NAME

unvis, **strunvis**, **strnunvis**, **strunvisx**, **strnunvisx** - decode a visual representation of characters

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <vis.h>

int
unvis(*char *cp, int c, int *astate, int flag*);

int
strunvis(*char *dst, const char *src*);

int
strnunvis(*char *dst, size_t dlen, const char *src*);

int
strunvisx(*char *dst, const char *src, int flag*);

int
strnunvisx(*char *dst, size_t dlen, const char *src, int flag*);

DESCRIPTION

The **unvis**(), **strunvis**() and **strunvisx**() functions are used to decode a visual representation of characters, as produced by the vis(3) function, back into the original form.

The **unvis**() function is called with successive characters in *c* until a valid sequence is recognized, at which time the decoded character is available at the character pointed to by *cp*.

The **strunvis**() function decodes the characters pointed to by *src* into the buffer pointed to by *dst*. The **strunvis**() function simply copies *src* to *dst*, decoding any escape sequences along the way, and returns the number of characters placed into *dst*, or -1 if an invalid escape sequence was detected. The size of *dst* should be equal to the size of *src* (that is, no expansion takes place during decoding).

The **strunvisx**() and **strnunvisx**() functions do the same as the **strunvis**() and **strnunvis**() functions, but take a flag that specifies the style the string *src* is encoded with. The meaning of the flag is the same as explained below for **unvis**().

The **unvis**() function implements a state machine that can be used to decode an arbitrary stream of bytes. All state associated with the bytes being decoded is stored outside the **unvis**() function (that is, a pointer to the state is passed in), so calls decoding different streams can be freely intermixed. To start decoding a stream of bytes, first initialize an integer to zero. Call **unvis**() with each successive byte, along with a pointer to this integer, and a pointer to a destination character. The **unvis**() function has several return codes that must be handled properly. They are:

0 (zero)	Another character is necessary; nothing has been recognized yet.
UNVIS_VALID	A valid character has been recognized and is available at the location pointed to by *cp*.
UNVIS_VALIDPUSH	A valid character has been recognized and is available at the location pointed to by *cp*; however, the character currently passed in should be passed in again.
UNVIS_NOCHAR	A valid sequence was detected, but no character was produced. This return code is necessary to indicate a logical break between characters.
UNVIS_SYNBAD	An invalid escape sequence was detected, or the decoder is in an unknown state. The decoder is placed into the starting state.

When all bytes in the stream have been processed, call **unvis**() one more time with flag set to UNVIS_END to extract any remaining character (the character passed in is ignored).

The *flag* argument is also used to specify the encoding style of the source. If set to VIS_NOESCAPE **unvis**() will not decode backslash escapes. If set to VIS_HTTPSTYLE or VIS_HTTP1808, **unvis**() will decode URI strings as specified in RFC 1808. If set to VIS_HTTP1866, **unvis**() will decode entity references and numeric character references as specified in RFC 1866. If set to VIS_MIMESTYLE, **unvis**() will decode MIME Quoted-Printable strings as specified in RFC 2045. If set to VIS_NOESCAPE, **unvis**() will not decode ‘\’ quoted characters.

The following code fragment illustrates a proper use of **unvis**().

```
int state = 0;
char out;

while ((ch = getchar()) != EOF) {
again:
        switch(unvis(&out, ch, &state, 0)) {
        case 0:
```

```
                case UNVIS_NOCHAR:
                        break;
                case UNVIS_VALID:
                        (void)putchar(out);
                        break;
                case UNVIS_VALIDPUSH:
                        (void)putchar(out);
                        goto again;
                case UNVIS_SYNBAD:
                        errx(EXIT_FAILURE, "Bad character sequence!");
                }
        }
        if (unvis(&out, '\0', &state, UNVIS_END) == UNVIS_VALID)
                (void)putchar(out);
```

ERRORS

The functions **strunvis**(), **strnunvis**(), **strunvisx**(), and **strnunvisx**() will return -1 on error and set *errno* to:

[EINVAL] An invalid escape sequence was detected, or the decoder is in an unknown state.

In addition the functions **strnunvis**() and **strnunvisx**() will can also set *errno* on error to:

[ENOSPC] Not enough space to perform the conversion.

SEE ALSO

unvis(1), vis(1), vis(3)

R. Fielding, *Relative Uniform Resource Locators*, RFC1808.

HISTORY

The **unvis**() function first appeared in 4.4BSD. The **strnunvis**() and **strnunvisx**() functions appeared in NetBSD 6.0 and FreeBSD 9.2.

BUGS

The names VIS_HTTP1808 and VIS_HTTP1866 are wrong. Percent-encoding was defined in RFC 1738, the original RFC for URL. RFC 1866 defines HTML 2.0, an application of SGML, from which it inherits concepts of numeric character references and entity references.