树莓派 3 操作系统开发 -- 技术报告

肖华冬

2018 年 6 月

目 录

前 言

本技术报告是对树莓派 3 操作系统开发过程中涉及的技术做一个总结。根据开发中涉及的工具不同，分为下面几个类别：

- Makefile
- Bash 脚本
- 链接程序
- Eclipse CDT
- C++
- ARMv8 体系结构
- Word 文档编辑

Makefile

命令转发

在 makefile 解析的过程中，make 会构造一些环境变量。其中，环境变量 MAKE 指向的是当前使用的 make 命令。通过在目标的命令中使用:$(MAKE) -f makefile TARGET 的形式，我们可以要求重新构造某个目标。

这个特性有用的地方在于，在转发的命令行上，我们可以重新定义某些环境变量，从而使得构造的同一个目标具有不同的上下文语境。

多构建系统协作

make 每次解析一个 makefile，都会形成一个基本的构建时运行环境。一般而言，一个工作空间可以定义许多构建环境来生成不同的文件。

多个构建系统协同工作的途径是强制命令转发。

测试： a.txt:touch a.txt && touch b.txt b.txt:a.txt echo making $@

可以知道 b.txt 一定比 a.txt 更新，因此 echo making $@的命令不会被执行。

为目标添加其他文件形式的依赖

为每个目标生成一个强制更新模式

如果目标不存在对应的文件，或者目标声明为伪目标，则目标的命令总是会被执行。强制更新使用的就是这个特性。

首先定义一个 FORCE 伪目标，然后，使用%通配符构造一个依赖。

确定一个目标及其依赖的具体原则

目标和依赖必须是独立的，也就是说，如果目标被删除了，它一定可以简单地通通过 make *目标*的方式重新构建出来，同时不引起其他部分的更新。

这就是最小依赖原则。

与此条款违背的一个具体例子就是链接程序链接过程中生成的 mapfile 和 elf 文件，它们具有相同的时间戳，并且 mapfile 实际上不能被单独构建出来。

对这种形式的依赖的改造就是，将 mapfile 与 elf 标记为同一组目标。去掉 mapfile 目标，将所有 mapfile 目标替换为 elf 目标。

参考

1. make 程序会将 makefile 中使用的 include 指令其后的文件参数作为依赖来检查。如果没有将该文件声明为一个目标，则检查其存在与否。如果该文件是一个目标，即文件的引用形式与某个目标的模式匹配，则首先检查的是该目标。
2. 具有相同时间戳的目标和依赖，目标不会被更新。

Bash 脚本

创建具有相同时间戳的文件

touch a b 创建的文件 a,b 具有相同的时间戳。
时间戳的具体指可通过 stat -c '%Y' FILE 的方式查看。

链接程序

导出链接时符号值

链接程序 ld 可接受-Map mapfile 作为符号映射的输出文件，结合 grep 可以得到每个输出符号的具体值。

注意，如果符号使用 PROVIDE 方式定义的，则其值可能未显示，因为没有引用。

确定未告知的分区

链接脚本中我们可能不会列出所有的分区名称，从而导致有的内容被放到我们期望的分区之前，或者其他地方。总之，这些行为是未定义的。

为了确定生成文件中有哪些是未定义的分区，可使用 readelf -S。

正确处理符号映射文件的时间戳

当使用 ld 的-Map mapfile 输出文件时，ld 的生成文件与 mapfile 具有确切相等的时间戳，这会导致依赖上产生的问题。因为 make 在遇到相同的时间戳时，会对依赖进行更新。

Eclipse CDT

Project

Linked Resource

Project

1. Resource
   1.1. Linked Resources
      1.1.1. Path Variables
         1.1.1.1. Define a New Path Variable 定义一个新的路径变量。所有的路径变量与构建变量是两个分离的名称空间。
      1.1.2. Linked Resource 此处可以删除引用的目录，但是不能增加。增加在 Paths & Symbols 中进行
   1.2. Resource Filters
      1.2.1. Filter
         1.2.1.1. Include Only
         1.2.1.2. Exclude All 添加的目录不会显示在工程中，但是该目录在文件系统中仍然是存在的。不同于 Exclude from Build，目录不会显示灰色，而是直接取消显示
         1.2.1.3. Filter Details 可以选择文件的属性、名称、相对路径作为过滤条件。比如，如果不希望 src/loader, 则选择相对路径作为条件。注意，当目标是目录时，需要勾选 recursive 选项
         1.2.1.4. 已知的问题：如果添加了 Filter，则导致编码变成 GBK（虽然配置可能仍然是 UTF8），从而导致文件不能正常显示。因此，不推荐添加 Filter。通过 Paths & Symbols 添加构建排除即可。
      1.2.2. Add Filter
      1.2.3. Add Group
2.

