

20

*Mach Functions***C-Thread Functions**

This section contains a summary of the C-thread functions, which are described in detail in the *NeXTSTEP Operating System Software* manual.

To use the C-thread functions, include in your source files the header file **mach/cthreads.h**:

```
#include <mach/cthreads.h>
```

Basic C-Thread Functions**Control a thread**

pthread_t	pthread_fork (any_t (* <i>function</i>)(), any_t <i>arg</i>)
any_t	pthread_join (pthread_t <i>t</i>)
void	pthread_detach (pthread_t <i>t</i>)
void	pthread_yield ()
kern_return_t	pthread_abort (pthread_t <i>t</i>)
void	pthread_exit (any_t <i>result</i>)

Get access to a thread

pthread_t	pthread_self ()
thread_t	pthread_thread (pthread_t <i>t</i>)

Associate a string with a thread

char *	pthread_name (pthread_t <i>t</i>)
void	pthread_set_name (pthread_t <i>t</i> , char * <i>name</i>)

Associate data with a thread

any_t	pthread_data (pthread_t <i>t</i>)
void	pthread_set_data (pthread_t <i>t</i> , any_t <i>data</i>)

Change the priority of a thread

kern_return_t	pthread_max_priority (pthread_t <i>t</i> , processor_set_t <i>processor_set</i> , int <i>max_priority</i>)
kern_return_t	pthread_priority (pthread_t <i>t</i> , int <i>priority</i> , boolean_t <i>set_max</i>)

Get or set the UNIX™ error number of this thread

int	pthread_errno()
void	pthread_set_errno_self (int <i>error</i>)

Get or set the maximum number of threads in this task

int	pthread_limit()
void	pthread_set_limit (int <i>limit</i>)
int	pthread_count()

Mutex Functions

Control a mutex

pthread_t	pthread_alloc()
void	pthread_init (struct mutex * <i>m</i>)
void	pthread_clear (struct mutex * <i>m</i>)
void	pthread_free (pthread_t <i>m</i>)

Associate a string with a mutex

char *	pthread_name (pthread_t <i>m</i>)
void	pthread_set_name (pthread_t <i>m</i> , char * <i>name</i>)

Synchronize a mutex

void	pthread_lock (pthread_t <i>m</i>)
int	pthread_try_lock (pthread_t <i>m</i>)
void	pthread_unlock (pthread_t <i>m</i>)

Condition Functions

Control a condition variable

condition_t	condition_alloc()
void	condition_init (struct condition * <i>c</i>)
void	condition_clear (struct condition * <i>c</i>)
void	condition_free (condition_t <i>c</i>)

Associate a string with a condition variable

char *	condition_name (condition_t <i>c</i>)
void	condition_set_name (condition_t <i>c</i> , char * <i>name</i>)

Synchronize a condition variable

void	condition_broadcast (condition_t <i>c</i>)
void	condition_signal (condition_t <i>c</i>)
void	condition_wait (condition_t <i>c</i> , pthread_t <i>m</i>)

Mach Kernel Functions

The Mach kernel is introduced in the *Operating System Software* manual. This section contains a summary of the Mach kernel functions, which are described in detail in *Operating System Software*.

To use the Mach kernel functions, include in your source files the header file **mach/mach.h**:

```
#include <mach/mach.h>
```

To use the Mach message functions, also include in your source files the header file **mach/message.h**:

```
#include <mach/mach.h>
#include <mach/message.h>
```

To use the Mach error string functions, include the header file **mach/error.h**, in addition to **mach/mach.h**:

```
#include <mach/mach.h>
#include <mach/error.h>
```

Task Functions

Control a task:

kern_return_t	task_create (task_t <i>parent_task</i> , boolean_t <i>inherit_memory</i> , task_t * <i>child_task</i>)
kern_return_t	task_suspend (task_t <i>target_task</i>)
kern_return_t	task_resume (task_t <i>target_task</i>)
kern_return_t	task_terminate (task_t <i>target_task</i>)

