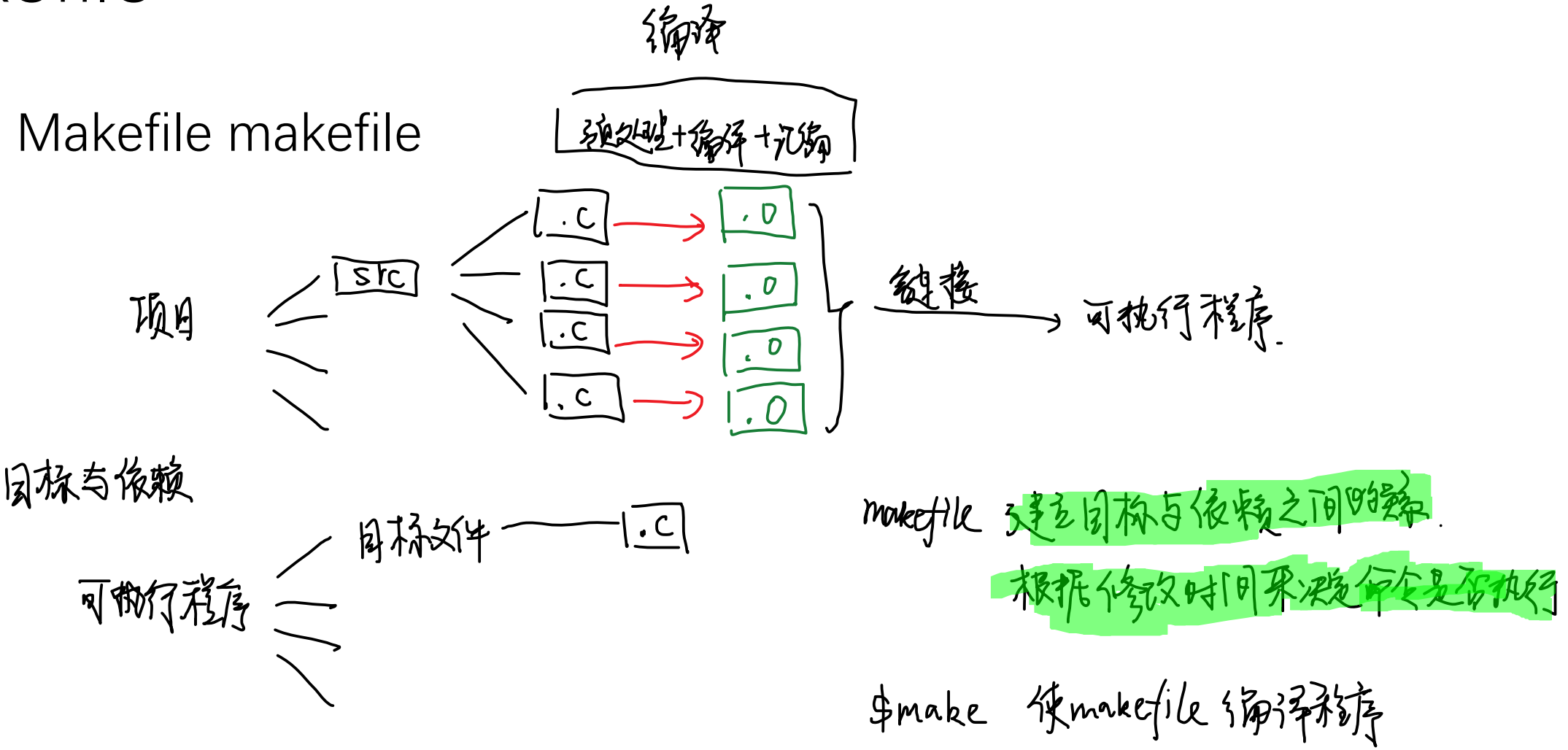
Makefile

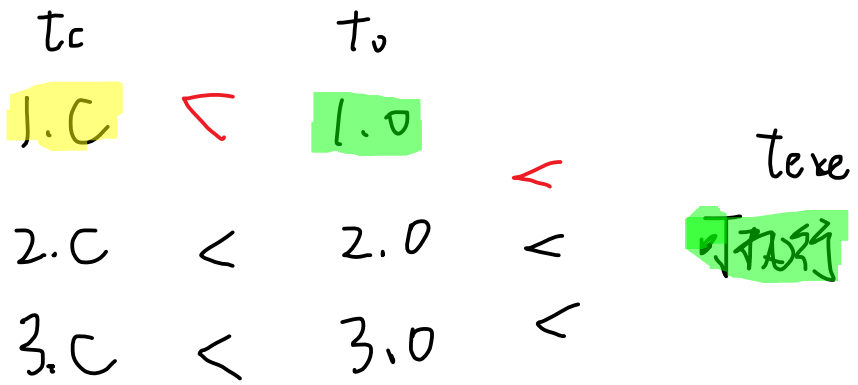
- 执行模式:　增量编译
- 规则：目标文件 依赖文件 命令
- 伪目标
- 变量
- 通配符 模式匹配
- 内置函数
- 循环结构

Makefile

- 文件 Makefile makefile



增量编译

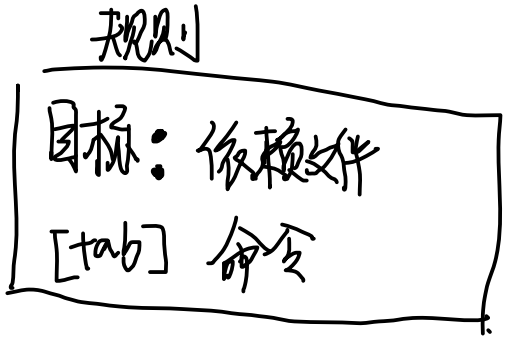


$t_c < t_0 < t_{exe}$

当修改1.C 以后.

$t_0 < t_{exe} < t_c$

规则

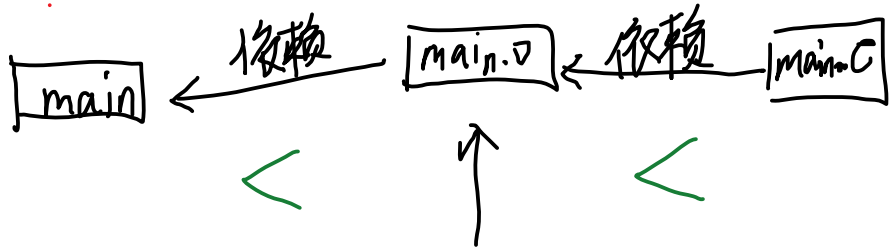


1. 命令是否执行. 取决目标与依赖的时间关系.

2. 命令可以写多个.

makefile

```
1 main:main.o
2     gcc main.o -o main
3 main.o:main.c
4     gcc -c main.c -o main.o
```



