www.linuxjournal.com /article/1124

Ncurses: portable screen-handling for Linux

Software

by Eric S. Raymond

on September 1, 1995

Since you are clueful enough to subscribe to *Linux Journal*, you may already know that Linux includes a clone of the Unix screen-handling library, curses(3). You may also know that Linux's ncurses(3), like its ancestor, is a general screen-handling library that supports all popular asynchronous terminal types, including the color-ANSI and vt100-like capabilities of most Linux consoles. You've certainly run a program that used curses, vi, perhaps, or any of a dozen screen-oriented games.

You may not know that there are actually two major flavors of curses, with differences that are important for portability. You probably don't know (unless you have read ahead) that post-1.9 releases of ncurses include dramatic new features that will make them far more reliable, powerful, and useful than their predecessors.

curses History

The first curses library was hacked together at the University of California at Berkeley in about 1980 to support a screen-oriented dungeon game called rogue. It leveraged an earlier facility called **termcap**, the terminal capability library, which was used in the vi editor and elsewhere.

Termcap consisted of a simple text database format for describing the control sequences and capabilities of serial terminals. Before the ANSI standard hundreds of mutually-incompatible terminal types competed in the market; termcap described them in a standard way that made it possible for vi's screen-painting code to be terminal-independent.

The curses library took this one step further, by defining a relatively clean C API (applications programming interface) to hide the details of termcap-based screen painting. This facility was a terrific success, and gets almost all the credit for lifting Unix out of the line-oriented interface style inherited from its early days on ASR-33 Teletypes.

Enhancements and Controversy

The success of curses did not go unnoticed at Bell Labs. Later System III releases and System V Release 1 included an enhanced curses library with many new features. These included:

- Support for multiple screen highlights (BSD curses could only handle one “standout” highlight, usually reverse-video).

- Support for line- and box-drawing using forms characters.
- Recognition of function keys on input.
- Color support (in later versions).
- Full support for sub-windows.
- Support for pads (windows of larger than screen size on which the screen or a sub-window defines a viewport).

The Bell Labs curses was upward-compatible from BSD curses, and much more powerful. However, its designers made one controversial major change; they scrapped the all-text termcap format for terminal capability descriptions, opting instead for a binary format called terminfo.

The most important reason for changing formats was that the BSD termcap database was getting so large that sequential searching took significant overhead. The terminfo format (binary capability blocks living in a bushy directory tree), by contrast, was optimized for fast lookup and fast loading.

In doing changing to a binary format, however, Bell Labs curses gave up two valuable features: extensibility and the ability to edit terminal descriptions with standard text tools. The termcap routines had never actually cared what the capabilities meant to the program using them, whether it was curses or something else. Thus, it was easy to add new capabilities and new interpretations; in fact, later BSD releases used the same code for printer- and modem-capability databases. In terminfo, by contrast, additions to the database format required a recompile of the library.

AT&T's no-source-code policy meant that Bell Labs curses itself never got a chance to outcompete the BSD version in the hacker community. Worse, the tradeoffs in the terminfo format meant that the largely BSD-centric hacker culture had a passable excuse to denigrate the major new capabilities in Bell Labs curses.

For more than five years, then, freeware authors writing screen-oriented programs were in the vexing situation of having to design around the older, less-capable BSD interface even on machines that supported the Bell Labs curses.

Source Code Freedom

Around 1982, noted hacker Pavel Curtis (formerly of Xerox PARC, now perhaps best known for his MOO project) attacked this problem head-on by starting work on a freeware clone of Bell Labs curses. This package was part of Pavel's personal toolkit; it was not widely distributed, or even very well known, until Zeyd Ben-Halim, zmbenhal@netcom.com, took over the development effort in late 1991. I got involved in late 1993 to support a screen-oriented multi-user Unix BBS I was developing.

The early ncurses versions had some serious drawbacks. The biggest problem was that a good many important Bell Labs curses capabilities were absent, leaving annoying and unpredictable gaps in the API. The documentation was poor and not suited to on-line browsing. Very little systematic compatibility testing against Bell Labs curses had been done. Last, but not least, all versions up to 1.8.5 had serious bugs.

Quality Improvements

Much has changed in the last year, out of sight of the Linux community at large. We have fixed many bugs and filled out the API. We've written more complete documentation, including man pages. We've done extensive compatibility testing against SVR4, linking identical programs to both libraries and comparing behavior. We've audited the code carefully for cross-platform portability and replaced a rather clunky home-brewed configuration system with GNU autoconf. We've tested the code for memory leaks with Purify and improved argument validation in many critical functions. The overall quality of the code has been dramatically improved.

