



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

## **Red Hat Enterprise Linux Release 9.2 Manual Pages on 'fstatvfs.3p' command**

**\$ man fstatvfs.3p**

FSTATVFS(3P)          POSIX Programmer's Manual          FSTATVFS(3P)

### PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

### NAME

fstatvfs, statvfs ? get file system information

### SYNOPSIS

```
#include <sys/statvfs.h>

int fstatvfs(int fildes, struct statvfs *buf);

int statvfs(const char *restrict path, struct statvfs *restrict buf);
```

### DESCRIPTION

The fstatvfs() function shall obtain information about the file system containing the file referenced by fildes.

The statvfs() function shall obtain information about the file system containing the file named by path.

For both functions, the buf argument is a pointer to a statvfs structure that shall be filled. Read, write, or execute permission of the named file is not required.

The following flags can be returned in the f\_flag member:

ST\_RDONLY   Read-only file system.

ST\_NOSUID   Setuid/setgid bits ignored by exec.

It is unspecified whether all members of the `statvfs` structure have meaningful values on all file systems.

## RETURN VALUE

Upon successful completion, `statvfs()` shall return 0. Otherwise, it shall return -1 and set `errno` to indicate the error.

## ERRORS

The `fstatvfs()` and `statvfs()` functions shall fail if:

**EIO** An I/O error occurred while reading the file system.

**EINTR** A signal was caught during execution of the function.

### EOVERFLOW

One of the values to be returned cannot be represented correctly in the structure pointed to by `buf`.

The `fstatvfs()` function shall fail if:

**EBADF** The `fdes` argument is not an open file descriptor.

The `statvfs()` function shall fail if:

**EACCES** Search permission is denied on a component of the path prefix.

**ELOOP** A loop exists in symbolic links encountered during resolution of the path argument.

### ENAMETOOLONG

The length of a component of a pathname is longer than `{NAME_MAX}`.

**ENOENT** A component of path does not name an existing file or path is an empty string.

### ENOTDIR

A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the path argument contains at least one non-`<slash>` character and ends with one or more trailing `<slash>` characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

The `statvfs()` function may fail if:

**ELOOP** More than `{SYMLOOP_MAX}` symbolic links were encountered during resolution of the path argument.

## ENAMETOOLONG

The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH\_MAX}.

The following sections are informative.

## EXAMPLES

### Obtaining File System Information Using fstatvfs()

The following example shows how to obtain file system information for the file system upon which the file named /home/cnd/mod1 resides, using the fstatvfs() function. The /home/cnd/mod1 file is opened with read/write privileges and the open file descriptor is passed to the fstatvfs() function.

```
#include <sys/statvfs.h>

#include <fcntl.h>

struct statvfs buffer;

int     status;

...

fildes = open("/home/cnd/mod1", O_RDWR);

status = fstatvfs(fildes, &buffer);
```

### Obtaining File System Information Using statvfs()

The following example shows how to obtain file system information for the file system upon which the file named /home/cnd/mod1 resides, using the statvfs() function.

```
#include <sys/statvfs.h>

struct statvfs buffer;

int     status;

...

status = statvfs("/home/cnd/mod1", &buffer);
```

## APPLICATION USAGE

None.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

#### SEE ALSO

chmod(), chown(), creat(), dup(), exec, fcntl(), link(), mknod(),  
open(), pipe(), read(), time(), unlink(), utime(), write()

The Base Definitions volume of POSIX.1-2017, <sys\_statvfs.h>

#### COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.opengroup.org/unix/online.html> .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see [https://www.kernel.org/doc/man-pages/reporting\\_bugs.html](https://www.kernel.org/doc/man-pages/reporting_bugs.html) .

IEEE/The Open Group

2017

FSTATVFS(3P)