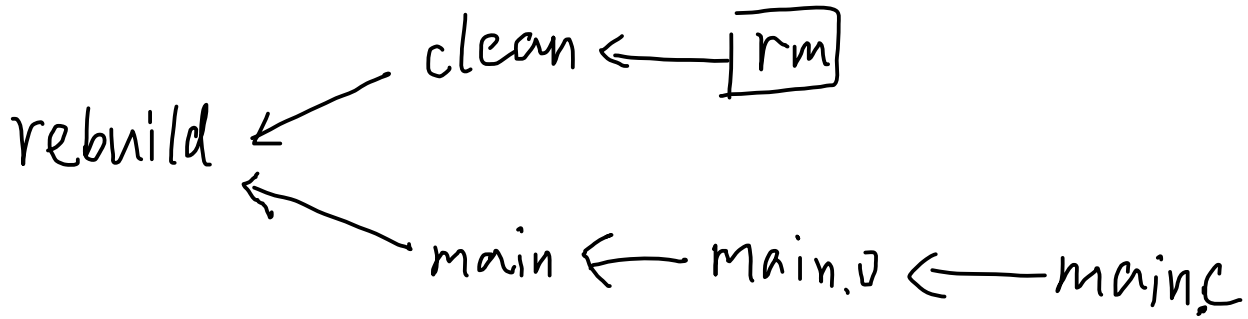
$make 目标名

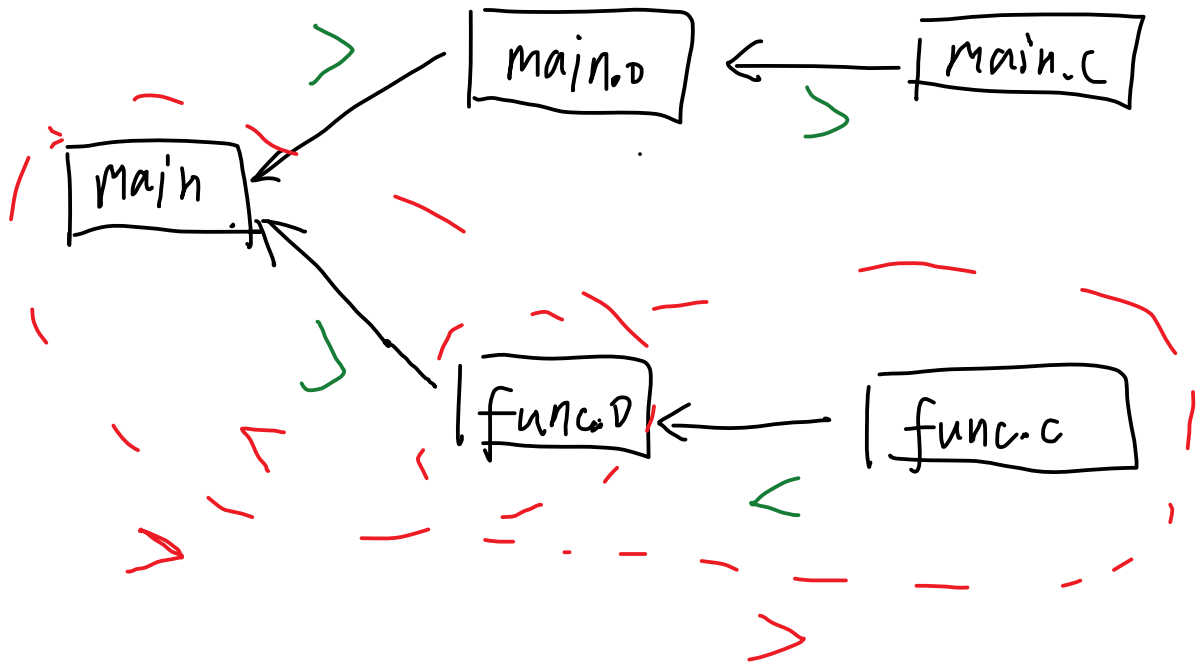
伪目标：这个目标不是文件

makefile

```
1 main:main.o
2     gcc main.o -o main
3 main.o:main.c
4     gcc -c main.c -o main.o
5 rebuild:clean main
6 clean:
7     rm -rf main.o main
~
```

$make rebuild.






```
1  main:main.o func.o
2      gcc main.o func.o -o main
3  main.o:main.c
4      gcc -c main.c -o main.o
5  func.o:func.c
6      gcc -c func.c -o func.o
7  .PHONY:clean rebuild
8  rebuild:clean main
9  clean:
10     rm -rf main.o main
```

变量

字符串替换

A = b 运行时替换

A := b 定义时替换

$(A) 取变量

```
1  out := main.exe
2  $(out):main.o func.o
3      gcc main.o func.o -o $(out)
4  main.o:main.c
5      gcc -c main.c -o main.o
6  func.o:func.c
7      gcc -c func.c -o func.o
8  .PHONY:clean rebuild
9  rebuild:clean $(out)
10 clean:
11     rm -rf main.o $(out)
```

