NAME
swapon, swapoff – start/stop swapping to file/device

LIBRARY
Standard C library (*libc*, *−lc*)

SYNOPSIS
#include <sys/swap.h>

int swapon(const char **path***, int** *swapflags***);**
int swapoff(const char **path***);**

DESCRIPTION
swapon() sets the swap area to the file or block device specified by *path*. **swapoff**() stops swapping to the
file or block device specified by *path*.

If the **SWAP_FLAG_PREFER** flag is specified in the **swapon**() *swapflags* argument, the new swap area
will have a higher priority than default. The priority is encoded within *swapflags* as:

    (prio << SWAP_FLAG_PRIO_SHIFT) & SWAP_FLAG_PRIO_MASK

If the **SWAP_FLAG_DISCARD** flag is specified in the **swapon**() *swapflags* argument, freed swap pages
will be discarded before they are reused, if the swap device supports the discard or trim operation. (This
may improve performance on some Solid State Devices, but often it does not.) See also NOTES.

These functions may be used only by a privileged process (one having the **CAP_SYS_ADMIN** capability).

Priority
Each swap area has a priority, either high or low. The default priority is low. Within the low-priority areas,
newer areas are even lower priority than older areas.

All priorities set with *swapflags* are high-priority, higher than default. They may have any nonnegative
value chosen by the caller. Higher numbers mean higher priority.

Swap pages are allocated from areas in priority order, highest priority first. For areas with different priori-
ties, a higher-priority area is exhausted before using a lower-priority area. If two or more areas have the
same priority, and it is the highest priority available, pages are allocated on a round-robin basis between
them.

As of Linux 1.3.6, the kernel usually follows these rules, but there are exceptions.

RETURN VALUE
On success, zero is returned. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS
EBUSY
    (for **swapon**()) The specified *path* is already being used as a swap area.

EINVAL
    The file *path* exists, but refers neither to a regular file nor to a block device;

EINVAL
    (**swapon**()) The indicated path does not contain a valid swap signature or resides on an in-memory
    filesystem such as **tmpfs**(5).

EINVAL (since Linux 3.4)
    (**swapon**()) An invalid flag value was specified in *swapflags*.

EINVAL
    (**swapoff**()) *path* is not currently a swap area.

ENFILE
    The system-wide limit on the total number of open files has been reached.

ENOENT
    The file *path* does not exist.

ENOMEM
    The system has insufficient memory to start swapping.

EPERM
    The caller does not have the **CAP_SYS_ADMIN** capability. Alternatively, the maximum number of swap files are already in use; see NOTES below.

STANDARDS

These functions are Linux-specific and should not be used in programs intended to be portable. The second *swapflags* argument was introduced in Linux 1.3.2.

NOTES

The partition or path must be prepared with **mkswap**(8).

There is an upper limit on the number of swap files that may be used, defined by the kernel constant **MAX_SWAPFILES**. Before Linux 2.4.10, **MAX_SWAPFILES** has the value 8; since Linux 2.4.10, it has the value 32. Since Linux 2.6.18, the limit is decreased by 2 (thus: 30) if the kernel is built with the **CONFIG_MIGRATION** option (which reserves two swap table entries for the page migration features of **mbind**(2) and **migrate_pages**(2)). Since Linux 2.6.32, the limit is further decreased by 1 if the kernel is built with the **CONFIG_MEMORY_FAILURE** option. Since Linux 5.14, the limit is further decreased by 4 if the kernel is built with the **CONFIG_DEVICE_PRIVATE** option.

Discard of swap pages was introduced in Linux 2.6.29, then made conditional on the **SWAP_FLAG_DIS-CARD** flag in Linux 2.6.36, which still discards the entire swap area when **swapon**() is called, even if that flag bit is not set.

SEE ALSO

mkswap(8), **swapoff**(8), **swapon**(8)