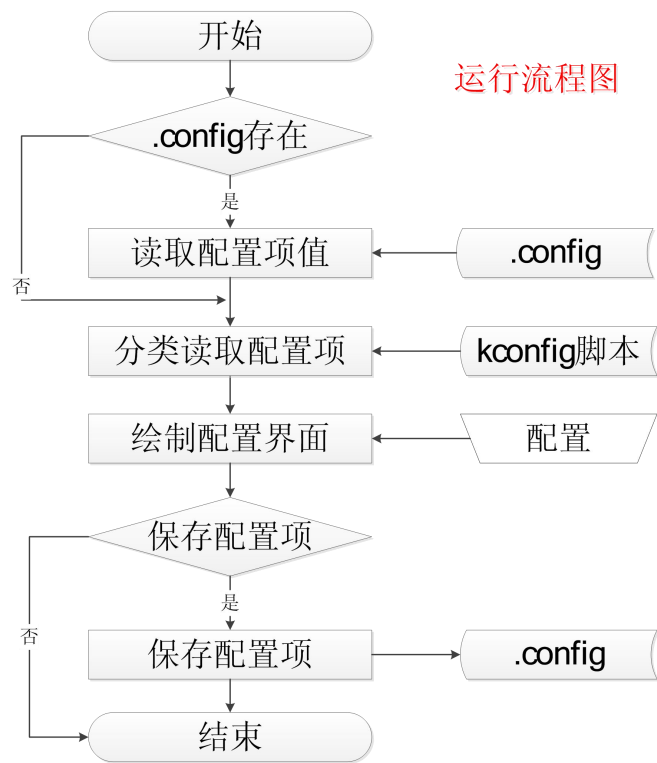


# Kconfig

Kconfig 的功能是实现对 Kernel 模块的分类和描述，它帮助 make menuconfig 命令实现 Kernel 的图形化配置。

■ 运行流程

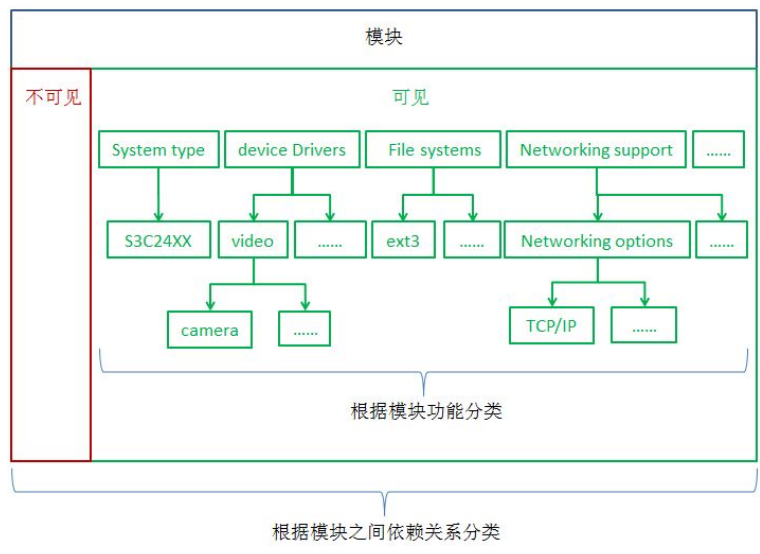


【图 B- 1】Kconfig 运行流程图

[说明]

- A. 用户根据 Kconfig 显示的图形菜单来修改模块配置，然后将修改后的配置以配置项的形式保存到.config 文件中
- B. Kconfig 脚本用来描述模块的分类和配置

