华清远见 FARSIGHT 嵌入式培训专家

嵌入式学院 华清远见旗下品牌

android移植概述

潘友华

版权

- 华清远见嵌入式培训中心版权所有；
- 未经华清远见明确许可，不能为任何目的以任何形式复制或传播此文档的任何部分；
- 本文档包含的信息如有更改，恕不另行通知；
- 保留所有权利。

内容提纲

- 1、android简介
- 2、移植方法概述
- 3、android分区
- 4、android源码结构
- 5、android编译系统
- 6、厂商定制
- 7、初始化脚本

android简介

- Android是一种基于Linux内核的开源、免费的操作系统，由Google公司和开放手机联盟主持开发。主流应用是智能终端，如智能手机、平板电脑、机顶盒、智能家居、穿戴设备。目前也开始在一些工业控制领域的得到应用。
- Android操作系统最初由Andy Rubin开发，主要支持手机。2005年8月由Google收购注资。2007年11月，Google与84家硬件制造商、软件开发商及电信营运商组建开放手机联盟共同研发改良Android系统。随后Google以Apache开源许可证的授权方式，发布了Android的源代码。
- 2013年09月24日谷歌开发的操作系统Android在迎来了5岁生日，全世界采用这款系统的设备数量已经达到10亿台。
- Android在正式发行之前，有两个内部测试版本，分别是：阿童木（AndroidBeta），发条机器人（Android 1.0）。后来由于涉及到版权问题，谷歌将其命名规则变更为用甜点作为它们系统版本的代号的命名方法。从Android 1.5发布的时候，代表每个版本代表的甜点的尺寸越变越大，分别是：纸杯蛋糕（Android 1.5），甜甜圈（Android 1.6），松饼（Android 2.0/2.1），冻酸奶（Android 2.2），姜饼（Android 2.3），蜂巢（Android 3.0），冰激凌三明治（Android 4.0），果冻豆（Jelly Bean，Android4.1和Android 4.2）。

