Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'scr_dump.5' command

$ man scr_dump.5

scr_dump(5) File Formats Manual scr_dump(5)

NAME

scr_dump - format of curses screen-dumps.

SYNOPSIS

scr_dump

DESCRIPTION

The curses library provides applications with the ability to write the contents of a window to an external file using scr_dump or putwin, and read it back using scr_restore or getwin.

The putwin and getwin functions do the work; while scr_dump and scr_re? store conveniently save and restore the whole screen, i.e., stdscr.

ncurses6

A longstanding implementation of screen-dump was revised with ncurses6 to remedy problems with the earlier approach:

? A ?magic number? is written to the beginning of the dump file, al? lowing applications (such as file(1)) to recognize curses dump files.

Because ncurses6 uses a new format, that requires a new magic num? ber was unused by other applications. This 16-bit number was un? used:

0x8888 (octal ?\210\210?)

but to be more certain, this 32-bit number was chosen:

0x88888888 (octal ?\210\210\210\210?)

This is the pattern submitted to the maintainers of the file pro?
gram:

```
#
# ncurses5 (and before) did not use a magic number,
# making screen dumps "data".
#
# ncurses6 (2015) uses this format, ignoring byte-order
0    string    \210\210\210\210ncurses    ncurses6 screen image
#
```

? The screen dumps are written in textual form, so that internal data sizes are not directly related to the dump-format, and enabling the library to read dumps from either narrow- or wide-character- con? figurations.

  The narrow library configuration holds characters and video at? tributes in a 32-bit chtype, while the wide-character library stores this information in the cchar_t structure, which is much larger than 32-bits.

? It is possible to read a screen dump into a terminal with a differ? ent screen-size, because the library truncates or fills the screen as necessary.

? The ncurses6 getwin reads the legacy screen dumps from ncurses5.

ncurses5 (legacy)

The screen-dump feature was added to ncurses in June 1995. While there were fixes and improvements in succeeding years, the basic scheme was unchanged:

? The WINDOW structure was written in binary form.

? The WINDOW structure refers to lines of data, which were written as an array of binary data following the WINDOW.

? When getwin restored the window, it would keep track of offsets into the array of line-data and adjust the WINDOW structure which was read back into memory.

This is similar to Unix SystemV, but does not write a ?magic number? to identify the file format.

PORTABILITY

There is no standard format for putwin. This section gives a brief de? scription of the existing formats.

X/Open Curses

Refer to X/Open Curses, Issue 7 (2009).

X/Open's documentation for enhanced curses says only:

> The getwin( ) function reads window-related data stored in the file by putwin( ). The function then creates and initializes a new win? dow using that data.
>
> The putwin( ) function writes all data associated with win into the stdio stream to which filep points, using an unspecified format. This information can be retrieved later using getwin( ).

In the mid-1990s when the X/Open Curses document was written, there were still systems using older, less capable curses libraries (aside from the BSD curses library which was not relevant to X/Open because it did not meet the criteria for base curses). The document explained the term ?enhanced? as follows:

> ? Shading is used to identify X/Open Enhanced Curses material, re? lating to interfaces included to provide enhanced capabilities for applications originally written to be compiled on systems based on the UNIX operating system. Therefore, the features de? scribed may not be present on systems that conform to XPG4 or to earlier XPG releases. The relevant reference pages may provide additional or more specific portability warnings about use of the material.

In the foregoing, emphasis was added to unspecified format and to XPG4 or to earlier XPG releases, for clarity.

Unix SystemV

Unix SystemV curses identified the file format by writing a ?magic num? ber? at the beginning of the dump. The WINDOW data and the lines of text follow, all in binary form.

The Solaris curses source has these definitions:

```
/* terminfo magic number */
```

#define MAGNUM 0432

/* curses screen dump magic number */

#define SVR2_DUMP_MAGIC_NUMBER 0433

#define SVR3_DUMP_MAGIC_NUMBER 0434

That is, the feature was likely introduced in SVr2 (1984), and improved in SVr3 (1987). The Solaris curses source has no magic number for SVr4 (1989). Other operating systems (AIX and HPUX) use a magic number which would correspond to this definition:

/* curses screen dump magic number */

#define SVR4_DUMP_MAGIC_NUMBER 0435

That octal number in bytes is 001, 035. Because most Unix vendors use big-endian hardware, the magic number is written with the high-order byte first, e.g.,

01 35

After the magic number, the WINDOW structure and line-data are written in binary format. While the magic number used by the Unix systems can be seen using od(1), none of the Unix systems documents the format used for screen-dumps.

The Unix systems do not use identical formats. While collecting infor? mation for for this manual page, the savescreen test-program produced dumps of different size (all on 64-bit hardware, on 40x80 screens):

? AIX (51817 bytes)

? HPUX (90093 bytes)

? Solaris 10 (13273 bytes)

? ncurses5 (12888 bytes)

Solaris

As noted above, Solaris curses has no magic number corresponding to SVr4 curses. This is odd since Solaris was the first operating system to pass the SVr4 guidelines. Solaris has two versions of curses:

? The default curses library uses the SVr3 magic number.

? There is an alternate curses library in /usr/xpg4. This uses a textual format with no magic number.

According to the copyright notice, the xpg4 Solaris curses library

was developed by MKS (Mortice Kern Systems) from 1990 to 1995.

Like ncurses6, there is a file-header with parameters. Unlike ncurses6, the contents of the window are written piecemeal, with coordinates and attributes for each chunk of text rather than writ? ing the whole window from top to bottom.

