**a. pthread\_attr\_init() / pthread\_attr\_destroy()**

The **pthread\_attr\_init**() function initializes the thread attributes object pointed to by *attr* with default attribute values. After this call, individual attributes of the object can be set using various related functions (listed under SEE ALSO), and then the object can be used in one or more [pthread\_create(3)](http://man7.org/linux/man-pages/man3/pthread_create.3.html) calls that create threads.

Calling **pthread\_attr\_init**() on a thread attributes object that has already been initialized results in undefined behavior.

When a thread attributes object is no longer required, it should be destroyed using the **pthread\_attr\_destroy**() function. Destroying a thread attributes object has no effect on threads that were created using that object.

Once a thread attributes object has been destroyed, it can be reinitialized using **pthread\_attr\_init**(). Any other use of a

destroyed thread attributes object has undefined results.

**b.pthread\_attr\_setaffinity\_np()/pthread\_attr\_getaffinity\_np()**

The **pthread\_attr\_setaffinity\_np**() function sets the CPU affinity mask attribute of the thread attributes object referred to by *attr* to the value specified in *cpuset*. This attribute determines the CPU affinity mask of a thread created using the thread attributes object *attr*.

The **pthread\_attr\_getaffinity\_np**() function returns the CPU affinity mask attribute of the thread attributes object referred to by *attr* in the buffer pointed to by *cpuset*.

The argument *cpusetsize* is the length (in bytes) of the buffer pointed to by *cpuset*. Typically, this argument would be specified as *sizeof(cpu\_set\_t)*.

For more details on CPU affinity masks, see [sched\_setaffinity(2)](http://man7.org/linux/man-pages/man2/sched_setaffinity.2.html).For a description of a set of macros that can be used to manipulate and inspect CPU sets, see [CPU\_SET(3)](http://man7.org/linux/man-pages/man3/CPU_SET.3.html).

**#define \_GNU\_SOURCE** /\* See feature\_test\_macros(7) \*/

**#include <pthread.h>**

**int pthread\_attr\_setaffinity\_np(pthread\_attr\_t \****attr***,**

**size\_t** *cpusetsize***, const cpu\_set\_t \****cpuset***);**

**int pthread\_attr\_getaffinity\_np(const pthread\_attr\_t \****attr***,**

**size\_t** *cpusetsize***, cpu\_set\_t \****cpuset***);**

**c.pthread\_attr\_setschedparam()/pthread\_attr\_getschedparam()**

The **pthread\_attr\_setschedparam**() function sets the scheduling parameter attributes of the thread attributes object referred to by *attr* to the values specified in the buffer pointed to by *param*. These attributes determine the scheduling parameters of a thread

created using the thread attributes object *attr*.

The **pthread\_attr\_getschedparam**() returns the scheduling parameter attributes of the thread attributes object *attr* in the buffer pointed to by *param*.Scheduling parameters are maintained in the following structure:

struct sched\_param {

int sched\_priority; /\* Scheduling priority \*/

};

As can be seen, only one scheduling parameter is supported. For details of the permitted ranges for scheduling priorities in each scheduling policy, see [sched(7)](http://man7.org/linux/man-pages/man7/sched.7.html).

**#include <pthread.h>**

**int pthread\_setschedparam(pthread\_t** *thread***, int** *policy***,**

**const struct sched\_param \****param***);**

**int pthread\_getschedparam(pthread\_t** *thread***, int \****policy***,**

**struct sched\_param \****param***);**

**d. pthread\_attr\_setstackaddr()/pthread\_attr\_getstackaddr()**

These functions are obsolete: **do not use them.** Use [pthread\_attr\_setstack(3)](http://man7.org/linux/man-pages/man3/pthread_attr_setstack.3.html) and [pthread\_attr\_getstack(3)](http://man7.org/linux/man-pages/man3/pthread_attr_getstack.3.html) instead. The **pthread\_attr\_setstackaddr**() function sets the stack address attribute of the thread attributes object referred to by *attr* to the value specified in *stackaddr*. This attribute specifies the location of the stack that should be used by a thread that is created using the thread attributes object *attr*. *stackaddr* should point to a buffer of at least **PTHREAD\_STACK\_MIN** bytes that was allocated by the caller. The pages of the allocated buffer should be both readable and writable.

