数字集成电路静态时序分析基础

邸志雄 博士, zxdi@home.swjtu.edu.cn

西南交通大学信息科学与技术学院

Part-1:TCL语言

第三部分 控制流

控制流-if

- 语法格式：

```
if {判断条件} {
脚本语句
} elseif {判断条件} {
脚本语句
} else {
脚本语句
}
```

```
(windows) 31 %  if {$a > $b} {
> puts $a
> } else {
> puts $b
> }
3
```

判断语句

脚本语句

- 注意，上例中脚本语句的'{'一定要写在上一行，因为如果不这样，TCL 解释器会认为if命令在换行符处已结束，下一行会被当成新的命令，从而导致错误

实例讲解

例题：我们如何判断一个列表{0 1 2 3 4}的长度是大于3，还是等3，还是小于3?

```
(System32) 24 % set list1 {0 1 2 3 4}
0 1 2 3 4
(System32) 25 % set length [llength $list1]
5
(System32) 26 % if {$length > 3} {
puts "The length of list1 is larger than 3"
} elseif {$length == 3} {
puts "The length of list1 is equal to 3"
} else {
puts "The length of list1 is less than 3"
}
The length of list1 is larger than 3
(System32) 27 %
```

循环指令-foreach

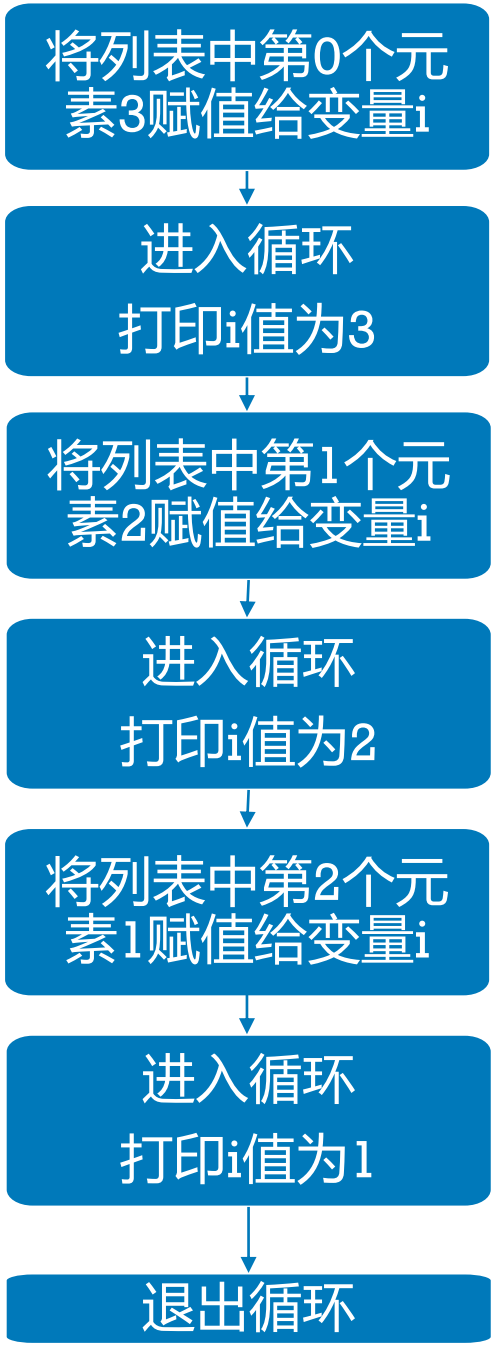
- 语法格式： foreach 变量 列表 循环主体
- 功能：从第0个元素开始，每次按顺序取得列表的一个元素，将其赋值给变量，然后执行循环主体一次，直到列表最后一个元素

```
(Windows) 36 % set list1 {3 2 1}
3 2 1
(Windows) 37 % foreach i $list1 {
> puts $i
> }
3
2
1
```