We've also added several Bell Labs-like utilities missing from older ncurses versions, including:

- infocmp(1) a terminfo entry lister and comparator.
- captoinfo(1) a termcap to terminfo translator.
- clear(1) a trivial screen clearer.
- tput(1) a terminfo capability access for shell scripts.

The captoinfo, clear, and tput utilities are based on code from Ross Ridge's mytinfo package (which we've effectively subsumed).

We chose System V Release 4.0 curses as our emulation target, and now support all of its very extensive features. We also include a clone of the SVR4 panels library, a curses extension that makes it easy to program stacks of windows with backing store.

The new ncurses package also includes a full and up-to-date set of man pages organized similarly to SVR4's.

In a few areas, we go beyond SVR4 curses, for example:

- The new cursor-movement optimizer and incremental-screen-update algorithms are smarter than the Bell Labs versions (and *much* smarter than the BSD versions) leading to significant update-speed gains for slow terminals.
- Unlike previous curses versions, ncurses can write the lower right hand corner cell of a terminal with automargin wrap (provided it has an insert-character capability).
- On Intel boxes, curses permits you to display not just the IBM high-half characters but also the ROM graphics in characters 0-31.
- Our terminfo tools recognize all the terminfo and termcap extensions found in GNU termcap, mytinfo, and the University of Waterloo libraries.
- We feature a C++ class derived from the libg++ CursesWindow class, but enhanced for ncurses.
- The ncurses suite now includes an ncurses program which allows you explicitly test most of ncurses's capabilities on any new platform.

- We've souped up the tracing feature. It is now possible to pick one of several levels of output tracing, and the trace log has been made easier to interpret (with symbolic dumping of screen attributes, colors, and so forth).
- Automatic fallback to the /etc/termcap file can be compiled in for systems without a terminfo tree. This feature is neither fast nor cheap, so you don't want to use it unless you have to.

Along with ncurses, you get a very complete terminfo file. I accepted the maintainer's baton for the 4.4BSD master termcap file from John Kunze in January 1995; I've since translated it to terminfo and added a lot of information from vendors like SCO, Digital Equipment Corporation, and Wyse. (The terminfo file is available separately from my WWW home page, www.ccil.org/~esr/home.html.)

How Do I Use ncurses?

Listing 1 gives a program skeleton that illustrates how to set up to use some of the new features.

Note the way we arrange for **endwin()** to get called by the signal catcher. This is necessary, otherwise a stray signal might leave the tty modes in a less than useful state.

Our next program fragment illustrates the use of the keypad mapping feature. This code, from an actual test program distributed with ncurses, is shown is Listing 2.

The example code illustrates how you can get function key tokens as single values outside the ASCII range from **getch()**. The call **keypad(stdscr, TRUE)** sets this up.

The ncurses distribution contains a more detailed tutorial in HTML form and also has several code examples in the test directory.

The Future of ncurses

The ncurses library seems to have achieved a stable plateau, suitable for production use, in the 1.9.1 and 1.9.2 releases. We're looking ahead to some interesting possibilities—most notably at supporting XPG4 Curses extended-level features for wide and multi-byte character sets. We're also thinking about MS-DOS and MS-Windows ports.

With the features we already have in a freeware library, you may be wondering whether there are good reasons for BSD curses to continue to exist. The answer is “probably not”.

Keith Bostic, maintainer of BSD curses, has agreed that if ncurses 1.9 proves stable and usable with nvi (new vi), he will switch to ncurses for the nvi distribution and pronounce BSD curses officially dead. The ncurses library has already been used to successfully support nvi for production, so it's a good bet that by the time you read this, Keith will have finished his testing—and BSD curses will be history.

This, in turn, means the Unix world will finally get a common freeware API that supports color, multiple highlights, and all the other capabilities common to PC consoles and today's character-cell terminals.

Update

Since this article was written, Keith Bostic tested ncurses with nvi, and has officially pronounced the old BSD curses dead. This means that ncurses will become the official standard curses for Linux, as well as for all the myriad free versions of BSD Unix.

Eric S. Raymond , esr@snark.thyrsus.com, is the volunteer editor of the Jargon File, a prolific author of freeware and FAQs, and a recent convert to Linux. His WWW home page is available at www.ccil.org/~esr/home.html.