ОПИСАНИЕ ИНТЕРФЕЙСА ВЫВОДА НА КОНСОЛЬ И В ТЕРМИНАЛ С БАЗОВОЙ ПОДДЕРЖКОЙ VT100

Версия библиотеки № 23 от 02.12.2015

Оглавление

Общие сведения

Библиотека интерфейса вывода на консоль и в терминал был разработан в качестве замены библиотек ncurses и Turbo Vision под QNX 6, с целью минимального потребления системных ресурсов и максимальной простоты API. Необходимым условием было функционирование не только в терминале, но и в текстовой консоли ОС QNX. В связи с этим, были наложены следующие ограничения:
- поддержка Esc-последовательностей только VT100;
- поддержка только простейших возможностей управления текстом (нормальный, жирный, подчеркнутый, мерцающий, инвертированный).

Библиотека интерфейса вывода на консоль реализована на языке C, компилятор GCC 4.8.2 (допускается компиляция при помощи как gcc, так и g++) в виде статически подключаемого модуля libminicurs.a. Все функции библиотеки объявлены в заголовочном файле minicurs.h.
Библиотека предназначена для использования в прикладных программах PBCPT, но не является частью API PBCPT.
При тестировании подтверждена совместимость со следующими типами терминалов:
- qansi (консоль QNX);
- vt100;
- xterm

и следующими терминальными программами:
- PuTTY;
- KiTTY;
- TeraTerm;
- DTelnet (частично, не выводятся заглавные русские буквы в кодировке CP866);
- Microsoft Telnet (частично, отображаются только символы латиницы).

Инициализация и закрытие интерфейса

Для инициализации интерфейса вывода на консоль служит функция mcurses_init. Вызов данной функции обязателен до обращения к другим функциям библиотеки. Данная функция:
- формирует структуру описания интерфейса;
- фиксирует текущие границы окна терминала;
- фиксирует тип терминала;
- размещает в указанной переменной типа uint32_t дескриптор интерфейса.

Если приложение предназначено только для отображения информации, то больше никаких действий по инициализации не требуется. Однако, если планируется обработка нажатия клавиш (например, для организации экранного меню, см ниже), необходимо инициализировать терминал соответствующим образом, при помощи функции mcurses_set_term. Данная функция переводит терминал в raw-режим, что позволяет обрабатывать Esc-последовательности при нажатии клавиш пользователем.
Процедуру инициализации интерфейса и терминала необходимо провести один раз, при запуске приложения.
При завершении работы приложения, необходимо восстановить предшествующее состояние терминала, при помощи функции mcurses_restore_term. Интерфейс вывода на консоль должен быть закрыт при помощи функции mcurses_close.
Функции инициализации и закрытия не могут использоваться в обработчиках сигналов или в обработчиках прерываний, т.к. содержат операции по выделению и освобождению памяти.

Основные действия по выводу информации на консоль

Очистка экрана производится при помощи функции mcurses_cls, очистка строки, в которой в настоящее время находится курсор – при помощи mcurses_clstr. Необходимо отметить, что строка очищается от начала и до конца.

Для установки курсора в определенную позицию на экране используется функция mcurses_set_cursor_to. Для установки курсора в определенную позицию в текущей строке предназначена функция mcurses_set_cursor_to_col, однако ее использование требует тщательного тестирования конкретного приложения на предмет работы в терминальных программах, т.к. соответствующая Esc-последовательность поддерживается не всеми терминалами.

Текущие границы окна терминала определяются при помощи функций mcurses_get_lines (количество строк на экране) и mcurses_get_columns (количество колонок в строке). При необходимости, может быть определена текущая позиция курсора на экране, при помощи функции mcurses_get_cursor_pos.

Для вывода строки на экран, с предварительной ее очисткой, используется функция mcurses_put_string. Указанный текст при этом выводится, начиная с 1 позиции в строке. Для вывода на экран текста, начиная с текущей позиции курсора, предназначена функция mcurses_put_text. В качестве необязательного параметра для обеих функций могут быть указаны атрибуты выводимого текста:

ta_normal – обычный текст;
ta_bold – жирный текст;
ta_underline – подчеркнутый текст;
ta_blink – мерцающий текст;
ta_inv – инвертированный текст (черный на белом).