变量

列表

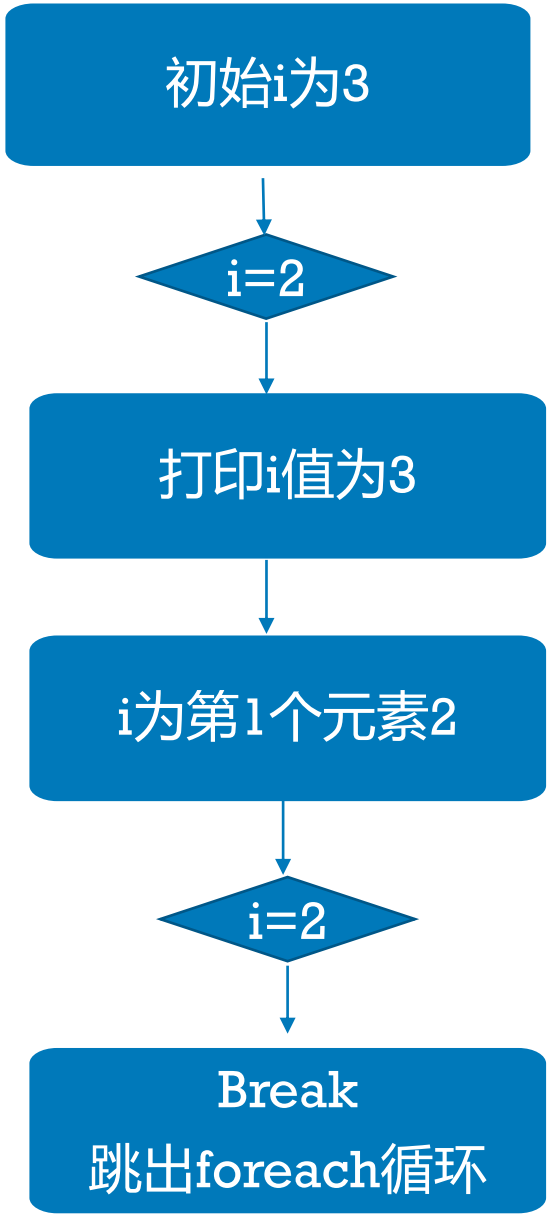
循环主体



循环控制指令-break

- 语法格式： break
- 功能:　　结束整个循环过程，并从循环中跳出

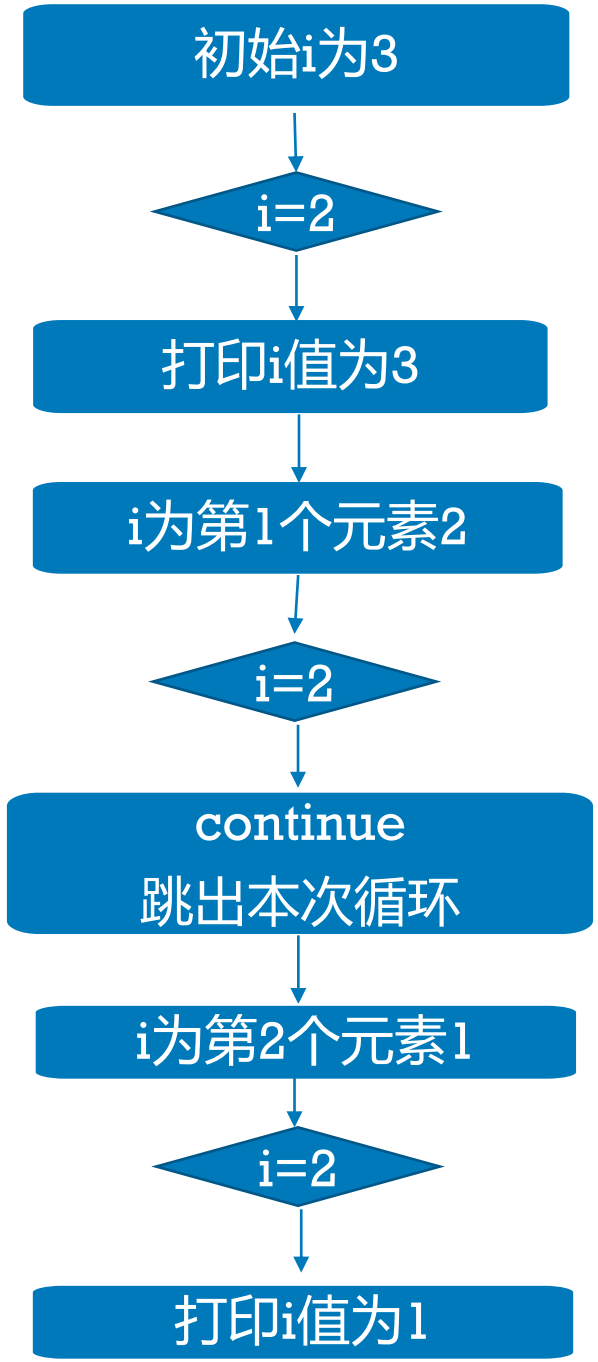
```
(Windows) 44 % set list1 {3 2 1}
3 2 1
(Windows) 45 % foreach i $list1 {
if {$i == 2} {
break
}
puts $i
}
3
```



