

目录

前言	1.1
Makefile简介	1.2
Makefile资料	1.3
Makefile实例	1.4
附录	1.5
参考资料	1.5.1

自动化利器：Makefile

- 最新版本：`v0.8`
- 更新时间：`20190530`

简介

简介Makefile的特点和用途，并给出Makefile实例，以便给其他人以参考，如何利用Makefile去将繁琐人工操作变成自动化处理的一键搞定，从而提高工作效率。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- [crifan/good_automation_tool_makefile: 自动化利器：Makefile](#)

如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook_template: demo how to use crifan gitbook template and demo](#)

在线浏览

- [自动化利器：Makefile book.crifan.com](#)
- [自动化利器：Makefile crifan.github.io](#)

离线下载阅读

- [自动化利器：Makefile PDF](#)
- [自动化利器：Makefile ePUB](#)
- [自动化利器：Makefile Mobi](#)

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 `crifan` 还写了其他 100+ 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2021-01-16
22:56:23

Makefile简介

什么是 Make 和 Makefile

- `make` 是一个 脚本工具 = script tool , 一般 (尤其是 Linux 类的系统中) 也叫做 命令 = command
 - 并且 Linux 类系统中的一般都已经内置安装了 `make`
- `Makefile` 是一个 文本文件 , 一般也叫做 脚本 = script
 - 是自己, 根据 `make` 的语法, 写的, 用于各种要做的事情, 实现对应的功能
- 用法 : 当你输入了 `make` 命令后, `make` 会去根据你写的 `Makefile` 中的规则去执行对应的动作
 - 最典型的用途是, Linux系统中用 `make` 根据 `*.c` 去编译生成 `*.o` 目标文件

Makefile 的用途和用法

类似于 `Makefile` 的工具, 主要用于实现:

- 将: 低效率的, 需要手动的多个步骤输入多个命令, 才能完成的工作
- 变成: 每次只需要, 高效率的, 运行 `make` 命令去完成各种复杂的功能
 - 当前, 前提是, 需要先写好对应的 `Makefile` 文件

即所谓的 自动化脚本

目的是: 提高工作效率

本人之前最早接触`Makefile`, 是在嵌入式开发期间, 有很多的系统、工具、软件, 都需要利用`makefile`去编译、配置。

比如: Linux内核的`make menuconfig`后再去`make`编译

后来在移动互联网开发期间遇到`makefile`, 更多的是:

将各种繁琐的人工要敲不同命令才能实现的功能, 用`makefile`去实现一键搞定的效果

Makefile 和其他自动化工具的区别

而自动化脚本, 其他也有很多工具和软件可以实现, 比如Linux中的 `shell` 等。而`Makefile`和`shell`等不同之处和侧重点是: 主要根据文件依赖关系, 主要用于文件编译。

而实际上可以利用文件依赖关系, 去实现更广泛的、做事情的先后顺序有依赖关系, 也可用 `Makefile` 去实现

比如自己的[Crifan的Gitbook的模板](#)中的`Makefile`, 就是用 `makefile` 脚本是实现:

- 本地调试gitbook
- 编译出不同格式的文件
- 发布到服务器上

等等功能, 其中内部利用了不同的步骤的先后依赖关系, 实现自动化的效果的。

详见帖子

[【已解决】提取Gitbook中Makefile公共部分](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-03-02
14:04:21

Makefile资料

官网资料

所有的资料中，最推荐的就是官网，解释的最权威和清楚。

不过是英文的，有能力的可以优先参考：

[GNU make: Top](#)

其中关于很多功能的解释，比较权威和清楚。

自己之前参考过官网的部分资料

比如之前就参考过：

- 函数：[GNU make: Functions](#)
- 想要输出信息，除了 `info`，还有 `error` 和 `warning`：[GNU make: Make Control Functions](#)

```
$(error text...)
$(warning text...)
$(info text...)
```

网上不错的 Makefile 教程

如果习惯看中文文档的，推荐一个网上之前流传甚广的教程：《跟我一起写Makefile》

其也是我最早参考学习的一份 `Makefile` 资料。之前是pdf版本的，现在出来有在线电子版了：

[概述 — 跟我一起写Makefile 1.0 文档](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-03-02
14:25:04

Makefile实例

下面列举一些自己用到过的 `Makefile` 去提高工作效率的实际的例子：

Makefile 中调用 Python 文件

[【已解决】Makefile中运行Python脚本文件](#)

Makefile 中实现自定义函数

[【已解决】Makefile中实现自己的函数并调用](#)

自定义函数的一些注意事项

- 自定义函数内部的内置函数需要加上 `eval` 才能正常运行
 - 详见： [【已解决】Makefile中define的自定义函数中的lastword不工作](#)
- 自定义函数开头不能是 `# xxx` 的注释，否则语法出错而无法运行
 - 详见： [【已解决】Makefile自定义函数调用出错： unterminated call to function 'notdir': missing '\)'. Stop](#)
- 自定义函数中打印变量，`echo`无效，需要用`info`
 - 详见： [【已解决】Makefile中define自定义函数中打印输出没效果](#)
- 自定义函数如何获取返回值
 - 详见： [【已解决】Makefile中如何获得define自定义函数的返回值](#)

提取出 gitbook 中公共部分的 Makefile

详见： [【已解决】提取Gitbook中Makefile公共部分](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-03-02
14:22:53

附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-03-02
13:50:27

参考资料

- 【已解决】 提取Gitbook中Makefile公共部分
- 【已解决】 Makefile中运行Python脚本文件
- 【已解决】 Makefile中实现自己的函数并调用
- 【已解决】 Makefile中define的自定义函数中的lastword不工作
- 【已解决】 Makefile自定义函数调用出错: unterminated call to function `notdir': missing)'. Stop`
- 【已解决】 Makefile中define自定义函数中打印输出没效果
- 【已解决】 Makefile中如何获得define自定义函数的返回值
- [GNU make: Top](#)
- [GNU make: Functions](#)
- [GNU make: Make Control Functions](#)
- [概述 — 跟我一起写Makefile 1.0 文档](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-03-02
14:29:04