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 'realpath.3p' command

$ man realpath.3p

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

realpath ? resolve a pathname

SYNOPSIS

```
#include <stdlib.h>
char *realpath(const char *restrict file_name,
    char *restrict resolved_name);
```

DESCRIPTION

The realpath() function shall derive, from the pathname pointed to by file_name, an absolute pathname that resolves to the same directory en? try, whose resolution does not involve '.', '..', or symbolic links. If resolved_name is a null pointer, the generated pathname shall be stored as a null-terminated string in a buffer allocated as if by a call to malloc(). Otherwise, if {PATH_MAX} is defined as a constant in the <limits.h> header, then the generated pathname shall be stored as a null-terminated string, up to a maximum of {PATH_MAX} bytes, in the buffer pointed to by resolved_name.

If resolved_name is not a null pointer and {PATH_MAX} is not defined as

a constant in the <limits.h> header, the behavior is undefined.

RETURN VALUE

Upon successful completion, realpath() shall return a pointer to the buffer containing the resolved name. Otherwise, realpath() shall re? turn a null pointer and set errno to indicate the error.

If the resolved_name argument is a null pointer, the pointer returned by realpath() can be passed to free().

If the resolved_name argument is not a null pointer and the realpath() function fails, the contents of the buffer pointed to by resolved_name are undefined.

ERRORS

The realpath() function shall fail if:

EACCES Search permission was denied for a component of the path prefix of file_name.

EINVAL The file_name argument is a null pointer.

EIO An error occurred while reading from the file system.

ELOOP A loop exists in symbolic links encountered during resolution of the file_name argument.

ENAMETOOLONG

The length of a component of a pathname is longer than {NAME_MAX}.

ENOENT A component of file_name does not name an existing file or file_name points to an empty string.

ENOTDIR

A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the file_name argument contains at least one non-<slash> character and ends with one or more trailing <slash> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

The realpath() function may fail if:

EACCES The file_name argument does not begin with a <slash> and none of the symbolic links (if any) processed during pathname resolution

of file_name had contents that began with a <slash>, and either search permission was denied for the current directory or read or search permission was denied for a directory above the cur? rent directory in the file hierarchy.

ELOOP More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the file_name argument.

ENAMETOOLONG

The length of a pathname exceeds {PATH_MAX}, or pathname resolu? tion of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

ENOMEM Insufficient storage space is available.

The following sections are informative.

EXAMPLES

Generating an Absolute Pathname

The following example generates an absolute pathname for the file iden? tified by the symlinkpath argument. The generated pathname is stored in the buffer pointed to by actualpath.

```
#include <stdlib.h>
...
char *symlinkpath = "/tmp/symlink/file";
char *actualpath;
actualpath = realpath(symlinkpath, NULL);
if (actualpath != NULL)
{
    ... use actualpath ...
    free(actualpath);
}
else
{
    ... handle error ...
}
```

APPLICATION USAGE

For functions that allocate memory as if by malloc(), the application

should release such memory when it is no longer required by a call to free(). For realpath(), this is the return value.

RATIONALE

Since realpath() has no length argument, if {PATH_MAX} is not defined as a constant in <limits.h>, applications have no way of determining how large a buffer they need to allocate for it to be safe to pass to realpath(). A {PATH_MAX} value obtained from a prior pathconf() call is out-of-date by the time realpath() is called. Hence the only reli? able way to use realpath() when {PATH_MAX} is not defined in <limits.h> is to pass a null pointer for resolved_name so that realpath() will al? locate a buffer of the necessary size.

FUTURE DIRECTIONS

None.

SEE ALSO

fpathconf(), free(), getcwd(), sysconf()

The Base Definitions volume of POSIX.1?2017, <limits.h>, <stdlib.h>