NAME

pivot_root - change the root filesystem

SYNOPSIS

int pivot_root(const char *_new_root_, const char *_put_old_);

Note: There is no glibc wrapper for this system call; see NOTES.

DESCRIPTION

pivot_root() moves the root filesystem of the calling process to the directory *put_old* and makes *new_root* the new root filesystem of the calling process.

The typical use of **pivot_root**() is during system startup, when the system mounts a temporary root filesystem (e.g., an **initrd**), then mounts the real root filesystem, and eventually turns the latter into the current root of all relevant processes or threads.

pivot_root() may or may not change the current root and the current working directory of any processes or threads which use the old root directory. The caller of**pi vot_root**() must ensure that processes with root or current working directory at the old root operate correctly in either case. An easy way to ensure this is to change their root and current working directory to *new_root* before invoking **pivot_root**().

The paragraph above is intentionally vague because the implementation of **pivot_root**() may change in the future. At the time of writing,**pi vot_root**() changes root and current working directory of each process or thread to *new_root* if they point to the old root directory. This is necessary in order to prevent kernel threads from keeping the old root directory busy with their root and current working directory, even if they never access the filesystem in any way. In the future, there may be a mechanism for kernel threads to explicitly relinquish any access to the filesystem, such that this fairly intrusive mechanism can be removed from **pivot_root**().

Note that this also applies to the calling process: **pivot_root**() may or may not affect its current working directory. It is therefore recommended to call**chdir("/")** immediately after **pi vot_root**().

The following restrictions apply to *new_root* and *put_old*:

- They must be directories.
- *new_root* and *put_old* must not be on the same filesystem as the current root.
- *put_old* must be underneath *new_root*, that is, adding a nonzero number of /.. to the string pointed to by *put_old* must yield the same directory as *new_root*.
- No other filesystem may be mounted on *put_old*.

See also pivot_root(8) for additional usage examples.

If the current root is not a mount point (e.g., after chroot(2) or **pivot_root**(), see also below), not the old root directory, but the mount point of that filesystem is mounted on *put_old*.

new_root does not have to be a mount point. In this case, */proc/mounts* will show the mount point of the filesystem containing *new_root* as root (/).

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set appropriately.

ERRORS

pivot_root() may return (in *errno*) any of the errors returned by stat(2). Additionally, it may return:

EBUSY

new_root or *put_old* are on the current root filesystem, or a filesystem is already mounted on *put_old*.

EINVAL

put_old is not underneath *new_root*.

ENOTDIR
new_root or *put_old* is not a directory.

EPERM
The calling process does not have the **CAP_SYS_ADMIN** capability.

VERSIONS
pivot_root() was introduced in Linux 2.3.41.

CONFORMING TO
pivot_root() is Linux-specific and hence is not portable.

NOTES
Glibc does not provide a wrapper for this system call; call it using syscall(2).

BUGS
pivot_root() should not have to change root and current working directory of all other processes in the system.

Some of the more obscure uses of **pivot_root**() may quickly lead to insanity.

SEE ALSO
chdir(2), chroot(2), stat(2), initrd(4), pivot_root(8)

COLOPHON
This page is part of release 4.10 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/.