Access a task or its threads:

kern_return_t	task_threads (task_t <i>target_task</i> , thread_array_t * <i>thread_list</i> , unsigned int * <i>thread_count</i>)
kern_return_t	task_info (task_t <i>target_task</i> , int <i>flavor</i> , task_info_t <i>task_info</i> , unsigned int * <i>task_info_count</i>)

Access a task's special ports:

kern_return_t	task_get_special_port (task_t <i>task</i> , int <i>which_port</i> , port_t * <i>special_port</i>)
kern_return_t	task_set_special_port (task_t <i>task</i> , int <i>which_port</i> , port_t <i>special_port</i>)
port_t	task_notify ()
task_t	task_self ()

Translate between Mach task and UNIX process ID:

kern_return_t	task_by_unix_pid (task_t <i>task</i> , int <i>pid</i> , task_t * <i>result_task</i>)
kern_return_t	unix_pid (task_t <i>target_task</i> , int * <i>pid</i>)

Set a task's scheduling priority:

kern_return_t	task_priority (task_t <i>task</i> , int <i>priority</i> , boolean_t <i>change_threads</i>)
---------------	--

Multiprocessor functions (not useful on single-processor systems):

kern_return_t	task_assign (task_t <i>task</i> , processor_set_t <i>new_processor_set</i> , boolean_t <i>assign_threads</i>)
kern_return_t	task_assign_default (task_t <i>task</i> , boolean_t <i>assign_threads</i>)
kern_return_t	task_get_assignment (task_t <i>task</i> , processor_set_t <i>*processor_set</i>)

Thread Functions

Control a thread:

kern_return_t	thread_create (task_t <i>parent_task</i> , thread_t <i>*child_thread</i>)
kern_return_t	thread_suspend (thread_t <i>target_thread</i>)
kern_return_t	thread_resume (thread_t <i>target_thread</i>)
kern_return_t	thread_terminate (thread_t <i>target_thread</i>)
kern_return_t	thread_abort (thread_t <i>target_thread</i>)

Get information about a thread:

kern_return_t	thread_info (thread_t <i>target_thread</i> , int <i>flavor</i> , thread_info_t <i>thread_info</i> , unsigned int <i>*thread_info_count</i>)
---------------	---

Access a thread's state:

kern_return_t	thread_get_state (thread_t <i>target_thread</i> , int <i>flavor</i> , thread_state_data_t <i>old_state</i> , unsigned int <i>*old_state_count</i>)
kern_return_t	thread_set_state (thread_t <i>target_thread</i> , int <i>flavor</i> , thread_state_data_t <i>new_state</i> , unsigned int <i>new_state_count</i>)

Access a thread's special ports:

kern_return_t	thread_get_special_port (thread_t <i>thread</i> , int <i>which_port</i> , port_t <i>*special_port</i>)
kern_return_t	thread_set_special_port (thread_t <i>thread</i> , int <i>which_port</i> , port_t <i>special_port</i>)
port_t	thread_reply ()
thread_t	thread_self ()

Affect the scheduling policy or priority of a thread:

kern_return_t	thread_policy (thread_t <i>thread</i> , int <i>policy</i> , int <i>data</i>)
kern_return_t	thread_priority (thread_t <i>thread</i> , int <i>priority</i> , boolean_t <i>set_max</i>)
kern_return_t	thread_max_priority (thread_t <i>thread</i> , processor_set_t <i>processor_set</i> , int <i>priority</i>)
kern_return_t	thread_switch (thread_t <i>new_thread</i> , int <i>option</i> , int <i>time</i>)

Multiprocessor functions (not useful on single-processor systems):

kern_return_t	thread_assign (thread_t <i>thread</i> , processor_set_t <i>new_processor_set</i>)
kern_return_t	thread_assign_default (thread_t <i>thread</i>)
kern_return_t	thread_get_assignment (thread_t <i>thread</i> , processor_set_t <i>*processor_set</i>)

