C

Summary of Kernel Support€Functions

This appendix summarizes the kernel support functions (and some macros that behave like functions) that loadable kernel servers can call. Within the general categories of ªGeneral Functionsº and ªNetwork Functions,º function declarations are further subgrouped to help you identify their interrelationships.

Chapter 10, ªKernel Support Functions,º contains full descriptions of all the functions listed here. In addition, loadable kernel servers can use many Mach kernel functions, which are described in a section of Chapter 4, ªMach Functions.º The Mach kernel functions are summarized in the manual, *NeXTSTEP Programmer Interface Summary*.

General Functions

This section contains a summary of the general purpose kernel support functions. Most of the functions and macros in this section are declared through either the **kernserv/kern_server_types.h** or **kernserv/prototypes.h** header file.

Time Functions

Busy-wait for a certain amount of time:

void **DELAY(**unsigned int *usecs***)**

Get or set the current time:

ns_time_t **clock_value(**clock_types_t *which_clock***)**
void **set_clock(**clock_types_t *which_clock*, ns_time_t *ns***)**

Get information about a clock:

chrono_attributes_t
clock_attributes(clock_types_t *which_clock***)**

Convert between **ns_time_t** and **timeval** data formats:

void **ns_time_to_timeval(**ns_time_t *ns*, struct timeval **tv***)**
ns_time_t **timeval_to_ns_time(**struct timeval **tv***)**

Schedule or unschedule a function to be called later:

void **ns_abstimeout(**func *function*, vm_address_t *arg*, ns_time_t *deadline*, int *priority***)**
void **ns_timeout(**func *function*, vm_address_t *arg*, ns_time_t *time*, int *priority***)**
boolean_t **ns_untimeout(**func *function*, vm_address_t *arg***)**

Memory Functions

Make addresses pageable or memory-resident:

kern_return_t **kern_serv_unwire_range(**kern_server_t **ksp*, vm_address_t *address*, vm_size_t€*size***)**
kern_return_t **kern_serv_wire_range(**kern_server_t **ksp*, vm_address_t *address*, vm_size_t *size***)**

Copy or initialize data:

void **bcopy(**void **from*, void **to*, int *length***)**
void **bytecopy(**void **from*, void **to*, int *length***)**
void **bzero(**void **address*, int *length***)**

Allocate or free memory:

void * **kalloc(**int *size***)**
void **kfree(**void **address*, int *size***)**
void * **kget(**int *size***)**

Critical Section and Synchronization Functions

Use read and write locks:

lock_t **lock_alloc(**void**)**
void **lock_free(**lock_t *lock***)**
void **lock_done(**lock_t *lock***)**
void **lock_init(**lock_t *lock*, boolean_t *can_sleep***)**
void **lock_read(**lock_t *lock***)**
void **lock_write(**lock_t *lock***)**

Use simple, nonsleeping locks:

void **simple_lock(**simple_lock_t *lock***)**
simple_lock_t **simple_lock_alloc(**void**)**
void **simple_lock_free(**simple_lock_t *lock***)**
void **simple_lock_init(**simple_lock_t *lock***)**
void **simple_unlock(**simple_lock_t *lock***)**

Cause a thread to sleep or wake up:

void **assert_wait(**int *event*, boolean_t *interruptible***)**
void **clear_wait(**thread_t *thread*, int *result*, boolean_t *interrupt_only***)**
void **thread_block(**void**)**
void **thread_set_timeout(**int *ticks***)**
void **thread_sleep(**int *event*, simple_lock_t *lock*, boolean_t *interruptible***)**
void **thread_wakeup(**int *event***)**

General Task and Thread Functions

Get information about this thread or task:

task_t **current_task(**void**)**
int **thread_wait_result(**void**)**

Create or kill a thread:

thread_t	**kernel_thread(**task_t *task*, void (**start*)(void)**)**
void	**thread_halt_self(**void**)**

Port and Message Functions

Request notification messages, such as port death notification:

kern_return_t	**kern_serv_notify(**kern_server_t **ksp*, port_t *reply_port*, port_t *request_port***)**

