Make 和 Makefile快速入门

克拉默与矩阵 2018-08-12 原文

前言

一个项目，拥有成百上千的源程序文件，编译链接这些源文件都是有规则的。Makefile是整个工程的编译规则集合，只需要一个make命令，就可以实现“自动化编译”。make是一个解释makefile中指令的命令工具，一般来说，大多数的IDE都有这个命令，比如：Delphi的make，Visual C++的nmake，Linux下GNU的make。

涉及到的编译（严格上讲是4个过程，预处理、编译、汇编、链接）在之前的文章中都有介绍。

make和makefile什么关系？

Make这个词，英语的意思是"制作"。Make命令直接用了这个意思，就是要做出某个文件。比如，要做出汽车（car），就可以执行下面的命令。

```
make car
```

但是，如果你真的输入这条命令，它并不会起作用。因为Make命令本身并不知道，如何做出汽车，需要有人告诉它，如何调用其他命令完成这个目标。

```
car: wheel engine gasoline
    echo wheel > car
    echo engine > car
    echo gasoline > car
```

也就是说，make car这条命令的背后，实际上分成两步：第一步，确认 wheel 、engine 和 gasoline必须已经存在，第二步使用echo 命令将汽车需要的轮子、发动机、汽油写入car内部。

像这样的规则，都写在一个叫做Makefile的文件中，Make命令依赖这个文件进行构建。Makefile文件也可以写为makefile， 或者用命令行参数指定为其他文件名。

```
$ make -f rules.txt
# 或者
$ make --file=rules.txt
```

因此可见，make是大多数Linux采用的构建工具。make规则存放的文件通常情况下叫Makefile，这只是一个大家都遵守的习惯而已。

make用法

make [options] [targets] ...

```
-b, -m                          忽略兼容性。
-B, --always-make               无条件 make 所有目标。
-C DIRECTORY, --directory=DIRECTORY 在执行前先切换到 DIRECTORY 目录。
-d                              打印大量调试信息。
--debug[=FLAGS]                 打印各种调试信息。
-e, --environment-overrides 环境变量覆盖 makefile 中的变量。
-f FILE, --file=FILE, --makefile=FILE 从 FILE 中读入 makefile。
-h, --help                      打印该消息并退出。
-i, --ignore-errors             Ignore errors from commands. //和-k参数结合使用能够得到所有的编译错误信息
-I DIRECTORY, --include-dir=DIRECTORY 在 DIRECTORY 中搜索被包含的 makefile。
-j [N], --jobs[=N]              同时允许 N 个任务；无参数表明允许无限个任务。
-k, --keep-going                当某些目标无法创建时仍然继续。
-l [N], --load-average[=N], --max-load[=N]
                                在系统负载高于 N 时不启动多任务。
-L, --check-symlink-times       使用软链接及软链接目标中修改时间较晚的一个。
-n, --just-print, --dry-run, --recon 不要实际运行任何命令；仅仅输出他们
-o FILE, --old-file=FILE, --assume-old=FILE 将 FILE 当做很旧，不必重新生成。
-p, --print-data-base           打印 make 的内部数据库。
```

```
  -q, --question                 不运行任何命令；退出状态说明是否已全部更新。
  -r, --no-builtin-rules         禁用内置隐含规则。
  -R, --no-builtin-variables     禁用内置变量设置。
  -s, --silent, --quiet          不显示命令。
  -S, --no-keep-going, --stop 关闭 -k。
  -t, --touch                    touch 目标而不是重新创建它们。
  -v, --version                  打印 make 的版本号并退出。
  -w, --print-directory          打印当前目录。
  --no-print-directory           关闭 -w，即使 -w 默认开启。
  -W FILE, --what-if=FILE, --new-file=FILE, --assume-new=FILE 将 FILE 当做
  最新。
  --warn-undefined-variables   当引用未定义变量的时候发出警告。
```

Makefile文件怎么生成的?

