

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.

Not all of the Linux filesystems implement all of the time fields. Some file system types allow mounting in such a way that file accesses do not cause an update of the *st_atime* field. (See ‘noatime’ in **mount(8)**.)

The field *st_atime* is changed by file accesses, e.g. by **execve(2)**, **mknod(2)**, **pipe(2)**, **utime(2)** and **read(2)** (of more than zero bytes). Other routines, like **mmap(2)**, may or may not update *st_atime*.

The field *st_mtime* is changed by file modifications, e.g. by **mknod(2)**, **truncate(2)**, **utime(2)** and **write(2)** (of more than zero bytes). Moreover, *st_mtime* of a directory is changed by the creation or deletion of files in that directory. The *st_mtime* field is *not* changed for changes in owner, group, hard link count, or mode.

The field *st_ctime* is changed by writing or by setting inode information (i.e., owner, group, link count, mode, etc.).

RETURN VALUE

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

NAME

strcpy, strncpy – copy a string

SYNOPSIS

```
#include <string.h>
```

```
char *strcpy(char *dest, const char *src);
```

```
char *strncpy(char *dest, const char *src, size_t n);
```

DESCRIPTION

The **strcpy()** function copies the string pointed to by *src* (including the terminating ‘\0’ character) to the array pointed to by *dest*. The strings may not overlap, and the destination string *dest* must be large enough to receive the copy.

The **strncpy()** function is similar, except that not more than *n* bytes of *src* are copied. Thus, if there is no null byte among the first *n* bytes of *src*, the result will not be null-terminated.

In the case where the length of *src* is less than that of *n*, the remainder of *dest* will be padded with null bytes.

RETURN VALUE

The **strcpy()** and **strncpy()** functions return a pointer to the destination string *dest*.

BUGS

If the destination string of a **strcpy()** is not large enough (that is, if the programmer was stupid/lazy, and failed to check the size before copying) then anything might happen. Overflowing fixed length strings is a favourite cracker technique.

CONFORMING TO

SVr4, 4.3BSD, C89, C99.

SEE ALSO

bcopy(3), **memcpy(3)**, **memmove(3)**, **wscpy(3)**, **wcsncpy(3)**