NAME

remap_file_pages − create a nonlinear file mapping

SYNOPSIS

#define _GNU_SOURCE /* See feature_test_macros(7) */
#include <sys/mman.h>

int remap_file_pages(void ****addr*, size_t** *size***, int** *prot***,**
size_t *pgoff***, int** *flags***);**

DESCRIPTION

Note: this system call was marked as deprecated starting with Linux 3.16. In Linux 4.0, the implementation was replaced by a slower in-kernel emulation. Those few applications that use this system call should consider migrating to alternatives. This change was made because the kernel code for this system call was complex, and it is believed to be little used or perhaps even completely unused. While it had some use cases in database applications on 32-bit systems, those use cases don't exist on 64-bit systems.

The **remap_file_pages**() system call is used to create a nonlinear mapping, that is, a mapping in which the pages of the file are mapped into a nonsequential order in memory. The advantage of using **remap_file_pages**() over using repeated calls to **mmap**(2) is that the former approach does not require the kernel to create additional VMA (Virtual Memory Area) data structures.

To create a nonlinear mapping we perform the following steps:

1. Use **mmap**(2) to create a mapping (which is initially linear). This mapping must be created with the **MAP_SHARED** flag.
2. Use one or more calls to **remap_file_pages**() to rearrange the correspondence between the pages of the mapping and the pages of the file. It is possible to map the same page of a file into multiple locations within the mapped region.

The *pgoff* and *size* arguments specify the region of the file that is to be relocated within the mapping: *pgoff* is a file offset in units of the system page size; *size* is the length of the region in bytes.

The *addr* argument serves two purposes. First, it identifies the mapping whose pages we want to rearrange. Thus, *addr* must be an address that falls within a region previously mapped by a call to **mmap**(2). Second, *addr* specifies the address at which the file pages identified by *pgoff* and *size* will be placed.

The values specified in *addr* and *size* should be multiples of the system page size. If they are not, then the kernel rounds *both* values *down* to the nearest multiple of the page size.

The *prot* argument must be specified as 0.

The *flags* argument has the same meaning as for **mmap**(2), but all flags other than **MAP_NONBLOCK** are ignored.

RETURN VALUE

On success, **remap_file_pages**() returns 0. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EINVAL
addr does not refer to a valid mapping created with the **MAP_SHARED** flag.

EINVAL
addr, *size*, *prot*, or *pgoff* is invalid.

VERSIONS

The **remap_file_pages**() system call appeared in Linux 2.5.46; glibc support was added in version 2.3.3.

CONFORMING TO

The **remap_file_pages**() system call is Linux-specific.

NOTES

Since Linux 2.6.23, **remap_file_pages**() creates non-linear mappings only on in-memory filesystems such as **tmpfs**(5), hugetlbfs or ramfs. On filesystems with a backing store, **remap_file_pages**() is not much

more efficient than using **mmap**(2) to adjust which parts of the file are mapped to which addresses.

SEE ALSO

getpagesize(2), **mmap**(2), **mmap2**(2), **mprotect**(2), **mremap**(2), **msync**(2)

COLOPHON

This page is part of release 5.02 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/.