对于小型项目，文件数量不多，这种情况你可以自己写makefile。但对于像Nginx这样的大项目，做法是由一个configure脚本根据用户提供的参数自动生成的。configure会检测你当前环境是否满足该软件的要求，然后根据你提供的选项参数自动生成makefile文件。一下是安装Nginx时的部分操作过程

```
[root@localhost nginx-1.14.]# ./configure \
> --prefix=/usr/local/nginx \
> --with-http_ssl_module
执行完这个命令会自动生成makefile
[root@localhost nginx-1.14.]# make && make install
```

对于采用CMake构建的项目来说，假如项目运行在Linux下，CMake可以根据当先环境的generator（Linux下就是gcc，g++）生成其构建所需要的makefile。

Makefile格式

概述

要想使make按照预期造出相应的东西，就要制定相应规则，而这些规则我们通常放在Makefile里面。规则的定义格式相当简单，如下

```
<target> : <prerequisites>
[tab]  <commands>
```

第一行冒号前面的部分，叫做"目标"（target），冒号后面的部分叫做"前置条件"（prerequisites）

第二行必须由一个tab键起首，后面跟着"命令"（commands）。

target是必需的，不可省略；prerequisites 和 commands都是可选的，但是两者之中必须至少存在一个。

每条规则就明确两件事：构建目标的前置条件是什么，以及如何构建。下面就详细讲解，每条规则的这三个组

成部分。

目标（target）

一个target就构成一条规则，target指明Make命令所要构建的对象。target可以是如下3种形式

①单个文件名

②多个文件名，使用空格分隔

③某个操作的名字，这也称为"伪目标"（phony target）

前置条件（prerequisites）

prerequisites通常是一组文件名，之间用空格分隔。它指定了target是否重新构建的判断标准：只要prerequisites在时间上新于target，即target过期了，就重新构建target

命令（commands）

命令（commands）表示如何更新目标文件，由一行或多行的Shell命令组成。它是构建"目标"的具体指令，它的运行结果通常就是生成目标文件。

每行命令之前必须有一个tab键。如果想用其他键，可以用内置变量**.RECIPEPREFIX**声明。

```
1.  .RECIPEPREFIX = >
2.  car:
3.  > echo wheel engine gasoline > car
```

上面代码用.RECIPEPREFIX指定，大于号（>）替代tab键。所以，每一行命令的起首变成了大于号，而不是tab键。

需要注意的是，每行命令在一个单独的shell中执行。这些Shell之间没有继承关系。

```
1.  var-lost:
2.      export foo=bar
3.      echo "foo=[$$foo]"
```

上面代码执行后（`make var-lost`），取不到foo的值。因为两行命令在两个不同的进程执行。一个解决办法是将两行命令写在一行，中间用分号分隔。

```
1.  var-kept:
2.      export foo=bar; echo "foo=[$$foo]"
```

另一个解决办法是在换行符前加反斜杠转义。

```
var-kept:
    export foo=bar; \
    echo "foo=[$$foo]"
```

最后一个方法是加上`.ONESHELL:`命令。

```
.ONESHELL:
var-kept:
    export foo=bar;
    echo "foo=[$$foo]"
```

参考资料

Isaac Schlueter的《Makefile文件教程

《GNU Make手册》。

Make 命令教程

Make 和 Makefile快速入门的更多相关文章

1. makefile快速入门
   前言 在linux上开发c/c++代码,基本都会使用make和makefile作为编译工具.我们也可以选择cmake或qmake来代替,不过它们只负责生成makefile,最终用来进行编译的依然是ma ...
2. Makefile 快速入门
   Makefile 速成 标签: Makefile编译器 2015-06-06 18:07 2396人阅读 评论(1) 收藏 举报 分类: C/C++ (132) Linux & MAC(19 ...

3. Linux快速入门04-扩展知识
   这部分是快速学习的最后一部分知识,其中最重要的内容就是源码的打包和软件的安装的学习,由于个人的Linux学习目的就是自己能在阿里云Ubuntu上搭建一个简单的nodejs发布环境. Linux系列文章 ...
4. CMake快速入门教程-实战
   http://www.ibm.com/developerworks/cn/linux/l-cn-cmake/ http://blog.csdn.net/dbzhang800/article/detai ...
5. 转:CMake快速入门教程-实战
   CMake快速入门教程:实战 收藏人:londonKu 2012-05-07 | 阅:10128 转:34 | 来源 | 分享 0. 前言一个多月 ...
6. Emacs快速入门
   Emacs 快速入门 Emacs 启动: 直接打emacs, 如果有X-windows就会开视窗. 如果不想用X 的版本, 就用 emacs -nw (No windows)起动. 符号说明 C-X ...
7. QuickJS 快速入门 (QuickJS QuickStart)
   1. QuickJS 快速入门 (QuickJS QuickStart) 1. QuickJS 快速入门 (QuickJS QuickStart) 1.1. 简介 1.2. 安装 1.3. 简单使用 ...
8. NOI Linux 快速入门指南
   目录 关于安装 NOI Linux 系统配置 网络 输入法 编辑器 1. gedit 打开 配置 外观展示 2. vim 打开 配置 使用 makefile 编译运行 1. 编写 makefile 2 ...
9. Web Api 入门实战 （快速入门+工具使用+不依赖IIS）
   平台之大势何人能挡? 带着你的Net飞奔吧!:http://www.cnblogs.com/dunitian/p/4822808.html 屁话我也就不多说了,什么简介的也省了,直接简单概括+demo ...