循环控制指令-continue

- 语法格式： continue
- 功能:　　仅结束本次循环

```
(Windows) 46 % set list1 {3 2 1}
3 2 1
(Windows) 47 % foreach i $list1 {
if {$i == 2} {
continue
}
puts $i
}
3
1
```



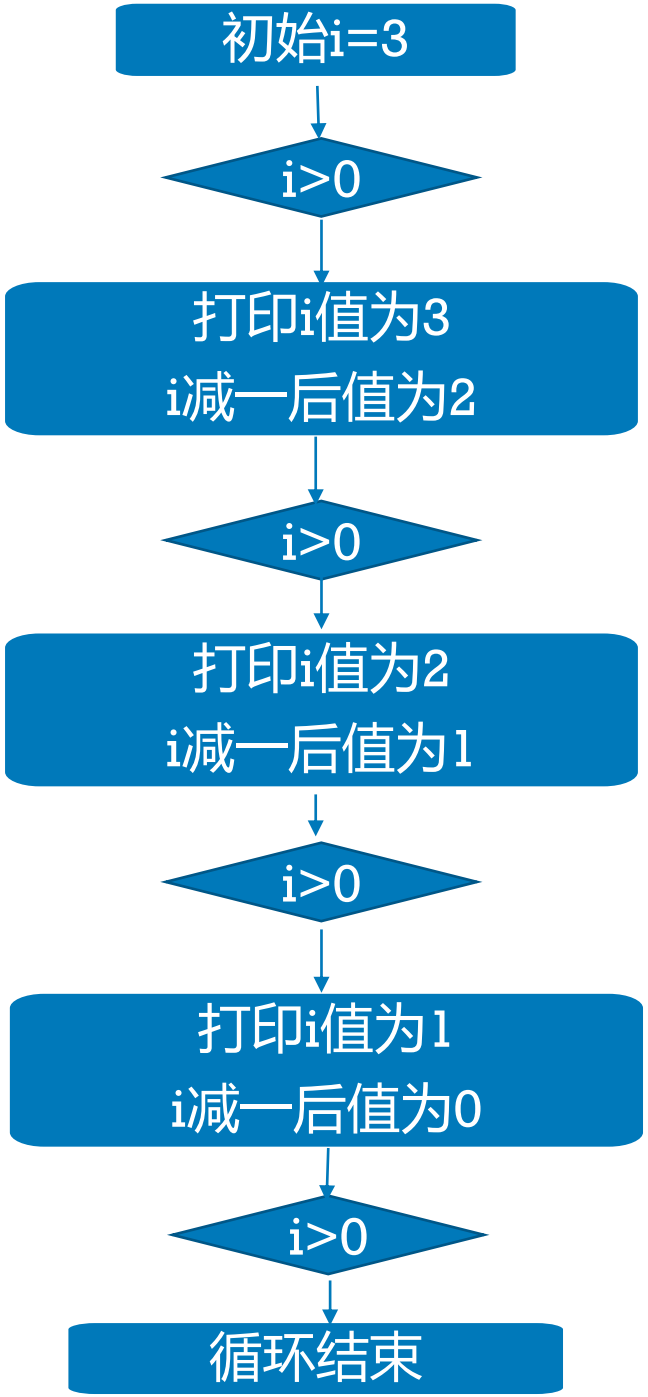
循环控制指令-while

- 语法格式： while 判断语句 循环主体
- 功能: 如果判断语句成立（返回值非0），就运行脚本，直到不满足判断条件停止循环，此时while命令中断并返回一个空字符串。

```
(Windows) 33 % set i 3
