NAME

getdents, getdents64 − get directory entries

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/syscall.h> /* Definition of **SYS_*** constants */
#include <unistd.h>

long syscall(SYS_getdents, unsigned int *fd***, struct linux_dir ent ***dirp***,**
unsigned int *count***);**

#define _GNU_SOURCE /* See feature_test_macros(7) */
#include <dirent.h>

ssize_t getdents64(int *fd***, v oid** *dirp***[.***count***], size_t** *count***);**

Note: glibc provides no wrapper for **getdents**(), necessitating the use of **syscall**(2).

Note: There is no definition of *struct linux_dirent* in glibc; see NOTES.

DESCRIPTION

These are not the interfaces you are interested in. Look at **readdir**(3) for the POSIX-conforming C library interface. This page documents the bare kernel system call interfaces.

getdents()

The system call **getdents**() reads several *linux_dirent* structures from the directory referred to by the open file descriptor *fd* into the b uffer pointed to by *dirp*. The argument *count* specifies the size of that buffer.

The *linux_dirent* structure is declared as follows:

```
struct linux_dirent {
    unsigned long  d_ino;     /* Inode number */
    unsigned long  d_off;     /* Offset to next linux_dirent */
    unsigned short d_reclen;  /* Length of this linux_dirent */
    char           d_name[];  /* Filename (null-terminated) */
                      /* length is actually (d_reclen − 2 −
                         offsetof(struct linux_dirent, d_name)) */
    /*
    char           pad;       // Zero padding byte
    char           d_type;    // File type (only since Linux
                              // 2.6.4); offset is (d_reclen − 1)
    */
}
```

d_ino is an inode number. *d_off* is the distance from the start of the directory to the start of the next *linux_dirent*. *d_reclen* is the size of this entire *linux_dirent*. *d_name* is a null-terminated filename.

d_type is a byte at the end of the structure that indicates the file type. It contains one of the following values (defined in*<dir ent.h>*):

DT_BLK	This is a block device.
DT_CHR	This is a character device.
DT_DIR	This is a directory.
DT_FIFO	This is a named pipe (FIFO).
DT_LNK	This is a symbolic link.
DT_REG	This is a regular file.
DT_SOCK	This is a UNIX domain socket.

DT_UNKNOWN
The file type is unknown.

The *d_type* field is implemented since Linux 2.6.4. It occupies a space that was previously a zero-filled padding byte in the *linux_dirent* structure. Thus, on kernels up to and including Linux 2.6.3, attempting to access this field always provides the value 0 (**DT_UNKNOWN**).

Currently, only some filesystems (among them: Btrfs, ext2, ext3, and ext4) have full support for returning the file type in *d_type*. All applications must properly handle a return of**DT_UNKNO WN**.

getdents64()

The original Linux **getdents**() system call did not handle large filesystems and large file offsets. Consequently, Linux 2.4 added **getdents64**(), with wider types for the *d_ino* and *d_off* fields. In addition, **getdents64**() supports an explicit *d_type* field.

The **getdents64**() system call is like **getdents**(), except that its second argument is a pointer to a buffer containing structures of the following type:

```
struct linux_dirent64 {
    ino64_t        d_ino;    /* 64-bit inode number */
    off64_t        d_off;    /* 64-bit offset to next structure */
    unsigned short d_reclen; /* Size of this dirent */
    unsigned char  d_type;   /* File type */
    char           d_name[]; /* Filename (null-terminated) */
};
```

RETURN VALUE

On success, the number of bytes read is returned. On end of directory, 0 is returned. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EBADF
Invalid file descriptor *fd*.

EFAULT
Argument points outside the calling process's address space.

EINVAL
Result buffer is too small.

ENOENT
No such directory.

ENOTDIR
File descriptor does not refer to a directory.

STANDARDS

SVr4.

NOTES

Library support for **getdents64**() was added in glibc 2.30; glibc does not provide a wrapper for **getdents**(); call **getdents**() (or **getdents64**() on earlier glibc versions) using **syscall**(2). In that case you will need to define the *linux_dirent* or *linux_dirent64* structure yourself.

Probably, you want to use **readdir**(3) instead of these system calls.

These calls supersede **readdir**(2).

EXAMPLES

The program below demonstrates the use of **getdents**(). The following output shows an example of what we see when running this program on an ext2 directory:

```
$ ./a.out /testfs/
--------------- nread=120 ---------------
inode#    file type  d_reclen  d_off   d_name
```

```
             2  directory    16         12  .
             2  directory    16         24  ..
            11  directory    24         44  lost+found
            12  regular      16         56  a
        228929  directory    16         68  sub
         16353  directory    16         80  sub2
        130817  directory    16       4096  sub3
```

Program source

```
#define _GNU_SOURCE
#include <dirent.h>     /* Defines DT_* constants */
#include <err.h>
#include <fcntl.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/syscall.h>
#include <unistd.h>

struct linux_dirent {
    unsigned long  d_ino;
    off_t          d_off;
    unsigned short d_reclen;
    char           d_name[];
};

#define BUF_SIZE 1024

int
main(int argc, char *argv[])
{
    int                  fd;
    char                 d_type;
    char                 buf[BUF_SIZE];
    long                 nread;
    struct linux_dirent  *d;

    fd = open(argc > 1 ? argv[1] : ".", O_RDONLY | O_DIRECTORY);
    if (fd == −1)
        err(EXIT_FAILURE, "open");

    for (;;) {
        nread = syscall(SYS_getdents, fd, buf, BUF_SIZE);
        if (nread == −1)
            err(EXIT_FAILURE, "getdents");

        if (nread == 0)
            break;

        printf("−−−−−−−−−−−−−−− nread=%ld −−−−−−−−−−−−−−−\n", nread);
        printf("inode#    file type  d_reclen  d_off   d_name\n");
        for (size_t bpos = 0; bpos < nread;) {
            d = (struct linux_dirent *) (buf + bpos);
            printf("%8lu  ", d->d_ino);
```

```
                d_type = *(buf + bpos + d->d_reclen − 1);
                printf("%−10s ", (d_type == DT_REG) ?  "regular" :
                                 (d_type == DT_DIR) ?  "directory" :
                                 (d_type == DT_FIFO) ? "FIFO" :
                                 (d_type == DT_SOCK) ? "socket" :
                                 (d_type == DT_LNK) ?  "symlink" :
                                 (d_type == DT_BLK) ?  "block dev" :
                                 (d_type == DT_CHR) ?  "char dev" : "???");
                printf("%4d %10jd  %s\n", d->d_reclen,
                       (intmax_t) d->d_off, d->d_name);
                bpos += d->d_reclen;
            }
        }

        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

readdir(2), **readdir**(3), **inode**(7)