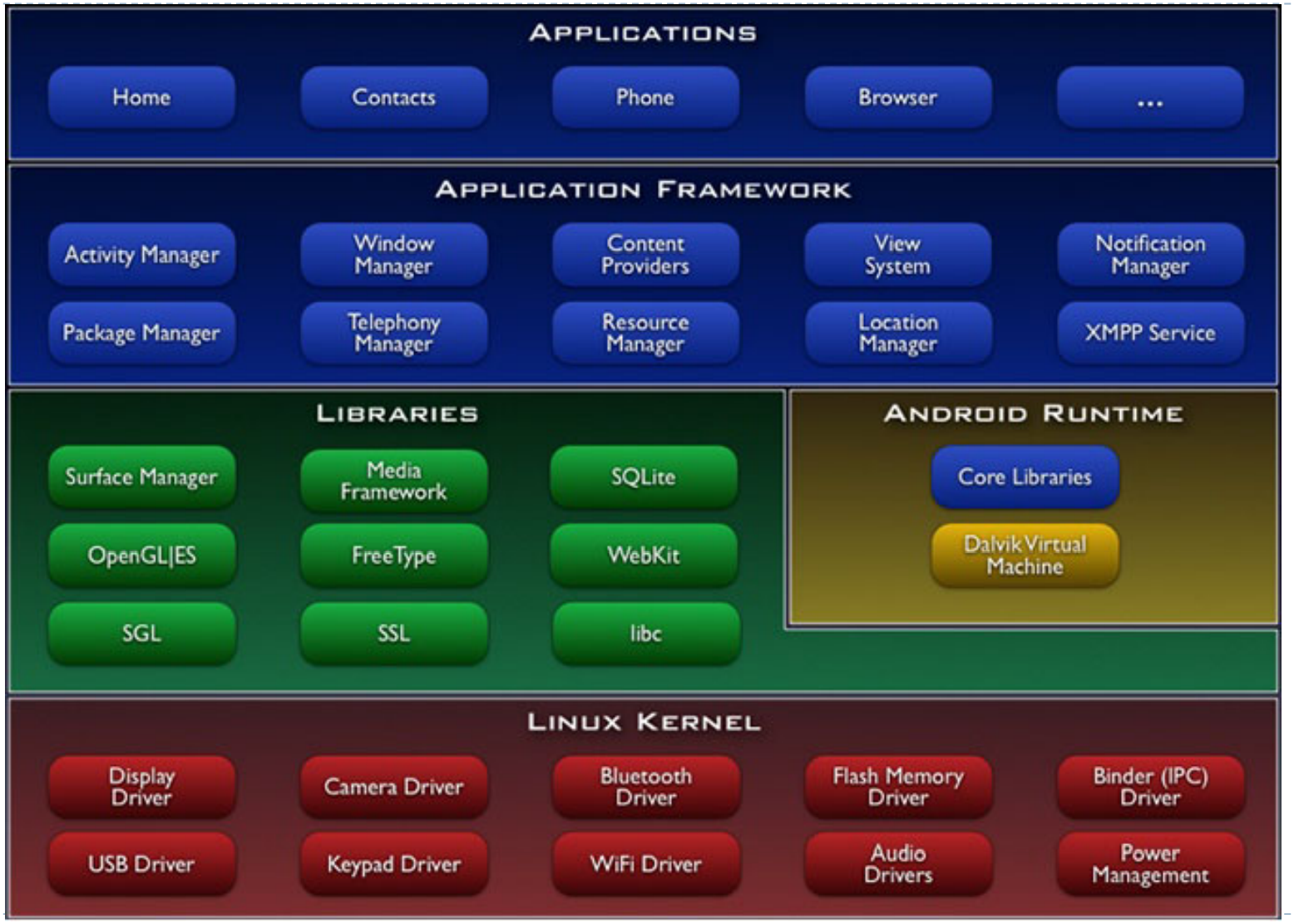
android简介


ANDROID
Cupcake
Android 1.5
Donut
Android 1.6
Eclair
Android 2.0/2.1
Froyo
Android 2.2
Gingerbread
Android 2.3
Honeycomb
Android 3.0
Ice Cream Sandwich
Android 4.0
Jelly Bean
Android 4.1
&Android 4.2

移植方法概述

- 一般的，基于成熟代码的移植，主要会有三个方面的工作：
  - A、有现成直接可用的代码，则选中编译
    - 主要是熟悉源码组织结构和编译系统。一般的，基于linux环境下的编译系统，一般都是由脚本及Makefile组成。
  - B、有现成的代码，但是部分代码不合适，则修改后选中编译
    - 除熟悉源码组织结构和编译系统外，还需要熟悉系统初始化流程，理解代码组织框架等等。
  - C、没有的代码，则编写添加
    - 一般的，需要添加的都是涉及设备控制的逻辑和功能模块。所以除熟悉源码组织结构和编译系统外，对该系统源码设计的思路要充分理解，按照其设计思路设计新的代码并加入到源码中。

移植方法概述


APPLICATIONS
Home
Contacts
Phone
Browser
...
APPLICATION FRAMEWORK
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
XMPP Service
LIBRARIES
Surface Manager
Media Framework
SQLite
OpenGL|ES
FreeType
WebKit
SGL
SSL
libc
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine
LINUX KERNEL
Display Driver
Camera Driver
Bluetooth Driver
Flash Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

移植方法概述

- 广义的，android系统包含bootloader、linux内核、及android原生版源码。所以android移植会涉及三个方面的工作：
  - A、bootloder移植
  - bootloder是引导程序，主要的工作是初始化好相应的硬件，把内核从外存读到内存上并跳转到内核入口地址，让内核运行起来。bootloder可以自己编写，也有很多免费、开源的，u-boot就是基于ARM平台的比较知名常用的一款bootloder。
  - B、linux内核移植
  - android kernel与linux主线版本的kernel有一定的差异，主要修改添加了：
    - Android IPC 系统 : Binder
    - Android 日志系统 : Logger
    - Android 电源管理 : Power
    - Android 闹钟管理 : Alarm
    - Android 内存控制台 : Ram_console
    - Android 时钟控制的 gpio: Timed_gpio
    - yaffs文件系统
  - C、android移植
  - android可以简单理解为文件系统的主要部分，包含了应用层APP、应用层框架、系统运行库、硬件抽象层、中介软件等等。

