邵俊

shaojun@hangsheng.com.cn

平台研发部

基础架构开发一组

Android日志记录模块

persistlog-xlog

修改履历

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 换版/修订记录 | 制/修订 | 审核 | 批准 | 生效日期 |
| V0.1 | 初版 | 制作 |  |  |  |
| V0.2 | 适配CCS5.0项目 | 戴盛 |  |  | 2022-05-11 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目 录

[1 目的 4](#_Toc45113528)

[2 适用范围 4](#_Toc45113529)

[3 术语与定义 4](#_Toc45113530)

[4 职责与权限 4](#_Toc45113531)

[5 正文内容 4](#_Toc45113532)

[5.1 Android日志系统结构 4](#_Toc45113533)

[5.1.1 日志系统框图 4](#_Toc45113534)

[5.1.2 logd启动及配置 6](#_Toc45113535)

[5.1.3 logd作用 6](#_Toc45113536)

[5.1.4 内核日志获取 6](#_Toc45113537)

[5.2 logcat模块 7](#_Toc45113538)

[5.2.1 程序依赖关系框图 7](#_Toc45113539)

[5.2.2 使用方法 7](#_Toc45113540)

[5.3 persistlog-xlog 8](#_Toc45113541)

[5.3.1 日志系统需求 8](#_Toc45113542)

[5.3.2 persistlog-xlog简介及开发背景 8](#_Toc45113543)

[5.3.3 persistlog-xlog自运行日志管理 8](#_Toc45113544)

[5.3.4 日志文件夹记录规则说明 8](#_Toc45113545)

[5.3.5 日志文件记录规则说明 9](#_Toc45113546)

[5.4 persistlog-xlog编译文件目录树 10](#_Toc45113547)

[5.5 编译打包 11](#_Toc45113548)

[5.6 开机启动及配置 12](#_Toc45113549)

[5.6.1 SELinux配置 12](#_Toc45113550)

[5.6.2 开机启动配置 12](#_Toc45113551)

[5.7 部署 13](#_Toc45113552)

[5.8 运行结果验证 13](#_Toc45113553)

[5.9 内核日志同步 14](#_Toc45113554)

[6 相关记录 14](#_Toc45113555)

[7 附录 15](#_Toc45113556)

# 目的

本文件主要包含Android日志持久化程序的结构、移植以及使用说明。

# 适用范围

Android产品开发人员，Android平台研发以及负责persistlog维护人员。

# 术语与定义

|  |  |
| --- | --- |
| 术语 | 说明 |
| persistlog-xlog | Android日志内容记录程序，负责将日志信息保存到文件中 |
|  |  |
|  |  |
|  |  |

# 职责与权限

TBD

# 正文内容

## Android日志系统结构

### 日志系统框图

下图是Android日志系统架构图。



图 5‑1 Android log系统架构图

模块说明：

|  |  |
| --- | --- |
| 名称 | 用途 |
| *framework* | 供Android应用打印日志使用 |
| *Application* | Android应用 |
| */system/lib64/liblog.so*  */system/lib/liblog.so* | 库文件，提供了logd的访问接口 |
| */system/bin/logd* | 日志系统后台程序，用于汇总、保存部分日志。 |
| */system/bin/logcatd* | 系统默认的日志持久化执行程序（默认在eng和userdebug打开） |
| */system/bin/logcat* | 日志查看程序，用于显示系统日志信息 |
| */system/bin/persistlog-xlog* | 航盛日志记录程序。 |
| */dev/socket/logd* | 控制命令通道 |
| */dev/socket/logdr* | 日志读取通道 |
| */dev/socket/logdw* | 日志写入通道 |

表 5‑1 模块说明

在图 5‑1中日志系统通过【*/system/lib64/liblog.so*】或者【*/system/lib/liblog.so*】为整个Android系统提供日志访问接口。【*/system/bin/logd*】用于记录整个系统的日志，包括内核日志。

### logd启动及配置

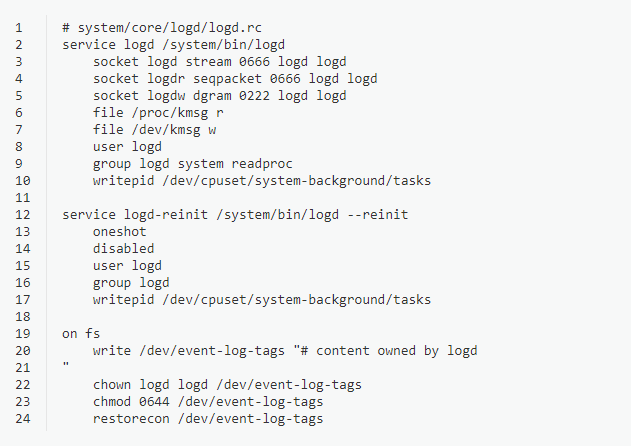


图 5‑2 logd启动参数

由上图我们可以看到，当【logd】启动时【init】会创建3个Unix域的socket，分别是【*/dev/socket/logd*】、【*/dev/socket/logdr*】、【*/dev/socket/logdw*】。这三个 socket 的功能如下：

