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

cthread_t	**cthread_fork(**any_t (***function*)(), any_t *arg***)**
any_t	**cthread_join(**cthread_t *t***)**
void	**cthread_detach(**cthread_t *t***)**
void	**cthread_yield()**
kern_return_t	**cthread_abort(**cthread_t *t***)**
void	**cthread_exit(**any_t *result***)**

Get access to a thread

cthread_t	**cthread_self()**
thread_t	**cthread_thread(**cthread_t *t***)**

Associate a string with a thread

char *	**cthread_name(**cthread_t *t***)**
void	**cthread_set_name(**cthread_t *t*, char **name***)*

Associate data with a thread

any_t	**cthread_data(**cthread_t *t***)**
void	**cthread_set_data(**cthread_t *t*, any_t *data***)**

Change the priority of a thread

kern_return_t	**cthread_max_priority(**cthread_t *t*, processor_set_t *processor_set*, int *max_priority***)**
kern_return_t	**cthread_priority(**cthread_t *t*, int *priority*, boolean_t *set_max***)**

Get or set the UNIX™ error number of this thread

int	**cthread_errno()**
void	**cthread_set_errno_self(**int *error***)**

Get or set the maximum number of threads in this task

int	**cthread_limit()**
void	**cthread_set_limit(**int *limit***)**
int	**cthread_count()**

Mutex Functions

Control a mutex

mutex_t	**mutex_alloc()**
void	**mutex_init(**struct mutex **m***)
void	**mutex_clear(**struct mutex **m***)
void	**mutex_free(**mutex_t *m***)**

Associate a string with a mutex

char *	**mutex_name(**mutex_t *m***)**
void	**mutex_set_name(**mutex_t *m*, char **name***)**

Synchronize a mutex

void	**mutex_lock(**mutex_t *m***)**
int	**mutex_try_lock(**mutex_t *m***)**
void	**mutex_unlock(**mutex_t *m***)**

Condition Functions

Control a condition variable

condition_t	**condition_alloc()**
void	**condition_init(**struct condition **c***)
void	**condition_clear(**struct condition **c***)
void	**condition_free(**condition_t *c***)**

Associate a string with a condition variable

char *	**condition_name(**condition_t *c***)**
void	**condition_set_name(**condition_t *c*, char **name***)**

Synchronize a condition variable

void	**condition_broadcast(**condition_t *c***)**
void	**condition_signal(**condition_t *c***)**
void	**condition_wait(**condition_t *c*, mutex_t *m***)**

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 *parent_task*, boolean_t *inherit_memory*, task_t **child_task***)*
kern_return_t **task_suspend(**task_t *target_task***)**
kern_return_t **task_resume(**task_t *target_task***)**
kern_return_t **task_terminate(**task_t *target_task***)**

Access a task or its threads:

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

Access a task's special ports:

kern_return_t **task_get_special_port(**task_t *task*, int *which_port*, port_t **special_port***)*
kern_return_t **task_set_special_port(**task_t *task*, int *which_port*, port_t *special_port***)**
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 *task*, int *pid*, task_t **result_task***)*
kern_return_t **unix_pid(**task_t *target_task*, int **pid***)*

Set a task's scheduling priority:

kern_return_t **task_priority(**task_t *task*, int *priority*, boolean_t *change_threads***)**

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

kern_return_t **task_assign(**task_t *task*, processor_set_t *new_processor_set*, boolean_t *assign_threads***)**
kern_return_t **task_assign_default(**task_t *task*, boolean_t *assign_threads***)**
kern_return_t **task_get_assignment(**task_t *task*, processor_set_t **processor_set***)**

Thread Functions

Control a thread:

kern_return_t **thread_create(**task_t *parent_task*, thread_t **child_thread***)**
kern_return_t **thread_suspend(**thread_t *target_thread***)**
kern_return_t **thread_resume(**thread_t *target_thread***)**
kern_return_t **thread_terminate(**thread_t *target_thread***)**
kern_return_t **thread_abort(**thread_t *target_thread***)**

