



Linux Ubuntu 26.04 Manual Pages on command 'execveat.2'

\$ man execveat.2

execveat(2) System Calls Manual execveat(2)

NAME

execveat - execute program relative to a directory file descriptor

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <linux/fcntl.h>     /* Definition of AT_* constants */
#include <unistd.h>

int execveat(int dirfd, const char *path,
             char *const _Nullable argv[],
             char *const _Nullable envp[],
             int flags);
```

DESCRIPTION

The `execveat()` system call executes the program referred to by the combination of `dirfd` and `path`. It operates in exactly the same way as `execve(2)`, except for the differences described in this manual page.

If `path` is relative, then it is interpreted relative to the directory referred to by the file descriptor `dirfd` (rather than relative to the current working directory of the calling process, as is done by `execve(2)` for a relative pathname).

If `path` is relative and `dirfd` is the special value `AT_FDCWD`, then `path` is interpreted relative to the current working directory of the calling process (like `execve(2)`).

If `path` is absolute, then `dirfd` is ignored.

If `path` is an empty string and the `AT_EMPTY_PATH` flag is specified, then the file descriptor

dirfd specifies the file to be executed (i.e., dirfd refers to an executable file, rather than a directory).

The flags argument is a bit mask that can include zero or more of the following flags:

AT_EMPTY_PATH

If path is an empty string, operate on the file referred to by dirfd (which may have been obtained using the open(2) O_PATH flag).

AT_SYMLINK_NOFOLLOW

If the file identified by dirfd and a non-NULL path is a symbolic link, then the call fails with the error ELOOP.

RETURN VALUE

On success, execveat() does not return. On error, -1 is returned, and errno is set to indicate the error.

ERRORS

The same errors that occur for execve(2) can also occur for execveat(). The following additional errors can occur for execveat():

path is relative but dirfd is neither AT_FDCWD nor a valid file descriptor.

EINVAL Invalid flag specified in flags.

ELOOP flags includes AT_SYMLINK_NOFOLLOW and the file identified by dirfd and a non-NULL path is a symbolic link.

ENOENT The program identified by dirfd and path requires the use of an interpreter program (such as a script starting with "#!"), but the file descriptor dirfd was opened with the O_CLOEXEC flag, with the result that the program file is inaccessible to the launched interpreter. See BUGS.

ENOTDIR

path is relative and dirfd is a file descriptor referring to a file other than a directory.

STANDARDS

Linux.

HISTORY

Linux 3.19, glibc 2.34.

NOTES

In addition to the reasons explained in openat(2), the execveat() system call is also needed to allow fexecve(3) to be implemented on systems that do not have the /proc filesystem

mounted.

When asked to execute a script file, the `argv[0]` that is passed to the script interpreter is a string of the form `/dev/fd/N` or `/dev/fd/N/P`, where `N` is the number of the file descriptor passed via the `dirfd` argument. A string of the first form occurs when `AT_EMPTY_PATH` is employed. A string of the second form occurs when the script is specified via both `dirfd` and `path`; in this case, `P` is the value given in `path`.

For the same reasons described in `fexecve(3)`, the natural idiom when using `execveat()` is to set the close-on-exec flag on `dirfd`. (But see `BUGS`.)

BUGS

The `ENOENT` error described above means that it is not possible to set the close-on-exec flag on the file descriptor given to a call of the form:

```
execveat(fd, "", argv, envp, AT_EMPTY_PATH);
```

However, the inability to set the close-on-exec flag means that a file descriptor referring to the script leaks through to the script itself. As well as wasting a file descriptor, this leakage can lead to file-descriptor exhaustion in scenarios where scripts recursively employ `execveat()`.

SEE ALSO

`execve(2)`, `openat(2)`, `fexecve(3)`