

Makefile

- 一个工程中的源文件不计其数，其按类型、功能、模块分别放在若干个目录中，
Makefile 文件定义了一系列的规则来指定哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，因为 Makefile 文件就像一个 Shell 脚本一样，也可以执行操作系统的命令。
- Makefile 带来的好处就是“自动化编译”，一旦写好，只需要一个 make 命令，整个工程完全自动编译，极大的提高了软件开发的效率。make 是一个命令工具，是一个解释 Makefile 文件中指令的命令工具，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。

■ 文件命名

makefile 或者 Makefile

■ Makefile 规则

□ 一个 Makefile 文件中可以有一个或者多个规则

目标 ...: 依赖 ...

命令 (Shell 命令)

...

- 目标：最终要生成的文件（伪目标除外）
- 依赖：生成目标所需要的文件或是目标
- 命令：通过执行命令对依赖操作生成目标（命令前必须 Tab 缩进）

□ Makefile 中的其它规则一般都是为第一条规则服务的。

- 命令在执行之前，需要先检查规则中的依赖是否存在
 - 如果存在，执行命令
 - 如果不存在，向下检查其它的规则，检查有没有一个规则是用来生成这个依赖的，如果找到了，则执行该规则中的命令
- 检测更新，在执行规则中的命令时，会比较目标和依赖文件的时间
 - 如果依赖的时间比目标的时间晚，需要重新生成目标
 - 如果依赖的时间比目标的时间早，目标不需要更新，对应规则中的命令不需要被执行

■ 自定义变量

变量名=变量值 var=hello

`$(var)`

■ 预定义变量

AR : 归档维护程序的名称, 默认值为 ar

CC : C 编译器的名称, 默认值为 cc

CXX : C++ 编译器的名称, 默认值为 g++

\$@ : 目标的完整名称

\$< : 第一个依赖文件的名称

\$^ : 所有的依赖文件

app:main.c a.c b.c

gcc -c main.c a.c b.c

#自动变量只能在规则的命令中使用

app:main.c a.c b.c

`$(CC) -c $^ -o $@`

■ 获取变量的值

`$(变量名)`

```
add.o:add.c
```

```
gcc -c add.c
```

```
div.o:div.c
```

```
gcc -c div.c
```

```
sub.o:sub.c
```

```
gcc -c sub.c
```

```
mult.o:mult.c
```

```
gcc -c mult.c
```

```
main.o:main.c
```

```
gcc -c main.c
```

```
%.o:%.c
```

- %: 通配符, 匹配一个字符串
- 两个%匹配的是同一个字符串

```
%.o:%.c
```

```
gcc -c $< -o $@
```

■ `$ (wildcard PATTERN...)`

- ❑ 功能：获取指定目录下指定类型的文件列表
- ❑ 参数：PATTERN 指的是某个或多个目录下的对应的某种类型的文件，如果有多个目录，一般使用空格间隔
- ❑ 返回：得到的若干个文件的文件列表，文件名之间使用空格间隔
- ❑ 示例：

```
$ (wildcard *.c ./sub/*.c)
```

返回值格式：a.c b.c c.c d.c e.c f.c

■ `$(patsubst <pattern>,<replacement>,<text>)`

□ 功能：查找<text>中的单词(单词以“空格”、“Tab”或“回车”“换行”分隔)是否符合模式<pattern>，如果匹配的话，则以<replacement>替换。

□ <pattern>可以包括通配符`%`，表示任意长度的字串。如果<replacement>中也包含`%`，那么，<replacement>中的这个`%`将是<pattern>中的那个%所代表的字串。(可以用`\\`来转义，以`\\%`来表示真实含义的`%`字符)

□ 返回：函数返回被替换过后的字符串

□ 示例：

```
$(patsubst %.c, %.o, x.c bar.c)
```

返回值格式：x.o bar.o



牛客大学

- 专业求职辅导 -

THANKS



关注【牛客大学】公众号
回复“牛客大学”获取更多求职资料