Возможны любые комбинации атрибутов по «ИЛИ», при этом необходимо тщательное тестирование конкретного приложения на предмет работы в терминальных программах, т.к. соответствующие Esc-последовательности могут по-разному обрабатываться. Так, Microsoft Telnet вместо подчеркнутого выводит текст красного цвета.

Сохранить текущее положение и параметры курсора можно при помощи функции mcurses_save_cursor, восстановить предшествующее состояние – при помощи mcurses_restore_cursor.

Рассмотрим пример простого приложения, выводящего текст на консоль

```
#include <locale.h>
#include "minicurs.h"

int main(int argc, char *argv[])
{
	uint32_t curses = -1;	///< Дескриптор консоли
	int lines = 25, columns = 80; ///< Количество строк и колонок
	char *str = "Тест консоли";
	char *press = "Для выхода нажмите любую клавишу...";

	setlocale(LC_CTYPE, "C-TRADITIONAL");

	// Инициализируем интерфейс
	if(mcurses_init(&curses) != EOK)
		exit(1);

	// Инициализируем терминал
```

```
	if(mcurses_set_term(curses) != EOK)
		exit(1);

	// Очищаем экран
	mcurses_cls(curses);

	// Определяем количество строк и колонок
	mcurses_get_lines(curses, &lines);
	mcurses_get_columns(curses, &columns);

	// Устанавливаем курсор на середину экрана
	mcurses_set_cursor_to(curses, lines/2, (columns/2)-(strlen(str)/2));

	// Выводим текст
	mcurses_put_text(curses, str, ta_bold | ta_inv);

	// Устанавливаем курсор в последнюю строку
	mcurses_set_cursor_to(curses, lines, 1);

	// Выводим текст и ждем нажатия клавиши
	mcurses_put_text(curses, press);
	getchar();

	// Восстанавливаем все и выходим
	mcurses_restore_term(curses);
	mcurses_close(&curses);
	return(0);
}
```

Организация экранного меню

Для организации экранного меню в библиотеке интерфейса вывода на консоль предусмотрен специальный набор функций. При этом экранное меню выводится в полный размер экрана. Если количество пунктов меню превышает количество строк на экране, на экран выводится только допустимая часть строк с прокруткой.

Экранное меню обрабатывает следующие клавиши:

- стрелки вверх-вниз перемещают курсор по пунктам меню на один пункт, с циклической прокруткой от конца к началу и наоборот;
- <PgUp> и <PgDn> перемещают курсор, соответственно, на первую или последнюю отображаемые строки;
- <Home> и <End> перемещают курсор, соответственно, на первый или последний пункты меню, с прокруткой при необходимости;
- <Enter> выбирает пункт меню под курсором;
- <Q> выходит из меню.

Кроме того, каждому пункту меню могут быть назначены клавиши быстрого доступа из числа цифр 0-9 и букв A-Z, за исключением Q (см выше). При нажатии соответствующей клавиши выбирается пункт меню, которому эта клавиша сопоставлена, даже если он находится за пределами экрана. Если двум пунктам меню назначены одинаковые клавиши, то выбран будет тот из них, который расположен ближе к началу списка. Клавиши быстрого доступа отображаются в списке слева от наименования пункта меню.

Каждому пункту меню могут быть сопоставлены:

- строковые данные, до 255 символов;

- целочисленные данные 32 бит;
- данные с плавающей точкой 32 бит.

В строке данных может быть указано специальное ключевое слово EXITMENU, при выборе данного пункта произойдет выход из экранного меню.

Для создания экранного меню используется функция mcurses_menu_init. Данная функция:
- формирует структуру описания меню;
- настраивает функцию-обработчик выбора пункта меню (нажатия клавиши Enter или клавиши быстрого доступа);
- при необходимости, настраивает отображаемый заголовок меню;
- при необходимости, настраивает функцию-обработчик нажатий клавиш, не входящих в стандартный набор (см выше);
- при необходимости, выделяет память под список пунктов меню;
- при необходимости, заполняет список пунктов меню из заданного массива типа mcurses_menu_item_t;
- помещает в указанную переменную типа uint32_t дескриптор экранного меню.

