

**NAME**

**sysctl, sysctlbyname, sysctlnametomib** - get or set system information

**LIBRARY**

Standard C Library (libc, -lc)

**SYNOPSIS**

```
#include <sys/sysctl.h>
```

*int*

```
sysctl(const int *name, u_int namelen, void *oldp, size_t *oldlenp, const void *newp, size_t newlen);
```

*int*

```
sysctlbyname(const char *name, void *oldp, size_t *oldlenp, const void *newp, size_t newlen);
```

*int*

```
sysctlnametomib(const char *name, int *mibp, size_t *sizep);
```

**DESCRIPTION**

The **sysctl()** function retrieves system information and allows processes with appropriate privileges to set system information. The information available from **sysctl()** consists of integers, strings, and tables. Information may be retrieved and set from the command interface using the **sysctl(8)** utility.

Unless explicitly noted below, **sysctl()** returns a consistent snapshot of the data requested. Consistency is obtained by locking the destination buffer into memory so that the data may be copied out without blocking. Calls to **sysctl()** are serialized to avoid deadlock.

The state is described using a ‘‘Management Information Base’’ (MIB) style name, listed in *name*, which is a *namelen* length array of integers.

The **sysctlbyname()** function accepts an ASCII representation of the name and internally looks up the integer name vector. Apart from that, it behaves the same as the standard **sysctl()** function.

The information is copied into the buffer specified by *oldp*. The size of the buffer is given by the location specified by *oldlenp* before the call, and that location gives the amount of data copied after a successful call and after a call that returns with the error code ENOMEM. If the amount of data available is greater than the size of the buffer supplied, the call supplies as much data as fits in the buffer provided and returns with the error code ENOMEM. If the old value is not desired, *oldp* and *oldlenp* should be set to NULL.

The size of the available data can be determined by calling `sysctl()` with the NULL argument for *oldp*. The size of the available data will be returned in the location pointed to by *oldlenp*. For some operations, the amount of space may change often. For these operations, the system attempts to round up so that the returned size is large enough for a call to return the data shortly thereafter.

To set a new value, *newp* is set to point to a buffer of length *newlen* from which the requested value is to be taken. If a new value is not to be set, *newp* should be set to NULL and *newlen* set to 0.

The `sysctlnametomib()` function accepts an ASCII representation of the name, looks up the integer name vector, and returns the numeric representation in the mib array pointed to by *mibp*. The number of elements in the mib array is given by the location specified by *sizep* before the call, and that location gives the number of entries copied after a successful call. The resulting *mib* and *size* may be used in subsequent `sysctl()` calls to get the data associated with the requested ASCII name. This interface is intended for use by applications that want to repeatedly request the same variable (the `sysctl()` function runs in about a third the time as the same request made via the `sysctlbyname()` function). The `sysctlnametomib()` function is also useful for fetching mib prefixes and then adding a final component. For example, to fetch process information for processes with pid's less than 100:

```
int i, mib[4];
size_t len;
struct kinfo_proc kp;

/* Fill out the first three components of the mib */
len = 4;
sysctlnametomib("kern.proc.pid", mib, &len);

/* Fetch and print entries for pid's < 100 */
for (i = 0; i < 100; i++) {
    mib[3] = i;
    len = sizeof(kp);
    if (sysctl(mib, 4, &kp, &len, NULL, 0) == -1)
        perror("sysctl");
    else if (len > 0)
        printkproc(&kp);
}
```

The top level names are defined with a CTL\_ prefix in `<sys/sysctl.h>`, and are as follows. The next and subsequent levels down are found in the include files listed here, and described in separate sections below.

