

**NAME**

**archive\_entry\_acl\_add\_entry**, **archive\_entry\_acl\_add\_entry\_w**, **archive\_entry\_acl\_clear**,  
**archive\_entry\_acl\_count**, **archive\_entry\_acl\_from\_text**, **archive\_entry\_acl\_from\_text\_w**,  
**archive\_entry\_acl\_next**, **archive\_entry\_acl\_reset**, **archive\_entry\_acl\_to\_text**,  
**archive\_entry\_acl\_to\_text\_w**, **archive\_entry\_acl\_types** - functions for manipulating Access Control Lists  
in archive entry descriptions

**LIBRARY**

Streaming Archive Library (libarchive, -larchive)

**SYNOPSIS**

```
#include <archive_entry.h>
```

*void*

```
archive_entry_acl_add_entry(struct archive_entry *a, int type, int permset, int tag, int qualifier,  
const char *name);
```

*void*

```
archive_entry_acl_add_entry_w(struct archive_entry *a, int type, int permset, int tag, int qualifier,  
const wchar_t *name);
```

*void*

```
archive_entry_acl_clear(struct archive_entry *a);
```

*int*

```
archive_entry_acl_count(struct archive_entry *a, int type);
```

*int*

```
archive_entry_acl_from_text(struct archive_entry *a, const char *text, int type);
```

*int*

```
archive_entry_acl_from_text_w(struct archive_entry *a, const wchar_t *text, int type);
```

*int*

```
archive_entry_acl_next(struct archive_entry *a, int type, int *ret_type, int *ret_permset, int *ret_tag,  
int *ret_qual, const char **ret_name);
```

*int*

```
archive_entry_acl_reset(struct archive_entry *a, int type);
```

*char \**

**archive\_entry\_acl\_to\_text**(*struct archive\_entry \*a, ssize\_t \*len\_p, int flags*);

*wchar\_t \**

**archive\_entry\_acl\_to\_text\_w**(*struct archive\_entry \*a, ssize\_t \*len\_p, int flags*);

*int*

**archive\_entry\_acl\_types**(*struct archive\_entry \*a*);

## DESCRIPTION

The "Access Control Lists (ACLs)" extend the standard Unix permission model. The ACL interface of **libarchive** supports both POSIX.1e and NFSv4 style ACLs. Use of ACLs is restricted by various levels of ACL support in operating systems, file systems and archive formats.

### POSIX.1e Access Control Lists

A POSIX.1e ACL consists of a number of independent entries. Each entry specifies the permission set as a bitmask of basic permissions. Valid permissions in the *permset* are:

ARCHIVE\_ENTRY\_ACL\_READ (**r**)

ARCHIVE\_ENTRY\_ACL\_WRITE (**w**)

ARCHIVE\_ENTRY\_ACL\_EXECUTE (**x**)

The permissions correspond to the normal Unix permissions.

The *tag* specifies the principal to which the permission applies. Valid values are:

ARCHIVE\_ENTRY\_ACL\_USER           The user specified by the name field.

ARCHIVE\_ENTRY\_ACL\_USER\_OBJ       The owner of the file.

ARCHIVE\_ENTRY\_ACL\_GROUP          The group specified by the name field.

ARCHIVE\_ENTRY\_ACL\_GROUP\_OBJ      The group which owns the file.

ARCHIVE\_ENTRY\_ACL\_MASK           The maximum permissions to be obtained via group permissions.

ARCHIVE\_ENTRY\_ACL\_OTHER          Any principal who is not the file owner or a member of the owning group.

The principals ARCHIVE\_ENTRY\_ACL\_USER\_OBJ, ARCHIVE\_ENTRY\_ACL\_GROUP\_OBJ and ARCHIVE\_ENTRY\_ACL\_OTHER are equivalent to user, group and other in the classic Unix permission model and specify non-extended ACL entries.

All files have an access ACL (ARCHIVE\_ENTRY\_ACL\_TYPE\_ACCESS). This specifies the permissions required for access to the file itself. Directories have an additional ACL (ARCHIVE\_ENTRY\_ACL\_TYPE\_DEFAULT), which controls the initial access ACL for newly-created directory entries.

