NAME

pthread_attr_init, pthread_attr_destroy, pthread_attr_setscope, pthread_attr_getscope, pthread_attr_setdetachstate, pthread_attr_getdetachstate, pthread_attr_setstacksize, pthread_attr_getstacksize, pthread_attr_setstackaddr, pthread_attr_getstackaddr, pthread_attr_setschedparam, pthread_attr_getschedparam, pthread_attr_setschedpolicy, pthread_attr_getschedpolicy, pthread_attr_setinheritsched, pthread_attr_getinheritsched – thread creation attributes

SYNOPSIS

#include <pthread.h>

int pthread_attr_init(pthread_attr_t **attr***);**
int pthread_attr_destroy(pthread_attr_t **attr***);**
int pthread_attr_setscope(pthread_attr_t **attr,* **int** *contentionscope***);**
int pthread_attr_getscope(const pthread_attr_t **attr,* **int** **contentionscope***);**
int pthread_attr_setdetachstate(pthread_attr_t **attr,* **int** *detachstate***);**
int pthread_attr_getdetachstate(const pthread_attr_t **attr,* **int** **detachstate***);**
int pthread_attr_setstacksize(pthread_attr_t **attr,* **size_t** *stacksize***);**
int pthread_attr_getstacksize(const pthread_attr_t **attr,* **size_t** **stacksize***);**
int pthread_attr_setstackaddr(pthread_attr_t **attr,* **void** **stackaddr***);**
int pthread_attr_getstackaddr(const pthread_attr_t **attr,* **void** ***stackaddr***);**
int pthread_attr_setschedparam(pthread_attr_t **attr,*
  const struct sched_param **param***);**
int pthread_attr_getschedparam(const pthread_attr_t **attr,*
  struct sched_param **param***);**
int pthread_attr_setschedpolicy(pthread_attr_t **attr,* **int** *policy***);**
int pthread_attr_getschedpolicy(const pthread_attr_t **attr,* **int** **policy***);**
int pthread_attr_setinheritsched(pthread_attr_t **attr,* **int** *inheritsched***);**
int pthread_attr_getinheritsched(const pthread_attr_t **attr,* **int** **inheritsched***);**

MT-LEVEL

MT-Safe

DESCRIPTION

The pthread approach to setting attributes for threads is to request the initialization of an attribute object, *attr*, and pass the initialized attribute object to **pthread_create**(3T). The convention in Solaris is to pass these attributes as flags to **thr_create**(3T).

All attributes in *attr* are independent of one another and may be singularly modified or retrieved. *attr*, itself, is independent of any thread and can be modified or used to create new threads. However, any change to *attr* after a thread is created will not affect that thread.

init

The **pthread_attr_init( )** function initializes a thread attributes object (*attr*) with the default value for each attribute as follows:

Attribute	Default	Value
contentionscope	**PTHREAD_SCOPE_PROCESS**	resource competition within process
detachstate	**PTHREAD_CREATE_JOINABLE**	joinable by other threads
stackaddr	**NULL**	stack allocated by system
stacksize	**NULL**	1 megabyte

priority	----	priority of parent (calling) thread
policy	**SCHED_OTHER**	determined by system
inheritsched	**PTHREAD_EXPLICIT_SCHED**	scheduling policy and parameters not inherited but explicitly defined by the attribute object

NOTE: Attribute objects should be destroyed before an initialized attribute object is re-initialized.

destroy

pthread_attr_destroy( ) destroys a thread attributes object (*attr*), which cannot be reused until it is reinitialized.

resource contentionscope

The **pthread_attr_setscope( )** and **pthread_attr_getscope( )** functions set and get the *contentionscope* thread attribute in the *attr* object. The *contentionscope* value may be set to the following:

PTHREAD_SCOPE_SYSTEM Indicates system scheduling contention scope. This thread is permanently "bound" to an LWP, and is also called a bound thread. This value is equivalent to **THR_BOUND** in Solaris threads (see **thr_create**(3T)).

PTHREAD_SCOPE_PROCESS Indicates process scheduling contention scope. This thread is not "bound" to an LWP, and is also called an unbound thread. **PTHREAD_SCOPE_PROCESS**, or unbound, is the default.

detachstate

The **pthread_attr_setdetachstate( )** and **pthread_attr_getdetachstate( )** functions set and get the *detachstate* attribute in the *attr* object. The *detachstate* attribute determines whether the thread is created in a detached state or not. The *detachstate* may be set to the following values:

PTHREAD_CREATE_DETACHED Creates a new detached thread. A detached thread disappears without leaving a trace. The thread ID and any of its resources are freed and ready for reuse. **pthread_join**(3T) and **thr_join**(3T) cannot wait for a detached thread.

PTHREAD_CREATE_JOINABLE Creates a new non-detached thread. The thread ID and its user-defined stack, if specified at thread creation time, is not freed until **pthread_join**(3T) or **thr_join**(3T) are called. **pthread_join**(3T) or **thr_join**(3T) must be called to release any resources associated with the terminated thread.

stacksize and stackaddr

The **pthread_attr_setstacksize( )** and **pthread_attr_getstacksize( )** functions set and get the *stacksize* thread attribute in the *attr* object. The *stacksize* default argument is **NULL**, and a thread default stack size is 1 megabyte.