Name	Next Level Names	Description
CTL_DEBUG	<sys/sysctl.h>	Debugging
CTL_VFS	<sys/mount.h>	File system
CTL_HW	<sys/sysctl.h>	Generic CPU, I/O
CTL_KERN	<sys/sysctl.h>	High kernel limits
CTL_MACHDEP	<sys/sysctl.h>	Machine dependent
CTL_NET	<sys/socket.h>	Networking
CTL_USER	<sys/sysctl.h>	User-level
CTL_VM	<vm/vm_param.h>	Virtual memory

For example, the following retrieves the maximum number of processes allowed in the system:

```
int mib[2], maxproc;
size_t len;

mib[0] = CTL_KERN;
mib[1] = KERN_MAXPROC;
len = sizeof(maxproc);
sysctl(mib, 2, &maxproc, &len, NULL, 0);
```

To retrieve the standard search path for the system utilities:

```
int mib[2];
size_t len;
char *p;

mib[0] = CTL_USER;
mib[1] = USER_CS_PATH;
sysctl(mib, 2, NULL, &len, NULL, 0);
p = malloc(len);
sysctl(mib, 2, p, &len, NULL, 0);
```

## CTL\_DEBUG

The debugging variables vary from system to system. A debugging variable may be added or deleted without need to recompile `sysctl()` to know about it. Each time it runs, `sysctl()` gets the list of debugging variables from the kernel and displays their current values. The system defines twenty (*struct ctldebug*) variables named *debug0* through *debug19*. They are declared as separate variables so that they can be individually initialized at the location of their associated variable. The loader prevents multiple use of the same variable by issuing errors if a variable is initialized in more than one place. For example, to export the variable *dospecialcheck* as a debugging variable, the following declaration would be used:

```
int dospecialcheck = 1;
struct ctldebug debug5 = { "dospecialcheck", &dospecialcheck };
```

### CTL\_VFS

A distinguished second level name, VFS\_GENERIC, is used to get general information about all file systems. One of its third level identifiers is VFS\_MAXTYPENUM that gives the highest valid file system type number. Its other third level identifier is VFS\_CONF that returns configuration information about the file system type given as a fourth level identifier (see `getvfsbyname(3)` as an example of its use). The remaining second level identifiers are the file system type number returned by a `statfs(2)` call or from VFS\_CONF. The third level identifiers available for each file system are given in the header file that defines the mount argument structure for that file system.

### CTL\_HW

The string and integer information available for the CTL\_HW level is detailed below. The changeable column shows whether a process with appropriate privilege may change the value.

Second Level Name	Type	Changeable
HW_MACHINE	string	no
HW_MODEL	string	no
HW_NCPU	integer	no
HW_BYTEORDER	integer	no
HW_PHYSMEM	integer	no
HW_USERMEM	integer	no
HW_PAGESIZE	integer	no
HW_FLOATINGPT	integer	no
HW_MACHINE_ARCH		
	string	no
HW_REALMEM	integer	no
HW_AVAILPAGES	integer	no

#### HW\_MACHINE

The machine class.

#### HW\_MODEL

The machine model

#### HW\_NCPU

The number of cpus.

#### HW\_BYTEORDER

The byteorder (4321 or 1234).

#### HW\_PHYSMEM

Amount of physical memory (in bytes), minus the amount used by the kernel, pre-loaded modules, and (on x86) the dcons buffer.

#### HW\_USERMEM

Amount of memory (in bytes) which is not wired.

#### HW\_PAGESIZE

The software page size.

#### HW\_FLOATINGPT

Nonzero if the floating point support is in hardware.

#### HW\_MACHINE\_ARCH

The machine dependent architecture type.

#### HW\_REALMEM

Amount of memory (in bytes) reported by the firmware. That value is sometimes not sane; in that case, the kernel reports the max memory address instead.

#### HW\_AVAILPAGES

The same value as HW\_PHYSMEM, measured in pages rather than bytes.

### CTL\_KERN

The string and integer information available for the CTL\_KERN level is detailed below. The changeable column shows whether a process with appropriate privilege may change the value. The types of data currently available are process information, system vnode, the open file entries, routing table entries, virtual memory statistics, load average history, and clock rate information.

