

**NAME**

getdents - get directory entries

**SYNOPSIS**

```
int getdents(unsigned int fd, struct linux_dirent *dirp,
             unsigned int count);
```

*Note:* There is no glibc wrapper for this system call; see NOTES.

**DESCRIPTION**

This is not the function you are interested in. Look at [readdir\(3\)](#) for the POSIX conforming C library interface. This page documents the bare kernel system call interface.

The system call **getdents()** reads several *linux\_dirent* structures from the directory referred to by the open file descriptor *fd* into the buffer 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 `<dirent.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 before 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\_UNKNOWN**.

**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 appropriately.

**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.

**CONFORMING TO**

SVr4.

**NOTES**

Glibc does not provide a wrapper for this system call; call it using [syscall\(2\)](#). You will need to define the *linux\_dirent* structure yourself. However, you probably want to use [readdir\(3\)](#) instead.

This call supersedes [readdir\(2\)](#).

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 employed in the *linux\_dirent* structure.

**EXAMPLE**

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 -----
i-node# 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 <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/syscall.h>

#define handle_error(msg)
do { perror(msg); exit(EXIT_FAILURE); } while (0)
```

```

struct linux_dirent {
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, nread;
char buf[BUF_SIZE];
struct linux_dirent *d;
int bpos;
char d_type;

fd = open(argc > 1 ? argv[1] : ., O_RDONLY | O_DIRECTORY);
if (fd == -1)
handle_error(open);

for ( ; ; ) {
nread = syscall(SYS_getdents, fd, buf, BUF_SIZE);
if (nread == -1)
handle_error(getdents);

if (nread == 0)
break;

printf(----- nread=%d -----n, nread);
printf(i-node# file type d_reclen d_off d_namen);
for (bpos = 0; bpos < nread;) {
d = (struct linux_dirent *) (buf + bpos);
printf(%8ld , 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_SOCKET) ? socket :
(d_type == DT_LNK) ? symlink :
(d_type == DT_BLK) ? block dev :
(d_type == DT_CHR) ? char dev : ???);
printf(%4d %10ld %sn, d->d_reclen,
(long long) d->d_off, d->d_name);
bpos += d->d_reclen;
}
}

exit(EXIT_SUCCESS);
}

```

**SEE ALSO**

[readdir\(2\)](#), [readdir\(3\)](#)

**COLOPHON**

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <http://www.kernel.org/doc/man-pages/>.