■ 模块分类原理

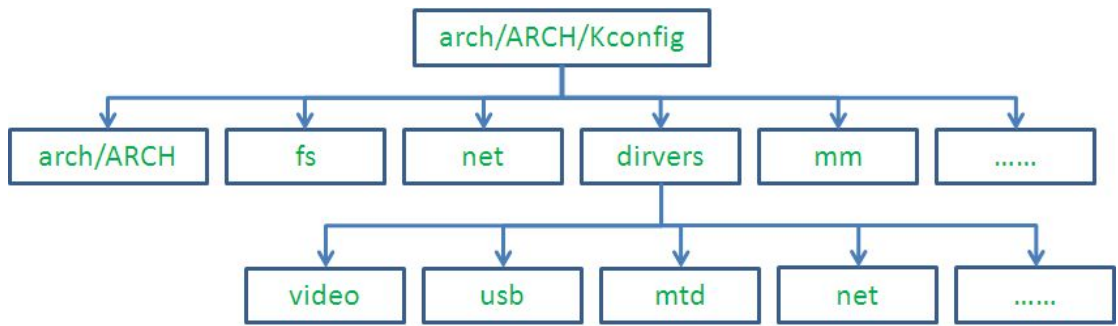


[说明]

Kconfig 对模块分类的方法为：

- [1] 根据模块之间的依赖关系将模块分成可见模块和不可见模块两大类
- [2] 根据模块功能将可见模块划分成若干类，并且使用分类菜单来描述这些类之间的关系

■ Kconfig 工作原理



[说明]

- [1] Kconfig 解析从 arch/ARCH/Kconfig 脚本开始
- [2] arch/ARCH/Kconfig 包含其他目录的 Kconfig，具体包含见 arch/ARCH/Kconfig 文件内容

■ Kconfig 脚本

脚本由配置项和分类菜单组成，配置项和分类菜单都使用一系列属性来描述，下表描述配置项、分类菜单和属性之间的关系：

| 分类菜单/配置项         | 属性                           |
|------------------|------------------------------|
| config           | 名称、类型、输入提示、数值范围、默认值、依赖、选择和帮助 |
| menu/endmenu     | 输入提示和依赖                      |
| menuconfig       | 名称、类型、输入提示、默认值、依赖、选择和帮助      |
| choice/endchoice | 名称、输入提示、默认值、optional 和依赖     |

|          |    |
|----------|----|
| if/endif | 依赖 |
| comment  | 帮助 |
| source   |    |

# 1. 依赖表达式

操作数 + 操作符

✓ 操作数

配置项(bool 或 tristate)、y(2)、m(1)、n(0)

✓ 操作符

| 操作符 | 说明                                       | 优先级 |
|-----|--|-----|
| =   | 表达式 1 = 表达式 2<br>两个表达式相等，结果为 y，不相等结果为 n  | 6   |
| !=  | 表达式 1 != 表达式 2<br>两个表达式相等，结果为 n，不相等结果为 t | 5   |
| ()  | (表达式)<br>提升表达的优先级                        | 4   |
| !   | ! 表达式<br>结果为 2 - 表达式                     | 3   |
| &&  | 表达式 1 && 表达式 2<br>结果为表达式 1 和表达 2 的最小值    | 2   |
|     | 表达式 1    表达式 2<br>结果为表达式 1 和表达 2 的最大值    | 1   |

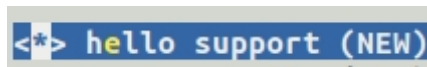
✓ 例

(CONFIG\_TEST1 = y) && CONFIG\_TEST2

## 2. 配置项脚本

```
config HELLO
tristate
prompt "hello support"
default y
depends on MODULES && ARM
select HELLO_TEST
help
    "Compile hello.c"
```

显示效果



是一个配置项描述脚本，脚本的每行描述一个属性，用关键字来说明描述的属性

### [1] 名称

配置项的名字，必须以“CONFIG\_”打头

#### ✓ 语法

“config” <标识符>

config 是关键字，表示配置项的开始；标识符是省略了“CONFIG\_”的配置项名称；

#### ✓ 例

例子中的配置项名称为“CONFIG\_HELLO”

### [2] 类型

配置项的类型，用来定义配置项的值域

#### ✓ 语法

<type>

用一个类型关键字表示

| type     | 值域              |
|----------|-----------------|
| bool     | y(静态编译)和 n(不选择) |
| tristate | y、m（动态加载）和 n    |
| string   | ASCII 码串        |
| hex      | 16 进制数字字符串      |
| int      | 10 进制字符串        |