Second Level Name	Type	Changeable
KERN_ARGMAX	integer	no
KERN_BOOTFILE	string	yes
KERN_BOOTTIME	struct timeval	no
KERN_CLOCKRATE	struct clockinfo	no
KERN_FILE	struct xfile	no
KERN_HOSTID	integer	yes
KERN_HOSTUUID	string	yes
KERN_HOSTNAME	string	yes

KERN_JOB_CONTROL	integer	no
KERN_MAXFILES	integer	yes
KERN_MAXFILESPPROC	integer	yes
KERN_MAXPROC	integer	no
KERN_MAXPROCPERUID	integer	yes
KERN_MAXVNODES	integer	yes
KERN_NGROUPS	integer	no
KERN_NISDOMAINNAME	string	yes
KERN_OSRELDATE	integer	no
KERN_OSRELEASE	string	no
KERN_OSREV	integer	no
KERN_OSTYPE	string	no
KERN_POSIX1	integer	no
KERN_PROC	node	not applicable
KERN_QUANTUM	integer	yes
KERN_SAVED_IDS	integer	no
KERN_SECURELVL	integer	raise only
KERN_UPDATEINTERVAL	integer	no
KERN_VERSION	string	no

**KERN\_ARGMAX**

The maximum bytes of argument to `execve(2)`.

**KERN\_BOOTFILE**

The full pathname of the file from which the kernel was loaded.

**KERN\_BOOTTIME**

A *struct timeval* structure is returned. This structure contains the time that the system was booted.

**KERN\_CLOCKRATE**

A *struct clockinfo* structure is returned. This structure contains the clock, statistics clock and profiling clock frequencies, the number of micro-seconds per hz tick and the skew rate.

**KERN\_FILE**

Return the entire file table. The returned data consists of an array of *struct xfile*, whose size depends on the current number of such objects in the system.

**KERN\_HOSTID**

Get or set the host ID.

**KERN\_HOSTUUID**

Get or set the host's universally unique identifier (UUID).

**KERN\_HOSTNAME**

Get or set the hostname.

**KERN\_JOB\_CONTROL**

Return 1 if job control is available on this system, otherwise 0.

**KERN\_MAXFILES**

The maximum number of files that may be open in the system.

**KERN\_MAXFILESPPROC**

The maximum number of files that may be open for a single process. This limit only applies to processes with an effective uid of nonzero at the time of the open request. Files that have already been opened are not affected if the limit or the effective uid is changed.

**KERN\_MAXPROC**

The maximum number of concurrent processes the system will allow.

**KERN\_MAXPROCPUID**

The maximum number of concurrent processes the system will allow for a single effective uid. This limit only applies to processes with an effective uid of nonzero at the time of a fork request. Processes that have already been started are not affected if the limit is changed.

**KERN\_MAXVNODES**

The maximum number of vnodes available on the system.

**KERN\_NGROUPS**

The maximum number of supplemental groups.

**KERN\_NISDOMAINNAME**

The name of the current YP/NIS domain.

**KERN\_OSRELDATE**

The kernel release version in the format *MmmRxx*, where *M* is the major version, *mm* is the two digit minor version, *R* is 0 if release branch, otherwise 1, and *xx* is updated when the available APIs change.

The userland release version is available from `<osreldate.h>`; parse this file if you need to get

the release version of the currently installed userland.

#### KERN\_OSRELEASE

The system release string.

#### KERN\_OSREV

The system revision string.

#### KERN\_OSTYPE

The system type string.

#### KERN\_POSIX1

The version of IEEE Std 1003.1 ("POSIX.1") with which the system attempts to comply.

#### KERN\_PROC

Return selected information about specific running processes.

For the following names, an array of *struct kinfo\_proc* structures is returned, whose size depends on the current number of such objects in the system.

<b>Third Level Name</b>	<b>Fourth Level</b>
KERN_PROC_ALL	None
KERN_PROC_PID	A process ID
KERN_PROC_PGRP	A process group
KERN_PROC_TTY	A tty device
KERN_PROC_UID	A user ID
KERN_PROC_RUID	A real user ID

If the third level name is KERN\_PROC\_ARGS then the command line argument array is returned in a flattened form, i.e., zero-terminated arguments follow each other. The total size of array is returned. It is also possible for a process to set its own process title this way. If the third level name is KERN\_PROC\_PATHNAME, the path of the process' text file is stored. For KERN\_PROC\_PATHNAME, a process ID of -1 implies the current process.