* socket logd 用于外接受控制命令。
* 客户端通过 logdr 读取 log 数据。使用 seqpacket，可以让用户在可靠地读取数据的同时，一次读取一条 log。
* 客户端通过 logdw 写入 log 数据。由于类型是 dgram，在非常繁忙的时候，log 可能会丢失。但是，这可以避免客户端阻塞在写 log 的调用上。

### logd作用

在Android系统中日志信息来源于各个进程以及内核，有些日志在时间上存在先后关联性，这就要求日志系统在日志输出时需要保持时间上的先后关系。

【logd】内部使用【ring buffer】来保存日志。当【buffer】存满时会从头开始覆盖【buffer】中时间最早的日志，然后以这种方式循序。

为了防止应用程序写入重复日志，【logd】在写入日志时会对相同的日志进行过滤。可以参考这篇博客：

<https://jekton.github.io/2018/05/17/logd-writing-part2/>

### 内核日志获取

当【logd】启动时会通过【/dev/kmsg】获取内核在logd启动前的日志。此外还会通过系统属性值来决定是否打开【/proc/kmsg】获取在logd启动后内核变化的日志。

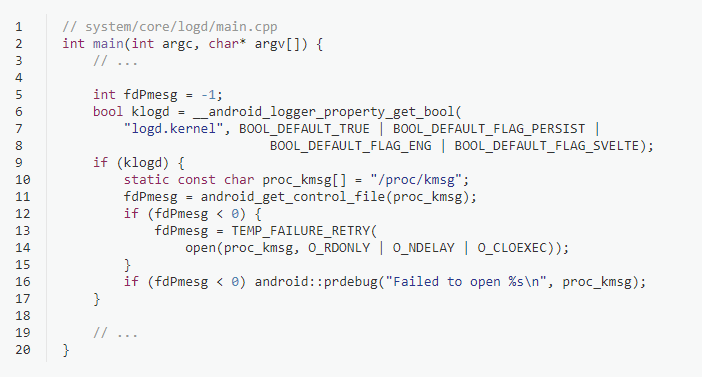


图 5‑3 内核日志控制代码

默认情况下，会读取内核 log。这里我们就直接假设 klogd 为 true。至于代码更加深层次的理解请参考如下这个网站：

<https://jekton.github.io/2018/05/11/logd-overview/>

## logcat模块

### 程序依赖关系框图



图 5‑4 logcat依赖关系

【logcat】：Android中打印系统日志的组件。

【logcatd】：Android中将系统日志保存成文件的组件。

【liblogcat.so】：提供访问logd的接口，并且提供了一个参数可以配置日志格式等。

### 使用方法

logcat提供了非常丰富的功能详情可以参考谷歌官方网站：

<https://developer.android.com/studio/command-line/logcat?hl=zh-cn>

## persistlog-xlog

### 日志系统需求

日志系统有如下需求：

* 记录内核日志
* 系统启动时以当前时间点创建带时间的软件夹。
* 合并不同日志ID，同时写入同一个文件。
* 可以单个日志文件大小
* 可以设置总日志文件大小
* 当日期被设置时，以当前日期重命名文件夹。
* 日志文件按照从小到大的顺序排列，数字越小代表日志文件在时间轴上越靠前，也就是时间越早。
* 日志文件夹同理。
* 日志需要支持压缩，以减少空间占用

### persistlog-xlog简介及开发背景

【persistlog-xlog】运行程序是扬州航盛平台研发部自行研发的应用程序，用于记录系统运行日志。

刚开始我们预想通过改进【logcatd】程序使之满足我们内部的需求，然而当我们改动【logcatd】时发现需要改动【liblogcat.so】，但是【logcat】也是使用【liblogcat.so】的，牵一发动全身。所以我们决定参考【liblogcat.so】的访问【liblog.so】的形式编写了【persistlog-xlog】

