

Android进程间的通信 - 耍流氓的方式保活Service



作者

红橙Darren (/u/35083fcb7747)

已关注

2017.03.30 11:27* 字数 2906 阅读 860 评论 13 喜欢 38

(/u/35083fcb7747)

1. 概述

上一期我们已经阅读了源码Android进程间的通信 - IPC(机制)Binder的原理和源码阅读 (<http://www.jianshu.com/p/810ecb80a96f>)，这一期我们就得用到它了。记得前几年在公司做购物商城，有一个倒计时的功能，上头非得要实现这个功能，当时的想法就是引咎辞职，因为APP应用一退出后台启动的Service根本不管用直接被杀掉了，前几年的事今天我们就来拯救拯救。

声明：

- 1.文章综合了很多其他的一些资料，所以非常感谢某些哥们的分享精神；
- 2.不能够保证所有的机型都适用，因为我们要面对的是各大Rom厂商；
- 3.不想带坏大家，我们应该把更多的时间放在内存管理和性能优化上面。



效果演示.gif

所有分享大纲：2017Android进阶之路与你同行
(<http://www.jianshu.com/p/c0ec2a7fc26a>)

视频讲解地址： <http://pan.baidu.com/s/1pLrvFj9> (<http://pan.baidu.com/s/1pLrvFj9>)

2. 论Service为什么会被杀死

Service被杀一般通常是三个方面，第一种现象是当手机内存不足，你的应用或者Service在后台这个时候你精力或者营养更不上可能就被杀了；第二种现象是用了第三方的一些管理软件比如流氓360，腾讯管家去清理进程他们也会对你下手；第三种现象就是各大Rom厂商，在你退出应用的时候会做一些清理工作。

2.1 进程的优先级：

(<https://developer.android.com/guide/components/processes-and-threads.html?hl=zh-cn>)

google的官网给进程划分了等级，并且明确告诉你哥们我不会乱来，回收进程的时候有自己套路。担心有些哥们不想翻墙这里我就Copy一下。

Android 系统将尽量长时间地保持应用进程，但为了新建进程或运行更重要的进程，最终需要移除旧进程来回收内存。为了确定保留或终止哪些进程，系统会根据进程中正在运行的组件以及这些组件的状态，将每个进程放入“重要性层次结构”中。必要时，系统会首先消除重要性最低的进程，然后是重要性略逊的进程，依此类推，以回收系统资源。

重要性层次结构一共有 5 级。以下列表按照重要程度列出了各类进程（第一个进程最重要，将是最后一个被终止的进程）：

1.前台进程

用户当前操作所必需的进程。如果一个进程满足以下任一条件，即视为前台进程：

- 托管用户正在交互的Activity已调用Activity的onResume()方法
- 托管某个Service，后者绑定到用户正在交互的 Activity
- 托管正在“前台”运行的Service服务已调用startForeground()
- 托管正执行一个生命周期回调的Service (onCreate()、onStart()或onDestroy())
- 托管正执行其onReceive()方法的BroadcastReceiver

通常，在任意给定时间前台进程都为数不多。只有在内存不足以支持它们同时继续运行这一万不得已的情况下，系统才会终止它们。此时，设备往往已达到内存分页状态，因此需要终止一些前台进程来确保用户界面正常响应。

2.可见进程

没有任何前台组件、但仍会影响用户在屏幕上所见内容的进程。如果一个进程满足以下任一条件，即视为可见进程：

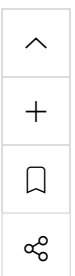
- 托管不在前台、但仍对用户可见的Activity已调用其onPause()方法。例如，如果前台 Activity 启动了一个对话框，允许在其后显示上一 Activity，则有可能发生这种情况。
- 托管绑定到可见（或前台）Activity 的Service。

可见进程被视为是极其重要的进程，除非为了维持所有前台进程同时运行而必须终止，否则系统不会终止这些进程。

3.服务进程

正在运行已使用startService()方法启动的服务且不属于上述两个更高类别进程的进程。尽管服务进程与用户所见内容没有直接关联，但是它们通常在执行一些用户关心的操作（例如，在后台播放音乐或从网络下载数据）。因此，除非内存不足以维持所有前台进程和可见进程同时运行，否则系统会让服务进程保持运行状态。