Get the kernel's task port:

port_t	**kern_serv_kernel_task_port(**void**)**

Get or set information about this server's ports:

port_t	**kern_serv_bootstrap_port(**kern_server_t **ksp***)**
port_t	**kern_serv_local_port(**kern_server_t **ksp***)**
port_t	**kern_serv_notify_port(**kern_server_t **ksp***)**
void	**kern_serv_port_gone(**kern_server_t **ksp*, port_name_t *port***)**
kern_return_t	**kern_serv_port_proc(**kern_server_t **ksp*, port_all_t *port*, port_map_proc_t*€function*, int *arg***)**
kern_return_t	**kern_serv_port_serv(**kern_server_t **ksp*, port_all_t *port*, port_map_proc_t*€function*, int *arg***)**
port_set_name_t	**kern_serv_port_set(**kern_server_t **ksp***)**

Hardware Interface Functions

Set up or remove an interrupt handler:

int	**install_polled_intr(**int *which*, int (**my_intr*)(void)**)**
int	**uninstall_polled_intr(**int *which*, int (**my_intr*)(void)**)**

Get or test a virtual address that corresponds to a hardware address:

caddr_t	**map_addr(**caddr_t *address*, int *size***)**
int	**probe_rb(**void **address***)**

Change or determine the processor level:

int	**curipl(**void**)**
int	**spl0(**void**)**, **spl1(**void**)**, **spl2(**void**)**, **spl3(**void**)**, **spl4(**void**)**, **spl5(**void**)**, **spl6(**void**)**, **spl7(**void**)**
void	**splx(**int *priority***)**

Logging and Debugging Functions

Kill the loadable kernel server:

void	**ASSERT(**int *expression***)**
kern_return_t	**kern_serv_panic(**port_t *bootstrap_port*, panic_msg_t *message***)**
void	**panic(**char **string***)**

Log a message:

void	**kern_serv_log(**kern_server_t **ksp*, int *log_level*, char **format*, *arg1*, ..., *arg5***)**
int	**log(**int *level*, char **format*, *arg*, ...**)**
int	**printf(**char **format*, *arg*, ...**)**

UNIX Support Functions

In a UNIX-style server, determine whether the user has root privileges:

int **suser(**void**)**

In a UNIX-style server, wait for I/O completion on a buffer:

void **biodone(**struct buf *_bp_**)**
void **biowait(**struct buf *_bp_**)**

In a UNIX-style server, copy data between user and kernel address space:

int **copyin(**void *_from_, void *_to_, int _length_**)**
int **copyout(**void *_from_, void *_to_, int _length_**)**

In a UNIX-style server, implement the **select()** system call:

int **selthreadcache(**void **_waiterPtr_**)**
void **selthreadclear(**void **_waiterPtr_**)**
int **selwakeup(**void *_waiter_, int _collided_**)**

Miscellaneous Functions

Modify or inspect a string:

int **sprintf(**char *_string_, char *_format_, _arg_, ...**)**
char * **strcat(**char *_string1_, char *_string2_**)**
int **strcmp(**char *_string1_, char *_string2_**)**
int **strncmp(**char *_string1_, char *_string2_, unsigned long _length_**)**
char * **strcpy(**char *_to_, char *_from_**)**
char * **strncpy(**char *_to_, char *_from_, unsigned long _length_**)**
int **strlen(**char *_string_**)**

Call a function from the main thread:

kern_return_t **kern_serv_callout(**kern_server_t *_ksp_, void (*_func_)(void *), void *_arg_**)**

Network Functions

This section contains a summary of the network-specific kernel support functions, which are described in detail in Chapter 10. A general discussion of networking drivers and protocols is in Chapter 8, ªNetwork Modules.º

Netif Functions

To use these functions, you need to include the header file **net/netif.h**.