android分区

- 一般的，Android系统磁盘分区如下：
- Boot分区：存储boot.img映像
- System分区：存放System.img映像
- UserData分区：存放userdata.img映像
- Recovery分区：存放recovery.img映像
- Cache分区：应用程序缓存分区，加快程序启动
- Misc分区：系统设置厂商硬件设置信息分区

android分区

分区/挂载对象	挂载点
Boot 分区	/boot
System 分区	/system
UserData 分区	/data
Recovery 分区	/recovery
Cache 分区	/cache
Misc 分区	/misc
sdcard 卡	/sdcard

android分区

- boot.img镜像不是普通意义上的文件系统，而是一种特殊的Android定制格式，由文件头信息boot header，压缩的内核，文件系统数据ramdisk以及second stage loader（可选）组成，它们之间非页面对齐部分用0填充。
- ramdisk.img是一个最基础的小型文件系统，是对root目录的打包和压缩，其中包含了启动android的很重要的文件，比如内核启动完后加载的第一个进程init、一些重要的配置文件等，总之它控制着整个android的启动。根据 init.rc、init.xxx.rc来初始化并装载系统库、程序等直到开机完成。以下是一个典型的ramdisk中包含的文件列表：
  - A、脚本配置文件及初始化程序
  - /init.trout.rc、/default.prop、/init.rc、/init.xxx.rc、/init
  - B、目录
  - /proc/、/dev/、/sys/、/sbin/、/system/、/data/
- android源码中提供mkbootimg工具来制作镜像包。格式如下：
  - mkbootimg --kernel 内核压缩包 --ramdisk ramdisk镜像 --board "xxx" --base 基地址 -o boot.img

android分区

- Android编译结果：
- 保存目录：out/target/product/<products>/
- 不同的平台其输出到out/target/product/不同的子目录下
- ramdisk.img：
  - 虚拟内存盘，用内存来模拟磁盘，它在Linux内核启动后被挂载，该映像里保存有Android最基本的文件系统以及一些命令
- system.img：
  - Android系统主要的文件系统映像，里面包含有Android系统运行必需的库、程序和配置文件
- userdata.img：
  - 用户数据映像，它是用户应用程序、用户信息保存目录
- recovery.img：
  - 系统恢复映像，当系统进入恢复模式时的启动映像

android源码结构

Project	Description
abi	abi相关代码。ABI：applicationbinary interface，应用程序二进制接口
bionic	C runtime: libc, libm, libdl, dynamic linker bionic含义为仿生，这里面是一些基础的库的源代码
bootloader/legacy	Bootloader reference code 启动引导相关代码
build	Build system build目录中存放的是编译系统mk文件，编译规则和generic产品基础配置文件
cts	Android兼容性测试套件标准
dalvik	Dalvik virtual machine JAVA虚拟机
development	High-level development and debugging tools 程序开发所需要的模板和工具 ，如AVD，示例源码
device	设备相关代码

android源码结构

frameworks	Core Android app framework libraries 核心框架——java及C++语言，是Android应用程序的框架
hardware	Hardware Abstraction Layer 主要是硬件抽象层HAL代码
ndk	Android NativeDevelopment Kit ndk相关代码
out	编译完成后的代码输出与此目录
prebuilt	Binaries to support Linux and Mac OS builds x86和arm架构下预编译的一些资源
packages	Android的各种系统级应用程序
sdk	Sdk工具源码及qemu相关源码
system	Android根文件系统相关源码，如：init、adb、toolbox及一些库
docs	介绍开源的相关文档
external	Android使用的一些开源的模块代码
libcore	核心库相关

Android 编译系统

- goole官网说明的编译过程：
  - A、初始化
    - $ source build/envsetup.sh
    - OR
    - $ ./build/envsetup.sh
  - B、选择编译平台
    - $ lunch <build-option>
  - C、执行编译
    - $ make

