NAME

archive_entry_stat, **archive_entry_copy_stat**, **archive_entry_filetype**, **archive_entry_set_filetype**, **archive_entry_mode**, **archive_entry_set_mode**, **archive_entry_size**, **archive_entry_size_is_set**, **archive_entry_set_size**, **archive_entry_unset_size**, **archive_entry_dev**, **archive_entry_set_dev**, **archive_entry_dev_is_set**, **archive_entry_devmajor**, **archive_entry_set_devmajor**, **archive_entry_devminor**, **archive_entry_set_devminor**, **archive_entry_ino**, **archive_entry_set_ino**, **archive_entry_ino_is_set**, **archive_entry_ino64**, **archive_entry_set_ino64**, **archive_entry_nlink**, **archive_entry_rdev**, **archive_entry_set_rdev**, **archive_entry_rdevmajor**, **archive_entry_set_rdevmajor**, **archive_entry_rdevminor**, **archive_entry_set_rdevminor**, — accessor functions for manipulating archive entry descriptions

LIBRARY

Streaming Archive Library (libarchive, -larchive)

SYNOPSIS

```
#include <archive_entry.h>

const struct stat *
archive_entry_stat(struct archive_entry *a);

void
archive_entry_copy_stat(struct archive_entry *a, const struct stat *sb);

mode_t
archive_entry_filetype(struct archive_entry *a);

void
archive_entry_set_filetype(struct archive_entry *a, unsigned int type);

mode_t
archive_entry_mode(struct archive_entry *a);

void
archive_entry_set_mode(struct archive_entry *a, mode_t mode);

int64_t
archive_entry_size(struct archive_entry *a);

int
archive_entry_size_is_set(struct archive_entry *a);

void
archive_entry_set_size(struct archive_entry *a, int64_t size);

void
archive_entry_unset_size(struct archive_entry *a);

dev_t
archive_entry_dev(struct archive_entry *a);

void
archive_entry_set_dev(struct archive_entry *a, dev_t dev);

int
archive_entry_dev_is_set(struct archive_entry *a);
```

dev_t
archive_entry_devmajor(*struct archive_entry *a*);

void
archive_entry_set_devmajor(*struct archive_entry *a*, *dev_t major*);

dev_t
archive_entry_devminor(*struct archive_entry *a*);

void
archive_entry_set_devminor(*struct archive_entry *a*, *dev_t minor*);

ino_t
archive_entry_ino(*struct archive_entry *a*);

void
archive_entry_set_ino(*struct archive_entry *a*, *unsigned long ino*);

int
archive_entry_ino_is_set(*struct archive_entry *a*);

int64_t
archive_entry_ino64(*struct archive_entry *a*);

void
archive_entry_set_ino64(*struct archive_entry *a*, *int64_t ino*);

unsigned int
archive_entry_nlink(*struct archive_entry *a*);

void
archive_entry_set_nlink(*struct archive_entry *a*, *unsigned int count*);

dev_t
archive_entry_rdev(*struct archive_entry *a*);

dev_t
archive_entry_rdevmajor(*struct archive_entry *a*);

dev_t
archive_entry_rdevminor(*struct archive_entry *a*);

void
archive_entry_set_rdev(*struct archive_entry *a*, *dev_t dev*);

void
archive_entry_set_rdevmajor(*struct archive_entry *a*, *dev_t major*);

void
archive_entry_set_rdevminor(*struct archive_entry *a*, *dev_t minor*);

DESCRIPTION

Copying to and from *struct stat*

The function **archive_entry_stat**() converts the various fields stored in the archive entry to the format used by stat(2). The return value remains valid until either **archive_entry_clear**() or **archive_entry_free**() is called. It is not affected by calls to the set accessor functions. It currently sets the following values in *struct stat*: *st_atime*, *st_ctime*, *st_dev*, *st_gid*, *st_ino*, *st_mode*, *st_mtime*, *st_nlink*, *st_rdev*, *st_size*, *st_uid*. In addition, *st_birthtime* and high-precision information for time-related fields will be included on platforms that support it.

The function **archive_entry_copy_stat**() copies fields from the platform's *struct stat*. Fields not provided by *struct stat* are unchanged.

General accessor functions

The functions **archive_entry_filetype**() and **archive_entry_set_filetype**() get respectively set the filetype. The file type is one of the following constants:

AE_IFREG	Regular file
AE_IFLNK	Symbolic link
AE_IFSOCK	Socket
AE_IFCHR	Character device
AE_IFBLK	Block device
AE_IFDIR	Directory
AE_IFIFO	Named pipe (fifo)

Not all file types are supported by all platforms. The constants used by stat(2) may have different numeric values from the corresponding constants above.

The functions **archive_entry_mode**() and **archive_entry_set_mode**() get/set a combination of file type and permissions and provide the equivalent of *st_mode*. Use of **archive_entry_filetype**() and **archive_entry_perm**() for getting and **archive_entry_set_filetype**() and **archive_entry_set_perm**() for setting is recommended.

The function **archive_entry_size**() returns the file size, if it has been set, and 0 otherwise. **archive_entry_size**() can be used to query that status. **archive_entry_set_size**() and **archive_entry_unset_size**() set and unset the size, respectively.

The number of references (hardlinks) can be obtained by calling **archive_entry_nlinks**() and set with **archive_entry_set_nlinks**().

Identifying unique files

The functions **archive_entry_dev**() and **archive_entry_ino64**() are used by archive_entry_linkify(3) to find hardlinks. The pair of device and inode is suppossed to identify hardlinked files.

The device major and minor number can be obtained independently using **archive_entry_devmajor**() and **archive_entry_devminor**(). The device can be set either via **archive_entry_set_dev**() or by the combination of major and minor number using **archive_entry_set_devmajor**() and **archive_entry_set_devminor**().

The inode number can be obtained using **archive_entry_ino**(). This is a legacy interface that uses the platform *ino_t*, which may be very small. To set the inode number, **archive_entry_set_ino64**() is the preferred interface.

Accessor functions for block and character devices

Block and character devices are characterised either using a device number or a pair of major and minor number. The combined device number can be obtained with **archive_device_rdev**() and set with **archive_device_set_rdev**(). The major and minor numbers are accessed by **archive_device_rdevmajor**(), **archive_device_rdevminor**() **archive_device_set_rdevmajor**() and **archive_device_set_rdevminor**().

The process of splitting the combined device number into major and minor number and the reverse process of combing them differs between platforms. Some archive formats use the combined form, while other formats use the split form.

SEE ALSO

archive(3), archive_entry_acl(3), archive_entry_perms(3), archive_entry_time(3), stat(2)