Initialize and install a new netif:

netif_t **if_attach(**if_init_func_t _init_func_, if_input_func_t _input_func_, if_output_func_t€_output_func_, if_getbuf_func_t _getbuf_func_, if_control_func_t _control_func_, const char *_name_, unsigned€int€_unit_, const char *_type_, unsigned int _mtu_, unsigned€int€_flags_,

netif_class_t€*class*, void **private*)
void **if_registervirtual(**if_attach_func_t *attach_func*, void **private***)*

Remove a netif:

void **if_detach(**netif_t *netif***)**

Get or set data for a netif:

unsigned int **if_collisions(**netif_t *netif***)**
void **if_collisions_set(**netif_t *netif*, unsigned int *collisions***)**
unsigned int **if_flags(**netif_t *netif***)**
void **if_flags_set(**netif_t *netif*, unsigned int *flags***)**
unsigned int **if_ierrors(**netif_t *netif***)**
void **if_ierrors_set(**netif_t *netif*, unsigned int *ierrors***)**
unsigned int **if_oerrors(**netif_t *netif***)**
void **if_oerrors_set(**netif_t *netif*, unsigned int *oerrors***)**
unsigned int **if_ipackets(**netif_t *netif***)**
void **if_ipackets_set(**netif_t *netif*, unsigned int *ipackets***)**
unsigned int **if_opackets(**netif_t *netif***)**
void **if_opackets_set(**netif_t *netif*, unsigned int *opackets***)**
unsigned int **if_mtu(**netif_t *netif***)**
const char * **if_name(**netif_t *netif***)**
void * **if_private(**netif_t *netif***)**
const char * **if_type(**netif_t *netif***)**
unsigned int **if_unit(**netif_t *netif***)**

Call a function implemented by a network module:

int **if_control(**netif_t *netif*, const char **command*, void **data***)**
netbuf_t **if_getbuf(**netif_t *netif***)**
int **if_init(**netif_t *netif***)**
int **if_ioctl(**netif_t *netif*, unsigned int *command*, void **data***)**
int **if_output(**netif_t *netif*, netbuf_t *packet*, void **address***)*

Get information about netifs:

netif_class_t **if_class(**netif_t *netif***)**
netif_t **iflist_first(**void**)**
netif_t **iflist_next(**netif_t *netif***)**

Dispatch a packet to a protocol handler:

int **if_handle_input(**netif_t *netif*, netbuf_t *packet*, void **extra***)*

Netbuf Functions

You should include the header file **net/netbuf.h** when you use these functions.

Allocate or free a netbuf or its wrapper:

netbuf_t **nb_alloc(**unsigned int *size***)**
netbuf_t **nb_alloc_wrapper(**void **data*, unsigned int *size*, void€*freefunc*(void€*), void€**freefunc_arg***)**
void **nb_free(**netbuf_t *nb***)**
void **nb_free_wrapper(**netbuf_t *nb***)**

Change the size of a netbuf:

int **nb_grow_bot(**netbuf_t *nb*, unsigned int *size***)**
int **nb_shrink_bot(**netbuf_t *nb*, unsigned int *size***)**

int	**nb_grow_top(**netbuf_t *nb*, unsigned int *size***)**
int	**nb_shrink_top(**netbuf_t *nb*, unsigned int *size***)**

Access the data in a netbuf:

char *	**nb_map(**netbuf_t *nb***)**
int	**nb_read(**netbuf_t *nb*, unsigned int *offset*, unsigned int *size*, void **target***)**
int	**nb_write(**netbuf_t *nb*, unsigned int *offset*, unsigned int *size*, void **source***)**
unsigned int	**nb_size(**netbuf_t *nb***)**

Miscellaneous Functions

For the host-network conversion functions, you need to include the header file **netinet/in.h**. For **inet_queue()**, you should include both **net/netif.h** and **net/netbuf.h**.

Convert values between host and network byte order:

u_long	**htonl(**u_long *hostlong***)**
u_short	**htons(**u_short *hostshort***)**
u_long	**ntohl(**u_long *netlong***)**
u_short	**ntohs(**u_short *netshort***)**

Give an IP input packet to the kernel for processing:

void	**inet_queue(**netif_t *netif*, netbuf_t *netbuf***)**