PDCurses

PDCurses added support for screen dumps in version 2.7 (2005). Like Unix SystemV and ncurses5, it writes the WINDOW structure in binary, but begins the file with its three-byte identifier ?PDC?, followed by a one-byte version, e.g.,

?PDC\001?

NetBSD

As of April 2017, NetBSD curses does not support scr_dump and scr_re? store (or scr_init, scr_set), although it has putwin and getwin.

Like ncurses5, NetBSD putwin does not identify its dumps with a useful magic number. It writes

? the curses shared library major and minor versions as the first two bytes (e.g., 7 and 1),

? followed by a binary dump of the WINDOW,

? some data for wide-characters referenced by the WINDOW structure, and

? finally, lines as done by other implementations.

EXAMPLE

Given a simple program which writes text to the screen (and for the sake of example, limiting the screen-size to 10x20):

```
#include <curses.h>
int
main(void)
{
    putenv("LINES=10");
    putenv("COLUMNS=20");
    initscr();
    start_color();
```

```
    init_pair(1, COLOR_WHITE, COLOR_BLUE);
    init_pair(2, COLOR_RED, COLOR_BLACK);
    bkgd(COLOR_PAIR(1));
    move(4, 5);
    attron(A_BOLD);
    addstr("Hello");
    move(5, 5);
    attroff(A_BOLD);
    attrset(A_REVERSE | COLOR_PAIR(2));
    addstr("World!");
    refresh();
    scr_dump("foo.out");
    endwin();
    return 0;
}
```

When run using ncurses6, the output looks like this:

```
\210\210\210\210ncurses 6.0.20170415
_cury=5
_curx=11
_maxy=9
_maxx=19
_flags=14
_attrs=\{REVERSE|C2}
flag=_idcok
_delay=-1
_regbottom=9
_bkgrnd=\{NORMAL|C1}\s
rows:
1:\{NORMAL|C1}\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s
2:\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s
3:\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s
4:\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s
5:\s\s\s\s\s\{BOLD}Hello\{NORMAL}\s\s\s\s\s\s\s\s\s\s\s
```

6:\s\s\s\s\s\{REVERSE|C2}World!\{NORMAL|C1}\s\s\s\s\s\s\s\s\s\s

7:\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s

8:\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s

9:\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s

10:\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s\s

The first four octal escapes are actually nonprinting characters, while the remainder of the file is printable text. You may notice:

? The actual color pair values are not written to the file.

? All characters are shown in printable form; spaces are ?\s? to en? sure they are not overlooked.

? Attributes are written in escaped curly braces, e.g., ?\{BOLD}?, and may include a color-pair (C1 or C2 in this example).

? The parameters in the header are written out only if they are non? zero. When reading back, order does not matter.

Running the same program with Solaris xpg4 curses gives this dump:

MAX=10,20

BEG=0,0

SCROLL=0,10

VMIN=1

VTIME=0

FLAGS=0x1000

FG=0,0

BG=0,0,

0,0,0,1,

0,19,0,0,

1,0,0,1,

1,19,0,0,

2,0,0,1,

2,19,0,0,

3,0,0,1,

3,19,0,0,

4,0,0,1,

4,5,0x20,0,Hello

4,10,0,1,

4,19,0,0,

5,0,0,1,

5,5,0x4,2,World!

5,11,0,1,

5,19,0,0,

6,0,0,1,

6,19,0,0,

7,0,0,1,

7,19,0,0,

8,0,0,1,

8,19,0,0,

9,0,0,1,

9,19,0,0,

CUR=11,5

Solaris getwin requires that all parameters are present, and in the same order. The xpg4 curses library does not know about the bce (back color erase) capability, and does not color the window background.

On the other hand, the SVr4 curses library does know about the back? ground color. However, its screen dumps are in binary. Here is the corresponding dump (using ?od -t x1?):

0000000 1c 01 c3 d6 f3 58 05 00 0b 00 0a 00 14 00 00 00

0000020 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00

0000040 00 00 b8 1a 06 08 cc 1a 06 08 00 00 09 00 10 00

0000060 00 00 00 80 00 00 20 00 00 00 ff ff ff ff 00 00

0000100 ff ff ff ff 00 00 00 00 20 80 00 00 20 80 00 00

0000120 20 80 00 00 20 80 00 00 20 80 00 00 20 80 00 00

*

0000620 20 80 00 00 20 80 00 00 20 80 00 00 48 80 00 04

0000640 65 80 00 04 6c 80 00 04 6c 80 00 04 6f 80 00 04

0000660 20 80 00 00 20 80 00 00 20 80 00 00 20 80 00 00

*

0000740 20 80 00 00 20 80 00 00 20 80 00 00 57 00 81 00

0000760 6f 00 81 00 72 00 81 00 6c 00 81 00 64 00 81 00

0001000 21 00 81 00 20 80 00 00 20 80 00 00 20 80 00 00

0001020 20 80 00 00 20 80 00 00 20 80 00 00 20 80 00 00

*

0001540 20 80 00 00 20 80 00 00 00 00 f6 d1 01 00 f6 d1

0001560 08 00 00 00 40 00 00 00 00 00 00 00 00 00 00 07

0001600 00 04 00 01 00 01 00 00 00 01 00 00 00 00 00 00

0001620 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

*

0002371

SEE ALSO

curs_scr_dump(3X), curs_util(3X).

AUTHORS

Thomas E. Dickey

extended screen-dump format for ncurses 6.0 (2015)

Eric S. Raymond

screen dump feature in ncurses 1.9.2d (1995)

scr_dump(5)