3
(Windows) 34 % while {$i > 0} {
> puts $i
> incr i -1; #set i [expr $i -1]
> }
3
2
1
```

判断语句

循环主体



循环控制指令-for

- 语法格式：

  for 参数初始化 判断语句 重新初始化参数 循环主体

- 功能:　　如果判断语句返回值非0就进入循环，执行循环主体后，再重新初始化参数。然后再次进行判断，直到判断语句返回值为0，循环结束。

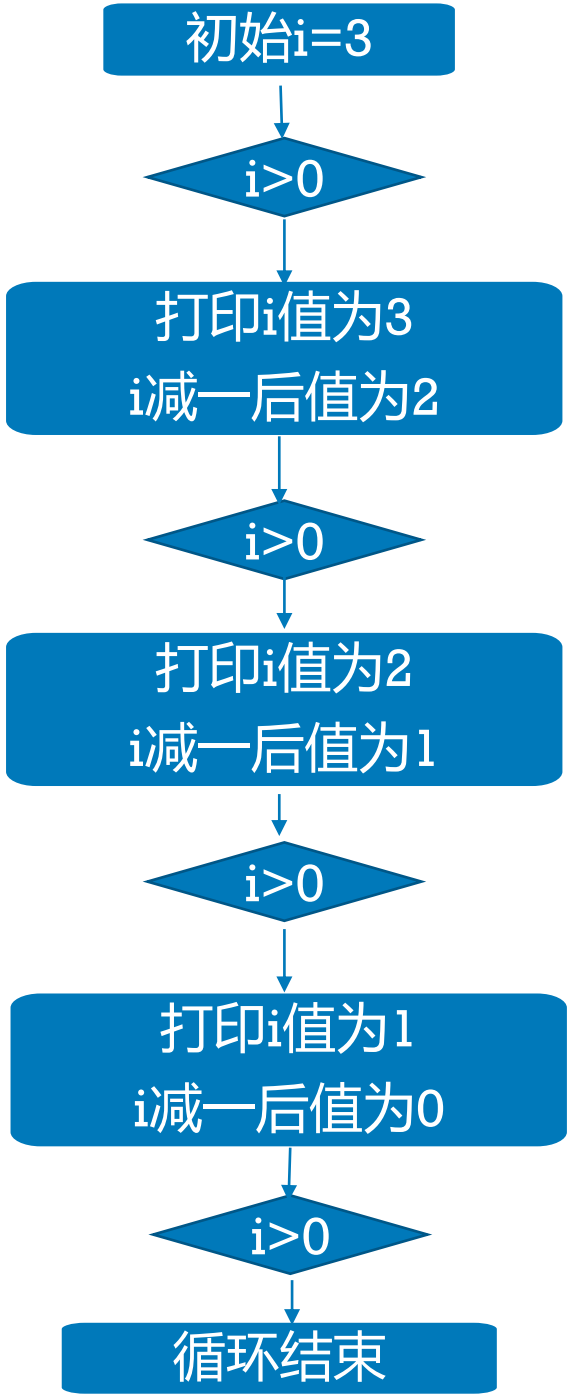
```
(Windows) 35 % for {set i 3} {$i >0} {incr i -1} {
puts $i
}
3
2
1
```

