Rocky Enterprise Linux 9.2 Manual Pages on command 'pkey_mprotect.2'

$ man pkey_mprotect.2

MPROTECT(2)   Linux Programmer's Manual   MPROTECT(2)

NAME

   mprotect, pkey_mprotect - set protection on a region of memory

SYNOPSIS

   #include <sys/mman.h>

   int mprotect(void *addr, size_t len, int prot);

   #define _GNU_SOURCE   /* See feature_test_macros(7) */

   #include <sys/mman.h>

   int pkey_mprotect(void *addr, size_t len, int prot, int pkey);

DESCRIPTION

   mprotect() changes the access protections for the calling process's memory pages contain?

   ing any part of the address range in the interval [addr, addr+len-1]. addr must be

   aligned to a page boundary.

   If the calling process tries to access memory in a manner that violates the protections,

   then the kernel generates a SIGSEGV signal for the process.

   prot is a combination of the following access flags: PROT_NONE or a bitwise-or of the

   other values in the following list:

   PROT_NONE

       The memory cannot be accessed at all.

   PROT_READ

       The memory can be read.

   PROT_WRITE

       The memory can be modified.

PROT_EXEC

    The memory can be executed.

PROT_SEM (since Linux 2.5.7)

    The memory can be used for atomic operations. This flag was introduced as part of

    the futex(2) implementation (in order to guarantee the ability to perform atomic

    operations required by commands such as FUTEX_WAIT), but is not currently used in

    on any architecture.

PROT_SAO (since Linux 2.6.26)

    The memory should have strong access ordering. This feature is specific to the

    PowerPC architecture (version 2.06 of the architecture specification adds the SAO

    CPU feature, and it is available on POWER 7 or PowerPC A2, for example).

Additionally (since Linux 2.6.0), prot can have one of the following flags set:

PROT_GROWSUP

    Apply the protection mode up to the end of a mapping that grows upwards.   (Such

    mappings are created for the stack area on architectures?for example, HP-PARISC?

    that have an upwardly growing stack.)

PROT_GROWSDOWN

    Apply the protection mode down to the beginning of a mapping that grows downward

    (which should be a stack segment or a segment mapped with the MAP_GROWSDOWN flag

    set).

Like mprotect(), pkey_mprotect() changes the protection on the pages specified by addr and

len.   The pkey argument specifies the protection key (see pkeys(7)) to assign to the mem?

ory. The protection key must be allocated with pkey_alloc(2) before it is passed to

pkey_mprotect(). For an example of the use of this system call, see pkeys(7).

RETURN VALUE

    On success, mprotect() and pkey_mprotect() return zero. On error, these system calls re?

    turn -1, and errno is set appropriately.

ERRORS

    EACCES The memory cannot be given the specified access. This can happen, for example, if

        you mmap(2) a file to which you have read-only access, then ask mprotect() to mark

        it PROT_WRITE.

    EINVAL addr is not a valid pointer, or not a multiple of the system page size.

    EINVAL (pkey_mprotect()) pkey has not been allocated with pkey_alloc(2)

EINVAL Both PROT_GROWSUP and PROT_GROWSDOWN were specified in prot.

EINVAL Invalid flags specified in prot.

EINVAL (PowerPC architecture) PROT_SAO was specified in prot, but SAO hardware feature is

    not available.

ENOMEM Internal kernel structures could not be allocated.

ENOMEM Addresses in the range [addr, addr+len-1] are invalid for the address space of the

    process, or specify one or more pages that are not mapped. (Before kernel 2.4.19,

    the error EFAULT was incorrectly produced for these cases.)

ENOMEM Changing the protection of a memory region would result in the total number of map?

    pings with distinct attributes (e.g., read versus read/write protection) exceeding

    the allowed maximum. (For example, making the protection of a range PROT_READ in

    the middle of a region currently protected as PROT_READ|PROT_WRITE would result in

    three mappings: two read/write mappings at each end and a read-only mapping in the

    middle.)

VERSIONS

    pkey_mprotect() first appeared in Linux 4.9; library support was added in glibc 2.27.

CONFORMING TO

    mprotect(): POSIX.1-2001, POSIX.1-2008, SVr4. POSIX says that the behavior of mprotect()

    is unspecified if it is applied to a region of memory that was not obtained via mmap(2).

    pkey_mprotect() is a nonportable Linux extension.

NOTES

    On Linux, it is always permissible to call mprotect() on any address in a process's ad?

    dress space (except for the kernel vsyscall area). In particular, it can be used to

    change existing code mappings to be writable.

    Whether PROT_EXEC has any effect different from PROT_READ depends on processor architec?

    ture, kernel version, and process state. If READ_IMPLIES_EXEC is set in the process's

    personality flags (see personality(2)), specifying PROT_READ will implicitly add

    PROT_EXEC.

    On some hardware architectures (e.g., i386), PROT_WRITE implies PROT_READ.

    POSIX.1 says that an implementation may permit access other than that specified in prot,

    but at a minimum can allow write access only if PROT_WRITE has been set, and must not al?

    low any access if PROT_NONE has been set.

    Applications should be careful when mixing use of mprotect() and pkey_mprotect(). On x86,

when mprotect() is used with prot set to PROT_EXEC a pkey may be allocated and set on the memory implicitly by the kernel, but only when the pkey was 0 previously.

On systems that do not support protection keys in hardware, pkey_mprotect() may still be used, but pkey must be set to -1. When called this way, the operation of pkey_mprotect() is equivalent to mprotect().

EXAMPLES

The program below demonstrates the use of mprotect(). The program allocates four pages of memory, makes the third of these pages read-only, and then executes a loop that walks up‐ward through the allocated region modifying bytes.

An example of what we might see when running the program is the following:

```
$ ./a.out
Start of region:        0x804c000
Got SIGSEGV at address: 0x804e000
```

Program source

```
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/mman.h>
#define handle_error(msg) \
    do { perror(msg); exit(EXIT_FAILURE); } while (0)
static char *buffer;
static void
handler(int sig, siginfo_t *si, void *unused)
{
    /* Note: calling printf() from a signal handler is not safe
       (and should not be done in production programs), since
       printf() is not async-signal-safe; see signal-safety(7).
       Nevertheless, we use printf() here as a simple way of
       showing that the handler was called. */
    printf("Got SIGSEGV at address: %p\n", si->si_addr);
```

```c
        exit(EXIT_FAILURE);
    }

    int
    main(int argc, char *argv[])
    {
        int pagesize;
        struct sigaction sa;

        sa.sa_flags = SA_SIGINFO;
        sigemptyset(&sa.sa_mask);
        sa.sa_sigaction = handler;
        if (sigaction(SIGSEGV, &sa, NULL) == -1)
            handle_error("sigaction");

        pagesize = sysconf(_SC_PAGE_SIZE);
        if (pagesize == -1)
            handle_error("sysconf");

        /* Allocate a buffer aligned on a page boundary;
           initial protection is PROT_READ | PROT_WRITE */

        buffer = memalign(pagesize, 4 * pagesize);
        if (buffer == NULL)
            handle_error("memalign");

        printf("Start of region:        %p\n", buffer);

        if (mprotect(buffer + pagesize * 2, pagesize,
                    PROT_READ) == -1)
            handle_error("mprotect");

        for (char *p = buffer ; ; )
            *(p++) = 'a';

        printf("Loop completed\n");     /* Should never happen */
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

mmap(2), sysconf(3), pkeys(7)

COLOPHON

This page is part of release 5.10 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/.