4.后台进程



包含目前对用户不可见的 Activity 的进程已调用 Activity 的onStop()方法。这些进程对用户体验没有直接影响，系统可能随时终止它们，以回收内存供前台进程、可见进程或服务进程使用。通常会有很多后台进程在运行，因此它们会保存在 LRU（最近最少使用）列表中，以确保包含用户最近查看的 Activity 的进程最后一个被终止。如果某个 Activity 正确实现了生命周期方法，并保存了其当前状态，则终止其进程不会对用户体验产生明显影响，因为当用户导航回该 Activity 时，Activity 会恢复其所有可见状态。有关保存和恢复状态的信息，请参阅Activity
(<https://developer.android.com/guide/components/activities.html?hl=zh-cn#SavingActivityState>)文档。

5.空进程

不含任何活动应用组件的进程。保留这种进程的的唯一目的是用作缓存，以缩短下次在其中运行组件所需的启动时间。为使总体系统资源在进程缓存和底层内核缓存之间保持平衡，系统往往会终止这些进程。

根据进程中当前活动组件的重要程度，Android 会将进程评定为它可能达到的最高级别。例如，如果某进程托管着服务和可见 Activity，则会将此进程评定为可见进程，而不是服务进程。

此外，一个进程的级别可能会因其他进程对它的依赖而有所提高，即服务于另一进程的进程其级别永远不会低于其所服务的进程。例如，如果进程 A 中的内容提供程序为进程 B 中的客户端提供服务，或者如果进程 A 中的服务绑定到进程 B 中的组件，则进程 A 始终被视为至少与进程 B 同样重要。

由于运行服务的进程其级别高于托管后台 Activity 的进程，因此启动长时间运行操作的 Activity 最好为该操作启动服务

(<https://developer.android.com/guide/components/services.html?hl=zh-cn>)，而不是简单地创建工作线程，当操作有可能比 Activity 更加持久时尤要如此。例如，正在将图片上传到网站的 Activity 应该启动服务来执行上传，这样一来，即使用户退出 Activity，仍可在后台继续执行上传操作。使用服务可以保证，无论 Activity 发生什么情况，该操作至少具备“服务进程”优先级。同理，广播接收器也应使用服务，而不是简单地将耗时冗长的操作放入线程中。

以上是google官网的进程的优先级划分，就是说越是前面越不容易被杀，越是后面越不容易被杀，那么5是最容易被杀的，下面我们来看一看杀进程回收内存的机制Low Memory Killer 到底是怎么对我们下手的：

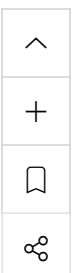
2.2 LowmemoryKiller的工作机制：

LowmemoryKiller会在内存不足的时候扫描所有的用户进程，找到不是太重要的进程杀死，至于LowmemoryKiller杀进程够不够狠，要看当前的内存使用情况，内存越少，下手越狠。在内核中，lowmemorykiller.c定义了几种内存回收等级如下：

```
static short lowmem_adj[6] = {
    0,
    1,
    6,
    12,
};
static int lowmem_adj_size = 4;

static int lowmem_minfree[6] = {
    3 * 512,
    2 * 1024,
    4 * 1024,
    16 * 1024,
};
static int lowmem_minfree_size = 4;
```

lowmem_adj中各项数值代表阈值的警戒级数，lowmem_minfree代表对应级数的剩余内存，两者一一对应，比如当系统的可用内存小于6MB时，警戒级数为0；当系统可用内存小于8M而大于6M时，警戒级数为1；当可用内存小于64M大于16MB时，警戒级数为12。LowmemoryKiller就是根据当前系统的可用内存多少来获取当前的警戒级数，如



果进程的oom_adj大于警戒级数并且占内存最大，将会被优先杀死。omm_adj越小，代表进程越重要。一些前台的进程，oom_adj会比较小，而后台的服务，omm_adj会比较大，所以当内存不足的时候，Lowmemorykiller先杀掉的是后台服务而不是前台的进程。对于LowmemoryKiller的杀死，这里有一句话很重要，就是：**具有相同omm_adj的进程，则杀死占用内存较多的**，因此，如果我们的APP进入后台，就尽量释放不必要的资源，以降低自己被杀的风险。简单看一下实现源码：

