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 'pthread_once.3p' command

$ man pthread_once.3p

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

pthread_once ? dynamic package initialization

SYNOPSIS

```
#include <pthread.h>
int pthread_once(pthread_once_t *once_control,
    void (*init_routine)(void));
pthread_once_t once_control = PTHREAD_ONCE_INIT;
```

DESCRIPTION

The first call to pthread_once() by any thread in a process, with a given once_control, shall call the init_routine with no arguments. Sub? sequent calls of pthread_once() with the same once_control shall not call the init_routine. On return from pthread_once(), init_routine shall have completed. The once_control parameter shall determine whether the associated initialization routine has been called.

The pthread_once() function is not a cancellation point. However, if init_routine is a cancellation point and is canceled, the effect on once_control shall be as if pthread_once() was never called.

If the call to init_routine is terminated by a call to longjmp(), _longjmp(), or siglongjmp(), the behavior is undefined.

The constant PTHREAD_ONCE_INIT is defined in the <pthread.h> header.

The behavior of pthread_once() is undefined if once_control has automatic storage duration or is not initialized by PTHREAD_ONCE_INIT.

RETURN VALUE

Upon successful completion, pthread_once() shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

The pthread_once() function shall not return an error code of [EINTR].

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

If init_routine recursively calls pthread_once() with the same once_control, the recursive call will not call the specified init_routine, and thus the specified init_routine will not complete, and thus the recursive call to pthread_once() will not return. Use of longjmp(), _longjmp(), or siglongjmp() within an init_routine to jump to a point outside of init_routine prevents init_routine from returning.

RATIONALE

Some C libraries are designed for dynamic initialization. That is, the global initialization for the library is performed when the first procedure in the library is called. In a single-threaded program, this is normally implemented using a static variable whose value is checked on entry to a routine, as follows:

```
static int random_is_initialized = 0;
extern void initialize_random(void);
int random_function()
{
    if (random_is_initialized == 0) {
        initialize_random();
        random_is_initialized = 1;
```

```
    }
    ... /* Operations performed after initialization. */
}
```

To keep the same structure in a multi-threaded program, a new primitive is needed. Otherwise, library initialization has to be accomplished by an explicit call to a library-exported initialization function prior to any use of the library.

For dynamic library initialization in a multi-threaded process, if an initialization flag is used the flag needs to be protected against mod? ification by multiple threads simultaneously calling into the library. This can be done by using a mutex (initialized by assigning PTHREAD_MU? TEX_INITIALIZER). However, the better solution is to use pthread_once() which is designed for exactly this purpose, as follows:

```
#include <pthread.h>
static pthread_once_t random_is_initialized = PTHREAD_ONCE_INIT;
extern void initialize_random(void);
int random_function()
{
    (void) pthread_once(&random_is_initialized, initialize_random);
    ... /* Operations performed after initialization. */
}
```

If an implementation detects that the value specified by the once_con? trol argument to pthread_once() does not refer to a pthread_once_t ob? ject initialized by PTHREAD_ONCE_INIT, it is recommended that the func? tion should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

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

COPYRIGHT