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_attr_getguardsize.3p' command

$ man pthread_attr_getguardsize.3p

PTHREAD_ATTR_GETGUARDSIZE(3POSIX Programmer's ManPTHREAD_ATTR_GETGUARDSIZE(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_attr_getguardsize, pthread_attr_setguardsize ? get and set the thread guardsize attribute

SYNOPSIS

```
#include <pthread.h>
int pthread_attr_getguardsize(const pthread_attr_t *restrict attr,
    size_t *restrict guardsize);
int pthread_attr_setguardsize(pthread_attr_t *attr,
    size_t guardsize);
```

DESCRIPTION

The pthread_attr_getguardsize() function shall get the guardsize attri? bute in the attr object. This attribute shall be returned in the guard? size parameter.

The pthread_attr_setguardsize() function shall set the guardsize attri? bute in the attr object. The new value of this attribute shall be ob? tained from the guardsize parameter. If guardsize is zero, a guard area shall not be provided for threads created with attr. If guardsize is

greater than zero, a guard area of at least size guardsize bytes shall be provided for each thread created with attr.

The guardsize attribute controls the size of the guard area for the created thread's stack. The guardsize attribute provides protection against overflow of the stack pointer. If a thread's stack is created with guard protection, the implementation allocates extra memory at the overflow end of the stack as a buffer against stack overflow of the stack pointer. If an application overflows into this buffer an error shall result (possibly in a SIGSEGV signal being delivered to the thread).

A conforming implementation may round up the value contained in guard? size to a multiple of the configurable system variable {PAGESIZE} (see <sys/mman.h>). If an implementation rounds up the value of guardsize to a multiple of {PAGESIZE}, a call to pthread_attr_getguardsize() specifying attr shall store in the guardsize parameter the guard size specified by the previous pthread_attr_setguardsize() function call.

The default value of the guardsize attribute is implementation-defined.

If the stackaddr attribute has been set (that is, the caller is allo? cating and managing its own thread stacks), the guardsize attribute shall be ignored and no protection shall be provided by the implementa? tion. It is the responsibility of the application to manage stack over? flow along with stack allocation and management in this case.

The behavior is undefined if the value specified by the attr argument to pthread_attr_getguardsize() or pthread_attr_setguardsize() does not refer to an initialized thread attributes object.

RETURN VALUE

If successful, the pthread_attr_getguardsize() and pthread_attr_set? guardsize() functions shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions shall fail if:

EINVAL The parameter guardsize is invalid.

These functions shall not return an error code of [EINTR].

The following sections are informative.

EXAMPLES

Retrieving the guardsize Attribute

This example shows how to obtain the guardsize attribute of a thread attribute object.

```
#include <pthread.h>
pthread_attr_t thread_attr;
size_t  guardsize;
int     rc;
/* code initializing thread_attr */
...
rc = pthread_attr_getguardsize (&thread_attr, &guardsize);
if (rc != 0)  {
    /* handle error */
    ...
}
else {
    if (guardsize > 0) {
    /* a guard area of at least guardsize bytes is provided */
    ...
    }
    else {
    /* no guard area provided */
    ...
    }
}
```

APPLICATION USAGE

None.

RATIONALE

The guardsize attribute is provided to the application for two reasons:

1. Overflow protection can potentially result in wasted system re? sources. An application that creates a large number of threads, and which knows its threads never overflow their stack, can save

system resources by turning off guard areas.

2. When threads allocate large data structures on the stack, large guard areas may be needed to detect stack overflow.

The default size of the guard area is left implementation-defined since on systems supporting very large page sizes, the overhead might be sub? stantial if at least one guard page is required by default.

If an implementation detects that the value specified by the attr argu? ment to pthread_attr_getguardsize() or pthread_attr_setguardsize() does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

FUTURE DIRECTIONS

None.

SEE ALSO

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

COPYRIGHT