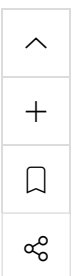
```
static int lowmem_shrink(int nr_to_scan, gfp_t gfp_mask)
{
    ...
    <!--获取free内存状况-->
    int other_free = global_page_state(NR_FREE_PAGES);
    int other_file = global_page_state(NR_FILE_PAGES);
    <!--找到min_adj -->
    for(i = 0; i < array_size; i++) {
        if (other_free < lowmem_minfree[i] &&
            other_file < lowmem_minfree[i]) {
            min_adj = lowmem_adj[i];
            break;
        }
    }
    <!--找到p->oomkilladj>min_adj并且oomkilladj最大，内存最大的进程-->
    for_each_process(p) {
        // 找到第一个大于等于min_adj的，也就是优先级比阈值低的
        if (p->oomkilladj < min_adj || !p->mm)
            continue;
        // 找到tasksize这个是什么呢
        tasksize = get_mm_rss(p->mm);
        if (tasksize <= 0)
            continue;
        if (selected) {
            // 找到优先级最低，并且内存占用大的
            if (p->oomkilladj < selected->oomkilladj)
                continue;
            if (p->oomkilladj == selected->oomkilladj &&
                tasksize <= selected_tasksize)
                continue;
        }
        selected = p;
        selected_tasksize = tasksize;
        lowmem_print(2, "select %d (%s), adj %d, size %d, to kill\n",
                     p->pid, p->comm, p->oomkilladj, tasksize);
    }
    if(selected != NULL) {...
        force_sig(SIGKILL, selected);
    }
    return rem;
}
```

那么现在我们可以针对性的做一些工作了比如：

- 1.提高进程的优先级，其实就是减小进程的p->oomkilladj（越小越重要），如启动Service调用startForeground()尽量提高进程的优先级；
- 2.当应用推到后台适当释放资源然后降低APP的内存占用量，因为在oom_adj相同的时候，会优先干掉内存消耗大的进程；
- 3.对于要一直在后台运行的Service我们一定要轻，不能太风骚。

3. 针对各大Rom厂商和清理软件：

我其实要面对的不是google原生的系统，在各大手机厂商以及各种流氓清理软件面前我们按套路出牌往往然并卵，我们要面对的也是他们，这也是说往往写的代码你会发现这里行那里不行，真是心中有一万只动物。会发现采用上面的方式根本没用，网上也有很多的解决方案，如QQ的1像素的Activity，某些第三方应用救救你，NDK方式hook子线程等等等等。我们这里决定采用双进程守护相互唤醒以及5.0以上的JobSheduler：



```

public class MessageService extends Service {

    private static final String TAG = "MessageService";

    private MessageServiceConnection mServiceConnection;
    private MessageBind mMessageBind;

    @Override
    public void onCreate() {
        super.onCreate();
        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {
                    Log.e(TAG, "等待接收消息");
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();

        if (mServiceConnection == null) {
            mServiceConnection = new MessageServiceConnection();
        }

        if (mMessageBind == null) {
            mMessageBind = new MessageBind();
        }
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        MessageService.this.bindService(new Intent(MessageService.this, GuardService.class),
            mServiceConnection, Context.BIND_IMPORTANT);
        return Service.START_STICKY;
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return mMessageBind;
    }

    private class MessageBind extends ProcessConnection.Stub {

        @Override
        public void processConnected() throws RemoteException {

        }

    }

    private class MessageServiceConnection implements ServiceConnection {

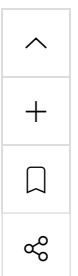
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            // 建立连接
            Toast.makeText(MessageService.this, "建立连接", Toast.LENGTH_LONG).show();
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            // 断开连接
            Toast.makeText(MessageService.this, "断开连接", Toast.LENGTH_LONG).show();
            Intent guardIntent = new Intent(MessageService.this, GuardService.class);
            // 发现断开我就从新启动和绑定
            startService(guardIntent);
            MessageService.this.bindService(guardIntent,
                mServiceConnection, Context.BIND_IMPORTANT);
        }

    }
}

```

双进程守护和唤醒在5.0以下还是很爽的，至少能保证各种流氓的清理软件不能够清理掉我们的Service，但是在5.0以后就不管用了，尤其在小米、魅族这些手机上面。不过柳暗花明又一村，我们可以利用JobScheduler。



```

/**
 * Created by Darren on 2017/3/25.
 * Email: 240336124@qq.com
 * Description:
 */
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class JobAwakenService extends JobService{

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        JobInfo.Builder builder = new JobInfo.Builder(1,new ComponentName(this,JobAwakenServ
        builder.setPeriodic(500);
        JobInfo jobInfo = builder.build();
        JobScheduler jobScheduler = (JobScheduler) getSystemService(Context.JOB_SCHEDULER_S
        jobScheduler.schedule(jobInfo);
        return START_STICKY;
    }

    @Override
    public boolean onStartJob(JobParameters params) {
        Log.e("TAG","onStartJob");

        // boolean isGuardAlive = isServiceWork(this,GuardService.class.getName());
        boolean isMessageAlive = isServiceWork(this,MessageService.class.getName());

        if(!isMessageAlive){
            // startService(new Intent(this,GuardService.class));
            startService(new Intent(this,MessageService.class));
        }

        return false;
    }

    @Override
    public boolean onStopJob(JobParameters params) {
        return false;
    }


    /**
     * 判断某个服务是否正在运行的方法
     *
     * @param mContext
     * @param serviceName
     *      是包名+服务的类名（例如： net.loonggg.testbackstage.TestService）
     * @return true代表正在运行，false代表服务没有正在运行
     */
    public boolean isServiceWork(Context mContext, String serviceName) {
        boolean isWork = false;
        ActivityManager myAM = (ActivityManager) mContext
            .getSystemService(Context.ACTIVITY_SERVICE);
        List<ActivityManager.RunningServiceInfo> myList = myAM.getRunningServices(100);
        if (myList.size() <= 0) {
            return false;
        }
        for (int i = 0; i < myList.size(); i++) {
            String mName = myList.get(i).service.getClassName().toString();
            if (mName.equals(serviceName)) {
                isWork = true;
                break;
            }
        }
        return isWork;
    }
}

