xingda920813 / **HelloDaemon**

Watch 35 Star 359 Fork 87

Code Issues 1 Pull requests 0 Projects 0 Wiki Insights

Android 服务保活/常驻 (Android service daemon using JobScheduler)

android alarmmanager job-scheduler android-service-daemon whitelist

107 commits **1** branch **0** releases **2** contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

liuhuo.xd 1.2.2 ver: Bug 修复. Latest commit 038027b Aug 14, 2017

.idea	RxJava 2.	May 3, 2017
gradle/wrapper	Update gradle version.	Mar 3, 2017
hellodaemon	1.2.2 ver: Bug 修复.	Aug 14, 2017
sample	1.2.2 ver: Bug 修复.	Aug 14, 2017
.gitignore	init.	Aug 14, 2016
Kill.apk	Update Kill.apk	Aug 13, 2017
LICENSE	Initial commit	Aug 14, 2016
README.md	Update README.md	Mar 21, 2017
build.gradle	加强 Kill 和 Revoke 的有效性.	Jul 19, 2017
gradle.properties	File encoding.	Feb 26, 2017
gradlew	init.	Aug 14, 2016
gradlew.bat	init.	Aug 14, 2016
settings.gradle	抽取为 Library 项目, 更易于引入.	Mar 4, 2017

README.md

HelloDaemon

Android 服务保活/常驻 (Android Service Daemon)

建议只在**App**的核心功能需要保活/常驻时使用。

本示例中使用的保活方法部分来源于下面的博客和库。启动前台服务而不显示通知来自于**D-clock**的**AndroidDaemonService**，对其他的一些非**native**层保活方法进行了实现。

Android 进程常驻（2）----细数利用android系统机制的保活手段

D-clock / AndroidDaemonService

实现了上面 **2** 个链接中的大多数保活思路：

1. 将**Service**设置为前台服务而不显示通知

> D-clock：

思路一：API < 18，启动前台Service时直接传入new Notification()；

思路二：API >= 18，同时启动两个id相同的前台Service，然后再将后启动的Service做stop处理；

//启动前台服务而不显示通知的漏洞已在 API Level 25 修复，大快人心！

前台服务相对于后台服务的优势，除了优先级的提升以外，还有一点：

在最近任务列表中划掉卡片时，前台服务不会停止；

(更新：经过测试，发现只是对于AOSP/CM/国际上对Framework层改动较小的Android系统是成立的；EMUI/MIUI等未加入白名单的情况下，划掉卡片，前台服务也会停止；加入白名单后划掉卡片的行为与国际厂商的系统相似。)

而后台服务会停止，并在稍后重新启动 (onStartCommand 返回 START_STICKY 时) 。

前台服务和后台服务被划掉卡片时，回调的都是 onTaskRemoved 方法。

onDestroy 方法只在 设置 -> 开发者选项 -> 正在运行的服务 里停止服务时才会回调。

2. 在 **Service** 的 **onStartCommand** 方法里返回 **START_STICKY**

3. 覆盖 **Service** 的 **onDestroy**/**onTaskRemoved** 方法**,** 保存数据到磁盘**,** 然后重新拉起服务

4. 监听 **8** 种系统广播**:**

CONNECTIVITY_CHANGE, USER_PRESENT, ACTION_POWER_CONNECTED, ACTION_POWER_DISCONNECTED, BOOT_COMPLETED, PACKAGE_ADDED, PACKAGE_REMOVED.

在网络连接改变, 用户屏幕解锁, 电源连接 / 断开, 系统启动完成, 安装 / 卸载软件包时拉起 Service.

Service 内部做了判断，若 Service 已在运行，不会重复启动.

5. 开启守护服务**:** 定时检查服务是否在运行，如果不在运行就拉起来

6. 守护 **Service** 组件的启用状态**,** 使其不被 **MAT** 等工具禁用

详见上面的 2 个链接。

增加实现：

+ 守护服务**: Android 5.0** 及以上版本使用 **JobScheduler**，效果比 **AlarmManager** 好

使用 JobScheduler, Android 系统能自动拉起被 Force Stop 的 Package，而 AlarmManager 无法拉起.

Android 4.4 及以下版本使用 AlarmManager.

+ 使用定时 **Observable :** 避免 **Android** 定制系统 **JobScheduler** / **AlarmManager** 唤醒间隔不稳定的情况

+ 增加停止服务并取消定时唤醒的快捷方法

+ 增加在不需要服务运行时取消 **Job** / **Alarm** / **Subscription** 的快捷方法 **(**广播 **Action)**

+ 增强对国产机型的适配**:** 防止华为机型按返回键回到桌面再锁屏后几秒钟进程被杀

测试机型 : 华为 荣耀6 Plus (EMUI 4.0 Android 6.0), 应用未加入白名单.

