S-LANG SCRIPTING FOR

MIDNIGHT COMMANDER

1 API: Functions exported to the interpreter.

The "*cure_*" prefix stands for "*current editor*". All functions are in the ***mc*** namespace, i.e.: **mc->func()**.

1.1 *Editor functions (cure_ prefix).*

1.2 *Movement – #1 function.*

cure_cursor_move (*offset*)

Moves the cursor offset bytes right or left (when offset is smaller than 0).

1.3 *Getting offsets – #3 functions.*

cure_cursor_offset (void);

Returns an integer that is the cursor position in the buffer.

cure_get_bol (void)

Gets an integer that is the offset of the beginning of current line.

cure_get_eol (void)

Gets an integer that is the offset of the end of current line.

1.4 *Getting data from buffer – #2 functions.*

cure_get_left_whole_word (*skip_space*)

Returns a string that is the word on left of cursor. If skip_space is true, then it jumps over a single block of white space if necessary.

cure_get_byte (*byte_index*)

Returns the byte at given byte offset.

1.5 *Editing functions – #3 functions.*

cure_delete (void)

Deletes the char under the cursor.

cure_backspace (void)

Deletes the char left of cursor.

cure_insert_ahead (*char*)

Inserts the given char right of cursor.

1.6 *Dialog functions – #4 functions.*

listbox (h, w, title, items)

Displays the list with given size, title and items.

listbox_with_data (*h*, *w*, *title*, *items*, *data*)

Displays the list with given size, title, items and the associated data elements.

listbox_auto (*title*, *items*)

Auto sized listbox.

message (*title*, *body*)

A press-any-key message dialog with given title and body text.

1.7 *Action hooks – #2 functions.*

set_action_hook (*action_name*, *func_name*, *user_data*)

Hooks up the given function to the given action.

add_new_action (*new_action_name*, *new_ck_id*)

Adds a new action with given name and numeric ID.

1.8 *Key bindings – #2 functions.*

editor_map_key_to_action (*key*, *action_name*)

Adds a key binding to the given action. TODO: use the enum ID.

editor_map_key_to_func (*new_action*, *key*, *func_name*)

Adds a key binding to an also newly added action. The key will invoke the given S-Lang function.

2 Implementation

1. New files

   There are 2 files added: **src/slang_api_functions.c** and **src/slang_engine.c**. The first one implements the interface functions, which are enumerated in its header, which is processed by Slirp, the S-Lang auto-export utility.

   The second implements:

   - script error catching and displaying,
   - interpreter initialization,
   - sourcing of *init.sl* script and of all plugins in *~/.config/mc/plugin*.

2. SLIRP automatic binding utility.

   One other file is being added – **src/slang_api_functions_glue.c**. It is the result of running the official S-Lang binding utility ↔ Slirp with the command: ***cd src***; ***slirp -rc ../slirprc -rename slang_api__ NULL -rename keybind_ NULL slang_api_functions.h***;. It contains the glue code between C and S-Lang interpreter. Besides linking it there has to be following call after SLang_init_all():

   ```
   /* Init the `mc` namespace. */
   init_slang_api_functions_module_ns ((char *) "mc");
   ```

   The call makes all the functions from slang_api_functions.h to become available to the interpreter and grouped in namespace ***mc***. So that one can then call, e.g.:

   ```
   variable cursor_position = mc->cure_cursor_offset();
   ```

3. Example plugin: *misc/grow_shrink_integer.plugin.sl*.
   The plugin uses the exported functions to implement integer advancing (Alt-a) and decreasing (Alt-x). It'll be automatically loaded if it'll be copied to *~/.config/mc/plugin*.