Port Functions

Control a port:

kern_return_t	port_allocate (task_t <i>task</i> , port_name_t * <i>port_name</i>)
kern_return_t	port_deallocate (task_t <i>task</i> , port_name_t <i>port_name</i>)
kern_return_t	port_rename (task_t <i>task</i> , port_name_t <i>old_name</i> , port_name_t <i>new_name</i>)
kern_return_t	port_set_backlog (task_t <i>task</i> , port_name_t <i>port_name</i> , int <i>backlog</i>)

Give a task rights to a port:

kern_return_t	port_insert_receive (task_t <i>task</i> , port_t <i>my_port</i> , port_name_t <i>its_name</i>)
kern_return_t	port_insert_send (task_t <i>task</i> , port_t <i>my_port</i> , port_name_t <i>its_name</i>)

Remove a task's rights to a port:

kern_return_t	port_extract_receive (task_t <i>task</i> , port_name_t <i>its_name</i> , port_t * <i>its_port</i>)
kern_return_t	port_extract_send (task_t <i>task</i> , port_name_t <i>its_name</i> , port_t * <i>its_port</i>)

Get information about a port:

kern_return_t	port_status (task_t <i>task</i> , port_name_t <i>port_name</i> , port_set_name_t * <i>port_set_name</i> , int * <i>num_msgs</i> , int * <i>backlog</i> , boolean_t * <i>owner</i> , boolean_t * <i>receiver</i>)
kern_return_t	port_type (task_t <i>task</i> , port_name_t <i>port_name</i> , port_type_t * <i>port_type</i>)

Get information about a task's port name space:

kern_return_t	port_names (task_t <i>task</i> , port_name_array_t * <i>port_names</i> , unsigned int * <i>port_names_count</i> , port_type_array_t * <i>port_types</i> , unsigned int * <i>port_types_count</i>)
---------------	---

Control a port set:

kern_return_t	port_set_allocate (task_t <i>task</i> , port_set_name_t * <i>set_name</i>)
kern_return_t	port_set_backup (task_t <i>task</i> , port_name_t <i>port_name</i> , port_t <i>backup</i> , port_t * <i>previous</i>)
kern_return_t	port_set_deallocate (task_t <i>task</i> , port_set_name_t <i>set_name</i>)

Access or modify a port set:

kern_return_t	port_set_add (task_t <i>task</i> , port_set_name_t <i>set_name</i> , port_name_t <i>port_name</i>)
kern_return_t	port_set_remove (task_t <i>task</i> , port_name_t <i>port_name</i>)
kern_return_t	port_set_status (task_t <i>task</i> , port_set_name_t <i>set_name</i> , port_name_array_t * <i>members</i> , unsigned int * <i>members_count</i>)

Message Functions

Send or receive a message:

msg_return_t	msg_send (msg_header_t * <i>header</i> , msg_option_t <i>option</i> , msg_timeout_t <i>timeout</i>)
msg_return_t	msg_receive (msg_header_t * <i>header</i> , msg_option_t <i>option</i> , msg_timeout_t <i>timeout</i>)
msg_return_t	msg_rpc (msg_header_t * <i>header</i> , msg_option_t <i>option</i> , msg_size_t <i>rcv_size</i> , msg_timeout_t <i>send_timeout</i> , msg_timeout_t <i>rcv_timeout</i>)

