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_);**
int readdir_r(DIR ***_dirp_, struct dirent ***_entry_**, struct dirent ****_result_**);**

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.

DESCRIPTION readdir_r

The **readdir_r()** function initializes the structure referenced by *entry* and stores a pointer to this structure in *result*. On successful return, the pointer returned at **result* will have the same value as the argument *entry*. Upon reaching the end of the directory stream, this pointer will have the value NULL.

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.

readdir_r() returns 0 if successful or an error number to indicate failure.

ERRORS

EACCES
Permission denied.

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

ENOTDIR
name is not a directory.

NAME

stat, lstat − get file status

SYNOPSIS

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

int stat(const char ***_path_, struct stat ***_buf_**);**
int lstat(const char ***_path_, 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 *path* 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 in the field *st_mode:*

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.

ERRORS

EACCES Search permisson is denied for one of the directories in the path prefix of *path*.

ENOENT A component of *path* does not exist, or *path* is an empty string.

ENOTDIR A component of the path prefix of *path* is not a directory.