### [3] 输入提示

用来在图形配置界面中代表配置项

#### ✓ 语法

"prompt" <提示字符> ["if" <依赖表达式>]

prompt 是关键字，表示后面写的是提示字符串；如果想在一定条件下才显示输入提示属性，可以加 if 依赖表达式；输入提示项可以直接放在类型后面，下面两个是等价的：

```
bool "Networking support"
```

和  
bool  
prompt "Networking support"

- ✓ 例  
例子中图形配置界面显示 hello support 来表示这个配置项

#### [4] 数字范围

当类型为 int 或 hex 时，用于限定配置项的值

- ✓ 语法  
"range" <symbol1> <symbol2> ["if" <依赖表达式>]  
symbol1 和 symbol2 只能是 int 和 hex 类型；配置项的值应该是大于或等于 symbol1，小于或等于 symbol2；如果想条件显示数值范围，可以加 if 依赖表达式

- ✓ 例  
config TEST2  
    int "Number of buffers(0-5)"  
    range 0 5  
表示这个配置项的值必须在 0 到 5 之间

#### [5] 默认值

在加载的.config 文件中不存在本配置项，且用户也从来没有设置过新值时，使用默认值，用户的任何改动会覆盖默认值

- ✓ 语法  
"default" <值> ["if" <依赖表达式>]  
配置选项可以有多个默认值，但是当多个默认值有效时，第一个定义的默认值被使用；默认值并不是只限于应用在定义他们的菜单选项；有可能定义的菜单选项已经在前面定义过默认值，在这里定义会被以前的值覆盖；如果想给默认值加一个选择条件，可以加 if 依赖表达式

- ✓ 例  
例子中 CONFIG\_HELLO 的默认值是 y

#### [6] 类型+默认值

类型和默认值两个可以合并书写

- ✓ 语法  
"def\_bool"/"def\_tristate" <值> ["if" <依赖表达式>]

- ✓ 例  
config TEST3  
    def\_bool y  
    prompt "test3 Kconfig"

#### [7] 依赖

依赖值用于决定配置项是否可见，依赖值为 y 时，配置项可见且类型和值域没有任何限制；依赖值为 m 时，配置项可见但是值只能为 m 和 n；依赖值为 n 时配置项不可见

- ✓ 语法  
“depends on” <值>  
值只能是 y、m 和 n
- ✓ 例  
例子中的 depends on MODULES && ARM 表示 MODULES 和 ARM 模块同时可见时，该配置项才可见

#### [8] 选择

用于强制某个配置项值为 y

- ✓ 语法  
"select" <symbol> ["if" <依赖表达式>]  
<symbol>只能是 “bool”和 “tristate”类型
- ✓ 例  
例子中 select HELLO\_TEST 表示该配置项值为 y 时，强制 CONFIG\_HELLO\_TEST 配置项的值为 y

#### [9] 帮助

图形界面中显示的帮助信息

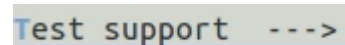
- ✓ 语法  
"help" or "---help---"  
缩进小于 help 关键字的行是帮助结束行
- ✓ 例  
例子中  
help  
"Compile hello.c"  
是帮助信息

### 3. 分类菜单

脚本

```
menu "Test support"
    depends on MODULES
    配置项或分类菜单
endmenu
```

显示效果



#### [1] 提示符

菜单标题，箭头前面的字符串

- ✓ 语法  
“menu” <提示符>  
menu 是分类菜单的关键字，用于标识分类菜单的开始位置，endmenu 用来和它配对表示分类菜单结束位置，他们中间的元素是显示在这个菜单里面的的内容

[2] 依赖

和配置项功能及语法一样，当依赖表达式的值为 y 或 m 时，整个菜单项可见，否则整个菜单不可见，当然它里面的配置项或者菜单项也不可见

4. 配置项菜单

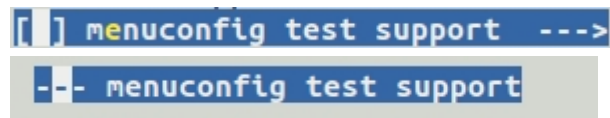
配置项菜单是一个配置和菜单组成，它从 `menuconfig` 关键字行开始，到脚本文件结束行为止，这中间的元素是显示在这个配置菜单的内容

