Makefile学习笔记

写在开始：我还是喜欢论文的排版方式，内容结构清晰，每部分内容存在合理，减少赘述

参考：

先验知识：

摘要：

1 概述

1.1 场景与需求

一个C/C++工程中包含大量存放在若干目录下的.h、.c和.cpp等文件。

各个文件需要通过编译和链接形成可执行文件。编译过程的功能为检查语法正确性、函数和变量等是否被声明，该过程会生成大量中间的.o文件；链接过程将各函数和全局变量等进行链接，即将各.o文件进行链接，同时为了方便，还可将中间件打包为.lib库文件，或.a文件。

同时我们还要满足以下需求：

1. 可重用性：在开发和测试等过程中会对工程中的文件进行多次编译和链接
2. 易用性：编译和链接规则足够简化
3. 全面覆盖性：编译和链接过程应覆盖所有需要的文件
4. 选择覆盖性：已经被编译过的.c/.cpp文件被修改后，只需要重新编译被修改的.c/.cpp文件即可；已经被编译过的.h文件被修改后，只需要重新编译引用了该.h文件的.c/.cpp文件即可。

1.2 已有解决方案

Windows环境下的很多IDE已经能够自动实现编译过程，但Unix环境下却仍只能用Makefile。

1.3 Makefile

Makefile以文件的形式存储，让用户可以在文件中指定规则，编译器可按照这些规则对程序进行编译。

2 Makefile

2.1 可重用性

目的：要完成从多个.c、.cpp和.h转换为一个可执行文件，即实现从prerequisites->target的映射。

* target: 目标的可执行文件
* prerequisites: 生成target所需要的源文件
* command: 编译和链接过程需要执行的命令

Makefile的语法如下：

|  |  |  |
| --- | --- | --- |
|  | target ... : prerequisites ...  command  ...  ... |  |

Makefile的规则为：Makefile执行时会先获取各文件的最近修改日期，如果prerequisites中存在一个及以上文件存在更改时，command将会被执行

Makefile的实现过程为：使用命令行模式，在包含Makefile文件的当前目录下执行make指令，该指令将实现以下功能：

1. 查找target文件：a)如果找到：将其作为最终目标文件；b)如果未找到：生成target文件
2. 查找target依赖的prerequisites文件（由.c、.cpp、.h编译生成的.o文件）：a)如果找到：链接生成target，结束；b)如果未找到：生成.o文件

分析：Makefile执行的过程采用递归的思想，从tar一次推回到.c、.cpp、.h文件

示例：

* target: tar
* prerequistes: preq\_1.c, preq\_2.c, preq\_3.c; preq\_13.h

Makefile文件结构：

* part1: .o -> tar
* part2: .c .h-> .o
* part3: clean

实现过程为：

|  |  |  |
| --- | --- | --- |
|  | tar : preq\_1.o preq\_2.o preq\_3.o  cc -o tar preq\_1.o preq\_2.o preq\_3.o  preq\_1.o: preq\_1.c preq\_13.h  cc -c pre\_1.o  preq\_2.o: preq\_2.c  cc -c pre\_2.o  preq\_3.o: preq\_3.c preq\_13.h  cc -c preq\_3.o  clean:  rm tar preq\_1.o preq\_2.o preq\_3.o |  |

2.2 易用性

2.1.1 变量法

场景：当某个文件更改后，需要对Makefile多处进行更改

解决方案：变量法

声明方式：Objects = preq\_1.o preq\_2.o preq\_3.o

使用方式：$(Objects)

分析：该种方式与宏定义方法思想类似

示例：

|  |  |  |
| --- | --- | --- |
|  | Objects = preq\_1.o preq\_2.o preq\_3.o  Target = tar  CC = cc  $(Target) : $(Objects)  $(CC) -o $(Target) $(Objects)  preq\_1.o: preq\_1.c preq\_13.h  cc -c pre\_1.o  preq\_2.o: preq\_2.c  cc -c pre\_2.o  preq\_3.o: preq\_3.c preq\_13.h  cc -c preq\_3.o  clean:  rm $(Target) $(Objects) |  |

2.2.2 自动关联法

场景：由于.o文件作为中间件，我们的最终目的是得到target，因此可采用同名的方法，自动实现name.c到name.o文件的关联

特例：该种方式可导致命令clean ：出现歧义，因此引入.HPONY声明clean是伪目标文件

示例：

|  |  |  |
| --- | --- | --- |
|  | Objects = preq\_1.o preq\_2.o preq\_3.o  Target = tar  CC = cc  $(Target) : $(Objects)  $(CC) -o $(Target) $(Objects)  preq\_1.o : preq\_13.h  cc -c pre\_1.o  preq\_2.o :  cc -c pre\_2.o  preq\_3.o : preq\_13.h  cc -c preq\_3.o  .PHONY : clean  clean :  rm $(Target) $(Objects) |  |

总结：因此，Makefile共包含以下五个部分：

1. 显式规则：指定依赖文件、目标文件之间的关系
2. 隐晦规则：Makefile执行时自动推导的规则
3. 变量定义
4. 注释：#行注释
5. 文件指示：在一个Makefile中引用另一个Makefile

2.2.3 默认文件法

默认情况下，执行make命令时会主动识别Makefile、makefile和GNUmakefile（.mk文件）文件

2.3 全面覆盖性