

ECurses : an Eiffel binding to Curses

Paul G. Crismer, Eric Fafchamps
<pgcrism@users.sourceforge.net>, <efa@users.sourceforge.net>

March 10, 2009

Abstract

ECurses is an Eiffel binding to Curses. It tries to be more than just a wrapping of the C library. It does not implement all the features of Curses, but provides enough functionality to allow development of nice console applications.
ECurses has been designed to be portable among different platforms, i.e. on Unix/Linux (Curses, NCurses) and Windows (PDCurses).

1 Introduction

Description

ECurses is an Eiffel binding to Curses. It is intended to be sufficiently complete to enable people write terminal/console applications in Eiffel.

The following table gives an overview of what ECurses provides, compared to the Curses library.

<i>Curses</i>	<i>Ecurses</i>	<i>Eiffel Class</i>
Curses core	Partial	CURSES_WINDOW
* color	yes	
* character attributes	yes	
* subwindows	yes	
* scrolling	yes	
* mouse interfacing	no	
* multiple terminal screen	no	
* testing for terminal capabilities	no	
Pads	Partial	CURSES_PAD
Soft label keys	yes	CURSES_SOFT_LABEL_KEYS
Panels	Yes	CURSES_PANEL
Menu	no	
Forms	no	

ECurses introduces some level of abstraction. Feature names try to be more explicit than their Curses counterpart. When possible pre-and post-conditions reflect the features specifications.

ECurses has not been ``extensively" tested. Though, it works sufficiently well to develop applications with color, scrolling, multiple panels, and basic forms.

This document is not a Curses manual. Please refer to appropriate man pages or to the excellent article ``Writing programs with NCURSES" by Eric S. Raymond and Zeyd M. Ben-Halim.

Software dependencies

Current version of eCurses depends on the following packages:

- Eiffel compiler
 - ISE EiffelStudio 6.3
 - or Gobo Eiffel Compiler 3.9
- Libraries
 - Gobo Eiffel 3.9
 - Curses library:
 - Linux: nCurses
 - windows: PDCurses

2 Installation

2.1 Unpack directory structure

Simply extract the tarball. Directory structure is described below :

base base Curses abstractions.

doc documentation; this file in various formats.

spec API and constants wrapping classes.

Clib c source files.

ise ISE specific files.

linux Linux specific files.

windows Windows specific files.

ge Gobo Eiffel specific files.

linux Linux specific files.

windows Windows specific files.

Utility some classes found useful to develop ECurses applications like an event manager and a frame panel.

test small test projects.

hello hello world !

- pad** exercise basic pad capabilities.
- slk** exercise soft label keys.
- curses** mimics the 'ncurses' standard test application.
- compile_test** compatibility test - forces compilation of all features.

2.2 Get the appropriate libraries

Get and install the appropriate Curses library.

NCURSES <http://www.gnu.org>. Work on Unix/Linux platforms.

PDCurses <http://pdcurses.sourceforge.net/>. Works on Windows platform. PDCurses also offer an X11 implementation for Unix, but it has not been tested with the ECurses library.

Get and install the GOBO library

GOBO <http://www.gobosoft.com>. Works on Unix/Linux/win32 platforms for ISE Eiffel and Gobo Eiffel.

2.3 Set some environment variable

GOBO Set to the root-directory of the gobo installation (e.g: ~/libs/eiffel/gobo)

ECURSES Set to the root-directory of the ecurses installation (e.g: ~/libs/eiffel/ecurses)

PDCURSES Set to the root-directory of the pdcurses installation (e.g: c:\libs\c\pdcurses)

2.4 Compile support library

C files are in the spec/Clib directory. The spec/<compiler>/<platform> directory contains ad hoc makefiles.

windows A makefile (Makefile) is provided for MSVC compiler. Library name is ecurses_c.lib.

linux A makefile (Makefile) is provided for Linux . Type "make all" in order to build the library libecurses_c.a.

This support library is necessary to compile and link ECurses programs .

3 Usage

This section gives an overview of what are the main abstractions of ECurses.

3.1 Hello World !

```
class
  HELLO_WORLD

inherit
  CURSES_APPLICATION

creation
  make

feature

  make is
    do
      initialize
      standard_window.move (standard_window.height // 2,
standard_window.width // 2)
      standard_window.put_string ("Hello, World")
      standard_window.move (standard_window.height-1, 0)
      standard_window.put_string ("Press any key....")
      standard_window.read_character
    end

end

end -- class  HELLO_WORLD
```

CURSES_APPLICATION must be inherited by an application's root class. The first feature to be called must be one of the initializers. Initialize is the basic one. Others initialize Curses so that it provides soft label keys at the bottom of the screen. An initialized ECurses application has a standard_window, a standard_panel, and soft_label_keys.

CURSES_WINDOW is the "basic" window class. It has many features which cannot all be enumerated here. They are related to window creation or destruction, positioning, update, status, cursor position, color setting, window input and output, scrolling, line and border drawing, graphic attributes setting, subwindows ...

3.2 Clusters description

3.2.1 Cluster BASE

CURSES_APPLICATION Inherited by the application's root class. It provides access to the standard (default) window and panel.

CURSES_WINDOW Main abstraction. A window is a rectangular portion on the screen. Internally, each window has a cell-memory where input/output operations occurs. Actual screen i/o occur only when a window is refreshed. A window can have several subwindows. Each subwindow is a window of its own excepted that it shares the same cell-memory as its parent: refreshing the parent refreshes all the subwindows.

