



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

\$ man get_mempolicy.2

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

NAME

get_mempolicy - retrieve NUMA memory policy for a thread

SYNOPSIS

```
#include <numaif.h>
```

```
long get_mempolicy(int *mode, unsigned long *nodemask,  
                  unsigned long maxnode, void *addr,  
                  unsigned long flags);
```

Link with -lnuma.

DESCRIPTION

get_mempolicy() retrieves the NUMA policy of the calling thread or of a memory address, depending on the setting of flags.

A NUMA machine has different memory controllers with different distances to specific CPUs. The memory policy defines from which node memory is allocated for the thread.

If flags is specified as 0, then information about the calling thread's default policy (as set by set_mempolicy(2)) is returned, in the buffers pointed to by mode and nodemask. The value returned in these arguments

may be used to restore the thread's policy to its state at the time of the call to `get_mempolicy()` using `set_mempolicy(2)`. When `flags` is 0, `addr` must be specified as `NULL`.

If `flags` specifies `MPOL_F_MEMS_ALLOWED` (available since Linux 2.6.24), the `mode` argument is ignored and the set of nodes (memories) that the thread is allowed to specify in subsequent calls to `mbind(2)` or `set_mempolicy(2)` (in the absence of any `mode` flags) is returned in `nodemask`. It is not permitted to combine `MPOL_F_MEMS_ALLOWED` with either `MPOL_F_ADDR` or `MPOL_F_NODE`.

If `flags` specifies `MPOL_F_ADDR`, then information is returned about the policy governing the memory address given in `addr`. This policy may be different from the thread's default policy if `mbind(2)` or one of the helper functions described in `numa(3)` has been used to establish a policy for the memory range containing `addr`.

If the `mode` argument is not `NULL`, then `get_mempolicy()` will store the policy mode and any optional mode flags of the requested NUMA policy in the location pointed to by this argument. If `nodemask` is not `NULL`, then the `nodemask` associated with the policy will be stored in the location pointed to by this argument. `maxnode` specifies the number of node IDs that can be stored into `nodemask`; that is, the maximum node ID plus one. The value specified by `maxnode` is always rounded to a multiple of `sizeof(unsigned long)*8`.

If `flags` specifies both `MPOL_F_NODE` and `MPOL_F_ADDR`, `get_mempolicy()` will return the node ID of the node on which the address `addr` is allocated into the location pointed to by `mode`. If no page has yet been allocated for the specified address, `get_mempolicy()` will allocate a page as if the thread had performed a read (load) access to that address, and return the ID of the node where that page was allocated.

If `flags` specifies `MPOL_F_NODE`, but not `MPOL_F_ADDR`, and the thread's current policy is `MPOL_INTERLEAVE`, then `get_mempolicy()` will return in the location pointed to by a non-`NULL` `mode` argument, the node ID of the next node that will be used for interleaving of internal kernel pages allocated on behalf of the thread. These allocations include pages for

memory-mapped files in process memory ranges mapped using the `mmap(2)` call with the `MAP_PRIVATE` flag for read accesses, and in memory ranges mapped with the `MAP_SHARED` flag for all accesses.

Other flag values are reserved.

For an overview of the possible policies see `set_mempolicy(2)`.

RETURN VALUE

On success, `get_mempolicy()` returns 0; on error, -1 is returned and `errno` is set to indicate the error.

ERRORS

EFAULT Part of all of the memory range specified by `nodemask` and `maxnode` points outside your accessible address space.

EINVAL The value specified by `maxnode` is less than the number of node IDs supported by the system. Or flags specified values other than `MPOL_F_NODE` or `MPOL_F_ADDR`; or flags specified `MPOL_F_ADDR` and `addr` is NULL, or flags did not specify `MPOL_F_ADDR` and `addr` is not NULL. Or, flags specified `MPOL_F_NODE` but not `MPOL_F_ADDR` and the current thread policy is not `MPOL_INTERLEAVE`. Or, flags specified `MPOL_F_MEMS_ALLOWED` with either `MPOL_F_ADDR` or `MPOL_F_NODE`. (And there are other **EINVAL** cases.)

VERSIONS

The `get_mempolicy()` system call was added to the Linux kernel in version 2.6.7.

CONFORMING TO

This system call is Linux-specific.

NOTES

For information on library support, see `numa(7)`.

SEE ALSO

`getcpu(2)`, `mbind(2)`, `mmap(2)`, `set_mempolicy(2)`, `numa(3)`, `numa(7)`, `numactl(8)`

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

