManager packageManager = this.context.getPackageManager();  
 Intent intent = packageManager.getLaunchIntentForPackage(packageName);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);  
 this.context.startActivity(intent);  
 }  
  
 */\*\*  
 \* 构造器类  
 \*/* public static class Builder {  
 private boolean debug;//是否调试  
 private Context contex; // 上下文  
 private String fileDir; // 文件路径  
 private String fileName; // 文件名  
 private String suffix;//后缀名  
 private String tip; // 提示信息  
  
 public Builder(Context context) {  
 this.contex = context;  
 this.debug = false;  
 this.tip = "I am sorry.";  
 this.fileDir = "crash";  
 this.fileName = "crash\_log";  
 this.suffix = ".trace";  
 }  
  
 */\*\*  
 \* 是否调试模式  
 \*  
 \** ***@param*** *isDebug  
 \** ***@return*** *\*/* public Builder debug(boolean isDebug) {  
 this.debug = isDebug;  
 return this;  
 }  
  
 */\*\*  
 \* 错误Toast消息  
 \*  
 \** ***@param*** *tip  
 \** ***@return*** *\*/* public Builder tip(String tip) {  
 this.tip = tip;  
 return this;  
 }  
  
 */\*\*  
 \* 存储路径文件名以及后缀名  
 \*  
 \** ***@param*** *fileDir  
 \** ***@param*** *fileName  
 \** ***@param*** *suffix  
 \** ***@return*** *\*/* public Builder file(String fileDir, String fileName, String suffix) {  
 this.fileDir = fileDir;  
 this.fileName = fileName;  
 this.suffix = suffix;  
 return this;  
 }  
  
 public CrashHandler build() {  
 return new CrashHandler(this);  
 }  
  
  
 }  
}

**注意：**异常日志存储在/sdcard/data/package\_name/cache/或则/data/data/package\_name/cache/目录下

# 三、使用方法

自定义AppContext继承Application在onCreate()方法中构建CrashHandler并catching()，如下所示：

public class AppContext extends Application {

@Override

public void onCreate() {

super.onCreate();

new CrashHandler.Builder(this)

.debug(false) // 是否调试模式

.tip("程序崩溃了") //崩溃时提示信息

.file("crash", "crash\_log", ".crash") //你懂的

.build()

.catching();//开始崩溃异常捕捉

}

}

然后AndroidManifest.xml中注册AppContext即可

<application

android:name=".AppContext"