Функция-обработчик выбора пункта меню должна быть объявлена в соответствии с описанием типа mcurses_menu_item_handler_t. Данная функция должна принимать на вход:
- строковый параметр (например, содержащий командную строку);
- целочисленный параметр;
- параметр с плавающей точкой.

Функция должна возвращать значение 1, если работа меню должна быть продолжена, и 0, если необходим выход из меню.

Функция-обработчик нажатий клавиш должна быть объявлена в соответствии с описанием типа mcurses_key_handler_t. Данная функция должна принимать на вход:
- дескриптор меню;
- указатель на структуру описания нажатой клавиши mcurses_key_t.

Функция должна возвращать значение 1, если работа меню должна быть продолжена, и 0, если необходим выход из меню.

Для добавления пункта в меню (с выделением/перераспределением памяти под список пунктов) используется функция mcurses_menu_add, которой должны быть переданы следующие параметры:
- наименование пункта меню, которое будет выведено на экран;
- символ быстрого доступа или 0, если быстрый доступ к пункту не предусматривается;
- строка данных до 255 символов;
- целое число 32 бита;
- число с плавающей точкой 32 бита.

Для вывода на экран меню и его запуска предназначена функция mcurses_menu_run. Эта функция реализует основной цикл работы меню. Выход из функции производится по выходу из меню, инициированному выбором соответствующего пункта, нажатием клавиши <Q> либо возвратом 0 из функции-обработчика, а также в случае ошибки при отображении меню. По выходу из меню экран очищается.

Функция mcurses_menu_run может быть вызвана приложением неоднократно, это дает возможность, например, добавлять пункты меню «на ходу».

Закрытие меню и очистка выделенной памяти производится при помощи функции mcurses_menu_close.

Рассмотрим пример организации меню.

```
#include <locale.h>
#include "minicurs.h"
```

```
int menu_item_handler(char *str, uint32_t int_data, float float_data)
{
system(str);
return 1;
}

int main(int argc, char *argv[])
{
      uint32_t curses = -1, menu = -1;
      int res = EINVAL;
      mcurses_menu_item_t menu_items[8] =
      {{"File list", '1', "ls -l;sleep 5", 0, 0.},
       {"Место на дисках", '2', "df -h;sleep 5 ", 0, 0.},
       {"Список процессов", '3', "ps -A;sleep 5 ", 0, 0.},
       {"Информация", '4', "pidin info;sleep 5 ", 0, 0.},
       {"Информация", '5', "pidin hwinfo;sleep 5", 0, 0.},
       {"Информация", '6', "pidin cpuinfo;sleep 5", 0, 0.},
       {"Информация", '7', "pidin asinfo;sleep 5", 0, 0.},
       {"Выход", '8', "EXITMENU", 0, 0.}};

      setlocale(LC_CTYPE, "C-TRADITIONAL");

      if(mcurses_init(&curses) != EOK)
            exit(1);

      if(mcurses_set_term(curses) != EOK)
            exit(1);

      res = mcurses_menu_init(&menu, curses, menu_item_handler, "Выберите пункт
меню", 8, menu_items);
      if(res == EOK)
      {
            res = mcurses_menu_run(menu);
            if(res != EOK)
                  printf("%s: %d\n", "Ошибка при отображении меню", res);
            mcurses_menu_close(&menu);
      }
      else
            printf("%s: %d\n", "Ошибка при инициализации меню", res);

      mcurses_restore_term(curses);
      mcurses_close(&curses);
      return(0);
}
```

Алфавитный указатель классов

Классы

Классы с их кратким описанием.

Список файлов

Файлы

Полный список документированных файлов.

Классы

Структура mcurses_key_t

Структура описания нажатой клавиши
`#include <minicurs.h>`

Открытые атрибуты

- int code
  Код символа
- int is_esc
  Это Esc-последовательность?