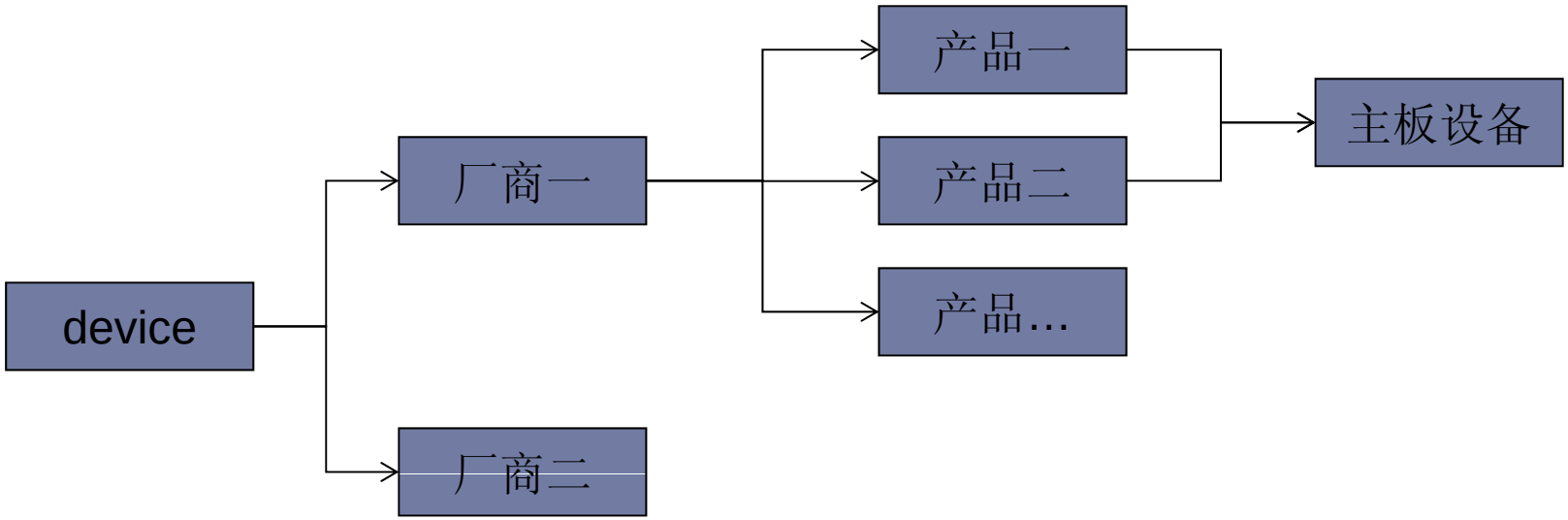
Android 编译系统

- build/envsetup.sh
  - A、定义了一些shell命令函数，这些命令主要是初始化编译环境，选择定制厂商等等。
  - B、加载vendor目录下自定义厂商产品编译项文件vendorsetup.sh，以此来让编译系统编译不同产品的定制目录。
- lunch命令
  - 打印出所有系统编译项，设置主要环境变量
  - TARGET_PRODUCT
  - TARGET_BUILD_VARIANT
  - TARGET_BUILD_TYPE

Android 编译系统

- 几个重要的Makefile
  - Android.mk
    - 编译源码Makefile文件，每个源码目录下都会有
  - main.mk
    - 定义了编译全部代码的依赖关系
  - config.mk
    - 用于配置编译系统，决定如何编译
  - envsetup.mk
    - 定义了编译环境配置
  - product_config.mk
    - 读取AndrodProducts.mk生成TARGET_DEVICE变量
  - AndroidProducts.mk
    - 定义某厂商所有产品文件列表
  - BoardConfig.mk
    - 定义开发板软件相关配置项，将来影响系统条件 编译。