### NFSv4 Access Control Lists

A NFSv4 ACL consists of multiple individual entries called Access Control Entries (ACEs).

There are four possible types of a NFSv4 ACE:

- ARCHIVE\_ENTRY\_ACL\_TYPE\_ALLOW Allow principal to perform actions requiring given permissions.
- ARCHIVE\_ENTRY\_ACL\_TYPE\_DENY Prevent principal from performing actions requiring given permissions.
- ARCHIVE\_ENTRY\_ACL\_TYPE\_AUDIT Log access attempts by principal which require given permissions.
- ARCHIVE\_ENTRY\_ACL\_TYPE\_ALARM Trigger a system alarm on access attempts by principal which require given permissions.

The *tag* specifies the principal to which the permission applies. Valid values are:

- ARCHIVE\_ENTRY\_ACL\_USER The user specified by the name field.
- ARCHIVE\_ENTRY\_ACL\_USER\_OBJ The owner of the file.
- ARCHIVE\_ENTRY\_ACL\_GROUP The group specified by the name field.
- ARCHIVE\_ENTRY\_ACL\_GROUP\_OBJ The group which owns the file.
- ARCHIVE\_ENTRY\_ACL\_EVERYONE Any principal who is not the file owner or a member of the owning group.

Entries with the ARCHIVE\_ENTRY\_ACL\_USER or ARCHIVE\_ENTRY\_ACL\_GROUP tag store the user and group name in the *name* string and optionally the user or group ID in the *qualifier* integer.

NFSv4 ACE permissions and flags are stored in the same *permset* bitfield. Some permissions share the same constant and permission character but have different effect on directories than on files. The following ACE permissions are supported:

- ARCHIVE\_ENTRY\_ACL\_READ\_DATA (**r**)  
Read data (file).
- ARCHIVE\_ENTRY\_ACL\_LIST\_DIRECTORY (**r**)  
List entries (directory).
- ARCHIVE\_ENTRY\_ACL\_WRITE\_DATA (**w**)  
Write data (file).
- ARCHIVE\_ENTRY\_ACL\_ADD\_FILE (**w**)  
Create files (directory).
- ARCHIVE\_ENTRY\_ACL\_EXECUTE (**x**)  
Execute file or change into a directory.
- ARCHIVE\_ENTRY\_ACL\_APPEND\_DATA (**p**)  
Append data (file).
- ARCHIVE\_ENTRY\_ACL\_ADD\_SUBDIRECTORY (**p**)

Create subdirectories (directory).

**ARCHIVE\_ENTRY\_ACL\_DELETE\_CHILD (D)**

Remove files and subdirectories inside a directory.

**ARCHIVE\_ENTRY\_ACL\_DELETE (d)**

Remove file or directory.

**ARCHIVE\_ENTRY\_ACL\_READ\_ATTRIBUTES (a)**

Read file or directory attributes.

**ARCHIVE\_ENTRY\_ACL\_WRITE\_ATTRIBUTES (A)**

Write file or directory attributes.

**ARCHIVE\_ENTRY\_ACL\_READ\_NAMED\_ATTRS (R)**

Read named file or directory attributes.

**ARCHIVE\_ENTRY\_ACL\_WRITE\_NAMED\_ATTRS (W)**

Write named file or directory attributes.

**ARCHIVE\_ENTRY\_ACL\_READ\_ACL (c)**

Read file or directory ACL.

**ARCHIVE\_ENTRY\_ACL\_WRITE\_ACL (C)**

Write file or directory ACL.

**ARCHIVE\_ENTRY\_ACL\_WRITE\_OWNER (o)**

Change owner of a file or directory.

**ARCHIVE\_ENTRY\_ACL\_SYNCHRONIZE (s)**

Use synchronous I/O.

The following NFSv4 ACL inheritance flags are supported:

**ARCHIVE\_ENTRY\_ACL\_ENTRY\_FILE\_INHERIT (f)**

Inherit parent directory ACE to files.

**ARCHIVE\_ENTRY\_ACL\_ENTRY\_DIRECTORY\_INHERIT (d)**

Inherit parent directory ACE to subdirectories.

**ARCHIVE\_ENTRY\_ACL\_ENTRY\_INHERIT\_ONLY (i)**

Only inherit, do not apply the permission on the directory itself.