```

所有分享大纲：2017Android进阶之路与你同行

(<http://www.jianshu.com/p/c0ec2a7fc26a>)

视频讲解地址：<http://pan.baidu.com/s/1pLrvFj9> (<http://pan.baidu.com/s/1pLrvFj9>)

 Android进阶之旅 (/nb/8948928)

举报文章 © 著作权归作者所有



红橙Darren (/u/35083fcb7747)

写了 50796 字，被 755 人关注，获得了 908 个喜欢
(/u/35083fcb7747)

✓ 已关注

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

赞赏支持



(/u/5ba565c5ecb6)

喜欢 38



更多分享

(http://cwb.assets.jianshu.io/notes/images/1076377



写下你的评论...

13条评论

只看作者

按喜欢排序 按时间正序 按时间倒序



wenld_ (/u/99f514ea81b3)

2楼 · 2017.03.30 16:54

(/u/99f514ea81b3)

赞赞赞

1人赞 回复

红橙Darren (/u/35083fcb7747) : @wenld_ (/users/99f514ea81b3) 😄😄😄

2017.03.30 17:21 回复

添加新评论



notsuitable (/u/cec32b82822f)

7楼 · 2017.04.10 06:25

(/u/cec32b82822f)

与两个app同时具有杀不死进程的功能代码，可是第二个应用就不会起作用，而且可以被杀死，前提是，这两个app同时安在一个手机上

1人赞 回复

notsuitable (/u/cec32b82822f) : 请问楼主您的到验证了吗，比如客户端和配送端同时存在在一个手机上，而且都具有杀不死进程功能的代码

2017.04.10 06:26 回复

红橙Darren (/u/35083fcb7747) : @notsuitable (/users/cec32b82822f) 不明白你的意思？写两个app干嘛？

2017.04.10 06:39 回复

红橙Darren (/u/35083fcb7747) : @notsuitable (/users/cec32b82822f) 没验证哪敢写文章，试了几款主流的机型没问题，是双进程，不是两个app，当然也不能保证所有都行，手上没那么多机型

2017.04.10 06:45 回复

添加新评论



阳光小熊 (/u/9657d11e6c3d)

3楼 · 2017.04.01 15:06

(/u/9657d11e6c3d)

很棒

赞 回复



辣公公 (/u/b3511d489d01)

4楼 · 2017.04.04 21:54

(/u/b3511d489d01)


真的有用吗？加入系统白名单才能真正解决问题

👍 赞 💬 回复

红橙Darren (/u/35083fcb7747) : @辣公公 (/users/b3511d489d01) 有没有用，得试过才知道，不能保证所有都有用，我试过几款主流的手机没问题。

2017.04.05 08:58 💬 回复

✎ 添加新评论

 小驴拉着小萝卜 (/u/920c748d0d2c)
5楼 · 2017.04.07 15:18
(/u/920c748d0d2c)


有没有demo，这个JobService我怎么启动不起来？

👍 赞 💬 回复

红橙Darren (/u/35083fcb7747) : @小驴拉着小萝卜 (/users/920c748d0d2c) 今晚8:30就会开始写了，很简单的

2017.04.07 16:22 💬 回复

✎ 添加新评论

 android0226 (/u/195e2f8264d6)
6楼 · 2017.04.08 07:59
(/u/195e2f8264d6)

processconnection.stub这个api在哪个包下，找不到。

👍 赞 💬 回复

红橙Darren (/u/35083fcb7747) : @android0226 (/users/195e2f8264d6) 要rebuild重新编译才行，要不然找不到

2017.04.08 08:06 💬 回复

✎ 添加新评论

被以下专题收入，发现更多相似内容

+

我的专题

Android知识

Android开发

Android..

service

Android开发

Android..

安卓知识

Android私房菜

^

+

🔖

🔗