NAME
>   alloc_hugepages, free_hugepages − allocate or free huge pages

SYNOPSIS
>   **void *alloc_hugepages(int** *key***, void ****addr***, size_t** *len***,**
>   **int** *prot***, int** *flag***);**
>
>   **int free_hugepages(void ****addr***);**

DESCRIPTION
>   The system calls **alloc_hugepages**() and **free_hugepages**() were introduced in Linux 2.5.36 and removed
>   again in 2.5.54. They existed only on i386 and ia64 (when built with **CONFIG_HUGETLB_PAGE**). In
>   Linux 2.4.20, the syscall numbers exist, but the calls fail with the error **ENOSYS**.
>
>   On i386 the memory management hardware knows about ordinary pages (4 KiB) and huge pages (2 or
>   4 MiB). Similarly ia64 knows about huge pages of several sizes. These system calls serve to map huge
>   pages into the process's memory or to free them again. Huge pages are locked into memory, and are not
>   swapped.
>
>   The *key* argument is an identifier. When zero the pages are private, and not inherited by children. When
>   positive the pages are shared with other applications using the same *key*, and inherited by child processes.
>
>   The *addr* argument of **free_hugepages**() tells which page is being freed: it was the return value of a call to
>   **alloc_hugepages**(). (The memory is first actually freed when all users have released it.) The *addr* argu-
>   ment of **alloc_hugepages**() is a hint, that the kernel may or may not follow. Addresses must be properly
>   aligned.
>
>   The *len* argument is the length of the required segment. It must be a multiple of the huge page size.
>
>   The *prot* argument specifies the memory protection of the segment. It is one of **PROT_READ**,
>   **PROT_WRITE**, **PROT_EXEC**.
>
>   The *flag* argument is ignored, unless *key* is positive. In that case, if *flag* is **IPC_CREAT**, then a new huge
>   page segment is created when none with the given key existed. If this flag is not set, then **ENOENT** is re-
>   turned when no segment with the given key exists.

RETURN VALUE
>   On success, **alloc_hugepages**() returns the allocated virtual address, and **free_hugepages**() returns zero.
>   On error, −1 is returned, and *errno* is set appropriately.

ERRORS
>   **ENOSYS**
>   The system call is not supported on this kernel.

FILES
>   */proc/sys/vm/nr_hugepages*
>   Number of configured hugetlb pages. This can be read and written.
>
>   */proc/meminfo*
>   Gives info on the number of configured hugetlb pages and on their size in the three variables
>   HugePages_Total, HugePages_Free, Hugepagesize.

CONFORMING TO
>   These calls are specific to Linux on Intel processors, and should not be used in programs intended to be
>   portable.

NOTES
>   These system calls are gone; they existed only in Linux 2.5.36 through to 2.5.54. Now the hugetlbfs
>   filesystem can be used instead. Memory backed by huge pages (if the CPU supports them) is obtained by
>   using **mmap**(2) to map files in this virtual filesystem.
>
>   The maximal number of huge pages can be specified using the **hugepages=** boot parameter.

COLOPHON

This page is part of release 5.05 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.