NAME

notifier − install callback function to display notification-messages in a custom manner

SYNOPSIS

#include "mjsu.h"

VOID notifier(VOID (*func)(UINT type, const CHAR *prefix, const CHAR *fmt, va_list ap));

DESCRIPTION

notifier() registers the supplied function to display messages resulting from calls to **error(3)**, **warning(3)**, **remark(3)**, or **usage(3)**; i.e. as a means of overriding the default display-method of those functions.

When the registered function is called:

type is one of ERROR_NOTIFY, WARNING_NOTIFY, INFO_NOTIFY, or USAGE_NOTIFY. This can be used to select a particular presentation for each type of message, such as a custom icon for fatal messages, and so on.

prefix is a simple string that should be used as the first part of the message.

fmt and *ap* are **printf()**-style format-specifier and values that should be used to format the remainder of the message.

The default behaviour can be restored by calling **notifier()** with a NULL *func*.

EXAMPLES

On Microsoft Windows using Visual Studio 2015 or later, display all such messages using stylised popup dialogs with the program-name as the title-bar caption and and an icon that depends on the message-type:

```
#include "mjsu.h"

static VOID my_txtmsg_show(UINT type,
        const CHAR *pfx, const CHAR *fmt, va_list ap)
{
CHAR *buf, *p;
size_t siz;
int n;
UINT mbtype = MB_OK|MB_TASKMODAL;

/* determine buffer-size needed to hold the formatted part */
n = vsnprintf(NULL, 0, fmt, ap);
if (n < 0)
    {
    /* vsnprintf cannot encode the message! */
    MessageBox(NULL, "internal error [vsnprintf]",
        whatami(), mbtype|MB_ICONERROR);
    exit(EXIT_FAILURE);
    }
siz = strlen(pfx) + n + 1;

/* allocate buffer large enough to hold the full message */
if (!(buf = malloc(siz)))
    {
    MessageBox(NULL, "error: out of memory",
        whatami(), mbtype|MB_ICONERROR);
    exit (EXIT_FAILURE);
    }

/* populate buffer with formatted message */
p = cpystr(buf, pfx, NULL);
```

```
        siz -= (p - buf);
        n = vsnprintf(p, siz-1, fmt, ap);
        p[n] = '\0';        /* in spite of MSDN documentation, vsnprintf() does
                             * not always terminate the output string even if
                             * there is room, so we do it explicitly, here.
                             */

        /* determine type of message-box we should use: */
        switch (type)
                {
        case INFO_MSG:
                mbtype |= MB_ICONINFORMATION;
                break;
        case WARNING_MSG:
                mbtype |= MB_ICONWARNING;
                break;
        default:
                mbtype |= MB_ICONERROR;
                break;
                }

        /* and finally display the message then deallocate the buffer */
        MessageBox(NULL, buf, whatami(), mbtype);
        buf = mem_free(buf);
        }

INT main(INT ac, CHAR **av)
        {
        ...
        /* install my custom message-display-function */
        notifier(my_display_func);
        ...
        }
```

SEE ALSO

error(3), **usage(3)**, **warning(3)**, **remark(3)**.

whatami(3), **mjsu(7)**.

fprintf(), as defined by ANSI X3.159-1989.

AVAILABILITY

This functions is written in C, conforming to ANSI X3.159-1989 (hosted program environment).

NOTES

Any callback function supplied to notifier() **must not** call **longjmp(3)**, **raise(3)** or **abort(3)**, or perform any other form of non-local goto, and **must not** cause a call to **error(3)**, **usage(3)**, **panic(3)**, **warning(3)** or **remark(3)**, otherwise arbitrary undefined behaviour may occur.