NAME

archive_entry_acl_add_entry, **archive_entry_acl_add_entry_w**, **archive_entry_acl_clear**, **archive_entry_acl_count**, **archive_entry_acl_next**, **archive_entry_acl_next_w**, **archive_entry_acl_reset**, **archive_entry_acl_text_w**, **archive_entry_atime**, **archive_entry_atime_nsec**, **archive_entry_clear**, **archive_entry_clone**, **archive_entry_copy_fflags_text**, **archive_entry_copy_fflags_text_w**, **archive_entry_copy_gname**, **archive_entry_copy_gname_w**, **archive_entry_copy_hardlink**, **archive_entry_copy_hardlink_w**, **archive_entry_copy_link**, **archive_entry_copy_link_w**, **archive_entry_copy_pathname_w**, **archive_entry_copy_sourcepath**, **archive_entry_copy_stat**, **archive_entry_copy_symlink**, **archive_entry_copy_symlink_w**, **archive_entry_copy_uname**, **archive_entry_copy_uname_w**, **archive_entry_dev**, **archive_entry_devmajor**, **archive_entry_devminor**, **archive_entry_filetype**, **archive_entry_fflags**, **archive_entry_fflags_text**, **archive_entry_free**, **archive_entry_gid**, **archive_entry_gname**, **archive_entry_hardlink**, **archive_entry_ino**, **archive_entry_mode**, **archive_entry_mtime**, **archive_entry_mtime_nsec**, **archive_entry_nlink**, **archive_entry_new**, **archive_entry_pathname**, **archive_entry_pathname_w**, **archive_entry_rdev**, **archive_entry_rdevmajor**, **archive_entry_rdevminor**, **archive_entry_set_atime**, **archive_entry_set_ctime**, **archive_entry_set_dev**, **archive_entry_set_devmajor**, **archive_entry_set_devminor**, **archive_entry_set_filetype**, **archive_entry_set_fflags**, **archive_entry_set_gid**, **archive_entry_set_gname**, **archive_entry_set_hardlink**, **archive_entry_set_link**, **archive_entry_set_mode**, **archive_entry_set_mtime**, **archive_entry_set_pathname**, **archive_entry_set_rdevmajor**, **archive_entry_set_rdevminor**, **archive_entry_set_size**, **archive_entry_set_symlink**, **archive_entry_set_uid**, **archive_entry_set_uname**, **archive_entry_size**, **archive_entry_sourcepath**, **archive_entry_stat**, **archive_entry_symlink**, **archive_entry_uid**, **archive_entry_uname** — functions for manipulating archive entry descriptions

SYNOPSIS

```
#include <archive_entry.h>

void
archive_entry_acl_add_entry(struct archive_entry *,int type,int permset,
    int tag,int qual,const char *name);

void
archive_entry_acl_add_entry_w(struct archive_entry *,int type,
    int permset,int tag,int qual,const wchar_t *name);

void
archive_entry_acl_clear(struct archive_entry *);

int
archive_entry_acl_count(struct archive_entry *, int type);

int
archive_entry_acl_next(struct archive_entry *,int want_type,int *type,
    int *permset,int *tag,int *qual,const char **name);

int
archive_entry_acl_next_w(struct archive_entry *,int want_type,int *type,
    int *permset,int *tag,int *qual,const wchar_t **name);
```

int
archive_entry_acl_reset(*struct archive_entry **, *int want_type*);

*const wchar_t **
archive_entry_acl_text_w(*struct archive_entry **, *int flags*);

time_t
archive_entry_atime(*struct archive_entry **);

long
archive_entry_atime_nsec(*struct archive_entry **);

*struct archive_entry **
archive_entry_clear(*struct archive_entry **);

*struct archive_entry **
archive_entry_clone(*struct archive_entry **);

*const char * **
archive_entry_copy_fflags_text_w(*struct archive_entry **, *const char **);

*const wchar_t **
archive_entry_copy_fflags_text_w(*struct archive_entry **, *const wchar_t **);

void
archive_entry_copy_gname(*struct archive_entry **, *const char **);

void
archive_entry_copy_gname_w(*struct archive_entry **, *const wchar_t **);

void
archive_entry_copy_hardlink(*struct archive_entry **, *const char **);

void
archive_entry_copy_hardlink_w(*struct archive_entry **, *const wchar_t **);

void
archive_entry_copy_sourcepath(*struct archive_entry **, *const char **);

void
archive_entry_copy_pathname_w(*struct archive_entry **, *const wchar_t **);

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

void
archive_entry_copy_symlink(*struct archive_entry **, *const char **);

void
archive_entry_copy_symlink_w(*struct archive_entry **, *const wchar_t **);

void
archive_entry_copy_uname(*struct archive_entry **, *const char **);

void
archive_entry_copy_uname_w(*struct archive_entry **, *const wchar_t **);

dev_t
archive_entry_dev(*struct archive_entry **);

dev_t
archive_entry_devmajor(*struct archive_entry **);

dev_t
archive_entry_devminor(*struct archive_entry **);

mode_t
archive_entry_filetype(*struct archive_entry **);

void
archive_entry_fflags(*struct archive_entry **, *unsigned long *set*, *unsigned long *clear*);

*const char **
archive_entry_fflags_text(*struct archive_entry **);

void
archive_entry_free(*struct archive_entry **);

*const char **
archive_entry_gname(*struct archive_entry **);

*const char **
archive_entry_hardlink(*struct archive_entry **);

ino_t
archive_entry_ino(*struct archive_entry **);

mode_t
archive_entry_mode(*struct archive_entry **);

time_t
archive_entry_mtime(*struct archive_entry **);

long
archive_entry_mtime_nsec(*struct archive_entry **);

unsigned int
archive_entry_nlink(*struct archive_entry **);