Virtual Memory Functions

Control virtual memory:

kern_return_t	vm_allocate (vm_task_t <i>target_task</i> , vm_address_t * <i>address</i> , vm_size_t <i>size</i> , boolean_t <i>anywhere</i>)
kern_return_t	vm_deallocate (vm_task_t <i>target_task</i> , vm_address_t <i>address</i> , vm_size_t <i>size</i>)
kern_return_t	vm_protect (vm_task_t <i>target_task</i> , vm_address_t <i>address</i> , vm_size_t <i>size</i> , boolean_t <i>set_maximum</i> , vm_prot_t <i>new_protection</i>)
kern_return_t	vm_inherit (vm_task_t <i>target_task</i> , vm_address_t <i>address</i> , vm_size_t <i>size</i> , vm_inherit_t <i>new_inheritance</i>)
kern_return_t	vm_deactivate (vm_task_t <i>target_task</i> , vm_address_t <i>address</i> , vm_size_t <i>size</i> , int <i>when</i>)
kern_return_t	vm_set_policy (vm_task_t <i>target_task</i> , vm_address_t <i>address</i> , vm_size_t <i>size</i> , int <i>policy</i>)

Access or modify the contents of virtual memory:

kern_return_t	vm_copy (vm_task_t <i>target_task</i> , vm_address_t <i>source_address</i> , vm_size_t <i>size</i> , vm_address_t <i>dest_address</i>)
kern_return_t	vm_read (vm_task_t <i>target_task</i> , vm_address_t <i>address</i> , vm_size_t <i>size</i> , pointer_t * <i>data</i> , unsigned int * <i>data_count</i>)
kern_return_t	vm_write (vm_task_t <i>target_task</i> , vm_address_t <i>address</i> , pointer_t <i>data</i> , unsigned int <i>data_count</i>)
kern_return_t	map_fd (int <i>fd</i> , vm_offset_t <i>offset</i> , vm_offset_t * <i>address</i> , boolean_t <i>find_space</i> , vm_size_t <i>size</i>)

Get information about virtual memory:

kern_return_t	vm_region (vm_task_t <i>target_task</i> , vm_address_t * <i>address</i> , vm_size_t * <i>size</i> , vm_prot_t * <i>protection</i> , vm_prot_t * <i>max_protection</i> , vm_inherit_t * <i>inheritance</i> , boolean_t * <i>shared</i> , port_t * <i>object_name</i> , vm_offset_t * <i>offset</i>)
kern_return_t	vm_statistics (vm_task_t <i>target_task</i> , vm_statistics_data_t * <i>vm_stats</i>)

Host Functions

Get this host's port:

host_t	host_self ()
host_priv_t	host_priv_self ()

Get information about a host:

kern_return_t	host_info (host_t <i>host</i> , int <i>flavor</i> , host_info_t <i>host_info</i> , unsigned int * <i>host_info_count</i>)
kern_return_t	host_kernel_version (host_t <i>host</i> , kernel_version_t <i>version</i>)

Get the name or privileged port of a processor set:

kern_return_t	host_processor_set_priv (host_priv_t <i>host_priv</i> , processor_set_t <i>processor_set_name</i> , processor_set_t * <i>processor_set</i>)
kern_return_t	host_processor_sets (host_t <i>host</i> , processor_set_name_array_t * <i>processor_set_list</i> , unsigned int * <i>processor_set_count</i>)

Get the ports of all processors on a host:

```
kern_return_t    host_processors(host_priv_t host_priv, processor_array_t *processor_list, unsigned
                                     int *processor_count)
```

Processor Functions

Get information about a processor:

```
kern_return_t    processor_info(processor_t processor, int flavor, host_t *host, processor_info_t
                                     processor_info, unsigned int *processor_info_count)
```

Get the name of the default processor set:

```
kern_return_t    processor_set_default(host_t host, processor_set_t *default_set)
```

Change the allowed scheduling policies of a processor set:

```
kern_return_t    processor_set_policy_enable(processor_set_t processor_set, int policy)
kern_return_t    processor_set_policy_disable(processor_set_t processor_set, int policy, boolean_t
                                     change_threads)
```

Get information about a processor set:

```
kern_return_t    processor_set_info(processor_set_t processor_set, int flavor, host_t *host,
                                     processor_set_info_t processor_set_info, unsigned int
                                     *processor_set_info_count)
kern_return_t    processor_set_tasks(processor_set_t processor_set, task_array_t *task_list,
                                     unsigned int *task_count)
kern_return_t    processor_set_threads(processor_set_t processor_set, thread_array_t *thread_list,
                                     unsigned int *thread_count)
```