<b>Third Level Name</b>	<b>Fourth Level</b>
KERN_PROC_ARGS	A process ID
KERN_PROC_PATHNAME	A process ID

#### KERN\_QUANTUM

The maximum period of time, in microseconds, for which a process is allowed to run without



being preempted if other processes are in the run queue.

#### KERN\_SAVED\_IDS

Returns 1 if saved set-group and saved set-user ID is available.

#### KERN\_SECURELVL

The system security level. This level may be raised by processes with appropriate privilege. It may not be lowered.

#### KERN\_VERSION

The system version string.

### CTL\_NET

The string and integer information available for the CTL\_NET level is detailed below. The changeable column shows whether a process with appropriate privilege may change the value.

Second Level Name	Type	Changeable
PF_ROUTE	routing messages	no
PF_INET	IPv4 values	yes
PF_INET6	IPv6 values	yes

#### PF\_ROUTE

Return the entire routing table or a subset of it. The data is returned as a sequence of routing messages (see route(4) for the header file, format and meaning). The length of each message is contained in the message header.

The third level name is a protocol number, which is currently always 0. The fourth level name is an address family, which may be set to 0 to select all address families. The fifth, sixth, and seventh level names are as follows:

Fifth level	Sixth Level	Seventh Level
NET_RT_FLAGS	rtflags	None
NET_RT_DUMP	None	None or fib number
NET_RT_IFLIST	0 or if_index	None
NET_RT_IFMALIST	0 or if_index	None
NET_RT_IFLISTL		

0 or `if_index` None  
NET\_RT\_NHOPS  
None fib number

The NET\_RT\_IFMALIST name returns information about multicast group memberships on all interfaces if 0 is specified, or for the interface specified by *if\_index*.

The NET\_RT\_IFLISTL is like NET\_RT\_IFLIST, just returning message header structs with additional fields allowing the interface to be extended without breaking binary compatibility. The NET\_RT\_IFLISTL uses 'l' versions of the message header structures: *struct if\_msghdrl* and *struct ifa\_msghdrl*.

NET\_RT\_NHOPS returns all nexthops for specified address family in given fib.

## PF\_INET

Get or set various global information about the IPv4 (Internet Protocol version 4). The third level name is the protocol. The fourth level name is the variable name. The currently defined protocols and names are:

Protocol	Variable	Type	Changeable
icmp	bmcastecho	integer	yes
icmp	maskrepl	integer	yes
ip	forwarding	integer	yes
ip	redirect	integer	yes
ip	ttl	integer	yes
udp	checksum	integer	yes

The variables are as follows:

icmp.bmcastecho

Returns 1 if an ICMP echo request to a broadcast or multicast address is to be answered.

icmp.maskrepl

Returns 1 if ICMP network mask requests are to be answered.

ip.forwarding

Returns 1 when IP forwarding is enabled for the host, meaning that the host is acting as a router.

**ip.redirect**

Returns 1 when ICMP redirects may be sent by the host. This option is ignored unless the host is routing IP packets, and should normally be enabled on all systems.

**ip.ttl** The maximum time-to-live (hop count) value for an IP packet sourced by the system. This value applies to normal transport protocols, not to ICMP.

**udp.checksum**

Returns 1 when UDP checksums are being computed and checked. Disabling UDP checksums is strongly discouraged.

For variables `net.inet.*.ipsec`, please refer to `ipsec(4)`.

**PF\_INET6**

Get or set various global information about the IPv6 (Internet Protocol version 6). The third level name is the protocol. The fourth level name is the variable name.

For variables `net.inet6.*` please refer to `inet6(4)`. For variables `net.inet6.*.ipsec6`, please refer to `ipsec(4)`.

**CTL\_USER**

The string and integer information available for the CTL\_USER level is detailed below. The changeable column shows whether a process with appropriate privilege may change the value.

