



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 'putenv.3p' command

\$ man putenv.3p

PUTENV(3P) POSIX Programmer's Manual PUTENV(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

putenv ? change or add a value to an environment

SYNOPSIS

```
#include <stdlib.h>

int putenv(char *string);
```

DESCRIPTION

The putenv() function shall use the string argument to set environment variable values. The string argument should point to a string of the form "name=value". The putenv() function shall make the value of the environment variable name equal to value by altering an existing variable or creating a new one. In either case, the string pointed to by string shall become part of the environment, so altering the string shall change the environment.

The putenv() function need not be thread-safe.

RETURN VALUE

Upon successful completion, putenv() shall return 0; otherwise, it shall return a non-zero value and set errno to indicate the error.

ERRORS

The `putenv()` function may fail if:

`ENOMEM` Insufficient memory was available.

The following sections are informative.

EXAMPLES

Changing the Value of an Environment Variable

The following example changes the value of the `HOME` environment variable to the value `/usr/home`.

```
#include <stdlib.h>

...

static char *var = "HOME=/usr/home";

int ret;

ret = putenv(var);
```

APPLICATION USAGE

The `putenv()` function manipulates the environment pointed to by `environ`, and can be used in conjunction with `getenv()`.

See `exec()` for restrictions on changing the environment in multithreaded applications.

This routine may use `malloc()` to enlarge the environment.

A potential error is to call `putenv()` with an automatic variable as the argument, then return from the calling function while string is still part of the environment.

Although the space used by string is no longer used once a new string which defines name is passed to `putenv()`, if any thread in the application has used `getenv()` to retrieve a pointer to this variable, it should not be freed by calling `free()`. If the changed environment variable is one known by the system (such as the locale environment variables) the application should never free the buffer used by earlier calls to `putenv()` for the same variable.

The `setenv()` function is preferred over this function. One reason is that `putenv()` is optional and therefore less portable. Another is that using `putenv()` can slow down environment searches, as explained in the `RATIONALE` section for `getenv()`.

RATIONALE

Refer to the RATIONALE section in `setenv()`.

FUTURE DIRECTIONS

None.

SEE ALSO

`exec`, `free()`, `getenv()`, `malloc()`, `setenv()`

The Base Definitions volume of POSIX.1-2017, `<stdlib.h>`

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.opengroup.org/unix/online.html>.

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.kernel.org/doc/man-pages/reporting_bugs.html.

IEEE/The Open Group

2017

PUTENV(3P)