各位看官好，上一篇文章我们聊了一下关于 OTA 升级过程中，新的软件包是如何从开发者的电脑上，安全的下载到嵌入式设备中的。

这个流程似乎很简单，不就是下载一个文件而已嘛，怎么还值得写成一篇文章呢?

其实这不仅仅是下载文件这么简单，这其中涉及到如何对众多的终端设备进行批量升级的策略问题。

如果你亲自在 AWS 的平台上操刀一次，就知道这其中有很多细节问题是需要考虑的。

一失足成千古恨哪！一旦设备升级策略忽略了一个小细节，也许某一天就是我们的深渊！

包括产品的生产过程也是如此，那些踩过的坑，真是一把鼻涕一把泪，这个问题后面有时间专门写一篇。

今天，我们继续 OTA升级过程中后续的阶段。

还记得我们之前的假设吗?

设备中正在执行的 V1 版本的程序，包括这 3 个文件，它们位于文件系统中的 /root/app 目录下：

> main: 主程序;
>
> config.ini: 配置文件(包括一个配置项：version=V1_0);
>
> mylib.so: 实现了某个算法的动态库，被 main 程序调用;

现在，新的版本 V2 优化了算法，压缩包名称是 app_V2.0.tgz，其中包括文件：

> main: 没有变化;
>
> config.ini: 配置项修改了：version=V2_0;
>
> mylib.so: 优化了算法，主要就是想升级这个动态库;
>
> upgrade.sh: 一个脚本程序，新增的文件;

升级包 app_V2.0.tgz 已经被下载到设备本地的文件系统中了，假设解压到目录 /root/upgrade 中。

现在需要做的事情就是：新版本程序，去替代 /root/app 目录中的旧版本程序。

upgrade.sh 升级脚本

我们首先要明白一个问题：执行升级指令、下载压缩包，都是此刻正在执行的 main 程序来执行的。

如果把复制替换的操作也让 main 程序来执行的话，肯定是会出问题的：它不可能去复制一个新的 main 文件，来把自己替换掉！

写过单片机程序的小伙伴肯定都知道：当新的固件下载到 flash 之后，一般都是重新启动设备，然后由 bootloader 来执行具体的文件复制操作。

那么对于带有文件系统的设备来说，也可以模仿类似的操作方式。

比如：当设备重新启动后，当执行 /etc/rc.local 时，此时 main 应用程序还没有启动。

此时就可以在 rc.local 这个文件中去做升级操作。

但是这样的方式，相当于是轻微的侵入了操作系统，总感觉这样做不太好。

此刻， upgrade.sh 升级脚本开始登场了！

这个脚本文件的主要作用就是用来控制升级过程。

这里隐藏这一个很重要的思想：`upgrade.sh` 是放在升级包中的，它并没有固化在终端设备中。

这样的话，每次执行升级任务时，都可以根据本次的升级需要，来灵活的编写升级脚本。

换句话说：只要能保证升级的通道没有问题，那么升级的过程就完全由这个脚本文件来控制，你想怎么搞，就怎么搞！

完全升级

所谓的完全升级，就是把旧版本的程序全部丢弃，把升级包中的新程序全部复制过去。

此时，升级脚本文件 `upgrade.sh` 就完成下面这几个主要工作：

> 1. 停止(kill)当前正在执行的 V1.0 版本的程序;
> 2. 删除 /root/app 目录下的所有旧文件;
> 3. 把升级包中所有的新版本文件 /root/upgrade/* 复制到 /root/app 目录下;

这样的完全升级方式是最无脑、最粗鲁的。

当然，还有一些细节问题是需要考虑的。比如：如果复制文件过程中出现错误怎么办?

还有一点，既然刚才提到了配置文件 `config.ini`，不知您是否会有这样一个疑问：

如果配置信息被用户修改了，那么升级之后，所有的配置信息又被恢复为默认值了，用户的私人配置信息全丢了怎么办?

关于这个问题，我们就继续来聊一下增量升级！

增量升级

所谓的增量升级：就是升级时并不会把所有的文件全部进行替换，而只是替换那些需要更新的文件。

对于我们假设的升级场景，只需要做 2 件事情：

> 1. 替换 mylib.so 库文件;
> 2. 把配置文件 config.ini 中的版本字段修改为：version=V2_0;

同样的，所有的升级过程仍然是写在 `upgrade.sh` 这个升级脚本中：

> 1. 停止(kill)当前正在执行的 V1.0 版本的程序;
> 2. 把 /root/upgrade/mylib.so 文件复制到 /root/app 目录下;
> 3. 使用 sed 命令来修改 config.ini 文件中的 version 字段;

PS：此时升级包中，只需要包含必要的文件就可以了，不需要把其他用不到的文件也放进去了。

从我描述的文字来看，似乎完全升级和增量升级差别不大。

这是因为这里的示例太简单，如果是一个比较复杂的、有多个模块相互配合的应用程序，增量升级的优势就明显了。

关于 OTA 升级过程，就先说这么多了，主要是以思想为主，毕竟每一个项目的需求场景是不一样的，从大方向上明白 OTA 的升级过程就可以了。

One more thing

为了表示我不是在胡说八道，这里提供一个很多年前的项目中，升级脚本文件的模板。

在公众号后台留言：601，即可收到。

另外，不知道是否有小伙伴对于 ESP32 中的升级流程感兴趣，下次再专门写一篇 ESP32 模组，如何与 AWS 后台通过 MQTT 指令进行交互，以及固件的下载、升级流程。

------ End ------

推荐阅读

【1】C语言指针-从底层原理到花式技巧，用图文和代码帮你讲解透彻
【2】一步步分析-如何用C实现面向对象编程
【3】原来gdb的底层调试原理这么简单
【4】内联汇编很可怕吗？看完这篇文章，终结它！
【5】都说软件架构要分层、分模块，具体应该怎么做

[这里插入：微信公众号]

星标公众号，能更快找到我！

扫码添加 道哥 私人微信

C/C++、物联网

Linux 操作系统

Linux 应用开发

喜欢请分享，满意点个赞，最后点在看。