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

$ man kexec_load.2

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

NAME

kexec_load, kexec_file_load - load a new kernel for later execution

SYNOPSIS

```
#include <linux/kexec.h>
long kexec_load(unsigned long entry, unsigned long nr_segments,
                struct kexec_segment *segments, unsigned long flags);
long kexec_file_load(int kernel_fd, int initrd_fd,
                     unsigned long cmdline_len, const char *cmdline,
                     unsigned long flags);
```

Note: There are no glibc wrappers for these system calls; see NOTES.

DESCRIPTION

The kexec_load() system call loads a new kernel that can be executed later by reboot(2).

The flags argument is a bit mask that controls the operation of the call. The following values can be specified in flags:

KEXEC_ON_CRASH (since Linux 2.6.13)

Execute the new kernel automatically on a system crash. This

"crash kernel" is loaded into an area of reserved memory that is determined at boot time using the crashkernel kernel command-line parameter. The location of this reserved memory is ex? ported to user space via the /proc/iomem file, in an entry la? beled "Crash kernel". A user-space application can parse this file and prepare a list of segments (see below) that specify this reserved memory as destination. If this flag is specified, the kernel checks that the target segments specified in segments fall within the reserved region.

KEXEC_PRESERVE_CONTEXT (since Linux 2.6.27)

Preserve the system hardware and software states before execut? ing the new kernel. This could be used for system suspend. This flag is available only if the kernel was configured with CONFIG_KEXEC_JUMP, and is effective only if nr_segments is greater than 0.

The high-order bits (corresponding to the mask 0xffff0000) of flags contain the architecture of the to-be-executed kernel. Specify (OR) the constant KEXEC_ARCH_DEFAULT to use the current architecture, or one of the following architecture constants KEXEC_ARCH_386, KEXEC_ARCH_68K, KEXEC_ARCH_X86_64, KEXEC_ARCH_PPC, KEXEC_ARCH_PPC64, KEXEC_ARCH_IA_64, KEXEC_ARCH_ARM, KEXEC_ARCH_S390, KEXEC_ARCH_SH, KEXEC_ARCH_MIPS, and KEXEC_ARCH_MIPS_LE. The architecture must be executable on the CPU of the system.

The entry argument is the physical entry address in the kernel image.

The nr_segments argument is the number of segments pointed to by the segments pointer; the kernel imposes an (arbitrary) limit of 16 on the number of segments. The segments argument is an array of kexec_segment structures which define the kernel layout:

```
struct kexec_segment {
    void   *buf;        /* Buffer in user space */
    size_t  bufsz;      /* Buffer length in user space */
    void   *mem;        /* Physical address of kernel */
    size_t  memsz;      /* Physical address length */
```

};

The kernel image defined by segments is copied from the calling process into the kernel either in regular memory or in reserved memory (if KEXEC_ON_CRASH is set). The kernel first performs various sanity checks on the information passed in segments. If these checks pass, the kernel copies the segment data to kernel memory. Each segment specified in segments is copied as follows:

* buf and bufsz identify a memory region in the caller's virtual ad? dress space that is the source of the copy. The value in bufsz may not exceed the value in the memsz field.
* mem and memsz specify a physical address range that is the target of the copy. The values specified in both fields must be multiples of the system page size.
* bufsz bytes are copied from the source buffer to the target kernel buffer. If bufsz is less than memsz, then the excess bytes in the kernel buffer are zeroed out.

In case of a normal kexec (i.e., the KEXEC_ON_CRASH flag is not set), the segment data is loaded in any available memory and is moved to the final destination at kexec reboot time (e.g., when the kexec(8) command is executed with the -e option).

In case of kexec on panic (i.e., the KEXEC_ON_CRASH flag is set), the segment data is loaded to reserved memory at the time of the call, and, after a crash, the kexec mechanism simply passes control to that ker? nel.

The kexec_load() system call is available only if the kernel was con? figured with CONFIG_KEXEC.

kexec_file_load()

The kexec_file_load() system call is similar to kexec_load(), but it takes a different set of arguments. It reads the kernel to be loaded from the file referred to by the file descriptor kernel_fd, and the initrd (initial RAM disk) to be loaded from file referred to by the file descriptor initrd_fd. The cmdline argument is a pointer to a buf? fer containing the command line for the new kernel. The cmdline_len

argument specifies size of the buffer. The last byte in the buffer must be a null byte ('\0').

The flags argument is a bit mask which modifies the behavior of the call. The following values can be specified in flags:

KEXEC_FILE_UNLOAD

Unload the currently loaded kernel.

KEXEC_FILE_ON_CRASH

Load the new kernel in the memory region reserved for the crash kernel (as for KEXEC_ON_CRASH). This kernel is booted if the currently running kernel crashes.

KEXEC_FILE_NO_INITRAMFS

Loading initrd/initramfs is optional. Specify this flag if no initramfs is being loaded. If this flag is set, the value passed in initrd_fd is ignored.

The kexec_file_load() system call was added to provide support for sys? tems where "kexec" loading should be restricted to only kernels that are signed. This system call is available only if the kernel was con? figured with CONFIG_KEXEC_FILE.

RETURN VALUE

On success, these system calls returns 0. On error, -1 is returned and errno is set to indicate the error.

ERRORS

EADDRNOTAVAIL

The KEXEC_ON_CRASH flags was specified, but the region specified by the mem and memsz fields of one of the segments entries lies outside the range of memory reserved for the crash kernel.

EADDRNOTAVAIL

The value in a mem or memsz field in one of the segments entries is not a multiple of the system page size.

EBADF kernel_fd or initrd_fd is not a valid file descriptor.

EBUSY Another crash kernel is already being loaded or a crash kernel is already in use.

EINVAL flags is invalid.

EINVAL The value of a bufsz field in one of the segments entries ex?
ceeds the value in the corresponding memsz field.

EINVAL nr_segments exceeds KEXEC_SEGMENT_MAX (16).

EINVAL Two or more of the kernel target buffers overlap.

EINVAL The value in cmdline[cmdline_len-1] is not '\0'.

EINVAL The file referred to by kernel_fd or initrd_fd is empty (length
zero).

ENOEXEC
kernel_fd does not refer to an open file, or the kernel can't
load this file. Currently, the file must be a bzImage and con?
tain an x86 kernel that is loadable above 4 GiB in memory (see
the kernel source file Documentation/x86/boot.txt).

ENOMEM Could not allocate memory.

EPERM The caller does not have the CAP_SYS_BOOT capability.

VERSIONS

The kexec_load() system call first appeared in Linux 2.6.13. The
kexec_file_load() system call first appeared in Linux 3.17.

CONFORMING TO

These system calls are Linux-specific.

NOTES

Currently, there is no glibc support for these system calls. Call them
using syscall(2).

SEE ALSO

reboot(2), syscall(2), kexec(8)

The kernel source files Documentation/kdump/kdump.txt and Documenta?
tion/admin-guide/kernel-parameters.txt

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/.