厂商定制



厂商定制

- Android源码可以被编译成不同平台的系统，为了方便对源码分类管理，将所有平台相关代码放到device目录下，由于不同平台归属于不同的厂商，同一厂商可能有多个不同产品平台，不同产品平台可能使用相同的主板设备，Android源码对平台有以下规定：
  - 不同的厂商在device下创建不同目录
  - 同一厂商可能生产多个目标产品（TARGET_PRODUCT），这些产品都通过一个产品配置文件来说明其产品细节（PRODUCT_NAME.mk）
  - 同一厂商的多个产品由产品列表文件配置说明AndroidProducts.mk
  - 同一厂商的多个产品可能使用相同主板设备 （TARGET_DEVICE）

厂商定制

- 1、在 device/目录下创建公司目录
  - $ mkdir device/<company>
- 2、创建一个产品名目录，用来存放产品相关文件
  - $ mkdir device/<company>/<product>/
- 3、创建一个产品Makefile
  - device/<company>/<product>/AndroidProducts.mk
  - 一般有如下内容：
  - A、修改或指定系统属性及自定义属性
  - B、拷贝文件（脚本、配置、资源、APP、库等等）
  - C、选配要编译的对象（应用程序、库）
  - D、指定产品、设备名称

厂商定制

- device/<company>/<product>/AndroidProducts.mk范例如下：
  - $(call inherit-product, device/.../xxx.mk)
  - DEVICE_PACKAGE_OVERLAYS := device/.../overlay
  - PRODUCT_COPY_FILES += \
  - source_path:destination_path
  - PRODUCT_PROPERTY_OVERRIDES +=\
  - key=value
  - PRODUCT_PACKAGES += \
  - xxx
  - PRODUCT_CHARACTERISTICS := tablet
  - # Overrides
  - PRODUCT_MODEL := demoPRODUCT_NAME := <first_product_name>
  - PRODUCT_DEVICE := <device_name>
  - 注：first_product_name 要和编译项中的产品名一致，device_name对应当前产品的设备目录，一般device_name和product_name一样。

厂商定制

- 4、创建设备Makefile，用来配置设备信息
- device/<company>/<product>/BoardConfig.mk ， 一般其内容有：
  - A、指定参数
    - 参数涉及代码中的参数、编译目录、编译参数等等
  - B、选配开关
    - Android.mk中的编译开关，以此配置来决定是否执行对应的编译
  - C、参数开关
    - 匹配开关后来指定参数
- 范例如下：

```
include device/.../xxx.mk
TARGET_NO_BOOTLOADER := true
TARGET_NO_KERNEL := true
TARGET_CPU_ABI := armeabi
BOARD_FLASH_BLOCK_SIZE := 4096
SW_BOARD_USR_WIFI := rtl8192cu
BOARD_HAVE_BLUETOOTH := true
```

厂商定制

- 5、将新的平台的编译选项添加到lunch菜单中
  - 创建device/<company>/vendorsetup.sh
  - 在里面添加新平台编译选项“<first_product_name>-eng”
  - add_lunch_combo <first_product_name>-eng

厂商定制

- Product目录树
- <company>
  - <product>
    - AndroidProducts.mk
    - <first_product_name>.mk
    - <second_product_name>.mk
    - [AndroidBoard.mk]
    - BoardConfig.mk
    - [system.prop]
    - vendorsetup.sh

初始化脚本

- init进程
  - init进程是一个由内核启动的用户级进程。启动过程就是代码init.c（system\core\init目录下）中main函数执行过程。init进程会解析执行init.rc，就是所谓的初始化脚本。
- init.rc脚本
  - init.rc文件由语句组成，主要包含了四种类型的语句：Action、Commands、Services、Options。在init.rc文件中一条语句通常是占据一行。单词之间是通过空格符来相隔的。如果需要在单词内使用空格，那么得使用转义字符“\”，如果在一行的末尾有一个反斜杠，那么是换行折叠符号，应该和下一行合并成一起来处理，这样做主要是为了避免一行的字符太长，与C语言中的含义是一致的。注释是以#号开头。Action和services显式声明了一个语句块，而commands和options属于最近声明的语句块。在第一个语句块之前 的commands和options会被忽略。
- 关键字
  - section：语句块，相当于C语言中大括号内的一个块。一个Section以Service或On开头的语句块。以Service开头的Section叫做服务，而以On开头的叫做动作(Action)
  - action：动作
  - services：服务
  - commands：命令
  - options：选项
  - trigger：触发器，或者叫做触发条件
  - class：所属类，即可以为多个service指定一个相同的类属，方便操作同时启动或停止