循环主体

初始化参数

判断语句

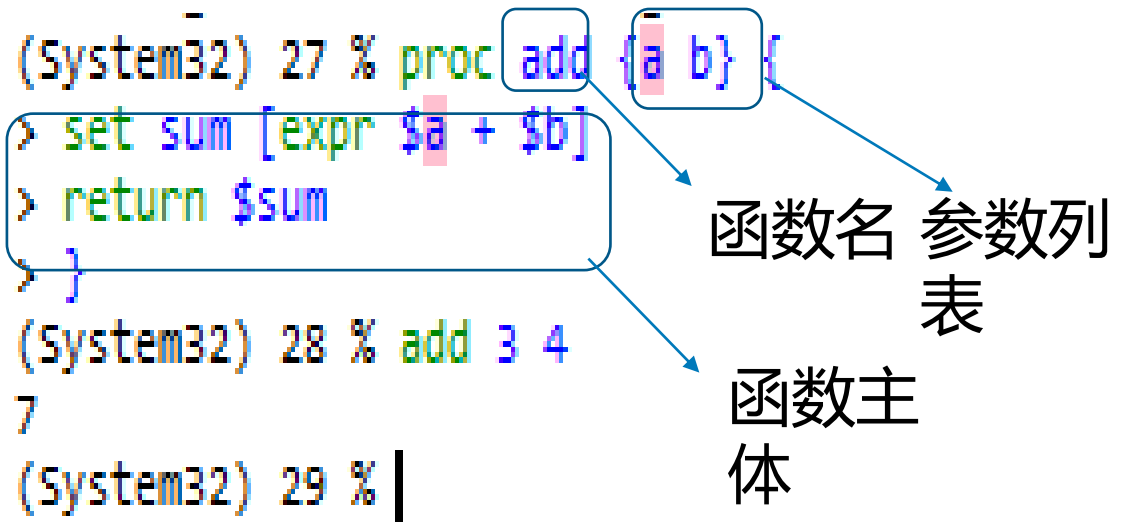
重新初始化参数



第四部分 过程函数

过程函数-proc

- 语法格式： proc 函数名 参数列表 函数主体
- 功能:　　类似于C语言中的函数。即用户自定义的功能，方便多次调用。



全局变量与局部变量

- 全局变量：在所有过程之外定义的变量。
- 局部变量：对于在过程中定义的变量，因为它们只能在过程中被访问，并且当过程退出时会被自动删除。
- 指令global，可以在过程内部引用全部变量

```
(Windows) 27 % set a 1
1
(Windows) 28 % proc sample {x} {
global a
> set a [expr $a + 1]
> return [expr $a + $x]
> }
(Windows) 29 % sample 3
5
(Windows) 30 %
```

```
(Windows) 30 % set a 1
1
(Windows) 31 % proc sample {x} {
set a [expr $a +1]
return [expr $a + $x]
}
(Windows) 32 % sample 3
can't read "a": no such variable
(Windows) 33 %
```

