NAME

str_dup − duplicate a string onto the heap

SYNOPSIS

#include "mjsu.h"

CHAR *str_dup(const CHAR *str, BOOL force);

DESCRIPTION

str_dup() attempts to copy the string at *str* (including the terminating nul character) into an automatically allocated cell of heap memory.

If the allocation request cannot be satisfied, and *force* is YES, the calling process is forcibly terminated by the call:

```
error("out of memory");
```

NOTE: the allocated memory should be subsequently explicitly deallocated by **mem_free()** or **free()** when it is no longer needed.

RETURNS

If successful, **str_dup()** returns a pointer to the allocated memory, which is also a pointer to the newly minted copy of the string at *str*. Otherwise, NULL (if anything) is returned.

EXAMPLE

To take a modifiable copy of a const (read-only) string:

```
BOOL parse(const CHAR *line)
        {
        const CHAR *delims = "\t ()[],;:.!?$%&\"´'";
        CHAR *buf;
        CHAR *w;

        /* although strtok() is handy, it modifies the
         * string being tokenised. Thus we must use
         * a COPY of the original string.
         */
        if (!(buf = str_dup(line, NO)))
                return (NO);

        /* use strtok() to break off individual words,
         * and process them one at a time...
         */
        for (w = strtok(buf, delims); w; w = strtok(NULL, delims))
                process_word(word);

        mem_free(buf);
        return (YES);
        }
```

SEE ALSO

mem_buy(3), **mem_free(3)**, **mjsu(7)**.

malloc() and **free()** as defined by ANSI X3.159-1989.

NOTES

Many compilation environments provide a functionally similar strdup(). However, strdup() is not defined by ANSI x3.159-1989, whereas **str_dup()** can be used in all ANSI C environments.

AVAILABILITY

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