1 General description.

This library is intended for creating the text user interfaces. It provides low-level functions for the work with the terminal and library widgets.

Supported platforms: Everything, where ncurses (it is tested on Linux) works, DOS with DJGPP, Windows 9x, Windows NT-based (2000, XP).

2 Installation.

Before the compilation it is necessary to recode sources to this platform's format (CRLF/LF). Set platform name (ncurses where it exists) in scr_cnf.h. Compile (using make). Headers and object modules (see later) are required to use it in your projects.

3 Using in your programs.

To use the base (low-level) functionality it's necessary to include header files "scr_ind.h" and "scr_drv.h", and to link modules scr_ind.o and scr_drv.o.

To use widgets it's necessary to include "scr_ml.h" and link "scr_ml.o". Also it is possible to use widget "special table" (scr_table).

3.1 Platforms specifics.

On Linux it is necessary to link with libncurses (gcc... -lncurses). Also, all definitions of types and functions from ncurses will be seen in the program using the library!

4 Principles of operations.

Screen coordinates begin at 0,0 (left upper corner).

Coordinates are always used in the form (X,Y).

There is a current output point (scr_printf, scr_addch will output there), after output it is moved to the right (after printed data).

There is a position of terminal (screen/hardware) cursor. It *does not depend* on the output point.

All output functions work *not* with the terminal screen, but with its copy! So, to display this copy on the screen, it is necessary to call scr_refresh.

Widgets *never* call scr_refresh.

Every widget which can process input from the terminal (keyboard), uses scr_{widget}_inject function for this . Its second parameter is the read symbol (key).

Widgets are created by scr_{widget}_create function.

———————

5 Low-level functions description.

5.1 Initializing and finalizing.

```
void scr_initscr(void);
void scr_stopscr(void);
```

It's necessary to call scr_initscr before working with the functions of the library, and scr_stopscr then program exits.

5.2 Output to the terminal.

```
void scr_cls(void);
void scr_addch(scr_char ch);
void scr_mvaddch(int x, int y, scr_char ch);
void scr_printf(char *format, ...);
void scr_mvprintf(unsigned char x, unsigned char y, char* format, ...);
```

Function scr_cls clears screen and sets output point and terminal cursor to the upper left corner of the screen.

Functions scr_(mv)addch output one char (of the type scr_char) into current or provided output position, and scr_(mv)printf are analogous to the usual printf, and they output the line with the current value of screen attributes.

5.3 Getting symbols from "screen".

```
scr_char scr_inch(void);
scr_char scr_mvinch(int x, int y);
```

They are analogous to output functions, but take symbols from "screen".

5.4 Getting and setting of the parameters of terminal and library.

```
void scr_locate(unsigned char x,unsigned char y);
void scr_setattr(scr_attr attr);
scr_attr scr_getattr(void);
unsigned char scr_maxx(void);
unsigned char scr_maxy(void);
```

Function scr_locate sets current output position, scr_(set—get)attr — set and get current screen attributes, scr_max(x—y) return width and height of the screen.

5.5 Working with the screen regions.

```
scr_char* scr_getarea(unsigned char x, unsigned char y,
           unsigned char lenx, unsigned char leny);
void scr_regetarea(unsigned char x, unsigned char y,
         unsigned char lenx, unsigned char leny, scr_char* area);
void scr_putarea(unsigned char x, unsigned char y,
           unsigned char lenx, unsigned char leny, scr_char* area);
void scr_freearea(scr_char* area);
```

Function scr_getarea allocates memory and copies there rectangular fragment of *current copy* of the screen, scr_regetarea copies region into previously malloc'ed area, scr_putarea copies region from memory to screen, scr_freearea frees memory allocated with scr_getarea.

5.6 Working with terminal cursor.

```
void scr_cursor(int x,int y);
void scr_hidecursor(void);
void scr_showcursor(void);
```

These are functions for putting the terminal cursor into the given point, hiding or redisplaying it. Movement of cursor occurs only after call to scr_refresh!

5.7 Getting terminal input.

```
int scr_getch(void);
```

It expects the input of the symbol (infinitely), returns the symbol as the code in range 0 - 256 or one of the constants SCR_KEY_thexxx, which are defined in scr_drv.h.

5.8 Type conversions.

```
unsigned char scr_char2char(scr_char ch);
scr_attr scr_char2attr(scr_char ch);
scr_char scr_to_scr_char(unsigned char ch, scr_attr attr);
scr_attr scr_color2attr(unsigned int color);
```

Function scr_char2char returns only symbol from the type ”screen symbol”, and so on, scr_color2attr converts color into the attribute. Look for according constants in scr_drv.h.