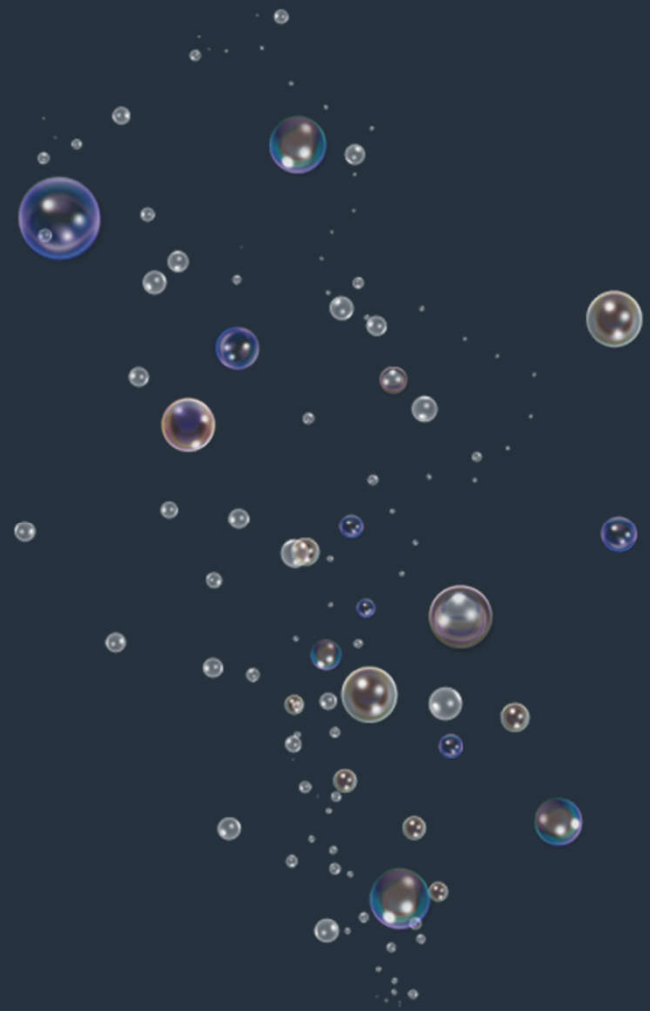


makefile设计

大连理工大学 赖晓晨

内容简介

- 隐含规则
- 生成多个文件
- makefile嵌套
- 使用变量
- 使用函数

隐含规则

- 什么是隐含规则

  一些经常使用而且使用频率很高的，事先已经约定好，不需显式书写出的规则。隐含规则是一种惯例，make会按照这种“惯例”心照不宣的运行。

隐含规则举例

```
foo : foo.o bar.o
        gcc -o foo foo.o bar.o
```

```
foo.o : foo.c
        gcc -c foo.c $(CFLAGS)
bar.o : bar.c
        gcc -c bar.c $(CFLAGS)
```

禁止make使用任何隐含规则：-r

make/hide/makefile

同时生成多个可执行文件

```
.PHONY: all
all: p1 p2 p3
p1:p1.c
        gcc p1.c -o p1
p2:p2.c
        gcc p2.c -o p2
p3:p3.c
        gcc p3.c -o p3
```

```
#include <stdio.h>   //p1.c
int main()
{
        printf("this is p1.\n");
        return 0;
}
```

make/prog/makefile

命令出错

- make运行时，make会检测每个命令的执行的返回码，如果命令返回成功，make会执行下一条命令，否则make终止。
- -：在makefile的命令行前加一个减号，此时不管命令是否出错，都认为是成功的。

命令出错

```
.PHONY: all
all: p1 p2 p3
p1:p1.c
        -gcc p1.c -o p1
p2:p2.c
        -gcc p2.c -o p2
p3:p3.c
        -gcc p3.c -o p3
```

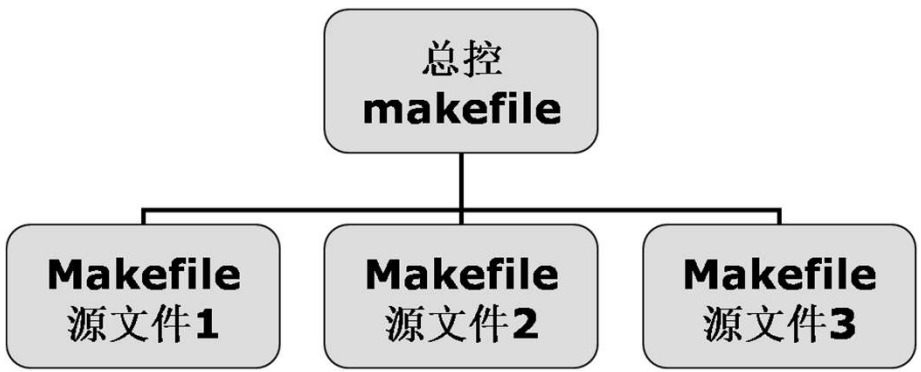
```
#include <stdio.h>   //p1.c
int main()
{
        printf("this is p1.\n")
        return 0;
}
```