第五部分 正则匹配

正则匹配

- 定义：正则表达式是一种特殊的字符串模式，用来去匹配符合规则的字符串。

- 正则表达式的\w，用来匹配一个字母、数字、下划线
- 正则表达式的\d ,用来匹配一个数字

字符串	abc123	
正则表达式	\w\w\w\d\d\d	✓
正则表达式	\d\w\w\d\d\d	X

正则匹配-量词

\w\w\w\d\d\d这种写法过于繁琐，我们可以用代替重复的量词进行表示。

在TCL中常用一下三种量词

符号	功能
*	零次或多次匹配
+	一次或者多次匹配
?	零次或者一次匹配

字符串　　abc123

正则表达式　\w+\d+ 或者 \w*\d*

正则匹配-量词

*和+的区别：*可以是零次，+一个至少是一次

字符串	abc123	✓
正则表达式	\d*\w*\d*	
正则表达式	\d+\w+\d+	X

正则匹配-量词

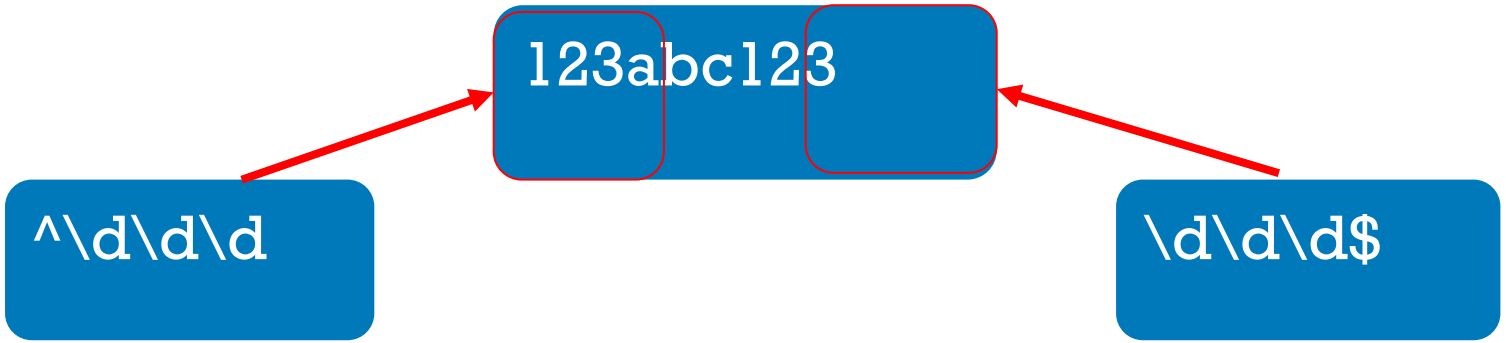
？表示零次或者一次匹配。

正则表达式	\w?\w?\d\d\w\w\d\d	
字符串	12ab34	
字符串	ab12ab34	

正则匹配-锚位

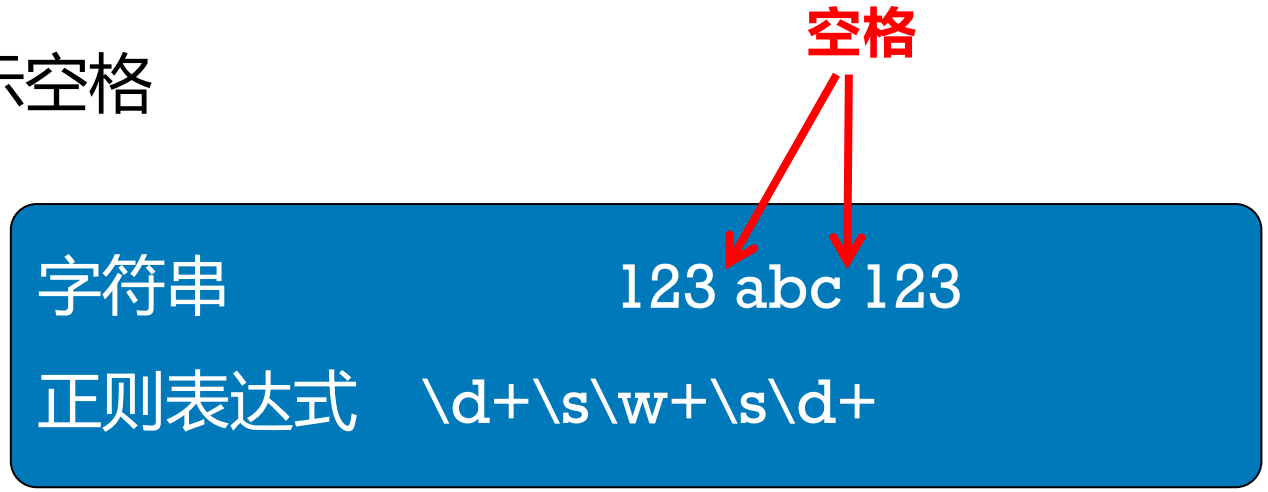
- 锚位，用来指示字符串当中的开头和结尾的位置，使我们能够匹配到正确的字符

符号	功能
^	字符串开头
$	字符串结尾



正则匹配-其他字符

- 常用的其他字符还有\s和.
- \s表示空格



- .表示任意一个字符

我们不确定具体是什么字符是就可以用.表示。

例如已知知字符串为”xxx空格xxx空格xxx”(x为未知字符)。

用.+\s.+\s.+就可以匹配

正则匹配指令-regexp

- 语法格式： regexp? switches? exp string? matchVar? ?subMatchVar subMatchVar …?
- 功能: 在字符串中使用正则表达式匹配。