Multiprocessor functions (not useful on single-processor systems):

```
kern_return_t    processor_assign(processor_t processor, processor_set_t new_processor_set,
                                     boolean_t wait)
kern_return_t    processor_control(processor_t processor, processor_info_t info, long *count)
kern_return_t    processor_exit(processor_t processor)
kern_return_t    processor_get_assignment(processor_t processor, processor_set_t *processor_set)
kern_return_t    processor_start(processor_t processor)
kern_return_t    processor_set_create(host_t host, port_t *new_set, port_t *new_name)
kern_return_t    processor_set_destroy(processor_set_t processor_set)
kern_return_t    processor_set_max_priority(processor_set_t processor_set, int max_priority,
                                     boolean_t change_threads)
```

Exception Functions

Raise or handle exceptions:

```
kern_return_t    exception_raise(port_t exception_port, port_t clear_port, port_t thread, port_t task,
                                     int exception, int code, int subcode)
```

void	mach_NeXT_exception (char * <i>string</i> , int <i>exception</i> , int <i>code</i> , int <i>subcode</i>)
char *	mach_NeXT_exception_string (int <i>exception</i> , int <i>code</i> , int <i>subcode</i>)
boolean_t	exc_server (msg_header_t * <i>in</i> , msg_header_t * <i>out</i>)

Error String Functions

Display or get a Mach error string:

void	mach_error (char * <i>string</i> , kern_return_t <i>errno</i>)
char *	mach_error_string (kern_return_t <i>errno</i>)

Network Name Server Functions

This section summarizes the Mach Network Name Server functions, which are not part of the Mach kernel. For more information see the *Operating System Software* manual. To use the Network Name Server functions, include in your source files the header files **mach/mach.h** and **servers/netname.h**:

```
#include <mach/mach.h>
#include <servers/netname.h>
```

Check a name into or out of the local name space:

kern_return_t	netname_check_in (port_t <i>server_port</i> , netname_name_t <i>port_name</i> , port_t <i>signature</i> , port_t <i>port_id</i>)
kern_return_t	netname_check_out (port_t <i>server_port</i> , netname_name_t <i>port_name</i> , port_t <i>signature</i>)

Look up a name on a specific host:

kern_return_t	netname_look_up (port_t <i>server_port</i> , netname_name_t <i>host_name</i> , netname_name_t <i>port_name</i> , port_t * <i>port_id</i>)
---------------	---

Bootstrap Server Functions

This section contains a summary of the Bootstrap Server functions, which are described in detail in the *Operating System Software* manual.

To use a Bootstrap Server function, include in your source files the header files **mach/mach.h** and **servers/bootstrap.h**:

```
#include <mach/mach.h>
#include <servers/bootstrap.h>
```

Look up a service:

kern_return_t	bootstrap_look_up (port_t <i>bootstrap_port</i> , name_t <i>service_name</i> , port_t * <i>service_port</i>)
kern_return_t	bootstrap_look_up_array (port_t <i>bootstrap_port</i> , name_array_t <i>service_names</i> , unsigned int <i>service_names_count</i> , port_array_t * <i>service_port</i> , unsigned int * <i>service_ports_count</i> , boolean_t

**all services known)*

Find out whether a service is active:

```
kern_return_t    bootstrap_status(port_t bootstrap_port, name_t service_name, boolean_t
                                *service_active°)
```

Get information about all known services:

```
kern_return_t bootstrap_info(port_t bootstrap_port, name_array_t *service_names, unsigned int
                             *service_names_count, name_array_t *server_names, unsigned int
                             *server_names_count, bool_array_t *service_active, unsigned int
                             *service_active_count)
```