Workspace

快捷键： Ctrl + P 设置成提示

在一个 Workspace 下定义多个共享源代码的工程

每个工程是一个基本的构建运行时。

为工程生成构建预配置文件

当我们删除了 Debug 目录时，Eclipse CDT 不会自动更新 Debug 目录以便生成 makefile。因此，实际上此时该工程没有构建系统。

为了生成一个构建系统，需要在 Eclipse CDT 可感知的范围内开始构建某些东西。为此，只需要添加一个 prepare 的伪目标，工程会自动生成构建系统。

从工程和目标的粒度进行依赖性分析

依赖是自动构建的本质所在。工程是指一个单独定义的构建运行时，目标则是这个构建环境的基本组成成分。每一个工程都会定义许多目标，工程之间存在相互依赖。Eclipse CDT 提供了一种自动构建的依赖模型：工程依赖。简单来说，每个工程都可以依赖其他工程。当某个工程构建任意一个目标时，它会首先构建依赖的目标，使用的是 make all。

但是这种粒度实际上是非常粗的，因为工程之间可以相互依赖。比如，一个工程会生成 elf 和 binary 两种格式文件，其中 binary 还依赖于另一个工程的 binary。而另一个工程的 elf 则依赖于此工程的 elf。显然，这种基于目标的依赖更加符合现实情况。但是，Eclipse CDT 不支持基于目标的依赖，因此，我们正确地添加依赖，我们需要从目标的角度确定工程之间的依赖。

如果从目标的角度分析出环形依赖，则这个依赖是无效的。因为要想构建 a，必须构建 b；要想构建 b，必须构建 a。其结果是，a 使用旧的 b 构建，b 使用旧的 a 构建，无限循环下去。如果没有分析出环形依赖，则可以确定出一个终点目标。该终点依赖其他目标，但是没有任何目标依赖该终点目标。于是， 这个终点目标所在的工程依赖于它依赖的目标所在的工程，即便中间目标存在工程之间的环形依赖，但那是伪环形依赖。

如果使用了转发，则依赖可以做得更轻，不需要依赖 make all，只需要依赖构建系统的存在即可，即 make prepare。

注意，最终是否依赖还需要检查参考 1 的隐式条件。

参考

1. Eclipse CDT 添加工程依赖后，以工程 A 依赖工程 B 为例，如果 A 的 Build Artifact 为 A.elf， 相应地 B 为 B.elf。则，添加依赖后，A 自动生成的构建文件中，A.elf 会自动依赖 B.elf。这一隐藏细节是决定我们是否选择 Eclipse 工程依赖的重要参考。

C++

使用前向声明避免无关依赖

结合模板和汇编立即数

假定我们希望将某条指令包装成函数调用，为其设定参数和返回值。如果这条指令只能接受立即数作为操作数，则一般的函数形式不能满足这条汇编指令的要求。一种可行的方法是，将立即数声明为模板参数，令其在编译时可知。然后，在动态内联汇编中，使用"i"作为输入操作数即可。

具体参见 svc_call 函数的实现。

在源代码级别声明构建文件集合

一般而言，通过定义不同的构建运行时，我们可以链接出不同的生成二进制文件。这是通过定义一个 OBJECT_LIST 变量来完成的。

但是，在源码级别，我们可以单独定义一个源文件，在其中使用#include 指令引入其他的源文件(.cpp)，这样实际上就会编译出一个很大的对象文件。

当然，从效率的角度考虑，我们推荐使用 OBJECT_LIST 的方式来实现。因为只要源代码是共享的，则对象文件也是可以共享的，可以避免额外的编译负担。

ARMv8 体系结构

生成 PSTATE 值

ARMv8 没有定义可直接访问 PSTATE 的单个寄存器，而是定义了一系列特殊的寄存器，包括 SPSel，CurrentEL，DAIF，NZCV，PAN，UAO 等特殊目的寄存器。通过将这些寄存器的值"逻辑或"，即可获取 PSTATE 的值，这对于进程状态的中断和恢复十分重要。

参考

1. ARMv8 仍然分为多个版本，分别为 ARMv8-a, ARMv8.1-a,ARMv8.2-a 等。在 g++的-march=选项中可以指定。比如，ARMv8-a 不支持 PAN 寄存器，但是 ARMv8.2-a 支持。

Word

样式

word 的文本格式是由样式控制的，每一份内容都有一个对应的样式。对样式的修改会反映到所有相应的新建内容上，但是对于已经存在的内容，样式保持不变。

当选择"更新某个样式以匹配所选的内容"时，其含义以当前内容为模板，更新样式设置，这样会使得所有该样式的内容得到更新。

中文排版常规要求

项目	内容
表格	水平对齐，垂直对齐
字体	标题：黑体 其他：宋体 数字、英文等：Times New Roman
页眉	

如何建立模板

主要是通过对样式的字体设置完成的。每种样式都定义了中文和西文所对应的字体、字号。

重复当前样式

在内容的段中使用换行符。