脚本

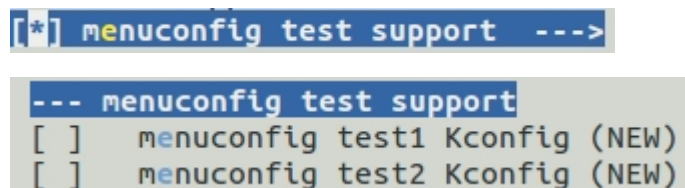
```
menuconfig TEST_MC
    bool "menuconfig test support"
    default n
    配置项或分类菜单
```

显示效果

未选中



选中



[1] 名称

配置项菜单中的配置项的名字

✓ 语法

“menuconfig” <标识符>

`menuconfig` 是关键字；标识符是省略了 “CONFIG\_” 的配置项菜单的名称；

✓ 例

例子中的配置项名称为 “CONFIG\_TEST\_MC”

[2] 类型

语法跟配置项相同，但是类型只能是 `bool` 和 `tristate`

[3] 其他

其他属性都跟配置项一样

选择

5. 选择菜单

从多个配置选项里面选择一个让它的值为 y，或者让多个配置选项的值同时为 m；选择菜单里面的配置项类型要一样，且只能为 `bool` 和 `tristate` 类型；

脚本

```
choice
    prompt "test choice"
    default CONFIG_TEST_C1
```

```

        help
            "test choice"

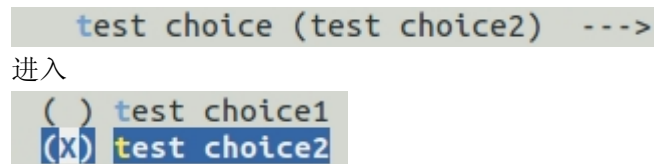
    config TEST_C1
        bool "test choice1"
        help
            "test choice1"

    config TEST_C2
        bool "test choice2"
        help
            "test choice2"

endchoice

```

显示效果



```

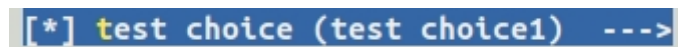
test choice (test choice2) --->
( ) test choice1
(X) test choice2

```

#### [1] optional

本身是一个关键字，加了它表示选择菜单中可以不选择任何配置项，否则至少要选择一项配置项，显示图：

选中



```

[*] test choice (test choice1) --->

```

没有选中



```

[ ] test choice

```

#### [2] 其它

其它属性和配置项一样

### 6. if/endif

根据依赖表达式的值来决定脚本是否有效  
脚本

```

if HELLO = y
    config TEST_C1
        bool "test choice1"
        help
            "test choice1"
    endif

```

显示效果

CONFIG\_HELLO 配置项选中



```

<*> hello support
< > test1 Kconfig (NEW)
< > test2 Kconfig (NEW)
    *** TEST is disable ***
< > test5 Kconfig (NEW)
[ ] test choice1 (NEW)
< > test choice two

```

没有选中

```

< > hello support
< > test1 Kconfig (NEW)
< > test2 Kconfig (NEW)
    *** TEST is disable ***
< > test5 Kconfig (NEW)
< > test choice two

```

#### [1] 依赖

例子中  $HELLO = y$  就是依赖表达式，表达式值为  $y$  时，中间的脚本执行；同时还可以加一个依赖属性，属性语法同配置项一样。

### 7. comment

在图形界面里面显示一行注释，通常的用法如下：

脚本

```

comment "TEST is disable"
    depends on !HELLO

```

显示效果

```

< > hello support
< > test1 Kconfig (NEW)
< > test2 Kconfig (NEW)
    *** TEST is disable ***
< > test5 Kconfig (NEW)
< > test choice two

```

#### A. 依赖

用于决定这个注释是否可以显示，语法同配置项

### 8. source

用 `source` 后面指定的文件内容替换 `source` 语句，相当于预处理命令里的头文件包含，`Kconfig` 工具主要采用文件包含的方式，来将各级目录下面的所有的 `Kconfig` 脚本文件合并成一个 `Kconfig` 脚本，然后显示成图形配置界面

```
source "drivers/test/Kconfig"
```