NAME

opendir − open a directory / readdir − read a directory

SYNOPSIS

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

#include <dirent.h>

DIR *opendir(const char *name);

struct dirent *readdir(DIR *dir);
```

DESCRIPTION opendir

The **opendir()** function opens a directory stream corresponding to the directory *name*, and returns a pointer to the directory stream. The stream is positioned at the first entry in the directory.

RETURN VALUE

The **opendir()** function returns a pointer to the directory stream or NULL if an error occurred.

DESCRIPTION readdir

The **readdir()** function returns a pointer to a dirent structure representing the next directory entry in the directory stream pointed to by *dir*. It returns NULL on reaching the end-of-file or if an error occurred.

The data returned by **readdir()** is overwritten by subsequent calls to **readdir()** for the same directory stream.

The *dirent* structure is defined as follows:

```
struct dirent {
    long           d_ino;          /* inode number */
    off_t          d_off;          /* offset to the next dirent */
    unsigned short d_reclen;       /* length of this record */
    unsigned char  d_type;      /* type of file */
    char           d_name[256]; /* filename */
};
```

RETURN VALUE

The **readdir()** function returns a pointer to a dirent structure, or NULL if an error occurs or end-of-file is reached.

ERRORS

EACCES
Permission denied.

EMFILE
Too many file descriptors in use by process.

ENFILE
Too many files are currently open in the system.

ENOENT
Directory does not exist, or *name* is an empty string.

ENOMEM
Insufficient memory to complete the operation.

ENOTDIR
name is not a directory.

SEE ALSO

open(2), **readdir**(3), **closedir**(3), **rewinddir**(3), **seekdir**(3), **telldir**(3), **scandir**(3)

NAME

qsort − sorts an array

SYNOPSIS

```
#include <stdlib.h>

void qsort(void *base, size_t nmemb, size_t size,
      int(*compar)(const void *, const void *));
```

DESCRIPTION

The **qsort**() function sorts an array with *nmemb* elements of size *size*. The *base* argument points to the start of the array.

The contents of the array are sorted in ascending order according to a comparison function pointed to by *compar*, which is called with two arguments that point to the objects being compared.

The comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second. If two members compare as equal, their order in the sorted array is undefined.

RETURN VALUE

The **qsort**() function returns no value.

SEE ALSO

sort(1), **alphasort**(3), **strcmp**(3), **versionsort**(3)

COLOPHON

This page is part of release 3.05 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.

NAME

stat, lstat − get file status

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char * file_name, struct stat *buf);
int lstat(const char * file_name, struct stat *buf);
```

DESCRIPTION

These functions return information about the specified file. You do not need any access rights to the file to get this information but you need search rights to all directories named in the path leading to the file.

stat stats the file pointed to by *file_name* and fills in *buf*.

lstat is identical to **stat**, except in the case of a symbolic link, where the link itself is stat-ed, not the file that it refers to.

They all return a *stat* structure, which contains the following fields:

```
struct stat {
    dev_t     st_dev;     /* device */
    ino_t     st_ino;     /* inode */
    mode_t    st_mode;    /* protection */
    nlink_t   st_nlink;   /* number of hard links */
    uid_t     st_uid;     /* user ID of owner */
    gid_t     st_gid;     /* group ID of owner */
    dev_t     st_rdev;    /* device type (if inode device) */
    off_t     st_size;    /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t  st_blocks;  /* number of blocks allocated */
    time_t    st_atime;   /* time of last access */
    time_t    st_mtime;   /* time of last modification */
    time_t    st_ctime;   /* time of last status change */
};
```

The value *st_size* gives the size of the file (if it is a regular file or a symlink) in bytes. The size of a symlink is the length of the pathname it contains, without trailing NUL.

The following POSIX macros are defined to check the file type:

S_ISREG(m)	is it a regular file?
S_ISDIR(m)	directory?

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set appropriately.

SEE ALSO

chmod(2), **chown**(2), **readlink**(2), **utime**(2), **capabilities**(7)

NAME

strcmp, strncmp − compare two strings

SYNOPSIS

```
#include <string.h>

int strcmp(const char *s1, const char *s2);

int strncmp(const char *s1, const char *s2, size_t n);
```

DESCRIPTION

The **strcmp**() function compares the two strings *s1* and *s2*. It returns an integer less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, to match, or be greater than *s2*.

The **strncmp**() function is similar, except it only compares the first (at most) *n* characters of *s1* and *s2*.

RETURN VALUE

The **strcmp**() and **strncmp**() functions return an integer less than, equal to, or greater than zero if *s1* (or the first *n* bytes thereof) is found, respectively, to be less than, to match, or be greater than *s2*.

CONFORMING TO

SVr4, 4.3BSD, C89, C99.

SEE ALSO

bcmp(3), **memcmp**(3), **strcasecmp**(3), **strcoll**(3), **strncasecmp**(3), **wcscmp**(3), **wcsncmp**(3)