

目 录

目 录

1. 概 述	1
1.1. Config	2
1.2. 编译选项	2
1.3. 编译选项	2
2. 语 法	3
2.1. 语 法	3
2.2. 语 法	4
3. 预 处 理	5
3.1. #if	5
3.2. #either	6
3.3. #switch	7
3.4. #case	8
3.5. #include	8
3.6. #do	9
3.7. #macro	9
3.8. #local	11
3.9. #reset	12
3.10. #process	12
3.11. #trace	13
4. 接口API	13
4.1. expand-directives	13

1. 概 述

Red是一个轻量级的Red引擎，它支持Red引擎的所有功能，Red引擎的所有功能都可以通过Red引擎的API来实现。Red引擎的API包括：
• directives 指令集
• directive 指令集

- Red引擎的API
- do 指令集
- block指令集 expand-directives 指令集

Red引擎的API包括：
• LOAD 指令集

Red引擎的API包括：

- 指令集
- 指令集

2. 宏

Red 宏

宏

宏是 Red 语言中一种特殊的函数，它可以在编译时执行一些操作，并生成代码。宏的定义格式如下：
`#macro name: func [arg1 arg2... /local word1 word2...][...code...]`

NOTE: 宏的定义必须在宏调用之前。宏的定义格式与函数定义类似，但宏定义必须以 `#macro` 开头，并且宏定义中的代码块必须以 `func` 开头。

宏的定义格式如下：

2.1. 宏的定义

宏的定义格式如下：

```
#macro name: func [arg1 arg2... /local word1 word2...][...code...]
```

宏的定义格式如下：
`name` 是宏的名称，`word` 是宏的局部变量。

1. 宏的定义
2. 宏的调用
3. 宏的局部变量
4. 宏的返回值

NOTE: 宏的定义必须在宏调用之前。

宏

```
Red []  
#macro make-KB: func [n][n * 1024]  
print make-KB 64
```

宏的调用

```
Red []  
print 65536
```

宏的返回值

```
Red []
#macro make-KB: func [n][n * 1024]
#macro make-MB: func [n][make-KB make-KB n]

print make-MB 1
```

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

```
Red []
print 1048576
```

2.2. □□□□□□□□

```
wordParse

```

□ □

[manual]

[illegible]

```
#macro <rule> func [<attribute> start end /local word1 word2...][...code...]
```

<rule> □□□□□□□□□□□□□□□□□□□□

- **lit-word!** `lit-word!word`
- **word!** `word!Parse` `skip`
- **block!** `block!Parse`

```
start end
```

```
<attribute> [] [manual] [][][][][][][][][][][][][][][][][][][][][][][][]
```

11

```
Red []

#macro integer! func [s e][s/1 + 1]
print 1 + 2
```

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

```
Red []
print 2 + 3
```

[illegible]

```
Red []

#macro integer! func [[manual] s e][s/1: s/1 + 1 next s]
print 1 + 2
```

block

```
Red []
#macro ['max some [integer!]] func [s e][
  first maximum-of copy/part next s e
]
print max 4 2 3 8 1
```

[illegible]

```
Red []
print 8
```

3. □□□□□□□

3.1. #if

```
#if <expr> [<body>]
```

<expr> : 00000000000000000000

<body> : if <expr> 0true0000000000000000

☐ true
 ☐ false
 ☐ `<body>`

```
Red []

#if config/OS = 'Windows [print "OS is Windows"]
```

Windows

```
Red []

print "OS is Windows"
```

Windows

```
Red []
```

#do word

```
Red []

#do [debug?: yes]

#if debug? [print "running in debug mode"]
```

```
Red []

print "running in debug mode"
```

3.2. #either

```
#either <expr> [<true>][<false>]

<expr>  : 
<true>  : if <expr> 
<false> : if <expr> 
```

```
Red []

print #either config/OS = 'Windows ["Windows"]["Unix"]
```

Windows

```
Red []

print "Windows"
```

Windows

```
Red []

print "Unix"
```

3.3. #switch

```
#switch <expr> [<value1> [<case1>] <value2> [<case2>] ...]
#switch <expr> [<value1> [<case1>] <value2> [<case2>] ... #default [<default>]]

<valueN> : 
<caseN>  : 
<default> : 
```

```
Red []

print #switch config/OS [
  Windows ["Windows"]
  Linux   ["Unix"]
  MacOSX  ["macOS"]
]
```

Windows

```
Red []

print "Windows"
```


3.6. #do

例

```
#do [<body>]
#do keep [<body>]

<body> : 100Red1000
```

例

例1: 1000個のRedを生成し、keepで保持し、bodyで100個のRedを生成する

例

```
Red []

#do [a: 1]

print ["2 + 3 =" #do keep [2 + 3]]

#if a < 0 [print "negative"]
```

例2: 100個のRedを生成する

```
Red []

print ["2 + 3 =" 5]
```

3.7. #macro

例

```
#macro <name> func <spec> <body>
#macro <pattern> func <spec> <body>

<name>      : 1000個のset-word!
<pattern> : 1000個のblock!, word!, lit-word!
<spec>     : 1000個のspec
<body>     : 1000個のbody
```

例

例1: 1000個のset-word!

例2: 1000個のspec、1000個のblock!、1000個のword!、1000個のlit-word!、1000個のbodyを生成する

□ □

- 1

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

3.8. #local

11

1

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

```
Red []
print 1.0
print [1 3 124]
print 2.0
```

3.9. #reset

11

11

word

3.10. #process

11

```
#process [on | off]
```

11

Red

#process off ##### #process on
#####

1

```
Red []

print "Conditional directives:"
#process off
foreach d [#if #either #switch #case][probe d]
#process on
```

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

```
Red []

print "Conditional directives:"
foreach d [#if #either #switch #case][probe d]
```

3.11. #trace

11

```
#trace [on | off]
```

11

Red

4. □□□□□API

```
RedXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX do   file!
XXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX do XXXXXXXXXXXXXXXXXXXX do load %file
```

4.1. expand-directives

11

```
expand-directives [<body>]
expand-directives/clean [<body>]
```

<body> : □□□□□□□□□□□□□□□□Red□□□

11

[illegible]

1

```
expand-directives [print #either config/OS = 'Windows ["Windows"] ["Unix"]]
```

Windows

```
[print "Windows"]
```