Using tcl with synopsys command

1. dc shell特有命令

```
source  rename  history exit
```

2.dc shell支持的tcl cmd

```
after exec history* open split
append expr if package string
array exit* incr pid subst
binary fblocked info proc switch
bgerror fconfigure interp puts tell
break fcopy join pwd time
catch file lappend read trace
cd fileevent lindex regexp unset
clock filename linsert regsub update
close flush list rename* uplevel
concat for llength return upvar
continue foreach lrange scan variable
encoding format lreplace seek vwait
eof gets lsearch set while
error glob lsort socket
eval global namespace source*
```

3.dc shell命令的使用

1. 交互式：enter dc shell cmd
2. 脚本：source script_file

4.基本命令的使用

1. 一条基本的命令通常是由cmd+arguments或是cmd+option+arguments
2. 命令之间通过分号或换行符进行分隔
3. 一条长命令可以在当前行尾处加\进行分隔，使一条命令分布在不同行

5.通配符的使用

1. * ： 匹配任意字符
2. ？ ： 匹配单个字符

6.大小写敏感

1. cmd对字符是大小写敏感的

7. 列出并重新运行先前输入的命令

1. 使用history命令可以列出之前运行的命令（默认20条）
2. history info number：列出之前number条执行的命令
3. history redo number：重新执行event number对应的命令
4. history redo -2：执行倒数两条命令

5. ！！：重新执行最后一条命令
6. ！6：执行event number为6的命令

8.命令状态

1. 命令状态是命令的返回值
2. 命令返回值重新设置
    - >：重定向符号将命令结果重定向到一个文件中
    - 内置重定向命令：redirect - tee file_name {命令变量}；命令结果重定向到文件并打印到屏幕；

9.echo/puts打印

- echo similar to bash
- puts similar to tcl

10.运行脚本

1. source script_name.tcl：加载并执行tcl脚本
2. source -echo -verbose script_name.tcl：显示执行的命令及命令的结果
3. source -echo -verbose script_name.tcl > filename：命令及结果重定向到一个文件

11.字符串

cmd	describe
format	格式化字符串
regexp	字符串内索引
regsub	正则替换
scan	字符串内字段赋值给变量
string	字符串操作函数
subst	执行替换

12.列表

1. 语法格式

```
set listname “value1 value2 ... valuen”
set listname [list value1 value2 ... valuen]
set listname {value1 value2 ... valuen}
set listname [list {value1 value2} {value3 value4} valuen]
```

2. example:

For example, if variable a is set to 5, the following commands generate very different results:

```
dc_shell> set a 5
5

dc_shell> set b {c d $a [list $a z]}
c d $a [list $a z]

dc_shell> set b [list c d $a [list $a z]]
c d 5 {5 z}
```

image.png

3. list command:

cmd	describe
concat	连接列表并返回一个新的列表。
join	将一个列表中的元素连接成一个字符串
lappend	将元素追加到一个列表中
lindex	从一个列表中返回一个特定的元素
linsert	列表中插入一个元素
list	返回一个由其参数形成的列表
llength	列表长度
lrange	列表中提取部分元素
lreplace	列表替换元素
lsearch	列表查找元素
lsort	列表排序
split	字符串分割为列表

13.数组

1. 语法格式

```
set arrayname(index) value
array set arrayname [list index0 value0 index1 value1 indexn valuen]
```

2. array command

cmd	describe
array size	返回数组大小
array names	返回数组索引值

14.if command

1. 语法格式

```

==========

```
if {condition} {
 commands
}
```

==========

```
if {commands} {
 commands
} else {
 commands
}
```

==========

```
if {commands} {
 commands
} elseif {condition} {
 commands
} else {
 commands
}
```

==========

```
## 15.while command

1. 语法格式

   ```tcl
while {condition} {
    commands
   }
```

16. for command

1. 语法格式

```
for {initial condition} {termination condition} {reinit condition} {
    commands
}
```

17.foreach command

1. 语法格式

```
foreach var $list {
    commands
}
```

18.break/continue

1. break：跳出整个循环块
2. continue：跳出当前循环条件，进入下一轮循环

19.switch command

1. 语法格式

```
switch var {
    pattern1 {commands1}
    pattern2 {commands2}
    ...
    default  {commands_default}
}
```

20.文件操作

1. cd/pwd same as linux command
2. 文件命令

cmd	describe
file dirname fname	返回文件路径
file exist fname	如果文件名存在，返回1，否则返回0
file extension fname	返回文件名的扩展部分
file isdirectory fname	如果文件名是一个目录，返回1，否则返回0
file isfile fname	如果文件名是一个文件，返回1，否则返回0
file readable fname	如果文件是可读的，返回1，否则返回0
file writable fname	如果文件是可写的，返回1，否则返回0
file rootname fname	返回文件名的名称部分
file size fname	返回一个文件的大小，单位是字节。
file tail fname	从文件路径字符串中返回文件名

3. open

```
open fname mode
```

cmd	describe
r	打开文件仅用于阅读,该文件必须已经存在,默认访问模式。
r+	打开文件进行读写；该文件必须已经存在。
w	打开文件，仅用于写入,如果该文件存在，则截断它。如果该文件不存在，则创建它
w+	打开文件进行读写。如果该文件存在，则截断它。如果文件不存在，则创建它
a	打开文件仅用于写入；新的数据被追加到文件中。该文件必须已经存在
a+	打开文件进行读写。如果该文件不存在，则创建 它。新的数据被追加到该文件中

4. close

```
close $fileID

set files [open main.tcl w+]
close $files
```

5. flush

```
flush $fileID

#使用flush命令来强制将缓冲输出写入文件
```

21.gets/puts

1. gets

```
#使用gets命令从一个文件中读取一个单行数据

get $fileID var

#$fid参数是文件的ID，该文件是通过打开命令获得的；var参数是接收数据行的变量。var参数是接收该行数据的变量。
```

2. puts

```
#puts命令，向文件中写入单行

puts $fileID var

#参数$fid是文件的ID,是通过打开命令获得的;参数var包含要写入的数据。var参数包含要写入的数据。puts命令在输出数据前在其上添加一个新行字符，然后再输出数据
```