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 vari? able 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 vari?able 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 envi?ron, and can be used in conjunction with getenv().

See exec() for restrictions on changing the environment in multi-threaded 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 applica?tion 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