Подробное описание

Структура описания нажатой клавиши

Объявления и описания членов структуры находятся в файле:

- minicurs/minicurs.h

Структура mcurses_menu_item_t

Структура элемента меню
`#include <minicurs.h>`

Открытые атрибуты

- char text [MAX_MENU_ITEM]
  Отображаемое имя элемента меню
- char sym
  Символ быстрого доступа
- struct {
- char str [MAX_MENU_STR]
  Хранимая строка
- uint32_t int_data
  Целочисленные данные
- float float_data
  Данные с плавающей точкой
- } item
  Содержимое элемента меню

Подробное описание

Структура элемента меню

Объявления и описания членов структуры находятся в файле:

- minicurs/minicurs.h

Структура mcurses_menu_t

Структура описания меню
`#include <minicurs.h>`

Открытые атрибуты

- minicurses_t * mcurses
  Указатель на описание интерйеса вывода на консоль
- char header [MAX_MENU_ITEM]
  Заголовок меню
- size_t count
  Количество элементов меню
- size_t allocated
  Количество элементов, под которое выделена память
- int selected
  Выбранный пункт меню
- int width
  Ширина меню
- int **first_line**
- int **last_line**
- int cur_line
  Первая, последняя, текущая строки окна меню
- int **first_item**
- int last_item
  Первый и последний отображаемые элементы меню
- mcurses_key_handler_t key_func
  Функция-обработчик нажатия клавиш
- mcurses_menu_item_handler_t item_func
  Функция-обработчик элемента меню
- mcurses_menu_item_t * items
  Массив элементов меню

Подробное описание

Структура описания меню

Объявления и описания членов структуры находятся в файле:

- minicurs/minicurs.h

Структура minicurses_t

Структура для хранения вспомогательной информации
`#include <minicurs.h>`

Открытые атрибуты

- struct termios newt oldt
  Переменные для инициализации терминала, для чтения вводимых символов
- int is_term_set
  Установлен ли терминал в raw.
- int lines
  Количество строк окна терминала
- int columns
  Количество колонок она терминала
- char term_type [80]
  Строка-идентификатор типа терминала
- int is_region_set
  Установлена ли областть прокрутки

Подробное описание

Структура для хранения вспомогательной информации

Объявления и описания членов структуры находятся в файле:

- minicurs/minicurs.h

Файлы

Файл minicurs/minicurs.h

Интерфейс для вывода на консоль с использованием Esc-последовательностей VT100.

