Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man fspick.2

fspick(2) System Calls Manual fspick(2)

NAME

fspick - select filesystem for reconfiguration

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <fcntl.h>          /* Definition of AT_* constants */
#include <sys/mount.h>
int fspick(int dirfd, const char *path, unsigned int flags);
```

DESCRIPTION

The fspick() system call is part of the suite of file-descriptor-based mount facilities in Linux.

fspick() creates a new filesystem configuration context for the extant filesystem instance associated with the path described by dirfd and path, places it into reconfiguration mode (similar to mount(8) with the -o remount option). A new file descriptor associated with the filesystem configuration context is then returned. The calling process must have the CAP_SYS_ADMIN capability in order to create a new filesystem configuration context.

The resultant file descriptor can be used with fsconfig(2) to specify the desired set of changes to filesystem parameters of the filesystem instance. Once the desired set of changes have been configured, the changes can be effectuated by calling fsconfig(2) with the FSCONFIG_CMD_RECONFIGURE command. In contrast to the behavior of MS_REMOUNT with mount(2), fspick() instantiates the filesystem configuration context with a copy of the extant filesystem's filesystem parameters; thus, subsequent FSCONFIG_CMD_RECONFIGURE operations

will only update filesystem parameters explicitly modified with fsconfig(2).

As with "*at()" system calls, fspick() uses the dirfd argument in conjunction with the path argument to determine the path to operate on, as follows:

? If the pathname given in path is absolute, then dirfd is ignored.

? If the pathname given in 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 open(2)).

? If the pathname given in path is relative, then it is interpreted relative to the direc? tory referred to by the file descriptor dirfd (rather than relative to the current work? ing directory of the calling process, as is done by open(2) for a relative pathname). In this case, dirfd must be a directory that was opened for reading (O_RDONLY) or using the O_PATH flag.

? If path is an empty string, and flags contains FSPICK_EMPTY_PATH, then the file descrip? tor dirfd is operated on directly. In this case, dirfd may refer to any type of file, not just a directory.

See openat(2) for an explanation of why the dirfd argument is useful.

flags can be used to control aspects of how path is resolved and properties of the returned file descriptor. A value for flags is constructed by bitwise ORing zero or more of the fol? lowing constants:

FSPICK_CLOEXEC

Set the close-on-exec (FD_CLOEXEC) flag on the new file descriptor. See the description of the O_CLOEXEC flag in open(2) for reasons why this may be use? ful.

FSPICK_EMPTY_PATH

If path is an empty string, operate on the file referred to by dirfd (which may have been obtained from open(2), fsmount(2), or open_tree(2)). In this case, dirfd may refer to any type of file, not just a directory. If dirfd is AT_FDCWD, fspick() will operate on the current working directory of the call? ing process.

FSPICK_SYMLINK_NOFOLLOW

Do not follow symbolic links in the terminal component of path. If path ref? erences a symbolic link, the returned filesystem context will reference the filesystem that the symbolic link itself resides on.

FSPICK_NO_AUTOMOUNT

Do not automount the terminal ("basename") component of path if it is a direc? tory that is an automount point. This allows you to reconfigure an automount point, rather than the location that would be mounted. This flag has no ef? fect if the automount point has already been mounted over.

As with filesystem contexts created with fsopen(2), the file descriptor returned by fspick() may be queried for message strings at any time by calling read(2) on the file descriptor. (See the "Message retrieval interface" subsection in fsopen(2) for more details on the mes? sage format.)

RETURN VALUE

On success, a new file descriptor is returned. On error, -1 is returned, and errno is set to indicate the error.

ERRORS

EACCES Search permission is denied for one of the directories in the path prefix of path. (See also path_resolution(7).)

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

EFAULT path is NULL or a pointer to a location outside the calling process's accessible ad? dress space.

EINVAL Invalid flag specified in flags.

ELOOP Too many symbolic links encountered when resolving path.

EMFILE The calling process has too many open files to create more.

ENAMETOOLONG

path is longer than PATH_MAX.

ENFILE The system has too many open files to create more.

ENOENT A component of path does not exist, or is a dangling symbolic link.

ENOENT path is an empty string, but FSPICK_EMPTY_PATH is not specified in flags.

ENOTDIR

A component of the path prefix of path is not a directory; or path is relative and dirfd is a file descriptor referring to a file other than a directory.

ENOMEM The kernel could not allocate sufficient memory to complete the operation.

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

STANDARDS

Linux.

HISTORY

Linux 5.2. glibc 2.36.

EXAMPLES

The following example sets the read-only flag on the filesystem instance referenced by the mount object attached at /tmp.

```
int fsfd = fspick(AT_FDCWD, "/tmp", FSPICK_CLOEXEC);
fsconfig(fsfd, FSCONFIG_SET_FLAG, "ro", NULL, 0);
fsconfig(fsfd, FSCONFIG_CMD_RECONFIGURE, NULL, NULL, 0);
```

The above procedure is roughly equivalent to the following mount operation using mount(2):

```
mount(NULL, "/tmp", NULL, MS_REMOUNT | MS_RDONLY, NULL);
```

With the notable caveat that in this example, mount(2) will clear all other filesystem parameters (such as MS_DIRSYNC or MS_SYNCHRONOUS); fsconfig(2) will only modify the ro parameter.

SEE ALSO

fsconfig(2), fsmount(2), fsopen(2), mount(2), mount_setattr(2), move_mount(2), open_tree(2), mount_namespaces(7)