<b>Second Level Name</b>	<b>Type</b>	<b>Changeable</b>
USER_BC_BASE_MAX	integer	no
USER_BC_DIM_MAX	integer	no
USER_BC_SCALE_MAX	integer	no
USER_BC_STRING_MAX	integer	no
USER_COLL_WEIGHTS_MAX	integer	no
USER_CS_PATH	string	no
USER_EXPR_NEST_MAX	integer	no
USER_LINE_MAX	integer	no
USER_LOCALBASE	string	no
USER_POSIX2_CHAR_TERM	integer	no
USER_POSIX2_C_BIND	integer	no
USER_POSIX2_C_DEV	integer	no
USER_POSIX2_FORT_DEV	integer	no
USER_POSIX2_FORT_RUN	integer	no
USER_POSIX2_LOCALEDEF	integer	no

USER_POSIX2_SW_DEV	integer	no
USER_POSIX2_UPE	integer	no
USER_POSIX2_VERSION	integer	no
USER_RE_DUP_MAX	integer	no
USER_STREAM_MAX	integer	no
USER_TZNAME_MAX	integer	no

#### USER\_BC\_BASE\_MAX

The maximum ibase/obase values in the bc(1) utility.

#### USER\_BC\_DIM\_MAX

The maximum array size in the bc(1) utility.

#### USER\_BC\_SCALE\_MAX

The maximum scale value in the bc(1) utility.

#### USER\_BC\_STRING\_MAX

The maximum string length in the bc(1) utility.

#### USER\_COLL\_WEIGHTS\_MAX

The maximum number of weights that can be assigned to any entry of the LC\_COLLATE order keyword in the locale definition file.

#### USER\_CS\_PATH

Return a value for the PATH environment variable that finds all the standard utilities.

#### USER\_EXPR\_NEST\_MAX

The maximum number of expressions that can be nested within parenthesis by the expr(1) utility.

#### USER\_LINE\_MAX

The maximum length in bytes of a text-processing utility's input line.

#### USER\_LOCALBASE

Return the value of localbase that has been compiled into system utilities that need to have access to resources provided by a port or package.

#### USER\_POSIX2\_CHAR\_TERM

Return 1 if the system supports at least one terminal type capable of all operations described in IEEE Std 1003.2 ("POSIX.2"), otherwise 0.

**USER\_POSIX2\_C\_BIND**

Return 1 if the system's C-language development facilities support the C-Language Bindings Option, otherwise 0.

**USER\_POSIX2\_C\_DEV**

Return 1 if the system supports the C-Language Development Utilities Option, otherwise 0.

**USER\_POSIX2\_FORT\_DEV**

Return 1 if the system supports the FORTRAN Development Utilities Option, otherwise 0.

**USER\_POSIX2\_FORT\_RUN**

Return 1 if the system supports the FORTRAN Runtime Utilities Option, otherwise 0.

**USER\_POSIX2\_LOCALEDEF**

Return 1 if the system supports the creation of locales, otherwise 0.

**USER\_POSIX2\_SW\_DEV**

Return 1 if the system supports the Software Development Utilities Option, otherwise 0.

**USER\_POSIX2\_UPE**

Return 1 if the system supports the User Portability Utilities Option, otherwise 0.

**USER\_POSIX2\_VERSION**

The version of IEEE Std 1003.2 ("POSIX.2") with which the system attempts to comply.

**USER\_RE\_DUP\_MAX**

The maximum number of repeated occurrences of a regular expression permitted when using interval notation.

**USER\_STREAM\_MAX**

The minimum maximum number of streams that a process may have open at any one time.

**USER\_TZNAME\_MAX**

The minimum maximum number of types supported for the name of a timezone.

**CTL\_VM**

The string and integer information available for the CTL\_VM level is detailed below. The changeable column shows whether a process with appropriate privilege may change the value.