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <termios.h>
#include <unistd.h>
#include <string.h>
#include <stdint.h>
#include <ctype.h>
```

Классы

- struct minicurses_t
  Структура для хранения вспомогательной информации
- struct mcurses_key_t
  Структура описания нажатой клавиши
- struct mcurses_menu_item_t
  Структура элемента меню
- struct mcurses_menu_t
  Структура описания меню

Макросы

- #define ESC_BEGIN "\033["
  Начало произвольной последовательности Esc[.
- #define ESC_NB "\033"
  Начало произвольной последовательности Esc.
- #define ESC_SGR1 ESC_BEGIN "1m"
  Жирный текст
- #define ESC_SGR4 ESC_BEGIN "4m"
  Подчеркнутый текст
- #define ESC_SGR5 ESC_BEGIN "5m"
  Мерцающий текст
- #define ESC_SGR7 ESC_BEGIN "7m"
  Инвертированный текст (черный на белом)
- #define ESC_SGR0 ESC_BEGIN "0m"
  Очистить изменения атрибутов текста
- #define ESC_ED2 ESC_BEGIN "H" ESC_BEGIN "2J"
  Очистка экрана с перемещением курсора в левый верхний угол
- #define ESC_EL0 ESC_BEGIN "K"
  Очистка текущей строки от курсора и до конца
- #define ESC_EL2 ESC_BEGIN "2K"
  Очистка текущей строки от начала и до конца
- #define ESC_HOM ESC_BEGIN "H"
  Перемещение курсора в левый верхний угол
- #define ESC_IND ESC_NB "D"

Перемещение курсора на одну строку вниз с прокруткой

- #define ESC_RI ESC_NB "M"
  Перемещение курсора на одну строку вверх с прокруткой
- #define ESC_CPL ESC_BEGIN "F"
  Перемещение курсора в начало предыдущей строки с прокруткой
- #define ESC_CNL ESC_BEGIN "E"
  Перемещение курсора в начало следующей строки с прокруткой
- #define ESC_DECSC ESC_NB "7"
  Сохранить позицию и параметры курсора
- #define ESC_DECRC ESC_NB "8"
  Восстановить позицию и параметры курсора
- #define ESC_RIS ESC_NB "c"
  Сбросить терминал к исходному состоянию
- #define ESC_CUU ESC_BEGIN "A"
  Перемещение курсора на одну строку вверх
- #define ESC_CUD ESC_BEGIN "B"
  Перемещение курсора на одну строку вниз
- #define ESC_SD ESC_BEGIN "S"
  Перемещение курсора на одну строку вниз с прокруткой
- #define ESC_SU ESC_BEGIN "T"
  Перемещение курсора на одну строку вверх с прокруткой
- #define ESC_HIDE_CUR ESC_BEGIN "?25l"
  Скрытие курсора
- #define ESC_SHOW_CUR ESC_BEGIN "?25h"
  Отображение курсора
- #define ESC_SYM '\033'
  Символ начала Esc-последовательности
- #define BRK_SYM '['
  Второй символ начала Esc-последовательности
- #define FIN_SYM '~'
  Возможный завершающий символ Esc-последовательности
- #define ENTER_SYM '\n'
  Символ выбора экранного элемента (<Enter>)
- #define UP_SYM 'A'
  Курсор вверх
- #define DN_SYM 'B'
  Курсор вниз
- #define RT_SYM 'C'
  Курсор вправо
- #define LT_SYM 'D'
  Курсор влево
- #define PGDN_SYM 'U'
  Страница вниз
- #define PGUP_SYM 'V'
  Страница вверх
- #define HOME_SYM 'H'
  В начало
- #define END_SYM 'Y'

В конец

- #define PGDN_DIG '6'
  Страница вниз (число)
- #define PGUP_DIG '5'
  Страница вверх (число)
- #define HOME_DIG '1'
  В начало (число)
- #define END_DIG '4'
  В конец (число)
- #define QUIT_SYM 'Q'
  Символ выхода из экранного элемента (Q)
- #define qUIT_SYM 'q'
  Символ выхода из экранного элемента (q)
- #define MAX_MENU_ITEM 72
  Максимальный размер отображаемого имени элемента меню
- #define MAX_MENU_STR 255
  Максимальный размер хранимой строки элемента меню
- #define MENU_CMD_EXIT "EXITMENU"
  Команда выхода из меню

Определения типов

- typedef int(* mcurses_key_handler_t) (uint32_t handle, mcurses_key_t *key)
- typedef int(* mcurses_menu_item_handler_t) (char *str, uint32_t int_data, float float_data)

Перечисления

- enum txt_attr { ta_normal = 0, ta_bold = 1, ta_underline = 2, ta_blink = 4, ta_inv = 8 }
  Атрибуты текста

Функции

- int minicurses_init (minicurses_t *mcurses)
- int minicurses_set_term (minicurses_t *mcurses)
- int minicurses_restore_term (minicurses_t *mcurses)
- int mcurses_init (uint32_t *handle)
- int mcurses_close (uint32_t *handle)
- int mcurses_set_term (uint32_t handle)
- int mcurses_restore_term (uint32_t handle)
- int mcurses_get_lines (uint32_t handle, int *lines)
- int mcurses_get_columns (uint32_t handle, int *columns)
- int mcurses_cls (uint32_t handle)
- int mcurses_clstr (uint32_t handle)
- int mcurses_set_region (uint32_t handle, int from, int to)
- int mcurses_reset_region (uint32_t handle)
- int mcurses_reset_term (uint32_t handle)
- int mcurses_move_next_line (uint32_t handle)
- int mcurses_move_prev_line (uint32_t handle)
