构建工具

约瑟夫·哈利特

2023 年 1 月 13 日

那么这一切是怎么回事呢？

我们用计算机做的很多事情都与格式转换有关,我们在编译代码时会这样做： ▶ cc -c library.c -o library.o ▶ cc hello.c library.o -o hello

当我们归档文件时：

▶ zip -r coursework.zip coursework 当我们绘

制图形时：

▶ dot -Tpdf 流程图.dot -O 流程图.pdf

我们可以自动化这个吗？

是的！

我们可以编写一个 shellscript 并将所有任务放在一个地方.....

```
#！ /usr/bin/env bash cc
-c 库.c -o 库.o cc hello.c 库.o -o hello
zip -r coursework.zip coursework 点
-Tpdf 流程图.dot -O 流程图.pdf
```

但我们能做得更好吗？

- 如果只是流程图发生了变化，我们真的需要重新编译C程序吗？
- 我们可以概括构建模式吗？

制作

制作一个用于自动化构建的古老工具。 ▶由Stuart Feldman于 1976 年开发▶采用告诉您如何构建文件的规则▶然后按照它们构建您需要的东西！

它的两种主要方言（现在）： BSD Make 更老式,POSIX GNU Make功能更丰富,Linux 上默认 在实践中,除非您开发 BSD,否则每个人都使用 GNU Make

▶如果您使用的是 Mac 或 BSD 机器,请安装 GNU Make,如果不起作用,请尝试 gmake

生成文件

Make 规则被放入Makefile中,如下所示：

你好:hello.c库.o
cc -o hello hello.c 库.o

库.o: 库.c cc -c -o 库.o 库.c

课程作业.zip:课程作业
zip -r 课程作业.zip 课程作业

流程图.pdf:流程图.dot 点-Tpdf 流程
图.dot -O 流程图.pdf

如果你要求 make 构建 hello,它会弄清楚它需要做什么： $ make hello cc -c -o

library.o
library.c cc -o hello hello.c
library.o

做出改变

如果您更改文件... Make 足够智能,仅重新运行您需要的步骤:例如,如果您编辑 hello.c 并重建： $ make hello cc -o hello hello.c library.o

但是如果你编辑library.c,它会发现它需要重建所有内容$ make hello cc -c -o library.o

library.c cc -o
hello hello.c library.o

虚假目标

除了如何创建文件的规则之外，您还可以拥有不依赖于文件的虚假目标，而只是告诉 make 在运行时要做什么

通常， Makefile 会包含一个 phony: all通常是

文件中的第一个规则（或标记为 .default）：
取决于您想要构建的所有内容

clean删除所有生成的文件

install安装程序 $ make cc -c -o

library.o
library.c cc -o hello.o hello.c
library.o zip -r coursework.zip
coursework dot -Tpdf Flowchart.dot -O
Flowchart.pdf

.PHONY:一切干净

全部:你好 coursework.zip 流程图.pdf

干净的：
git clean-dfx

你好:hello.c库.o
cc -o hello hello.c 库.o

库.o: 库.c cc -c -o 库.o 库.c

课程作业.zip:课程作业
zip -r 课程作业.zip 课程作业

流程图.pdf:流程图.dot 点-Tpdf 流程
图.dot -O 流程图.pdf

图案规则

（到目前为止，一切都应该在 GNU 和 BSD Make 中运行……现在我们已经进入了 GNU 的土地）
如果我们想向 hello 程序添加一个额外的库怎么办？我们可以去更新 Makefile，但最好概括一下！

```
CC=叮当声
CFLAGS=-墙-O3

.PHONY:一切干净

所有:你好 coursework.zip 流程图.pdf 干净：

	git clean-dfx

你好:hello.c library.o extra-library.o

%.o: %.c $
	(CC) $(CFLAGS) -c -o $@ $<

%： %。C
	$(CC) $(CFLAGS) -o $@ $<

%.zip: % zip
	-r $@ $<

%.pdf: %.dot 点
	-Tpdf $< -O $@
```

隐式模式规则

实际上,因为 Make 太老了,它已经知道如何编译 C（和 Fortran/Pascal...)代码：

.PHONY:一切干净

所有:你好 coursework.zip 流程图.pdf 干净：

```
	git clean-dfx
```

你好:hello.c library.o extra-library.o

%.zip: % zip

```
	-r $@ $<
```

%.pdf: %.dot 点

```
	-Tpdf $< -O $@
```

让我们变得更一般！

假设我们想添加更多图形……我们可以添加对所有图形的依赖关系来构建它们，或者……

```
.PHONY：所有干净的数字=$
(patsubst .dot,.pdf,$(通配符 *.dot))

全部：你好 coursework.zip ${figures} 干净：

	git clean-dfx

你好：hello.c library.o extra-library.o

%.zip: % zip
	-r $@ $<

%.pdf: %.dot 点
	-Tpdf $< -O $@
```

Make 非常强大

我爱让…

▶我滥用它来编译所有内容▶用于分发可重复的科学研究▶用于构建和部署网站 模式规则和高级内容很简洁……

▶ …但如果你从不使用它,我不会生气▶ Make 是你在职业生涯中会一次又一次使用的工具之一。 ▶ …还有很多技巧我还没有向你展示;-)

去阅读GNU Make 手册

▶作为技术文档来说已经很不错了

综上所述

当您获得一些软件时...并且您会在其中找到一个 Makefile...
只需输入

make！ ▶ （并确保您的项目以相同的方式构建！）

（实际上,通常您必须输入 ./configure 然后 make,原因我们下次会谈到。）

▶不,我不会教你自动工具,别担心！