<b>Second Level Name</b>	<b>Type</b>	<b>Changeable</b>
--------------------------	-------------	-------------------

VM_LOADAVG	struct loadavg	no
VM_TOTAL	struct vmtotal	no
VM_SWAPPING_ENABLED	integer	maybe
VM_V_FREE_MIN	integer	yes
VM_V_FREE_RESERVED	integer	yes
VM_V_FREE_TARGET	integer	yes
VM_V_INACTIVE_TARGET	integer	yes
VM_V_PAGEOUT_FREE_MIN	integer	yes
VM_OVERCOMMIT	integer	yes

### VM\_LOADAVG

Return the load average history. The returned data consists of a *struct loadavg*.

### VM\_TOTAL

Return the system wide virtual memory statistics. The returned data consists of a *struct vmtotal*.

### VM\_SWAPPING\_ENABLED

1 if process swapping is enabled or 0 if disabled. This variable is permanently set to 0 if the kernel was built with swapping disabled.

### VM\_V\_FREE\_MIN

Minimum amount of memory (cache memory plus free memory) required to be available before a process waiting on memory will be awakened.

### VM\_V\_FREE\_RESERVED

Processes will awaken the pageout daemon and wait for memory if the number of free and cached pages drops below this value.

### VM\_V\_FREE\_TARGET

The total amount of free memory (including cache memory) that the pageout daemon tries to maintain.

### VM\_V\_INACTIVE\_TARGET

The desired number of inactive pages that the pageout daemon should achieve when it runs. Inactive pages can be quickly inserted into process address space when needed.

### VM\_V\_PAGEOUT\_FREE\_MIN

If the amount of free and cache memory falls below this value, the pageout daemon will enter

"memory conserving mode" to avoid deadlock.

## VM\_OVERCOMMIT

Overcommit behaviour, as described in `tuning(7)`.

## RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable `errno` is set to indicate the error.

## FILES

<code>&lt;sys/sysctl.h&gt;</code>	definitions for top level identifiers, second level kernel and hardware identifiers, and user level identifiers
<code>&lt;sys/socket.h&gt;</code>	definitions for second level network identifiers
<code>&lt;sys/gmon.h&gt;</code>	definitions for third level profiling identifiers
<code>&lt;vm/vm_param.h&gt;</code>	definitions for second level virtual memory identifiers
<code>&lt;netinet/in.h&gt;</code>	definitions for third level IPv4/IPv6 identifiers and fourth level IPv4/v6 identifiers
<code>&lt;netinet/icmp_var.h&gt;</code>	definitions for fourth level ICMP identifiers
<code>&lt;netinet/icmp6.h&gt;</code>	definitions for fourth level ICMPv6 identifiers
<code>&lt;netinet/udp_var.h&gt;</code>	definitions for fourth level UDP identifiers

## ERRORS

The following errors may be reported:

[EFAULT]	The buffer <i>name</i> , <i>oldp</i> , <i>newp</i> , or length pointer <i>oldlenp</i> contains an invalid address.
[EINVAL]	The <i>name</i> array is less than two or greater than CTL_MAXNAME.
[EINVAL]	A non-null <i>newp</i> is given and its specified length in <i>newlen</i> is too large or too small.
[ENOMEM]	The length pointed to by <i>oldlenp</i> is too short to hold the requested value.
[ENOMEM]	The smaller of either the length pointed to by <i>oldlenp</i> or the estimated size of the returned data exceeds the system limit on locked memory.
[ENOMEM]	Locking the buffer <i>oldp</i> , or a portion of the buffer if the estimated size of the data to be returned is smaller, would cause the process to exceed its per-process locked memory limit.

- [ENOTDIR]        The *name* array specifies an intermediate rather than terminal name.
- [EISDIR]        The *name* array specifies a terminal name, but the actual name is not terminal.
- [ENOENT]        The *name* array specifies a value that is unknown.
- [EPERM]        An attempt is made to set a read-only value.
- [EPERM]        A process without appropriate privilege attempts to set a value.

**SEE ALSO**

confstr(3), kvm(3), sysconf(3), sysctl(8)

**HISTORY**

The `sysctl()` function first appeared in 4.4BSD.