NAME

syslog, klogctl − read and/or clear kernel message ring buffer; set console_loglevel

SYNOPSIS

int syslog(int *type***, char ****bufp***, int** *len***);**
/* No wrapper provided in glibc */

/* The glibc interface */
#include <sys/klog.h>

int klogctl(int *type***, char ****bufp***, int** *len***);**

DESCRIPTION

If you need the C library function **syslog**() (which talks to **syslogd**(8)), then look at **syslog**(3). The system call of this name is about controlling the kernel *printk*() buffer, and the glibc version is called **klogctl**().

The *type* argument determines the action taken by this function.

Quoting from *kernel/printk.c*:

```
/*
 * Commands to sys_syslog:
 *
 *      0 -- Close the log.  Currently a NOP.
 *      1 -- Open the log. Currently a NOP.
 *      2 -- Read from the log.
 *      3 -- Read all messages remaining in the ring buffer.
 *      4 -- Read and clear all messages remaining in the ring buffer
 *      5 -- Clear ring buffer.
 *      6 -- Disable printk to console
 *      7 -- Enable printk to console
 *      8 -- Set level of messages printed to console
 *      9 -- Return number of unread characters in the log buffer
 *     10 -- Return size of the log buffer
 */
```

Only command types 3 and 10 are allowed to unprivileged processes. Type 9 was added in 2.4.10; type 10 in 2.6.6.

The kernel log buffer

The kernel has a cyclic buffer of length **LOG_BUF_LEN** in which messages given as arguments to the kernel function **printk**() are stored (regardless of their loglevel). In early kernels, **LOG_BUF_LEN** had the value 4096; from kernel 1.3.54, it was 8192; from kernel 2.1.113 it was 16384; since 2.4.23/2.6 the value is a kernel configuration option. In recent kernels the size can be queried with command type 10.

The call *syslog(2,buf,len)* waits until this kernel log buffer is non-empty, and then reads at most *len* bytes into the buffer *buf*. It returns the number of bytes read. Bytes read from the log disappear from the log buffer: the information can only be read once. This is the function executed by the kernel when a user program reads */proc/kmsg*.

The call *syslog(3,buf,len)* will read the last *len* bytes from the log buffer (non-destructively), but will not read more than was written into the buffer since the last "clear ring buffer" command (which does not clear the buffer at all). It returns the number of bytes read.

The call *syslog(4,buf,len)* does precisely the same, but also executes the "clear ring buffer" command.

The call *syslog(5,dummy,dummy)* executes just the "clear ring buffer" command. (In each call where *buf* or

len is shown as "dummy", the value of the argument is ignored by the call.)

The call *syslog(6,dummy,dummy)* sets the console log level to minimum, so that no messages are printed to the console.

The call *syslog(7,dummy,dummy)* sets the console log level to default, so that messages are printed to the console.

The call *syslog(8,dummy,level)* sets the console log level to *level*, which must be an integer between 1 and 8 (inclusive). See the **loglevel** section for details.

The call *syslog(9,dummy,dummy)* returns the number of bytes currently available to be read on the kernel log buffer.

The call *syslog(10,dummy,dummy)* returns the total size of the kernel log buffer.

The loglevel

The kernel routine **printk**() will only print a message on the console, if it has a loglevel less than the value of the variable *console_loglevel*. This variable initially has the value **DEFAULT_CONSOLE_LOGLEVEL** (7), but is set to 10 if the kernel command line contains the word "debug", and to 15 in case of a kernel fault (the 10 and 15 are just silly, and equivalent to 8). This variable is set (to a value in the range 1-8) by the call *syslog(8,dummy,value)*. The calls *syslog(type,dummy,dummy)* with *type* equal to 6 or 7, set it to 1 (kernel panics only) or 7 (all except debugging messages), respectively.

Every text line in a message has its own loglevel. This level is *DEFAULT_MESSAGE_LOGLEVEL − 1* (6) unless the line starts with <d> where *d* is a digit in the range 1-7, in which case the level is *d*. The conventional meaning of the loglevel is defined in <*linux/kernel.h*> as follows:

```
#define KERN_EMERG    "<0>"  /* system is unusable               */
#define KERN_ALERT    "<1>"  /* action must be taken immediately */
#define KERN_CRIT     "<2>"  /* critical conditions              */
#define KERN_ERR      "<3>"  /* error conditions                 */
#define KERN_WARNING  "<4>"  /* warning conditions               */
#define KERN_NOTICE   "<5>"  /* normal but significant condition */
#define KERN_INFO     "<6>"  /* informational                    */
#define KERN_DEBUG    "<7>"  /* debug-level messages             */
```

RETURN VALUE

For *type* equal to 2, 3, or 4, a successful call to **syslog**() returns the number of bytes read. For *type* 9, **syslog**() returns the number of bytes currently available to be read on the kernel log buffer. For *type* 10, **syslog**() returns the total size of the kernel log buffer. For other values of *type*, 0 is returned on success.

In case of error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EINVAL

Bad arguments (e.g., bad *type*; or for *type* 2, 3, or 4, *buf* is NULL, or *len* is less than zero; or for *type* 8, the *level* is outside the range 1 to 8).

EPERM

An attempt was made to change console_loglevel or clear the kernel message ring buffer by a process without sufficient privilege (more precisely: without the **CAP_SYS_ADMIN** capability).

ERESTARTSYS

System call was interrupted by a signal; nothing was read. (This can be seen only during a trace.)

ENOSYS
This **syslog**() system call is not available, because the kernel was compiled with the **CONFIG_PRINTK** kernel-configuration option disabled.

CONFORMING TO
This system call is Linux-specific and should not be used in programs intended to be portable.

NOTES
From the very start people noted that it is unfortunate that a system call and a library routine of the same name are entirely different animals. In libc4 and libc5 the number of this call was defined by **SYS_klog**. In glibc 2.0 the syscall is baptized **klogctl**().

SEE ALSO
syslog(3)

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/.