- int mcurses_set_cursor_to (uint32_t handle, int line, int col)
- int mcurses_set_cursor_to_col (uint32_t handle, int col)
- int mcurses_get_cursor_pos (uint32_t handle, int *line, int *col)
- int mcurses_move_down (uint32_t handle)
- int mcurses_move_up (uint32_t handle)
- int mcurses_scroll_down (uint32_t handle, int lines=1)

- int mcurses_scroll_up (uint32_t handle, int lines=1)
- int mcurses_save_cursor (uint32_t handle)
- int mcurses_restore_cursor (uint32_t handle)
- int mcurses_hide_cursor (uint32_t handle)
- int mcurses_show_cursor (uint32_t handle)
- int mcurses_put_text (uint32_t handle, char *text, int attr=ta_normal)
- int mcurses_put_string (uint32_t handle, char *text, int attr=ta_normal)
- int mcurses_menu_key_handler (uint32_t handle, mcurses_key_t *key)
- int mcurses_menu_init (uint32_t *handle, uint32_t mcurses, mcurses_menu_item_handler_t item_func, char *header, size_t items_count=0, mcurses_menu_item_t *items=NULL, mcurses_key_handler_t key_func=mcurses_menu_key_handler)
- int mcurses_menu_close (uint32_t *handle)
- int mcurses_menu_add (uint32_t handle, char *text, char sym, char *str, uint32_t int_data, float float_data)
- mcurses_menu_item_t * mcurses_menu_item (uint32_t handle, int item)
- mcurses_menu_item_t * mcurses_menu_by_text (uint32_t handle, char *text)
- mcurses_menu_item_t * mcurses_menu_by_sym (uint32_t handle, char sym)
- int mcurses_menu_run (uint32_t handle)
- int mcurses_menu_display (uint32_t handle)
- int mcurses_menu_item_display (uint32_t handle, int item, int selected)
- int mcurses_menu_define_width (uint32_t handle)

Подробное описание

Интерфейс для вывода на консоль с использованием Esc-последовательностей VT100.

Интерфейс для вывода на консоль с использованием Esc-последовательностей VT100, аналог ncurses без излишеств и извращений.

Типы

typedef int(* mcurses_key_handler_t) (uint32_t handle, mcurses_key_t *key)

Тип функции-обработчика нажатий клавиш

Аргументы:

in	*uint32_t*	handle - дескриптор экранного элемента
in	*mcurses_key_t*	*key - описание нажатой клавиши

Возвращает:

Функция должна возвращать 1 для продолжения работы экранного элемента, либо 0 для выхода

typedef int(* mcurses_menu_item_handler_t) (char *str, uint32_t int_data, float float_data)

Тип функции-обработчика, вызываемой при выборе пользователем элемента меню

Аргументы:

in	*char*	*str - хранимая строка элемента меню
in	*uint32_t*	int_data - хранимые целочисленные данные элемента меню
in	*float*	float_data - хранимые данные с плавающей точкой элемента меню

Возвращает:

Функция должна возвращать 1 для продолжения работы меню, либо 0 для выхода

Перечисления

enum txt_attr

Атрибуты текста

Элементы перечислений

ta_normal Обычный текст

ta_bold Жирный

ta_underline Подчеркнутый

ta_blink Мерцающий

ta_inv Инвертированный (черный на белом)

Функции

int mcurses_close (uint32_t * *handle*)

Закрытие интерфейса вывода на консоль с созданием дескриптора

Аргументы:

in,out	*uint32_t*	handle - указатель на дескриптор интерфейса

Возвращает:

EOK в случае успеха
EINVAL, если указатель не инициализирован
признак ошибки

int mcurses_cls (uint32_t *handle*) `[inline]`

Очистка экрана

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_clstr (uint32_t *handle*) `[inline]`

Очистка строки

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_get_columns (uint32_t *handle*, int * *columns*) `[inline]`

Получение текущего количества колонок на экране

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
out	*int*	*columns - указатель на переменную под количество строк

Возвращает:

EOK в случае успеха
EINVAL, если указатель не инициализирован

int mcurses_get_cursor_pos (uint32_t *handle*, int * *line*, int * *col*)

Получить текущую позицию курсора

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
out	*int*	*line - указатель на буфер под номер строки
out	*int*	*col - указатель на буфер под номер колонки

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_get_lines (uint32_t *handle*, int * *lines*) `[inline]`

Получение текущего количества строк на экране

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
out	*int*	*lines - указатель на переменную под количество строк

Возвращает:

EOK в случае успеха
EINVAL, если указатель не инициализирован

int mcurses_hide_cursor (uint32_t *handle*) `[inline]`

Скрыть курсор

