NAME

getdirentries, **getdents** - get directory entries in a file system independent format

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/types.h>
#include <dirent.h>

ssize_t
getdirentries(*int fd, char *buf, size_t nbytes, off_t *basep*);

ssize_t
getdents(*int fd, char *buf, size_t nbytes*);

DESCRIPTION

The **getdirentries**() and **getdents**() system calls read directory entries from the directory referenced by the file descriptor *fd* into the buffer pointed to by *buf*, in a file system independent format. Up to *nbytes* of data will be transferred. The *nbytes* argument must be greater than or equal to the block size associated with the file, see stat(2). Some file systems may not support these system calls with buffers smaller than this size.

The data in the buffer is a series of *dirent* structures each containing the following entries:

```
ino_t       d_fileno;
off_t       d_off;
uint16_t    d_reclen;
uint8_t     d_type;
uint16_t    d_namlen;
char        d_name[MAXNAMLEN + 1];          /* see below */
```

The *d_fileno* entry is a number which is unique for each distinct file in the file system. Files that are linked by hard links (see link(2)) have the same *d_fileno*. The *d_off* field returns a cookie which, if non-zero, can be used with lseek(2) to position the directory descriptor to the next entry. The *d_reclen* entry is the length, in bytes, of the directory record. The *d_type* entry is the type of the file pointed to by the directory record. The file type values are defined in *<sys/dirent.h>*. The *d_name* entry contains a null terminated file name. The *d_namlen* entry specifies the length of the file name excluding the null byte. Thus the actual size of *d_name* may vary from 1 to MAXNAMLEN + 1.

Entries may be separated by extra space. The *d_reclen* entry may be used as an offset from the start of a *dirent* structure to the next structure, if any.

The actual number of bytes transferred is returned. The current position pointer associated with *fd* is set to point to the next block of entries. The pointer may not advance by the number of bytes returned by **getdirentries**() or **getdents**(). A value of zero is returned when the end of the directory has been reached.

If the *basep* pointer value is non-NULL, the **getdirentries**() system call writes the position of the block read into the location pointed to by *basep*. Alternatively, the current position pointer may be set and retrieved by lseek(2). The current position pointer should only be set to a value returned by lseek(2), a value returned in the location pointed to by *basep* (**getdirentries**() only), a value returned in the *d_off* field if it is non-zero, or zero.

IMPLEMENTATION NOTES

The *d_off* field is currently set to 0 by the NFS client, since the directory offset cookies returned by an NFS server cannot be used by lseek(2) at this time.

RETURN VALUES

If successful, the number of bytes actually transferred is returned. Otherwise, -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **getdirentries**() system call will fail if:

[EBADF]	The *fd* argument is not a valid file descriptor open for reading.
[EFAULT]	Either *buf* or non-NULL *basep* point outside the allocated address space.
[EINVAL]	The file referenced by *fd* is not a directory, or *nbytes* is too small for returning a directory entry or block of entries, or the current position pointer is invalid.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[EINTEGRITY]	Corrupted data was detected while reading from the file system.
[ENOENT]	Directory unlinked but still open.

SEE ALSO

lseek(2), open(2)

HISTORY

The **getdirentries**() system call first appeared in 4.4BSD. The **getdents**() system call first appeared in FreeBSD 3.0.