缺少分号

make/prog2/makefile

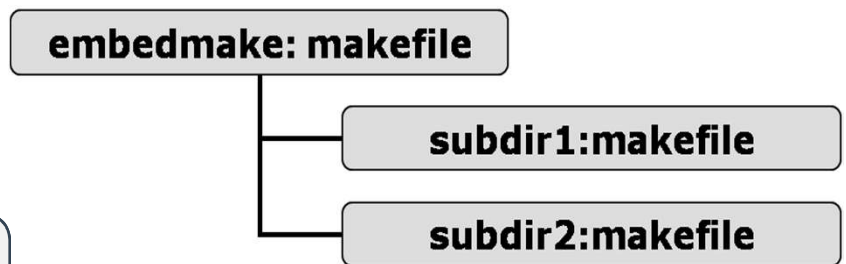
make嵌套执行

- make嵌套的含义，目录树
- 总控makefile



make嵌套执行举例

- 目录结构
- 总控makefile




```
.PHONY: both
both: a b c
a:
	@gcc a.c -o a
b:
	@cd subdir1;make
c:
	@cd subdir2;make
```

make\embedmake\makefile

make嵌套执行举例

- subdir1/makefile：

```
b:b.c
	@gcc b.c -o b
```

- subdir2/makefile

```
c:c.c
	@gcc c.c -o c
```

make\embedmake\makefile

使用变量

- makefile中可以使用变量
- 变量类似于C语言的宏，但值可修改
- 变量名大小写敏感
- 变量名不应该包含:#=或空格
- 变量使用时用$(var)形式

变量（两种方式）

- 递归展开变量：使用“=”表示，可先使用，后定义。弊病：有可能递归定义，导致无限展开。

```
foo = $(bar)
bar = $(ugh)
ugh = Huh?
all:
        echo $(foo)
```

```
a = $(b)
b = $(a)
all:
        echo $(a)
```

- 简单展开变量：使用“:=”表示，须先定义再使用。

```
x:=foo
y:=$(x) bar
x:=later
all:
        @echo $(y)
```

```
y:=$(x) bar
x:=foo
all:
        @echo $(y)
```

其他变量

- Makefile中，除了可使用用户自定义变量之外，还可以使用以下变量：
  - 系统环境变量
  - 自动化变量
  - 预定义变量

环境变量

- make运行时的系统环境变量可以在make开始运行时被载入到makefile中。

```
all:
        @echo $(PATH)
```

- 但是如果makefile定义了这个变量，则环境变量的值被覆盖（除非make运行时指定了-e参数）

```
PATH :=  " home"
all:
        @echo $(PATH)
```

自动化变量

- $@：表示规则中的目标文件集

```
hello : a.c b.c c.c
        @echo $@
```

- $<：表示依赖目标中的第一个目标名字。

```
hello : a.c b.c c.c
        @echo $<
```

- 其他自动化变量：

$%、$?、$^、$*、$+

make/let/m7、m8

预定义变量

- AR：归档维护程序，默认为ar
- AS：汇编程序，默认为as
- CC：C编译程序，默认为cc
- CPP：C预处理程序， 默认为cpp
- RM：文件删除程序，默认为：rm –f

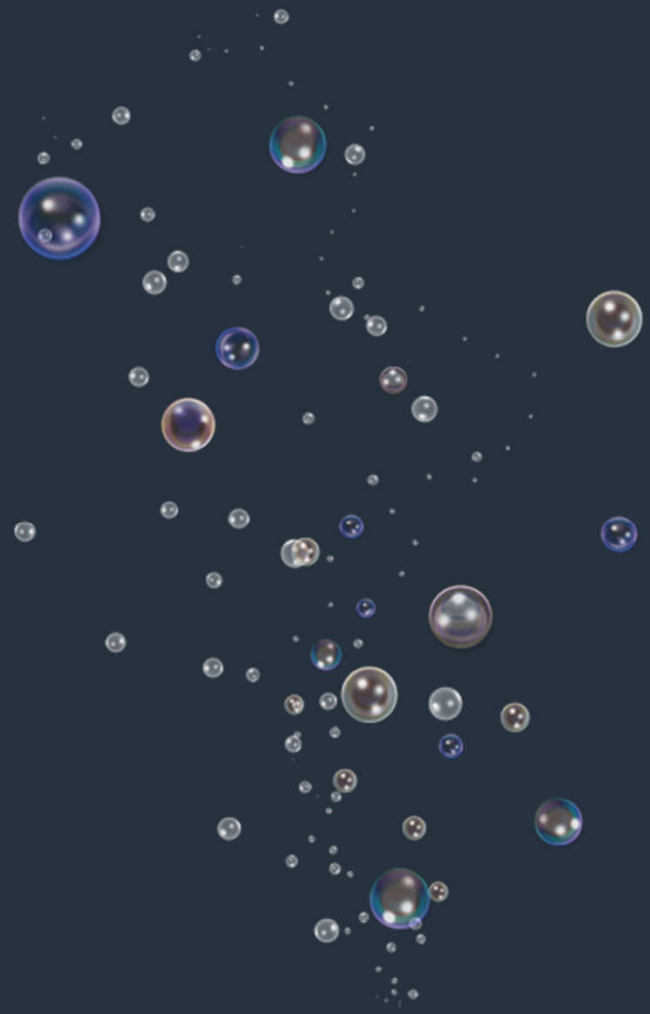
函数

makefile支持的函数语法如下：

$(function arg1,arg2,...)

```
comma:=,
empty:=
space:=$(empty) $(empty)
foo:=a b c
bar:=$(subst $(space),$(comma),$(foo))
all:
	@echo $(bar)
```

字符串替换



嵌入式软件设计

大连理工大学 赖晓晨