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_init (uint32_t * *handle*)

Инициализация интерфейса вывода на консоль с созданием дескриптора

Аргументы:

in,out	*uint32_t*	*handle - указатель на дескриптор интерфейса

Возвращает:

EOK в случае успеха
EINVAL, если указатель не инициализирован
признак ошибки

int mcurses_menu_add (uint32_t *handle*, char * *text*, char *sym*, char * *str*, uint32_t *int_data*, float *float_data*)

Добавление пункта меню

Аргументы:

in	*uint32_t*	handle - дескриптор меню
in	*char*	*text - отображаемый текст
in	*char*	sym - символ быстрого доступа (0 - отключает быстрый доступ)
in	*char*	*str - хранимая строка

in	*uint32_t*	int_data - хранимые целочисленные данные
in	*float*	float_data - хранимые данные с плавающей точкой

Возвращает:

EOK в случае успеха
EINVAL, если указатели не инициализированы
признак ошибки

mcurses_menu_item_t* mcurses_menu_by_sym (uint32_t *handle*, char *sym*)

Доступ к пункту меню по символу быстрого доступа

Аргументы:

in	*uint32_t*	handle - дескриптор меню
in	*char*	sym - символ быстрого доступа

Возвращает:

указатель на пункт меню
NULL в случае ошибки

mcurses_menu_item_t* mcurses_menu_by_text (uint32_t *handle*, char * *text*)

Доступ к пункту меню по отображаемому тексту

Аргументы:

in	*uint32_t*	handle - дескриптор меню
in	*char*	*text - отображаемый текст

Возвращает:

указатель на пункт меню
NULL в случае ошибки

int mcurses_menu_close (uint32_t * *handle*)

Закрытие экранного меню

Аргументы:

in,out	*uint32_t*	*handle - указатель на дескриптор меню

Возвращает:

EOK в случае успеха
EINVAL, если указатели не инициализированы
признак ошибки

int mcurses_menu_define_width (uint32_t *handle*)

Определение ширины меню

Аргументы:

in	*uint32_t*	handle - дескриптор меню

Возвращает:

EOK в случае успеха
признак ошибки

int mcurses_menu_display (uint32_t *handle*)

Отображение меню

Аргументы:

in	*uint32_t*	handle - дескриптор меню

Возвращает:

EOK в случае успеха

признак ошибки

int mcurses_menu_init (uint32_t * *handle*, uint32_t *mcurses*, mcurses_menu_item_handler_t *item_func*, char * *header*, size_t *items_count* = 0, mcurses_menu_item_t * *items* = NULL, mcurses_key_handler_t *key_func* = mcurses_menu_key_handler)

Инициализация экранного меню с созданием дескриптора

Аргументы:

in,out	*uint32_t*	*handle - указатель на дескриптор меню
in	*uint32_t*	mcurses - дескриптор интерфейса
in	*mcurses_menu_item_handler_t*	item_func - функция-обработчик пункта меню
in	*char*	*header - заголовок меню
in	*size_t*	items_count - количество элементов меню (по умолчанию 0)
in	*mcurses_menu_item_t*	*items - массив элементов меню (по умолчанию не инициализирован)
in	*mcurses_key_handler_t*	key_func - функция-обработчик нажатий клавиш (по умолчанию - стандартный)

Возвращает:

EOK в случае успеха
EINVAL, если указатели не инициализированы
признак ошибки

mcurses_menu_item_t* mcurses_menu_item (uint32_t *handle*, int *item*)

Доступ к пункту меню по порядковому номеру

Аргументы:

in	*uint32_t*	handle - дескриптор меню
in	*int*	item - порядковый номер пункта

Возвращает:

указатель на пункт меню
NULL в случае ошибки

int mcurses_menu_item_display (uint32_t *handle*, int *item*, int *selected*)

Отображение пункта меню

Аргументы:

in	*uint32_t*	handle - дескриптор меню
in	*int*	item - порядковый номер пункта
in	*int*	selected - выделить (1) или нет (0) пункт

Возвращает:

EOK в случае успеха
признак ошибки

int mcurses_menu_key_handler (uint32_t *handle*, mcurses_key_t * *key*) [inline]

Стандартная функция-обработчик нажатий клавиш в меню

Аргументы:

in	*uint32_t*	handle - дескриптор меню
in	*mcurses_key_t*	*key - описание нажатой клавиши

Возвращает:

1 для продолжения работы экранного элемента, либо 0 для выхода

int mcurses_menu_run (uint32_t *handle*)

Отображение и запуск меню

Аргументы:

in	*uint32_t*	handle - дескриптор меню

Возвращает:

EOK в случае успеха
признак ошибки

int mcurses_move_down (uint32_t *handle*) `[inline]`

Переместить курсор на следующую строку с прокруткой

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_move_next_line (uint32_t *handle*) `[inline]`

Переместить курсор в начало следующей строки с прокруткой

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_move_prev_line (uint32_t *handle*) `[inline]`

Переместить курсор в начало предыдущей строки с прокруткой

