

**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_nlink()` and set with `archive_entry_set_nlink()`.

### 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 supposed 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 combining them differs between platforms. Some archive formats use the combined form, while other formats use the split form.

**SEE ALSO**

stat(2), archive\_entry\_acl(3), archive\_entry\_perms(3), archive\_entry\_time(3), libarchive(3)