mode

sets or queries the mode echoing Scilab instructions in the console

Syntax

```
mode(k)
k = mode()
```

Arguments

k

integer from -1 to 6: chosen or current execution / echoing mode.

Description

`mode(k)` allows to choose how informations are displayed in the console during the execution of Scilab instructions. If these instructions include a `mode` one, following ones in the same environment are echoed according to the new mode. A semicolon appended to any instruction always cancels the display of its result, whatever is the current execution `mode`.

Contexts

`mode(..)` and other instructions can be used and executed in various contexts:

- **Functions (F)**: a function written in Scilab language may include `mode` instructions. After being compiled and called, effects of an inner `mode` instruction are tagged with a **F** in the table herebelow. By default, instructions in functions are run in silent mode `mode(-1)`, whatever is the current mode in the calling environment.
- **Scripts (S)**: Scilab instructions written in a file (typically with the .sce extension) out of any function definition may include `mode` instructions. When such a file is run with `exec(filename)` or `exec(filename, mode_k)`, effects of a `mode` instruction in the executed file are tagged with a **S** in the table herebelow. By default, scripts are run in `mode(3)` mode, *whatever is the current mode in the calling environment*. This is overridden with the `mode_k` option.
- **Console (C)**: Scilab instructions directly entered in the console are always displayed as entered. Effects of the current `mode` or of any forthcoming `mode` instruction entered in the console are tagged with a **C** in the table herebelow. By default, results of instructions run in the console are displayed in `mode(2)`.
- **execstr(T)**: This function accepts a matrix of text T. Each component is executed as a series of Scilab instructions, that may include `mode` ones. Effects of any forthcoming `mode` instruction met in the matrix are tagged with a **T** (as Text) in the table herebelow. By default, all instructions are run in silent mode `mode(-1)`, *whatever is the current mode in the calling environment* running `execstr()`.
- **Callbacks (K)**: a callback is a unique string in which Scilab instructions are written. This string is assigned to an interactive component such as the item of a menu, a checkbox, etc. The instructions are executed when the component is activated by an interaction: the menu is selected, the checkbox is checked or unchecked, etc. A callback may include some `mode` instructions. The instructions of a callback are always executed directly at the console level. Their effects remain in the console after the callback is completed. Effects of a `mode` instruction used in a callbacK are tagged with a **K** in the table herebelow.

Features

mode #	**-1**	**0**	**1**	**2**	**3**	**4**	**6**
Displays instructions [a]	C	C	C S	C	C S	C S	C S K
Displays results [b]		always	always	always	always	always	always
Step by step [s]						S F K	S F T K
Compact [c]	C++	+	++		SFT +	CK++ SFT+	S+
Comments	[d]		[e]	[f]	[g]	[h]	[h,i]

Comments

[a]: In normal modes, instructions are displayed with the `-->` heading prompt. In step-by-step modes, `>>` is used instead.

[b]: provided that no semicolon is appended.

[c]: "+" means: no extra blank line after results. "++" means: no extra blank line neither after completed instructions, nor after results.

[d]: Default silent mode in functions and with `execstr()`.

[e]: `mode(5)` is equivalent to `mode(1)` but must not be used.

[f]: Default mode in the console.

[g]: Default `exec()` mode.

[h]:
- Any comment `//` is displayed without prompting and being stepped.
- Some parasitic `-->` prompts and extra blank lines may be sometimes displayed (bug).
- A callback is always made of a unique string of instructions, as if they were specified and run on a single row. Therefore, both available stepping execution modes are activable but useless in any callback.

[i]: `mode(7)` does the same but must not be used.

[s]: The step-by-step mode stops after each line of instruction(s) and waits for the user pressing the `<enter>` or `p<enter>` keys to go on. Entering `p` enters the pause mode. These modes may be used for instance in demos, or as a raw debugging mode.

The mode in the calling environment is never changed after using `mode(..)` in a called function, in an executed script.sce or as an `execstr()` input, after the execution is completed and returns. When `mode(k)` is used in a *callback* that is executed, it becomes and remains the actual echoing mode in the console after the end of the callback.

Output intentionnally displayed by functions like `disp()` or `mprinf()` are never cancelled, even with `mode(-1)`.

`mode(5)`, `mode(7)`, and other unregistered values may be accepted but should not be used: they could be removed or redefined in the future.

Examples

In a function():

```
function example_mode(level_mode)
   disp(mode());
   mode(level_mode)
   a = 3
endfunction

mode(2)
example_mode(0)
mode()
example_mode(1)
example_mode(2)
```

With exec(script, mode):

```
ins = [
    "mprintf(""Default execution mode: %d\n"", mode())"
    "mode(i)"
    "mprintf(""New active mode: %d\n"", mode())"
    "// A new comment"
    "a = rand(2,4)"
    "b = %pi;"
    "c = %s;"
    ];
fn = TMPDIR + "\test_mode.sce";
mputl(ins, fn);
//
mode(2)
i = 1;
exec(fn)
mode()
exec(fn, 0)
i = 3; // instructions are displayed
exec(fn, 3)
i = 4; // displayed instructions + stepped mode. "p<enter>" enters the paused mode
exec(fn, 4)
```

With execstr():

```
ins = [
    "mprintf(""Default execution mode: %d\n"", mode())"
    "mode(1)        // Entering the compact mode"
    "mprintf(""New active mode: %d\n"", mode())"
    "a = rand(2,4)"
    "b = 1"
    "c = %pi"
    ];
mode(2)
execstr(ins)
mode()       // The initial mode is restored
```

In a callback (here a menu):

```
mode(2)
uimenu("parent",0,"Label","mode_test",..
       "callback", "disp(mode()); mode(1); a = rand(2,4), pwd(),");
// Click on the "mode_test" menu and see what is displayed in the console
mode()
delmenu mode_test
```

See also

- exec
- execstr
- semicolon
- debug
- pause
- getscilabmode
- warning mode
- funcprot
- ieee

History

Version	Description
6.0	• mode(4) is now stepped and can be paused, in scripts as well as in functions. • For/in scripts, mode(4) now displays each line of instructions, and displays results in a compact way. It can be used for demos. • Callbacks were always executed in silent mode(-1). They are now executed by default in the current `mode()`.