### persistlog-xlog自运行日志管理

【persistlog-xlog】正如它名称一样是用于保存系统日志的。那么【persistlog-xlog】自身的日志该如何输出呢？我们是否可以将日志输出到【logd】中呢？答案是否定的，因为在【persistlog-xlog】出现问题时将不能将日志持久化的保存，那么自身的日志信息也就不会被保存下来。为了解决这个问题【persistlog-xlog】内部集成了一个日志保存模块，用于记录程序本身运行时的信息。

【persistlog-xlog】日志文件名称为【persistlog-xlog.pl】。这个日志模块会控制日志文件大小。默认一共保存五个文件，当个日志文件大小为【1MB】，当日志文件大于设置值时日志文件会做翻转，命名为【persistlog-xlog-1.pl】依次往后。

### 日志文件夹记录规则说明

默认日志文件保存在【/hsaelog/logd】中，保存日志的文件夹以如下规则命名：

{序号}@{年-月-日}

如：1@2019-03-02、2@2019-03-02

当日志记录模块启动时，会扫描指定的文件夹的子文件夹（通过运行参数设置），匹配并记录相应的文件夹。根据文件夹的名称识别文件夹的记录时间，并将文件夹名称中序号按照自然数的顺序重新命名，以防止序号一直增加。如图 5‑5所示。

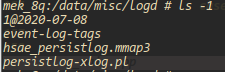


图 5‑5日志文件夹

日志文件夹序号最大的代表本次日志记录程序启动时将日志文件保存的文件夹。如图 5‑6。文件夹序号是日志文件保存先后的唯一凭证，这是由于系统时间在上电时并不一定是正确的，因此文件夹名称上的时间戳有可能是乱序的。



图 5‑6文件夹时间轴

时间轴上我们可以看到有两个文件夹时间戳是一样的，这是由于日志记录模块在每次启动时比对当前日期与目录下的最后一次文件夹时间（序号最大），两者相同则不会新建文件夹，如果两者不同则新建文件夹。车机在重启时时间是通过MCU传递过来的，如果在【persistlog-xlog】服务启动时时间还未设置，比对最后文件夹时间与当前不符，就会新建文件夹。文件夹上的时间戳在系统时间被设置时会重新命名，这样可以保证当时间更新时文件夹的日期可以得到同步。当前是通过时间设置时的日志信息来判断的。

### 日志文件记录规则说明

日志内容包含：【main】、【events】、【radio】、【kernel】、【security】五大类，这些日志都会都会同时写在同一个文件中，这样有利于提高数据压缩比。

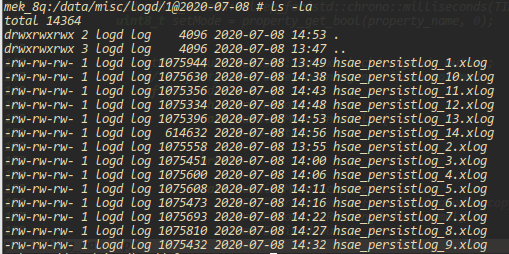


图 5‑7日志文件

日志文件按照上述规则保存日志信息。日志文件中的序号按照从小到大的序号以此排列，序号最大的文件日志在时间轴上越靠后。如图 5‑8。



图 5‑8日志文件在时间轴上排布

## persistlog-xlog编译文件目录树

【persistlog-xlog】编译需要如下文件，在平台项目中【persistlog-xlog】被放置在【packages/services/persistlog-xlog】路径下：

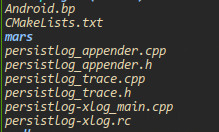


图 5‑9编译文件

【persistlog-xlog】需要

|  |
| --- |
| Android.bp 编译脚本文件  CMakeLists.txt 可忽略  mars 腾讯mars模块使用了xlog组件  persistlog\_appender.cpp  persistlog\_appender.h  persistlog\_trace.cpp  persistlog\_trace.h  persistlog-xlog\_main.cpp  persistlog-xlog.rc 启动脚本  spdlog 日志模块，用于保存persistlog-xlog自身日志 |

## 编译打包

Porting【persistlog-xlog】代码后需要修改打包相关的配置，在QXP AOSP中修改【device/qcom/common/base.mk】文件的末尾，添加如下信息：