The **pthread\_attr\_getstackaddr**() function returns the stack address attribute of the thread attributes object referred to by *attr* in the buffer pointed to by *stackaddr*.

**int pthread\_attr\_setstackaddr(pthread\_attr\_t \****attr***, void \****stackaddr***);**

**int pthread\_attr\_getstackaddr(const pthread\_attr\_t \****attr***, void \*\****stackaddr***);**

**e. pthread\_attr\_setstack()/pthread\_attr\_getstack()**

The **pthread\_attr\_setstack**() function sets the stack address and stack size attributes of the thread attributes object referred to by *attr* to the values specified in *stackaddr* and *stacksize*, respectively. These attributes specify the location and size of the stack that should be used by a thread that is created using the thread attributes object *attr*. *stackaddr* should point to the lowest addressable byte of a buffer of *stacksize* bytes that was allocated by the caller. The pages of the allocated buffer should be both readable and writable.

The **pthread\_attr\_getstack**() function returns the stack address and stack size attributes of the thread attributes object referred to by *attr* in the buffers pointed to by *stackaddr* and *stacksize*, respectively.

**int pthread\_attr\_setstack(pthread\_attr\_t \****attr***,** **void \****stackaddr***, size\_t** *stacksize***);**

**int pthread\_attr\_getstack(const pthread\_attr\_t \****attr***,** **void \*\****stackaddr***, size\_t \****stacksize***);**

**f. pthread\_attr\_setstacksize()/pthread\_attr\_getstacksize()**

The **pthread\_attr\_setstacksize**() function sets the stack size attribute of the thread attributes object referred to by *attr* to the value specified in *stacksize*.The stack size attribute determines the minimum size (in bytes) that will be allocated for threads created using the thread attributes object *attr*.

The **pthread\_attr\_getstacksize**() function returns the stack size attribute of the thread attributes object referred to by *attr* in the buffer pointed to by *stacksize*.

**int pthread\_attr\_setstacksize(pthread\_attr\_t \****attr***, size\_t** *stacksize***);**

**int pthread\_attr\_getstacksize(const pthread\_attr\_t \****attr***, size\_t \****stacksize***);**

**g. pthread\_attr\_setscope()/pthread\_attr\_getscope()**

The **pthread\_attr\_setscope**() function sets the contention scope attribute of the thread attributes object referred to by *attr* to the value specified in *scope*. The contention scope attribute defines the set of threads against which a thread competes for resources such as the CPU. POSIX.1 specifies two possible values for *scope*:

**PTHREAD\_SCOPE\_SYSTEM** The thread competes for resources with all other threads in all processes on the system that are in the same scheduling allocation domain (a group of one or more processors).

**PTHREAD\_SCOPE\_SYSTEM** threads are scheduled relative to one another according to their scheduling policy and priority.

**PTHREAD\_SCOPE\_PROCESS** The thread competes for resources with all other threads in the same process that were also created with the **THREAD\_SCOPE\_PROCESS** contention scope. **PTHREAD\_SCOPE\_PROCESS**

threads are scheduled relative to other threads in the process

according to their scheduling policy and priority. POSIX.1 leaves it unspecified how these threads contend with other threads in other process on the system or with other threads in the same process that were created with the **PTHREAD\_SCOPE\_SYSTEM** contention scope.

POSIX.1 requires that an implementation support at least one of these contention scopes. Linux supports **PTHREAD\_SCOPE\_SYSTEM**, but not **PTHREAD\_SCOPE\_PROCESS**. On systems that support multiple contention scopes, then, in order for the parameter setting made by **pthread\_attr\_setscope**() to have effect when calling [pthread\_create(3)](http://man7.org/linux/man-pages/man3/pthread_create.3.html), the caller must use [pthread\_attr\_setinheritsched(3)](http://man7.org/linux/man-pages/man3/pthread_attr_setinheritsched.3.html) to set the inherit-scheduler attribute of the attributes object *attr* to **PTHREAD\_EXPLICIT\_SCHED**.

The **pthread\_attr\_getscope**() function returns the contention scope attribute of the thread attributes object referred to by *attr* in the buffer pointed to by *scope*.

**int pthread\_attr\_setscope(pthread\_attr\_t \**attr*, int *scope*);**

**int pthread\_attr\_getscope(const pthread\_attr\_t \****attr***, int \****scope***);**