- switches：

  -nocase将字符串中的大写都当成小写看待。
- exp 正则表达式
- string 用来进行匹配的字符串
- matchstring表示用正则表示式匹配的所有字符串
- sub1表示正则表达式中的第一个子表达式匹配的字符串
- sub2表示正则表达式中的第二个子表达式匹配的字符串

实例讲解

例题：　　如何匹配字符串”abc456”

```
(Tcl) 39 % regexp {\w+\d+} "abc456"
1
```

例题：　　如何匹配一个以数字开头并且以数字结尾的字符串？

```
(Tcl) 50 % regexp {^\d.*\d$} "1 dfsal 1"
1
```

捕获变量

- 通过()可以捕获字符串

  例如如何将字符串“Snow is 30 years old”中30捕获出来?

```
(Tcl) 68 % regexp {\s(\d+).*} "Snow is 30 years old" total age
1
(Tcl) 69 % puts $total
 30 years old
(Tcl) 70 % puts $age
30
```

- 一次捕获多个字符串

  例如如何将字符串“Snow is 30 years old”中Snow 和30一次捕获?

```
(Tcl) 80 % regexp {^(\w+)\s\w+\s(\d+).*} "Snow is 30 years old" total name age
1
(Tcl) 81 % puts $total
Snow is 30 years old
(Tcl) 82 % puts $name
Snow
(Tcl) 83 % puts $age
30
```

第六部分 文本处理

文本处理

用TCL处理文本在工作中十分常用。

主要掌握以下三个指令。

- open
- gets
- close

文本处理

- open
  - 语法格式　　open 文件 打开方式（打开方式 r表示读模式，w表示写模式。）
  - 功能　　打开文件
- gets
  - 语法格式　　gets fileId 变量名
  - 功能　　gets读fileId标识的文件的下一行，并把该行赋给变量，并返回该行的字符数（文件尾返回-1）
- close
  - 语法格式　　close fileid
  - 功能　　关闭文件

实例讲解

例题：　　整个读入文件过程

```
(Tcl) 32 % set INPUTFILE [open file.txt r]
file354b580
(Tcl) 33 % while {[gets $INPUTFILE line] >=0} {
> puts "$line"
> }
a
b
c
(Tcl) 34 % close $INPUTFILE
(Tcl) 35 %
```