CURSES_ERROR_HANDLING Interface through which each low-level Curses call transits. Provides handling of return codes.

CURSES_PAD A pad is a rectangular area which is viewed through a view, i.e. a smaller-than-the-pad rectangle. The view is displayed on the screen. A pad can be larger than the terminal screen, but its view cannot.

CURSES_PANEL A panel is a window which can be moved, hidden, shown. Panels can be overlapped, while windows cannot easily.

CURSES_SOFT_LABEL_KEYS Soft label keys, viewable at the bottom of the screen. Several soft label keys organizations are available at initialization., but they are limited to the capabilities of the library (PDCurses or NCurses) or of the terminal.

CURSES_SYSTEM Curses state an behaviour not related to a pad, panel or window.

SHARED_CURSES_SYSTEM The ``Curses system" is a singleton accessible to all ECurses abstractions.

3.2.2 Cluster SPEC

CURSES_ATTRIBUTE_CONSTANTS Output attribute constants, like blink, underline.

CURSES_CHARACTER_CONSTANTS Constants reflecting special terminal characters like graphics blocks, etc...

CURSES_COLOR_CONSTANTS Color constants

CURSES_KEY_CONSTANTS Constants that reflect key codes like function keys, cursor keys, keypad keys.

`CURSES_*_API` Those classes encapsulate the external features accessing the various aspects of the Curses library.

3.2.3 Cluster UTILITY

`CURSES_EVENT_MANAGER` Basic event management logic. This class is intended to be inherited and redefined.

`CURSES_FRAME_PANEL` Panel with a border and a clipped client area (borders are not overwritten).

4 Questions and Answers

4.1 Status of the library

It is the first publicly available version. Work still has to be done in various directions like : assertions, documentation, tests. Still, it works...

ECurses has not been tested systematically. It has been used to develop some nice terminal applications. Only those features necessary to these applications has been tested.

ECurses does not provide access to all the features of Curses. Applications need to be developed to get interesting feedback and evolution of this library.

4.2 C wrapping

On the C side, Curses functions are implemented either as functions or as macros. Library implementors make their own choice : a function implemented as a macro in PDCurses is actually a function in NCurses. That's why we have chosen for simple C wrapping rules applicable to both implementations.

We also wanted this wrapping to be as portable as possible among Eiffel compilers.

For each "function" F that can be called, a `c_ecurses_F` function is defined. This function basically cares that argument types are converted the right way.

This mechanism is tedious (to type) but rather straightforward. External clauses are simple, like defined in ETL (hope that VE can handle it...) :

Eiffel side (example)

```
F (w : POINTER; y, x : INTEGER): INTEGER is
  external "C"
  alias "c_ecurses_F"
end
```

C side

```
EIF_INTEGER c_ecurses_F (EIF_POINTER w, EIF_INTEGER y, EIF_INTEGER x)
{
  return (EIF_INTEGER) F((WINDOW*)w, (int)y, (int) x);
}
```

This C wrapping has been implemented for all CURSES_*_API classes.

4.3 Portability

There are two kind of portability :

1. Curses library portability
2. Eiffel portability

Ecurses has been tested with PDCurses on Windows and NCurses on Linux.

Ecurses has been compiled using ISE Eiffel version 4.4 on Linux (Caldera 2.3, kernel 2.210, glibc2.1, egcs 2.91.66), version 4.3 and 4.4 on Windows (MSVC Compiler).

Ecurses has been compiled using SmartEiffel version 1.1 on Linux (Caldera 2.3, kernel 2.210, glibc2.1, egcs 2.91.66), version 1.1 on Windows (MSVC Compiler).

4.4 Console and X terminal emulations

Curses applications are verrrry sensible to the choice of good TERMCAPS entries. Before saying ``it does not work'', verify your console or terminal settings.

Some X terminal emulations won't give expected behaviour to your applications. Some characters do not appear because they do not exist in the used font. Some keys are not recognized because of an incorrect termcap entry.

Terminal emulators like xterm or xterm-color are known to work best. Running your application in console mode can help you see how it should work in your X terminal emulator.

4.5 Reading from the keyboard

A CURSES_WINDOW object has features like `read_character` and `read_line`.
A `read_character` always give two results : `last_character`, and `last_key`,
which is an integer representing the key code.

If you want to test for key codes, do not forget to `enable_metacharacters`
on a window so that function or cursor keys end up in one code value.
Otherwise, you should get an escape sequence.

If the keypad is to be used, use the feature `enable_keypad`.

5 TODO

- * Software contracts could be extended and better reflect features specifications.
- * Documentation and sample programs should allow people to use ECurses, without knowing much of Curses.

6 Copyright

This document and the ECurses library are licensed under the terms
of the Eiffel Forum Freeware License, version 2. Please refer to the file `licence.txt` in this
package or at <http://www.gnu.org/licences/eiffel-forum-licence-2.html>

References

NCurses

<http://www.gnu.org/software/ncurses/ncurses.html>

<http://en.wikipedia.org/wiki/NCurses>

``Writing programs with NCURSES" by Eric S. Raymond and Zeyd M. Ben-Halim.

<http://invisible-island.net/ncurses/ncurses-intro.html>

man pages for NCurses

A Hacker's Guide to NCURSES by Eric S. Raymond

<http://invisible-island.net/ncurses/hackguide.html>

NCurses FAQ

<http://invisible-island.net/ncurses/ncurses.faq.html>