execute command

Synopsis
command(command, timeout)

Description
Executes any Bourne shell command. Remember to escape any special characters which have meaning to the shell.

Arguments

command	Command line to be executed by Bourne shell
timeout	Timeout interval (seconds): 0 for no timeout Allowed:Int Default: 0

Returns
execution result record
Fields in the returned record are

lines String array of output written by command, one string per line of output.

status Integer valued exit status from command execution

errlines Array of lines written to stderr by command

---

sh.done - Function

Package	**utility**
Module	**misc**
Tool	**sh**

destroy the tool

Synopsis
done()

1.6.16 substitute - Tool

Package utility
Module misc

Functions to substitute Glish variables and expressions
include "substitute.g"

Constructors

Functions

substitute	substitute Glish variables and expressions
substitutename	return the value of a variable name
substitutestring	return the value of a string variable
substitutevar	return the value of a variable

Description
The substitute function allows the user to embed Glish variables and expressions in a string and have them substituted. It is, for instance, used in TaQL (Table Query Language). E.g.

```
table.query ("column > $glishvar")
table.query ("column > $(var1+var2)")
```

Substitute allows the user also to use a Glish variable representing a tool (e.g. a table tool). E.g.

```
t1:=table("table1.dat")
t2:=table("table2.dat")
t1.query ("time in [select time from $t2 where windspeed<10]")
```

Further below are some more detailed examples.
The following rules apply:

1. Glish variables can be substituted by preceeding their name with a $ (as shown in the examples above). A variable name must start with an underscore or alphabetic, followed by zero or more alphanumerics and underscores.

2. Parts enclosed in single or double quotes are left untouched. Furthermore a $ can be escaped by a backslash, which is useful when an environment variable is used. Note that Glish requires an extra backslash to escape the backslash. The output contains the quotes and backslashes.

3. When name has a vector value, its substitution is enclosed in square brackets and separated by commas. It means that the result can directly be used in a TaQL command (or in a Glish command).

4. A string value is enclosed in double quotes. When the value contains a double quote, that quote is enclosed in single quotes. This is in line with the TaQL way of specifying string constants.

5. When the variable has a record value representing a tool (e.g. table), it is substituted by `$n` (n is a sequence number), while the table-id is added to the argument `idrec`. The first field in `idrec` is `nr` containing the number of substituted table-id's. The other fields contain the table-id's at each sequence number. The fields are unnamed, but can be accessed with the index operator such that `idrec[n+1]` contains the table-id of `$n`.
   The argument type tells which tool types are recognized. The default is `''` meaning that no tools are recognized.
   The argument `startseqnr` tells the first sequence number to use.
   For a tool to be valid the following have to be met:

   The tool must have a function `type()` returning a string containing the tool type.

   The tool must have a function `id()` returning the object-id of the tool.

   The tool types `table` and `region` do not meet these requirements. However, they are recognized and handled specifically.

6. When the name is unknown or has an unknown type, it is left untouched (thus the result is simply `$name`).

Furthermore it substitutes `$(expression)` by the expression result. It correctly handles parentheses and quotes in the expression. E.g.

```
$(a+b)
$(a)
$((a+b)*(a+b))
$(len("ab cd( de"))
```

Similar escape rules as above apply.
Substitution is NOT recursive. E.g. if `a:=1` and `b:="$a"`, the result of `substitute("$a")` is `"$a"` and not `1`.
Substitute has one problem due to the rules used by the eval function of Glish. Eval searches a variable only in the global scope. So if the substitution mechanism is used in a function, one has to use global variables if they are to be substituted. At the of the function they should be deleted. One has to be sure to use a unique name for the variable, e.g. by using the function name a s a suffix. E.g.

```
myfunc := function() {
  tab := table('mytable')
  global coldata_in_myfunc;
  coldata_in_myfunc := tab.getcol ('col');
  seltab := tab.query ('col > $(sum(coldata)/len(coldata))')
  symbol_delete ('coldata_in_myfunc');
}
```

Example

The following examples shows how variables and expressions are substituted.

```
- a:=10
- b:=20
- substitute("$a+$b")          #substitute both variables
10+20
- substitute("$(a+b)")         #substitute the expression
30
- substitute("$a$b")           #substitute both variables
1020
- substitute("$c")             #c is unknown, thus $c is returned
$c
- substitute("'$a'$b")         #$a is quoted, thus not substituted
'$a'20
- substitute("\\$a$b")         #first $ is escaped, thus $a not substituted
                               #note that \\ is needed to escape \ in glish
\$a20
- substitute('ab $(len("ab cd ef")) cd')    #substitute the $(len...
ab 3 cd
- substitute('$("ab cd ef")')  #a vector is enclosed in [] and separated by ,
                               #strings are enclosed in ""
["ab","cd","ef"]
- str:='ab"cd'
- substitute('$str')           #a " in a string is enclosed in ''
"ab"'"'"cd"
```

The following example shows how substitute is used in TaQL.

```
#   Open 2 tables.
- t:=table("/aips++/gvandiep/9800617.MS")
- t1:=table("/aips++/gvandiep/9800618.MS")
#   Show their handles
- t.handle()
[type=table, id=0, file=/aips++/gvandiep/9800617.MS]
- t1.handle()
[type=table, id=1, file=/aips++/gvandiep/9800618.MS]
#   Create a record (substitute clears it if not empty).
```

```
- idrec:=[=]
#   Substitute the various variables.
#   The table tools are replaced by sequence numbers (1 is first one).
- substitute("select from $t1 $t where col>$a+1",'table',1,idrec)
select from $1 $2 where col>10+1
#   idrec contains the number of sequence numbers (2) and the table id
#   belonging to each sequence number. They are used by the table client
#   to get the correct table for $1 and $2 in the query command.
- idrec
[nr=2, *25=1, *26=0]
```

substitute.substitute - Function

Package	**utility**
Module	**misc**
Tool	**substitute**

substitute Glish variables and expressions

Synopsis

substitute(string, type, startseqnr, idrec)

Description

It analyzes a string and substitutes embedded Glish variables and expressions by their value as explained in the general description. If `var` is a valid Glish variable, `$var` is substituted by its value. If `expr` is a valid Glish expression, `$(expr)` is substituted by the result of the expression.

Arguments

string	in	string to be analyzed
type	in	tool type recognized (default none)
startseqnr	in	first sequence number to use for substituted tools
idrec	out	record containing id's of substituted tools

Returns

input string with appropriate parts substituted

---

substitute.substitutename - Function

Package	**utility**
Module	**misc**
Tool	**substitute**

return the value of a variable name

Synopsis
substitutename(name, type, idrec)

Description
If `name` is the name of a valid Glish variable, the value of the variable is returned as a string. (string values are enclosed in "", vectors are enclosed in `[]`).
When `name` is a recognized tool (i.e. its type matches), a sequence number is returned as `$n`. The tool id is added to `idrec`.

Arguments

name	in	name of variable to be substituted
type	in	tool type recognized (default none)
idrec	inout	record containing id's of substituted tools

Returns
string with value of Glish variable

---

substitute.substitutevar - Function

Package	**utility**
Module	**misc**
Tool	**substitute**

return the value of a variable

Synopsis
substitutevar(var)

Description
If `var` has a known value (i.e. numeric ,string, boolean), its value is returned as a string. (string values are enclosed in "", vectors are enclosed in `[]`).

Arguments