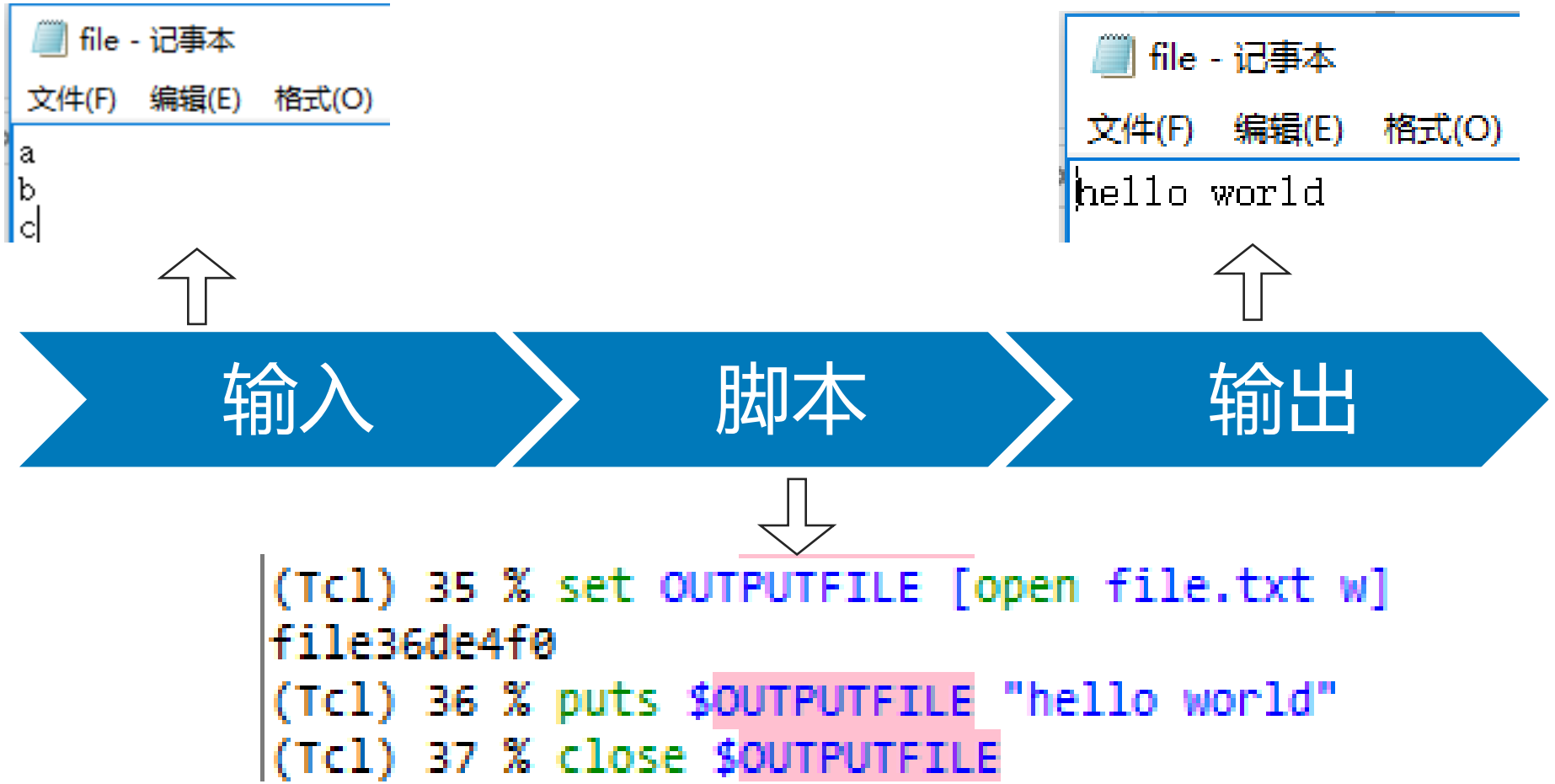
以只读模式打开文件 file.txt

从第一行开始逐行读取文件

关闭文件file.txt

实例讲解

例题：　　一个完整写入文件过程



例题

现有文本file.txt其内容如下。请写一TCL脚本求出所有Slack值之和。

```
Slack = -0.051
Slack = -0.234
Slack = -0.311
Slack = -0.056
Slack = -0.434
Slack = -0.316
Slack = -0.151
Slack = -0.524
```

例题

现有文本file.txt其内容如下。请写一TCL脚本求出所有Slack值之和。

```
Slack = -0.051
Slack = -0.234
Slack = -0.311
Slack = -0.056
Slack = -0.434
Slack = -0.316
Slack = -0.151
Slack = -0.524
```

```
set sum 0
set INFILE [open file.txt r]
while {[gets $INFILE line] >= 0} {
if {[regexp {^Slack\s+=\s+(-?\d+\.?\d+)}
$line total slack]} {
set sum [expr $sum +$slack]
}
}
close $INFILE
puts $sum
```

参考书目

- Using Tcl with Synopsys Tools. Version B-2008.09, March 2011. Synopsys.
- 集成电路静态时序分析与建模. 刘峰, 机械工业出版社.出版时间：2016-07-01.

谢谢聆听！

个人教学工作主页https://customizablecomputinglab.github.io/