*struct archive_entry **
archive_entry_new(*void*);

*const char **
archive_entry_pathname(*struct archive_entry **);

*const wchar_t **
archive_entry_pathname_w(*struct archive_entry **);

dev_t
archive_entry_rdev(*struct archive_entry **);

dev_t
archive_entry_rdevmajor(*struct archive_entry **);

dev_t
archive_entry_rdevminor(*struct archive_entry **);

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

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

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

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

void
archive_entry_set_fflags(*struct archive_entry **,*unsigned long set*,
unsigned long clear);

void
archive_entry_set_gid(*struct archive_entry **, *gid_t*);

void
archive_entry_set_gname(*struct archive_entry **, *const char **);

void
archive_entry_set_hardlink(*struct archive_entry **, *const char **);

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

void
archive_entry_set_link(*struct archive_entry **, *const char **);

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

void
archive_entry_set_mtime(*struct archive_entry **, *time_t*, *long nanos*);

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

void
archive_entry_set_pathname(*struct archive_entry **, *const char **);

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

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

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

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

void
archive_entry_set_symlink(*struct archive_entry **, *const char **);

void
archive_entry_set_uid(*struct archive_entry **, *uid_t*);

void
archive_entry_set_uname(*struct archive_entry **, *const char **);

int64_t
archive_entry_size(*struct archive_entry* *);

const char *
archive_entry_sourcepath(*struct archive_entry* *);

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

const char *
archive_entry_symlink(*struct archive_entry* *);

const char *
archive_entry_uname(*struct archive_entry* *);

DESCRIPTION

These functions create and manipulate data objects that represent entries within an archive. You can think of a struct archive_entry as a heavy-duty version of struct stat: it includes everything from struct stat plus associated pathname, textual group and user names, etc. These objects are used by libarchive(3) to represent the metadata associated with a particular entry in an archive.

Create and Destroy

There are functions to allocate, destroy, clear, and copy *archive_entry* objects:

archive_entry_clear()
: Erases the object, resetting all internal fields to the same state as a newly-created object. This is provided to allow you to quickly recycle objects without thrashing the heap.

archive_entry_clone()
: A deep copy operation; all text fields are duplicated.

archive_entry_free()
: Releases the struct archive_entry object.

archive_entry_new()
: Allocate and return a blank struct archive_entry object.

Set and Get Functions

Most of the functions here set or read entries in an object. Such functions have one of the following forms:

archive_entry_set_XXXX()
: Stores the provided data in the object. In particular, for strings, the pointer is stored, not the referenced string.

archive_entry_copy_XXXX()
: As above, except that the referenced data is copied into the object.

archive_entry_XXXX()
: Returns the specified data. In the case of strings, a const-qualified pointer to the string is returned.

String data can be set or accessed as wide character strings or normal *char* strings. The functions that use wide character strings are suffixed with **_w**. Note that these are different representations of the same data: For example, if you store a narrow string and read the corresponding wide string, the object will transparently convert formats using the current locale. Similarly, if you store a wide string and then store a narrow string for the same data, the previously-set wide string will be discarded in favor of the new data.

There are a few set/get functions that merit additional description:

archive_entry_set_link()
: This function sets the symlink field if it is already set. Otherwise, it sets the hardlink field.

File Flags

File flags are transparently converted between a bitmap representation and a textual format. For example, if you set the bitmap and ask for text, the library will build a canonical text format. However, if you set a text format and request a text format, you will get back the same text, even if it is ill-formed. If you need to canonicalize a textual flags string, you should first set the text form, then request the bitmap form, then use that to set the bitmap form. Setting the bitmap format will clear the internal text representation and force it to be reconstructed when you next request the text form.

The bitmap format consists of two integers, one containing bits that should be set, the other specifying bits that should be cleared. Bits not mentioned in either bitmap will be ignored. Usually, the bitmap of bits to be cleared will be set to zero. In unusual circumstances, you can force a fully-specified set of file flags by setting the bitmap of flags to clear to the complement of the bitmap of flags to set. (This differs from `fflagstostr`(3), which only includes names for set bits.) Converting a bitmap to a textual string is a platform-specific operation; bits that are not meaningful on the current platform will be ignored.

The canonical text format is a comma-separated list of flag names. The **`archive_entry_copy_fflags_text`**() and **`archive_entry_copy_fflags_text_w`**() functions parse the provided text and sets the internal bitmap values. This is a platform-specific operation; names that are not meaningful on the current platform will be ignored. The function returns a pointer to the start of the first name that was not recognized, or NULL if every name was recognized. Note that every name--including names that follow an unrecognized name--will be evaluated, and the bitmaps will be set to reflect every name that is recognized. (In particular, this differs from `strtofflags`(3), which stops parsing at the first unrecognized name.)

ACL Handling

XXX This needs serious help. XXX

An “Access Control List” (ACL) is a list of permissions that grant access to particular users or groups beyond what would normally be provided by standard POSIX mode bits. The ACL handling here addresses some deficiencies in the POSIX.1e draft 17 ACL specification. In particular, POSIX.1e draft 17 specifies several different formats, but none of those formats include both textual user/group names and numeric UIDs/GIDs.

XXX explain ACL stuff XXX

SEE ALSO

`archive`(3)

HISTORY

The **`libarchive`** library first appeared in FreeBSD 5.3.

AUTHORS

The **`libarchive`** library was written by Tim Kientzle ⟨kientzle@acm.org⟩.