Check in a service:

```
kern_return_t    bootstrap_check_in(port_t bootstrap_port, name_t service_name, port_all_t
                                     *service_port)
```

```
kern_return_t bootstrap_create_service(port_t bootstrap_port, name_t service_name, port_t
                                     *service_port)
```

```
kern_return_t bootstrap_register(port_t bootstrap_port, name_t service_name, port_t
                                service_port)
```

Create a new bootstrap port:

```
kern return t    bootstrap_subset(port t bootstrap port, port t requestor port, port t *subset port)
```

Kernel-Server Loader Functions

This section contains a summary of the kernel-server loader functions, which are described in detail in the *Operating System Software* manual. Use these functions in a user-level program to communicate with the kernel-server loader, which controls all loadable kernel servers.

To use these functions, include in your source files the header files **`mach/mach.h`** and **`kernserv/kern_loader_types.h`**. Most functions also require that you include the header file **`kernserv/kern_loader.h`**; the error functions require that you include the header file **`kernserv/kern_loader_error.h`**. Two other header files may be necessary: **`kernserv/kern_loader_reply_handler.h`** and **`kernserv/kern_loader_reply.h`**.

To use these functions, you must compile with the kernload library. For example:

```
cc myprog.c -lkernload
```

Get a port to use in other kernel-server loader functions:

```
kern return t      kern loader look up(port t*loader port)
```

```
kern_return_t kern_loader_server_com_port(port_t loader_port, port_t task_port, server_name_t
server name, port_t *server_com_port)
```

Get or display an error string:

```
void      kern_loader_error(const char *string, kern_return_t errno)
```

```
const char *      kern_loader_error_string(kern return t_errno)
```

Affect the runnability of a loadable kernel server:

kern_return_t	kern_loader_add_server (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_reloc_t <i>server_reloc</i>)
kern_return_t	kern_loader_delete_server (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_name_t <i>server_name</i>)
kern_return_t	kern_loader_load_server (port_t <i>loader_port</i> , server_name_t <i>server_name</i>)
kern_return_t	kern_loader_unload_server (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_name_t <i>server_name</i>)

Get information from or about a loaded kernel server:

kern_return_t	kern_loader_get_log (port_t <i>loader_port</i> , port_t <i>server_com_port</i> , port_t <i>reply_port</i>)
kern_return_t	kern_loader_log_level (port_t <i>loader_port</i> , port_t <i>server_com_port</i> , int <i>log_level</i>)
kern_return_t	kern_loader_server_info (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_name_t <i>server_name</i> , server_state_t * <i>server_state</i> , vm_address_t * <i>load_address</i> , vm_size_t * <i>load_size</i> , server_reloc_t <i>relocatable</i> , server_reloc_t <i>loadable</i> , port_name_array_t * <i>port_list</i> , unsigned int * <i>port_list_count</i> , port_name_string_array_t * <i>port_names</i> , unsigned int * <i>port_names_count</i> , boolean_array_t * <i>advertised</i> , unsigned int * <i>advertised_count</i>)
kern_return_t	kern_loader_server_task_port (port_t <i>loader_port</i> , port_t <i>kernel_port</i> , server_name_t <i>server_name</i> , port_t * <i>server_task_port</i>)

Get general information from the kernel-server loader:

kern_return_t	kern_loader_server_list (port_t <i>loader_port</i> , server_name_array_t * <i>server_names</i> , unsigned int * <i>server_names_count</i>)
kern_return_t	kern_loader_status_port (port_t <i>loader_port</i> , port_t <i>listen_port</i>)

Request or handle an asynchronous message from the kernel-server loader:

kern_return_t	kern_loader_ping (port_t <i>loader_port</i> , port_t <i>ping_port</i> , int <i>id</i>)
kern_return_t	kern_loader_reply_handler (msg_header_t * <i>msg</i> , kern_loader_reply_t * <i>kern_loader_reply</i>)

Shut down or reconfigure the kernel-server loader:

kern_return_t	kern_loader_abort (port_t <i>loader_port</i> , port_t <i>priv_port</i> , boolean_t <i>restart</i>)
---------------	--