预定义变量

变量名	功能	默认含义
AR	打包库文件	ar
AS	汇编程序	as
CC	C编译器	cc
CPP	C预编译器	$(CC) -E
CXX	C++编译器	g++
RM	删除	rm –f
ARFLAGS	库选项	无
ASFLAGS	汇编选项	无
CFLAGS	C编译器选项	无
CPPFLAGS	C预编译器选项	无
CXXFLAGS	C++编译器选项	无

自动变量

$@ 目标文件

$< 第一个依赖文件

$^ 所有依赖文件

$? 新于目标的依赖文件（修改比较晚）

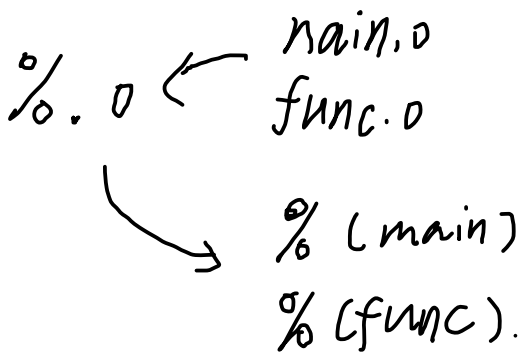
$(@D) 目标文件的目录部分

$(@F) 目标文件的文件名部分

} 某一个规则内

匹配字符%

objs = main.o func.o



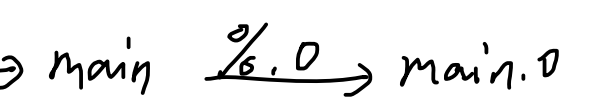
%可以匹配字符串的子串

```
1  CC := gcc
2  out := main.exe
3  objs := main.o func.o
4  $(out):$(objs)
5      $(CC) $^ -o $@
6  %.o:%.c
7      $(CC) -c $^ -o $@
8  .PHONY:clean rebuild
9  rebuild:clean $(out)
10 clean:
11     rm -rf $(objs) $(out)
```

如果%在规则的目标中，对上一个依赖文件进行匹配

匹配字符%

```
1 CC := gcc
2 out := main.exe
3 srcs := main.c func.c new.c
4 objs := $(srcs:%.c=%.o)
5 $(out):$(objs)
6     $(CC) $^ -o $@
7 %.o:%.c
8     $(CC) -c $^ -o $@
9 .PHONY:clean rebuild
10 rebuild:clean $(out)
11 clean:
12     rm -rf $(objs) $(out)
~
```



func → func.o

new → new.o

内置函数

wildcard 对文件系统使用通配符.

$(函数名 空格 参数1,参数2)

subst from,to,text.

文本替换.

```
var := "how are you"
newvar := $(subst you,me,$(var))
```

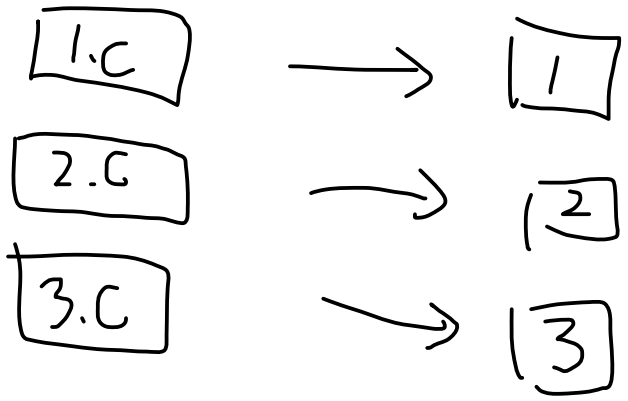
```
how are me
```

patsubst pattern,replacement,text.

循环

```
1 LIST = one two three
2 all:
3     @for i in $(LIST); do echo $$i; done
```

实例




```
1 CC = gcc
2 srcs = $(wildcard *.c)
3 # 1.c 2.c 3.c
4 targets = $(patsubst %.c,%,$(srcs))
5 # 1 2 3
6 all:$(targets)
7     @for i in $(targets); do $(CC) -o $$i $$i.c; done
8 .PHONY:clean rebuild
9 rebuild:clean all
10 clean:
11     rm -f $(targets)
```