Get information about a thread:

kern_return_t **thread_info(**thread_t *target_thread*, int *flavor*, thread_info_t *thread_info*, unsigned int **thread_info_count***)**

Access a thread's state:

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

Access a thread's special ports:

kern_return_t **thread_get_special_port(**thread_t *thread*, int *which_port*, port_t **special_port***)**
kern_return_t **thread_set_special_port(**thread_t *thread*, int *which_port*, port_t *special_port***)**
port_t **thread_reply()**
thread_t **thread_self()**

Affect the scheduling policy or priority of a thread:

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

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

kern_return_t **thread_assign(**thread_t *thread*, processor_set_t *new_processor_set***)**
kern_return_t **thread_assign_default(**thread_t *thread***)**
kern_return_t **thread_get_assignment(**thread_t *thread*, processor_set_t **processor_set***)**

Port Functions

Control a port:

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

Give a task rights to a port:

kern_return_t **port_insert_receive(**task_t *task*, port_t *my_port*, port_name_t *its_name***)**
kern_return_t **port_insert_send(**task_t *task*, port_t *my_port*, port_name_t *its_name***)**

Remove a task's rights to a port:

kern_return_t **port_extract_receive(**task_t *task*, port_name_t *its_name*, port_t **its_port***)*
kern_return_t **port_extract_send(**task_t *task*, port_name_t *its_name*, port_t **its_port***)*

Get information about a port:

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

Get information about a task's port name space:

kern_return_t **port_names(**task_t *task*, port_name_array_t **port_names*, unsigned int **port_names_count*, port_type_array_t **port_types*, unsigned int **port_types_count***)*

Control a port set:

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

Access or modify a port set:

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

Message Functions

Send or receive a message:

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

Virtual Memory Functions

Control virtual memory:

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

Access or modify the contents of virtual memory:

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

Get information about virtual memory:

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

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

Get the name or privileged port of a processor set:

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

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 **string*, int *exception*, int *code*, int *subcode***)**
char * **mach_NeXT_exception_string(**int *exception*, int *code*, int *subcode***)**
boolean_t **exc_server(**msg_header_t **in*, msg_header_t **out***)**

Error String Functions

Display or get a Mach error string:

void **mach_error(**char **string*, kern_return_t *errno***)**
char * **mach_error_string(**kern_return_t *errno***)**

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 *server_port*, netname_name_t *port_name*, port_t *signature*, port_t *port_id***)**
kern_return_t **netname_check_out(**port_t *server_port*, netname_name_t *port_name*, port_t *signature***)**

Look up a name on a specific host:

kern_return_t **netname_look_up(**port_t *server_port*, netname_name_t *host_name*, netname_name_t *port_name*, port_t ***port_id***)**

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 *bootstrap_port*, name_t *service_name*, port_t ***service_port***)**
kern_return_t **bootstrap_look_up_array(**port_t *bootstrap_port*, name_array_t *service_names*, unsigned int *service_names_count*, port_array_t ***service_port*, unsigned int ***service_ports_count*, 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 *loader_port*, port_t *task_port*, server_reloc_t *server_reloc***)**
kern_return_t **kern_loader_delete_server(**port_t *loader_port*, port_t *task_port*, server_name_t *server_name***)**
kern_return_t **kern_loader_load_server(**port_t *loader_port*, server_name_t *server_name***)**
kern_return_t **kern_loader_unload_server(**port_t *loader_port*, port_t *task_port*, server_name_t *server_name***)**

Get information from or about a loaded kernel server:

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

Get general information from the kernel-server loader:

kern_return_t **kern_loader_server_list(**port_t *loader_port*, server_name_array_t **server_names*, unsigned int **server_names_count***)**
kern_return_t **kern_loader_status_port(**port_t *loader_port*, port_t *listen_port***)**

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

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

Shut down or reconfigure the kernel-server loader:

kern_return_t **kern_loader_abort(**port_t *loader_port*, port_t *priv_port*, boolean_t *restart***)**