Say Holle World走入Makefile

Makefile系列

所有编程语言的入门都从Holle World开始，这几乎成了惯例，本文也将以输出Holle World走入Makefile的世界

# Makefile所需环境

环境的建立有多种方式：

1. 在Windows上安装Cgywin
2. 在Windows上安装Linux虚拟机
3. 准备一台Linux卡片机，例如树莓派

对环境进行测试，在命令行窗口输入如下命令

make -v

输出

GNU Make 4.1

Built for x86\_64-pc-linux-gnu

Copyright (C) 1988-2014 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

本文的实验环境为在VMware Workstation中建立的ubuntu虚拟机

# 输出Hello World

## 文件编写

现在我们编写一个最简单的Makefile的例子，在命令终端输出Hello World

选择或新建一个目录，在目录中新建一个文件，命名为Makefile，在Makefile文件中输入如下内容

all:

echo "Hello World"

echo前面必须至少有一个TAB键，并且不能出现空格。这种错误很隐蔽，在大多数编辑器中TAB和空格并没有明显区别，所以建议使用可以区分两者的编辑器，比如sublime

## 文件运行

编写完文件之后接下来是运行Makefile，有两种方法运行，一种是在Makefile所在目录运行make命令，第二种是在Makefile所在目录运行make all命令。

ubuntu@liubo:~/my\_project/Makefile\_project/$ make

echo "Hello World"

Hello World

ubuntu@liubo:~/my\_project/Makefile\_project/$ make all

echo "Hello World"

Hello World

到这里我们迈出了万里长征的第一步，我们在命令终端输出了Hello World。

# Makefile的规则概念

规则是Makefile的细胞，不管多么复杂的Makefile都是用规则构成的。

规则的语法格式：

target1,target2... depend1, depend2, ...

command

......

......

每条规则由三个部分组成分别是目标(target),依赖(depend)和命令(command)。

## 目标(target)：

目标可以是object file(目标文件)，也可以是执行文件，还可以是标签（Label）即伪目标。如果目标是文件，对应的规则为生成目标文件的命令；如果目标是伪目标，那么命令可以执行动作，不生成任何文件。目标和命令都可以有多条。目标的名字可以由字母和下划线组成。

## 依赖(depend):

依赖就是要生成那个target所需要的文件或是目标。依赖也称为先决条件，make会在构建目标之前先构建它的依赖。

在规则的命令中可以使用这些依赖。例如：生成可执行文件的目标文件（\*.o）可以作为依赖，如果规则的命令中不需要任何依赖，那么规则的依赖可以为空。如果当前规则中的依赖是其他规则中的某个目标，这样就形成了规则之间的嵌套。依赖可以根据要执行的命令的实际需求，指定很多个。

depend中如果有一个以上的文件比target文件要新的话，command所定义的命令就会被执行。这就是makefile的规则，也是makefile中最核心的内容。

## 命令(command):

命令是当前这条规则需要执行的动作，动作可以是多个，一般情况下这个动作就是一个shell命令，例如：通过某个命令编译文件、生成库文件、进入目录等。command前面必须至少有一个TAB键缩进，不能出现空格，并且独占占一行

## Hello World规则说明

以一下例子说明

all:

echo "Hello World"

这里all为目标，all并不是需要生成的文件名，是伪目标；depend被省略了；echo "Hello World"为command，是一条shell命令

# 对Hello World的改造

## 最初版本

all:

echo "Hello World!"

运行结果

ubuntu@liubo:~/my\_project/Makefile\_project/$ make

echo "Hello World"

Hello World

ubuntu@liubo:~/my\_project/Makefile\_project/$ make another

make: \*\*\* No rule to make target 'another'. Stop.

语法总结：

1、如果make的目标不存在，结果会报错，make: \*\*\* No rule to make target 'another'. Stop.

## 增加另一个规则

如下：

all:

echo "Hello World!"

another:

echo "show another"

运行结果

ubuntu@liubo:~/my\_project/Makefile\_project/$ make

echo "Hello World!"

Hello World!

ubuntu@liubo:~/my\_project/Makefile\_project/$ make another

echo "show another"

show another

语法总结：

1、在一个Makefile文件中可以定义多个规则

2、如果不指明目标make会执行默认目标

3、需要执行非默认规则执行make + 目标

## 更改规则顺序

如下：

another:

echo "show another"

all:

echo "Hello World!"

运行结果

ubuntu@liubo:~/my\_project/Makefile\_project/$ make

echo "show another"

show another

语法总结：

1、默认规则为Makefile文件第一个规则，和名字无关

## 涉及的面试问题

make和Makefile的关系是什么？

make是编译项目工程的工具，Makefile是被make解析的文件。输入make命令后查找当前目录的MAkefile文件，然后进行解析并依据解析结果进行编译工作。

## 参考资料

版权声明：本文参考了其他CSDN博主的文章，遵循CC 4.0 BY-SA版权协议，现附上原文出处链接及本声明。

1. Makefile入门一、helloworld 原文链接：https://blog.csdn.net/hjxu2016/article/details/100672126
2. https://www.cnblogs.com/qytan36/archive/2010/05/25/1743955.html