观察到：

在未加入白名单的情况下，按Back键回到桌面再锁屏后几秒钟即会杀掉进程；

但是按Home键返回桌面的话，即使锁屏，也不会杀掉进程。

(更新：经过测试，在EMUI系统上，『即使锁屏，也不会杀掉进程』只对App的卡片还在多任务屏幕的第一屏时有效，一旦被挤到第二页及以后，锁屏后几秒钟即会杀掉进程；加入白名单后，回到桌面再锁屏后不会杀进程。)

因此，重写了onBackPressed方法，使其只是返回到桌面，而不是将当前Activity finish/destroy掉。

测试机型 : 红米1S 4G (MIUI 8 Android 4.4.2), 应用未加入白名单.

观察到：

在未加入白名单的情况下，回到桌面再锁屏后不会杀进程；

但划掉卡片，进程死亡并不再启动；加入白名单后，划掉卡片，服务不会停止，与CM的行为相似。

可以看出，若不想使用Native保活，引导用户加入白名单可能是比较可行的方法。

+ 用 **Intent** 跳转

- Android Doze 模式
- 华为 自启管理
- 华为 锁屏清理
- 小米 自启动管理
- 小米 神隐模式
- 三星 5.0/5.1 自启动应用程序管理
- 三星 6.0+ 未监视的应用程序管理
- 魅族 自启动管理
- 魅族 待机耗电管理
- Oppo 自启动管理
- Vivo 后台高耗电
- 金立 应用自启
- 金立 绿色后台
- 乐视 自启动管理
- 乐视 应用保护
- 酷派 自启动管理
- 联想 后台管理
- 联想 后台耗电优化
- 中兴 自启管理
- 中兴 锁屏加速受保护应用

配合 android.support.v7.AlertDialog 引导用户将 App 加入白名单.

+ 守护服务和**BroadcastReceiver**运行在**:watch**子进程中，与主进程分离

+ 工作服务运行在主进程中，免去与服务通信需使用**AIDL**或其他**IPC**方式的麻烦

参考了 Poweramp, 启动的前台服务与 UI 运行在同一进程中。

+ 做了防止重复启动**Service**的处理，可以任意调用**startService(Intent i)**

若服务还在运行，就什么也不做；若服务不在运行就拉起来。

+ 在子线程中运行定时任务，处理了运行前检查和销毁时保存的问题

开始任务前，先检查磁盘中是否有上次销毁时保存的数据；定期将数据保存到磁盘。

引入

1. 添加二进制

build.gradle 中添加

```
compile 'com.xdandroid:hellodaemon:+'
```

2. 继承 AbsWorkService, 实现 6 个抽象方法

```
/**
 * 是否 任务完成, 不再需要服务运行?
 * @return 应当停止服务, true; 应当启动服务, false; 无法判断, null.
 */
Boolean shouldStopService();

/**
 * 任务是否正在运行?
 * @return 任务正在运行, true; 任务当前不在运行, false; 无法判断, null.
```

```
     */
    Boolean isWorkRunning();

    void startWork();

    void stopWork();

    //Service.onBind(Intent intent)
    @Nullable IBinder onBind(Intent intent, Void unused);

    //服务被杀时调用, 可以在这里面保存数据.
    void onServiceKilled();
```

别忘了在 Manifest 中注册这个 Service.

3. 自定义 **Application**

在 Application 的 `onCreate()` 中, 调用

```
DaemonEnv.initialize(
  Context app,  //Application Context.
  Class<? extends AbsWorkService> serviceClass, //刚才创建的 Service 对应的 Class 对象.
  @Nullable Integer wakeUpInterval);  //定时唤醒的时间间隔(ms), 默认 6 分钟.

Context.startService(new Intent(Context app, Class<? extends AbsWorkService> serviceClass));
```

别忘了在 Manifest 中通过 android:name 使用这个自定义的 Application.

4. API 说明

启动 **Service:**

```
Context.startService(new Intent(Context c, Class<? extends AbsWorkService> serviceClass))
```

停止 **Service:**

在 ? extends AbsWorkService 中, 添加 `stopService()` 方法:

1.操作自己维护的 flag, 使 `shouldStopService()` 返回 `true`；

2.调用自己的方法或第三方 SDK 提供的 API, 停止任务;

3.调用 `AbsWorkService.cancelJobAlarmSub()` 取消 Job / Alarm / Subscription.

需要停止服务时, 调用 ? extends AbsWorkService 上的 `stopService()` 即可.

处理白名单:

以下 API 全部位于 IntentWrapper 中:

```
List<IntentWrapper> getIntentWrapperList();

//弹出 android.support.v7.AlertDialog, 引导用户将 App 加入白名单.
void whiteListMatters(Activity a, String reason);

//防止华为机型未加入白名单时按返回键回到桌面再锁屏后几秒钟进程被杀.
//重写 MainActivity.onBackPressed(), 只保留对以下 API 的调用.
void onBackPressed(Activity a);
```

为节省用户的电量, 当不再需要服务运行时, 可以调用 **`AbsWorkService.cancelJobAlarmSub()`** 取消定时唤醒的 **Job / Alarm / Subscription**, 并调用 **`stopService()`** 停止服务.

详见代码及注释。