The **pthread_attr_setstackaddr( )** and **pthread_attr_getstackaddr( )** functions set and get the *stackaddr* thread attribute in the *attr* object. The *stackaddr* default is **NULL**. (See **pthread_create**(3T).)

schedparam (priority)

The **pthread_attr_setschedparam( )** and **pthread_attr_getschedparam( )** functions set and get the scheduling parameter thread attributes in the *attr* argument, determined by the scheduling policy set in the *attr* object. The only required member of the *param* structure for the **SCHED_OTHER**, **SCHED_FIFO**, and **SCHED_RR** policies is *sched_priority* (see **NOTES** section below). You can use these functions to get and set the priority of the thread to be created. The **sched_priority** of the *param* structure is **NULL**, by default, which means the newly created thread inherits the priority of its parent thread.

schedpolicy

The **pthread_attr_setschedpolicy( )** and **pthread_attr_getschedpolicy( )** functions set and get the *schedpolicy* thread attribute in the *attr* argument.

Values for the *policy* attribute are **SCHED_FIFO**, **SCHED_RR**, or the default value **SCHED_OTHER** (see **NOTES** section below.

RETURN VALUES

Upon successful completion, the following functions return **0**; otherwise, an error number is returned to indicate the error: **pthread_attr_init( )**, **pthread_attr_destroy( )**, **pthread_attr_setstacksize( )**, **pthread_attr_getstacksize( )**, **pthread_attr_setstackaddr( )**, **pthread_attr_getstackaddr( )**, **pthread_attr_setdetachstate( )**, **pthread_attr_getdetachstate( )**, **pthread_attr_setscope**( ), **pthread_attr_getscope( )**, **pthread_attr_setinheritsched( )**, **pthread_attr_getinheritsched( )**, **pthread_attr_setschedpolicy( )**, and **pthread_attr_getschedpolicy( )**.

ERRORS

If any of the following conditions occur, **pthread_attr_init( )** returns the corresponding error number:

ENOMEM Insufficient memory exists to create the thread attributes object.

If any of the following conditions occur, **pthread_attr_setstacksize( )** returns the corresponding error number:

EINVAL The value of *stacksize* is less than **PTHREAD_STACK_MIN** or exceeds a system-imposed limit.

If any of the following conditions occur, **pthread_attr_destroy( )**, **pthread_attr_setstacksize( )**, **pthread_attr_getstacksize( )**, **pthread_attr_setstackaddr( )**, **pthread_attr_getstackaddr( )**, **pthread_attr_setdetachstate( )**, **pthread_attr_getdetachstate( )**, **pthread_attr_setscope( )**, **pthread_attr_getscope( )**, **pthread_attr_setschedparam( )**, **pthread_attr_getschedparam( )**, **pthread_attr_setinheritsched( )**, **pthread_attr_getinheritsched( )**, **pthread_attr_setschedpolicy( )**, and **pthread_attr_getschedpolicy( )** return the corresponding error number:

EINVAL The value of *attr* is not valid.

If any of the following conditions occur, **pthread_attr_setstacksize( )** returns the corresponding error number:

EINVAL The value of *stacksize* is less than **PTHREAD_STACK_MIN**.

If any of the following conditions occur, **pthread_attr_setdetachstate( )** returns the corresponding error number:

EINVAL The value of *detachstate* is not valid.

If any of the following conditions occur, **pthread_attr_setscope( )** returns the corresponding error number:

EINVAL The value of *contentionscope* is not valid.

If any of the following conditions occur, **pthread_attr_setschedparam( )** returns the corresponding error number:

EINVAL The value of the **sched_priority** member of the *param* structure is less than or equal to **0**.

If any of the following conditions occur, **pthread_attr_getstacksize( )** returns the corresponding error number:

EINVAL The value of *stacksize* is **NULL**.

If any of the following conditions occur, **pthread_attr_getstackaddr( )** returns the corresponding error number:

EINVAL The value of *stackaddr* is **NULL**.

If any of the following conditions occur, **pthread_attr_getdetachstate( )** returns the corresponding error number:

EINVAL The value of *detachstate* is **NULL**.

If any of the following conditions occur, **pthread_attr_getscope( )** returns the corresponding error number:

EINVAL The value of *contentionscope* is **NULL**.

If any of the following conditions occur, either **pthread_attr_setschedparam( )** and **pthread_attr_getschedparam( )** returns the corresponding error number:

EINVAL The value of *param* is **NULL**.

For each of the following conditions, if the condition is detected, **pthread_attr_setinheritsched( )** and **pthread_attr_setschedpolicy( )** return the corresponding error number:

ENOTSUP An attempt was made to set the attribute to an unsupported *policy* or *inheritsched*.

For each of the following conditions, if the condition is detected, **pthread_attr_getinheritsched( )** and **pthread_attr_getschedpolicy( )** return the corresponding error number:

EINVAL *policy* or *inheritsched* is **NULL**.

SEE ALSO

pthread_create(3T), **pthread_join**(3T), **thr_create**(3T).

NOTES

Currently, the only policy supported is **SCHED_OTHER**. Attempting to set policy as **SCHED_FIFO** or **SCHED_RR** will result in the error **ENOSUP**.

The attribute object is part of the POSIX threads interface. There is no Solaris threads counterpart to the POSIX threads attribute object.