初始化脚本

- 动作(Action)
  - 动作表示了一组命令(commands)组成。动作包含一个触发器，决定了何时执行这个动作。当触发器的条件满足时，这个动作会被加入到已被执行的队列尾。如果此动作在队列中已经存在，那么它将不会执行。
  - 一个动作所包含的命令将被依次执行。动作的语法如下所示：

```
on <trigger>
    <command>
    <command>
    <command>
```

初始化脚本

- 服务(services)
  - 服务是指那些需要在系统初始化时就启动或退出时自动重启的程序。
  - 它的语法结构如下所示：

```
service <name> <pathname> [ <argument> ]*
        <option>
        <option>
        ...
```

初始化脚本

- 选项（options)
  - 选项是用来修改服务的，它们影响如何及何时运行这个服务。

选项	描述
critical	据设备相关的关键服务，如果在4分钟内，此服务重复启动了4次，那么设备将会重启进入还原模式。
disabled	服务不会自动运行，必须显式地通过服务器来启动。
setenv <name> <value>	设置环境变量
socket <name> <type> <perm> [<user> [<group>]]	在/dev/socket/下创建一个 unix domain 的 socket，并传递创建的文件描述符 fd 给服务进程，其中 type 必须为 dgram 或 stream、seqpacket，用户名和组名默认为0
user <username>	在执行此服务之前先切换用户名，当前默认为 root
group <groupname> [<groupname>]*	类似于 user，切换组名
oneshot	当此服务退出时不会自动重启
class <name>	给服务指定一个类属，这样方便操作多个服务同时启动或停止。默认情况下为 default
onrestart	当服务重启时执行一条指令

初始化脚本

- 触发器(trigger)
  - 触发器用来描述一个触发条件，当这个触发条件满足时可以执行动作。

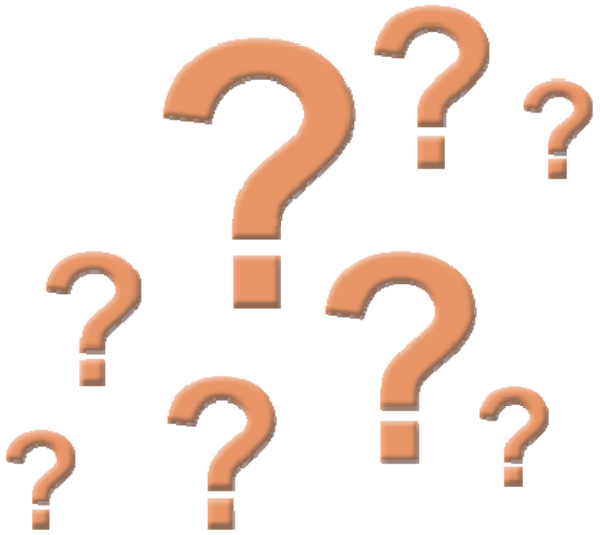
触发器	描述
boot	当 init 程序执行，并载入 init.conf 文件时触发
<name>=<value>	当属性名对应的值设置为指定值时触发
device-added-<path>	当添加设备时触发
device-removed-<path>	当设备移除时触发
service-exited-<name>	当指定的服务退出时触发

初始化脚本

- 属性(Properties)
  - init程序在运行时会更新属性系统的一些属性，提供程序内部正在执行的信息。

属性名	描述
init.action	当前正在执行的动作，如果没有则为空字符串""
init.command	当前正在执行的命令.没有则为空字符串.
init.svc.<name>	当前某个服务的状态，可为"stopped"，"running"，"restarting"

Q&A



谢谢！