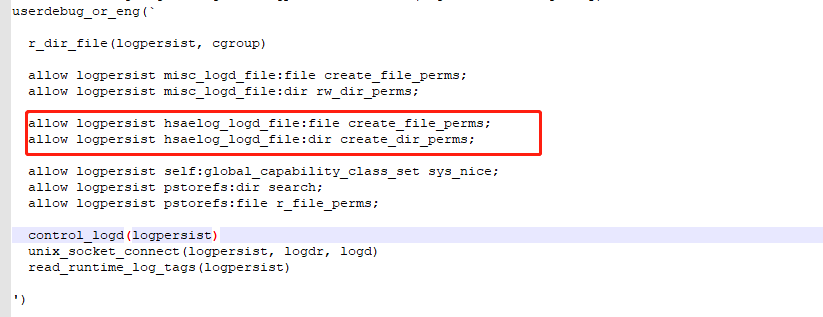
|  |
| --- |
| PRODUCT\_PACKAGES += \  persistlog-xlog |

## 开机启动及配置

### SELinux配置

在【system/sepolicy/private/**file\_contexts**】与【system/sepolicy/prebuilts/api/30.0/private/file\_contexts】文件中，添加如下信息：

|  |
| --- |
| /system/bin/persistlog-xlog -- u:object\_r:logcat\_exec:s0 |

在【system/sepolicy/private/logpersist.te】与【system/sepolicy/prebuilts/api/30.0/private/logpersist.te】文件中修改如下配置：

### 开机启动配置

【persistlog-xlog】通过【persistlog-xlog.rc】来描述启动配置。

|  |
| --- |
| *# hsae persistlog service rc*  *# author : AnaBit*  *# email : shaojun@hangsheng.com.cn*  *# date : 2020-07-02*  *# enable, prep and start persistlog-xlog*  *on load\_persist\_props\_action*  *setprop hsae.persist.log.xlog.enable true*  *# enable persistlog-xlog service*  *on property:hsae.persist.log.xlog.enable=true*  *mkdir /hsaelog/logd 0760 nhlog nhlog*  *exec - nhlog nhlog -- /system/bin/chmod -R 760 /hsaelog/logd*  *start persistlog-xlog*  *# stop persistlog-xlog service*  *on property:hsae.persist.log.xlog.enable=false*  *stop persistlog-xlog*  *# persistlog-xlog service*  *service persistlog-xlog /system/bin/persistlog-xlog -s 200 -r 1*  *class late\_start*  *disabled*  *# logd for write to /data/misc/logd, log group for read from log daemon*  *user nhlog*  *group log*  *writepid /dev/cpuset/system-background/tasks*  *oom\_score\_adjust -600* |

表格 1 persistlog-xlog配置文件

其中：

|  |
| --- |
| *service persistlog-xlog /system/bin/persistlog-xlog -s 200 -r 1* |

|  |  |
| --- | --- |
| 参数 | 描述 |
| -s | 限制日志文件总大小，单位是：Mbyte。 |
| -r | 单个日志文件大小，单位是：Mbyte。 |

## 部署

AOSP源码编译后烧录或者通过OTA升级的方式将【persistlog-xlog】组件部署到目标板上测试。

如果【persistlog-xlog】功能正确导入，那么在烧录完镜像后在系统文件夹中将存在如下文件：

|  |  |
| --- | --- |
| 文件 | 说明 |
| system/bin/persistlog-xlog | 执行程序 |
| /system/etc/init/persistlog-xlog.rc | 开机启动脚本 |

