makefile 语法

makefile 基本应用

示例：

```
proc:proc.c
    gcc proc.c -o proc
.PHONY:clean
clean:
    rm -f proc
```

1. **[目标文件]：[依赖项1][依赖项2].....**
   tab + 命令
   解释：依赖项是对生成目标文件的限制，当依赖项不全存在时，**一般**无法生成目标文件
2. 生成目标文件**proc**：**make proc**
3. 如果使用**make**,且后面不加文件名，默认生成**makefile**文件里的第一个目标文件。
4. **.PHONY:【目标文件】**
   使目标文件避免时间检查，保证对应命令一定执行
   【举个例子】：当 **ModifyTime(proc) > ModifyTime(proc.c)** 此时**proc**是用最新的**proc.c**编译的，无需重复生成，若使用`gcc proc.c -o proc`，将不执行该命令
   【使用场景】：一般希望**清理一定执行**

makefile 定义变量

```
bin=proc
src=proc.c
$(bin):$(src)
    gcc $(src) -o $(bin)
.PHONY:clean
clean:
    rm -f $(bin)
```

1. **[变量名]=[文件名]**
2. **$(变量名)**：表示文件名

makefile 高级应用

$语法

```
bin=proc
src=proc.c
$(bin):$(src)
    gcc $^ -o $@
```

```
.PHONY:clean
clean:
    rm -f $(bin)
```

1. **$^**：表达所有依赖项
2. **$@**：表达目标文件

makefile 一次性编译多个文件(每个文件单独形成一个程序)

知识点：若需要生成的目标文件的依赖项不存在，**make** 会尝试利用**makefile**文件里的规则生成依赖项。这个规则是递归式定义的，即当需要生成的依赖项作为目标文件时，其对应的依赖项不存在，会重复上述操作。

```
test:proc proc1

proc:proc.c
        gcc proc.c -o proc
proc1:proc1.c
        gcc proc1.c -o proc1
```

使用`make test` 将会触发

1. `gcc proc.c -o proc`
2. `gcc proc1.c -o proc1`

解释：**test**的依赖项**proc proc1** 不存在，尝试生成这两个文件，**makefile**中有这两个文件的生成方法，然后**makefile**尝试使用，则触发这两条命令。