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_move_up (uint32_t *handle*) `[inline]`

Переместить курсор на предыдущую строку с прокруткой

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_put_string (uint32_t *handle*, char * *text*, int *attr* = `ta normal`)

Вывести строку на консоль с указанными атрибутами, с предварительной очисткой строки

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
in	*char*	*text - текстовая строка для вывода
in	*int*	attr - атрибуты текста (по умолчанию обычный текст)

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_put_text (uint32_t *handle*, char * *text*, int *attr* = `ta normal`)

Вывести текст на консоль с указанными атрибутами

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
in	*char*	*text - текстовая строка для вывода
in	*int*	attr - атрибуты текста (по умолчанию обычный текст)

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_reset_region (uint32_t *handle*) `[inline]`

Сбросить границы прокрутки экрана

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_reset_term (uint32_t *handle*) `[inline]`

Сбросить терминал к исходному состоянию

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_restore_cursor (uint32_t *handle*) `[inline]`

Восстановить позицию и параметры курсора

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_restore_term (uint32_t *handle*) `[inline]`

Восстановление настроек терминала при завершении считывания нажатий клавиш

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха

int mcurses_save_cursor (uint32_t *handle*) `[inline]`

Сохранить позицию и параметры курсора

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_scroll_down (uint32_t *handle*, int *lines* = 1)[inline]

Прокрутить список вниз

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
in	*int*	lines - количество строк (по умолчанию 1)

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_scroll_up (uint32_t *handle*, int *lines* = 1)[inline]

Прокрутить список вверх

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
in	*int*	lines - количество строк (по умолчанию 1)

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_set_cursor_to (uint32_t *handle*, int *line*, int *col*)[inline]

Переместить курсор в указанную позицию

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
in	*int*	line - номер строки, от 1 до 24
in	*int*	col - номер колонки, от 1 до 80

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_set_cursor_to_col (uint32_t *handle*, int *col*)[inline]

Переместить курсор в указанную колонку в текущей строке

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
in	*int*	col - номер колонки, от 1 до 80

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int mcurses_set_region (uint32_t *handle*, int *from*, int *to*)[inline]

Установить границы прокрутки экрана

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса
in	*int*	from - номер первой строки, начиная с 1

in	*int*	to - номер последней строки, начиная с 1, больше чем from+1

Возвращает:

EOK в случае успеха
EINVAL в случае неверных параметров
ошибку вывода на консоль

int mcurses_set_term (uint32_t *handle*) `[inline]`

Настройка терминала для считывания нажатий клавиш

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха

int mcurses_show_cursor (uint32_t *handle*) `[inline]`

Показать курсор

Аргументы:

in	*uint32_t*	handle - дескриптор интерфейса

Возвращает:

EOK в случае успеха
ошибку вывода на консоль

int minicurses_init (minicurses_t * *mcurses*)

Инициализация интерфейса вывода на консоль

Аргументы:

in,out	*minicurses_t*	*mcurses - указатель на структуру описания интерфейса

Возвращает:

EOK в случае успеха
EINVAL, если указатель не инициализирован

int minicurses_restore_term (minicurses_t * *mcurses*)

Восстановление настроек терминала при завершении считывания нажатий клавиш

Аргументы:

in	*minicurses_t*	*mcurses - указатель на структуру описания интерфейса

Возвращает:

EOK в случае успеха
EINVAL, если указатель не инициализирован или терминал не был настроен
ошибка настройки терминала

int minicurses_set_term (minicurses_t * *mcurses*)

Настройка терминала для считывания нажатий клавиш

Аргументы:

in	*minicurses_t*	*mcurses - указатель на структуру описания интерфейса

Возвращает:

EOK в случае успеха
EINVAL, если указатель не инициализирован или терминал уже настроен
ошибка настройки терминала