NAME

sysctl − read/write system parameters

SYNOPSIS

#include <unistd.h>
#include <linux/sysctl.h>

int _sysctl(struct __sysctl_args *_args_);

DESCRIPTION

Do not use this system call! See NOTES.

The **_sysctl**() call reads and/or writes kernel parameters. For example, the hostname, or the maximum number of open files. The argument has the form

```
struct __sysctl_args {
    int    *name;    /* integer vector describing variable */
    int     nlen;    /* length of this vector */
    void   *oldval;  /* 0 or address where to store old value */
    size_t *oldlenp; /* available room for old value,
                        overwritten by actual size of old value */
    void   *newval;  /* 0 or address of new value */
    size_t  newlen;  /* size of new value */
};
```

This call does a search in a tree structure, possibly resembling a directory tree under */proc/sys*, and if the requested item is found calls some appropriate routine to read or modify the value.

RETURN VALUE

Upon successful completion, **_sysctl**() returns 0. Otherwise, a value of −1 is returned and *errno* is set to indicate the error.

ERRORS

EFAULT
The invocation asked for the previous value by setting *oldval* non-NULL, but allowed zero room in *oldlenp*.

ENOTDIR
name was not found.

EPERM
No search permission for one of the encountered "directories", or no read permission where *oldval* was non-zero, or no write permission where *newval* was non-zero.

CONFORMING TO

This call is Linux-specific, and should not be used in programs intended to be portable. A **sysctl**() call has been present in Linux since version 1.3.57. It originated in 4.4BSD. Only Linux has the */proc/sys* mirror, and the object naming schemes differ between Linux and 4.4BSD, but the declaration of the **sysctl**() function is the same in both.

NOTES

Glibc does not provide a wrapper for this system call; call it using **syscall**(2).

Or rather... don't call it: use of this system call has long been discouraged, and it is so unloved that **it is likely to disappear in a future kernel version**. Remove it from your programs now; use the */proc/sys* interface instead.

BUGS

The object names vary between kernel versions, making this system call worthless for applications.

Not all available objects are properly documented.

It is not yet possible to change operating system by writing to */proc/sys/kernel/ostype*.

EXAMPLE

```
#define _GNU_SOURCE
#include <unistd.h>
#include <sys/syscall.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <linux/sysctl.h>

int _sysctl(struct __sysctl_args *args );

#define OSNAMESZ 100

int
main(void)
{
   struct __sysctl_args args;
   char osname[OSNAMESZ];
   size_t osnamelth;
   int name[] = { CTL_KERN, KERN_OSTYPE };

   memset(&args, 0, sizeof(struct __sysctl_args));
   args.name = name;
   args.nlen = sizeof(name)/sizeof(name[0]);
   args.oldval = osname;
   args.oldlenp = &osnamelth;

   osnamelth = sizeof(osname);

   if (syscall(SYS__sysctl, &args) == -1) {
      perror("_sysctl");
      exit(EXIT_FAILURE);
   }
   printf("This machine is running %*s\n", osnamelth, osname);
   exit(EXIT_SUCCESS);
}
```

SEE ALSO

proc(5)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.