随机推荐

1. Lambda应用设计模式
   前言 在使用 Lambda 表达式时,我们常会碰到一些典型的应用场景,而从常用场景中抽取出来的应用方式可以描述为应用模式.这些模式可能不全是新的模式,有的参考自 JavaScript 的设计模式,但至 ...
2. 1068: [SCOI2007]压缩 - BZOJ
   Description 给一个由小写字母组成的字符串,我们可以用一种简单的方法来压缩其中的重复信息.压缩后的字符串除了小写字母外还可以(但不必)包含大写字母R与M,其中M标记重复串的开始,R重复从上一 ...
3. ntp---时钟同步服务
   NTP--时钟同步服务 地球分为东西十二个区域,共计 24 个时区 格林威治作为全球标准时间即 (GMT 时间),东时区以格林威治时区进行加,而西时区则为减. 地球的轨道并非正圆,在加上自转速度逐年 ...
4. Spring系列(一) Spring的核心
   Spring 简介 Spring 是一个开源轻量级企业应用架构,目的是为了简化企业级应用开发.(1)Spring 框架可以帮我们管理对象的生命周期,帮助我们管理对象间的依赖关系,相互协作:(2)Spr ...
5. Android 网络编程的陷阱
   陷阱一,不要在主线程或者UI线程中建立网络连接 Androd4.0以后,不允许在主线程中建立网络连接,不然会出现莫名其妙的程序退出情况.正确的做法是在主线程中,创建新的线程来运行网络连接程序. // ...

6. VScode启动后cup100%占用的解决方法
   新安装的vscode,版本1.29.1.启动后,cpu占用一直是100%,非常的卡.百度以下,找到了解决方法,整理一下. 解决方法:在VScode中文件->首选项->设置->搜索-& ...
7. vue里ref ($refs)用法
   ref 有三种用法: 1.ref 加在普通的元素上,用this.ref.name 获取到的是dom元素 2.ref 加在子组件上,用this.ref.name 获取到的是组件实例,可以使用组件的所有方 ...
8. windows 7 中使用命令行创建WiFi热点
   就是让你的电脑可以作为WiFi热点,然后供其它支持WiFi的设备上网 首先你的电脑中必须有正常使用的无线网卡 幺幺幺切克闹,开始命令吧,(注:命令是在windows中的命令提示符中运行的) 禁用承载网 ...
9. EBS Webservice Timeout，HTTP Server Return "500 Internal Server Error"
   http://blog.itpub.net/26687597/viewspace-1207571/ 基于Oracle EBS R12,开发了一个Webservice用于返回某项主数据,当请求的数据量非 ...
10. [转载]Java操作Excel文件的两种方案
    微软在桌面系统上的成功,令我们不得不大量使用它的办公产品,如:Word,Excel.时至今日,它的源代码仍然不公开已封锁了我们的进一步应用和开发.在我们实际开发企业办公系统的过程中,常常有客户这样子要 ...

热门专题

GIT 删除目录的版本控制 | PD建立物理模型字段批量设置 | 如何通过SUPER调用父类方法
对P1.O和P2.O连接 显示MULTIPLE | 多次请求,显示 该请求来自于缓存
LINUX 重定向 实时刷新 UNBUFFER | JAVA多线程并发致使服务器宕机 | HTMLRADIUS怎么用
SOCKS5服务端 WINDOWS | WINFORM 毛玻璃 | CENTOS7上SNMP如何配置平均CPU使用率
ASP.NET 菜单 | C# 将动态链接库生成到DEBUG目录 | WIN10 编译的DEBUG程序 WIN7不能运行
@FEIGNCLIENT 动态URL | FREAD函数与FSEEK | REACT NATIVE WEBVIEW 遮住
JSP中JS CSS样式不显示 | GULP-LOAD-PLUGINS 执行 SH | EXT4.1绕过登录