## 运行结果验证

首先输入如下指令：

|  |
| --- |
| *pidof persistlog-xlog* |

如果【persistlog-xlog】成功运行将出现如下信息：

|  |
| --- |
| *1|mek\_8q:/ # pidof persistlog-xlog*  *3350* |

其中【3350】表示【persistlog-xlog】进程ID，间接代表了【persistlog-xlog】已经正常运行了。

然后进入【/hsaelog/logd/】查看是否有{序号}@{年-月-日}的文件夹，如果存在则代表已经porting成功。

## 内核日志同步

【Android O】默认代码中存在内核日志不同步的问题，需要在源码中改动部分代码。左边为原始代码，右边为修改后的代码。

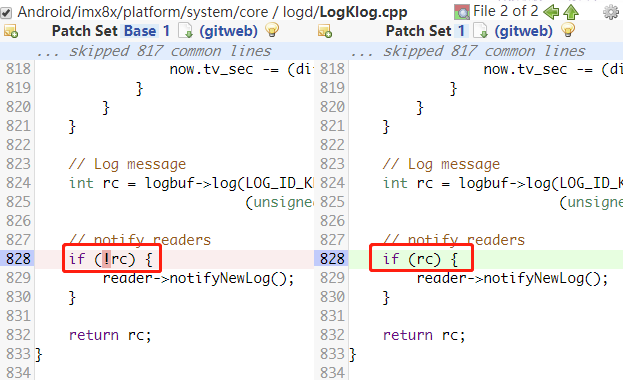


图 5‑10 内核同步补丁

在图 5‑10有标注相关改动位置。

此问题在【Android P】的版本中得到了修正。

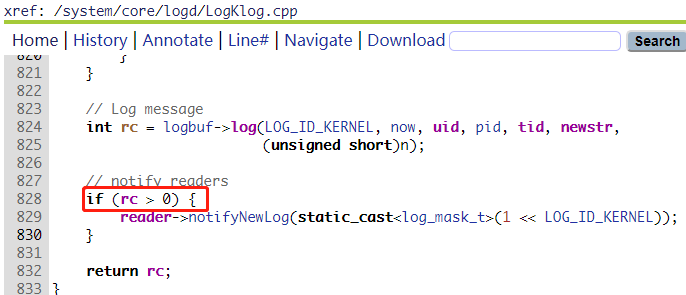


图 5‑11 Android P内核日志同步

# ~~从persistlog导入persistlog-xlog~~

~~如果之前已经成功导入了persistlog模块从可以参考本节内容。~~

## ~~selinux权限配置~~

~~原有的关于【persistlog】的权限内容不需要变动，只需要在【system/sepolicy/private/~~**~~file\_contexts~~**~~】与【system/sepolicy/prebuilts/api/28.0/private/file\_contexts】文件中，添加如下信息：~~

|  |
| --- |
| ~~/system/bin/persistlog-xlog -- u:object\_r:logcat\_exec:s0~~ |

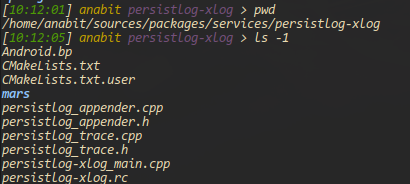
~~上述内容用于给【persistlog-xlog】进程增加selinux识别标签。【persistlog-xlog】权限与【logcat】、【persistlog】一致；这样可以继承这些进程原有权限。~~

## ~~运行参数修改~~

~~【persistlog-xlog】通过【persistlog-xlog.rc】文件实现开机自启动以及启动配置，详情参考【5.6.2】章节。默认配置为总共【200MB】日志空间，单个日志文件【1MB】。~~

## ~~集成编译~~

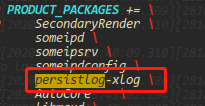
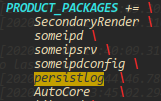
~~【persistlog-xlog】以独立的仓库提供，内容见5.4；项目组集成可以将此仓库放置在【packages/services】目录中，可以参考我的编译服务器位置如下：~~

~~~~

~~图 6‑1仓库位置~~

## ~~将img打包设置~~

~~【persistlog】与【persistlog-xlog】在功能是重复的只需要保留一个即可。由于之前的项目中在打包时已经将【persistlog】打包进system.img，所以需要将之前的【persistlog】打包配置，默认配置文件在【device/fsl/imx8q/mek\_8q/mek\_8q.mk】中，搜索【persistlog】将其修改成【persistlog-xlog】，实例如下：~~

~~~~

~~修改完成后，整个编译(非增量编译)即可将【persistlog】替换成【persistlog-xlog】。~~

## 日志文件解压

使用【persistlog-xlog】后生成的日志信息都是通过压缩的，查看日志时就需要通过对应的脚本解压。解压工具默认在【persistlog-xlog/mars/log/crypt/decode\_mars\_nocrypt\_log\_file.py】路径下，工具使用python2.7编写，使用时需要运行系统支持【python2.7】的环境。这里以【ubuntu】系统为例。

将日志拷贝到【ubuntu】系统中运行脚本，通过【python2.7 decode\_mars\_nocrypt\_log\_file.py hsae\_persistlog\_4.xlog】指令解压日志文件，如下：

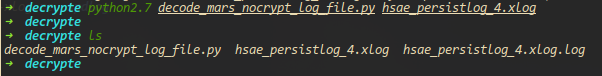


图 6‑2文件解压

解压后的文件以输入文件名开头，以【.log】结尾。

# 相关记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 记录名称 | 记录编号 | 保存部门 | 保存期限 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 附录

相关资料

logcatd默认启动配置

* https://blog.csdn.net/shusuanly/article/details/80577306

logd机制

* https://jekton.github.io/2018/05/11/logd-overview/
  + https://jekton.github.io/2018/05/16/logd-writing-part1/
  + https://jekton.github.io/2018/05/17/logd-writing-part2/

启动流程

* https://blog.csdn.net/kc58236582/article/details/78364759
* https://github.com/aosp-mirror/platform\_system\_core/blob/master/logcat/logcatd.rc