**ARCHIVE\_ENTRY\_ACL\_ENTRY\_NO\_PROPAGATE\_INHERIT (n)**

Do not propagate inherit flags. Only first-level entries inherit ACLs.

**ARCHIVE\_ENTRY\_ACL\_ENTRY\_SUCCESSFUL\_ACCESS (S)**

Trigger alarm or audit on successful access.

**ARCHIVE\_ENTRY\_ACL\_ENTRY\_FAILED\_ACCESS (F)**

Trigger alarm or audit on failed access.

**ARCHIVE\_ENTRY\_ACL\_ENTRY\_INHERITED (I)**

Mark that ACE was inherited.

## Functions

**archive\_entry\_acl\_add\_entry()** and **archive\_entry\_acl\_add\_entry\_w()** add a single ACL entry. For the

access ACL and non-extended principals, the classic Unix permissions are updated. An archive entry cannot contain both POSIX.1e and NFSv4 ACL entries.

**archive\_entry\_acl\_clear()** removes all ACL entries and resets the enumeration pointer.

**archive\_entry\_acl\_count()** counts the ACL entries that have the given type mask. *type* can be the bitwise-or of

ARCHIVE\_ENTRY\_ACL\_TYPE\_ACCESS  
ARCHIVE\_ENTRY\_ACL\_TYPE\_DEFAULT

for POSIX.1e ACLs and

ARCHIVE\_ENTRY\_ACL\_TYPE\_ALLOW  
ARCHIVE\_ENTRY\_ACL\_TYPE\_DENY  
ARCHIVE\_ENTRY\_ACL\_TYPE\_AUDIT  
ARCHIVE\_ENTRY\_ACL\_TYPE\_ALARM

for NFSv4 ACLs. For POSIX.1e ACLs if ARCHIVE\_ENTRY\_ACL\_TYPE\_ACCESS is included and at least one extended ACL entry is found, the three non-extended ACLs are added.

**archive\_entry\_acl\_from\_text()** and **archive\_entry\_acl\_from\_text\_w()** add new (or merge with existing) ACL entries from (wide) text. The argument *type* may take one of the following values:

ARCHIVE\_ENTRY\_ACL\_TYPE\_ACCESS  
ARCHIVE\_ENTRY\_ACL\_TYPE\_DEFAULT  
ARCHIVE\_ENTRY\_ACL\_TYPE\_NFS4

Supports all formats that can be created with **archive\_entry\_acl\_to\_text()** or respectively **archive\_entry\_acl\_to\_text\_w()**. Existing ACL entries are preserved. To get a clean new ACL from text **archive\_entry\_acl\_clear()** must be called first. Entries prefixed with "default:" are treated as ARCHIVE\_ENTRY\_ACL\_TYPE\_DEFAULT unless *type* is ARCHIVE\_ENTRY\_ACL\_TYPE\_NFS4. Invalid entries, non-parseable ACL entries and entries beginning with the '#' character (comments) are skipped.

**archive\_entry\_acl\_next()** return the next entry of the ACL list. This functions may only be called after **archive\_entry\_acl\_reset()** has indicated the presence of extended ACL entries.

**archive\_entry\_acl\_reset()** prepare reading the list of ACL entries with **archive\_entry\_acl\_next()**. The function returns 0 if no non-extended ACLs are found. In this case, the access permissions should be obtained by **archive\_entry\_mode(3)** or set using **chmod(2)**. Otherwise, the function returns the same value as **archive\_entry\_acl\_count()**.

**archive\_entry\_acl\_to\_text()** and **archive\_entry\_acl\_to\_text\_w()** convert the ACL entries for the given type into a (wide) string of ACL entries separated by newline. If the pointer *len\_p* is not NULL, then the function shall return the length of the string (not including the NULL terminator) in the location

pointed to by *len\_p*. The *flag* argument is a bitwise-or.

The following flags are effective only on POSIX.1e ACL:

ARCHIVE\_ENTRY\_ACL\_TYPE\_ACCESS

Output access ACLs.

ARCHIVE\_ENTRY\_ACL\_TYPE\_DEFAULT

Output POSIX.1e default ACLs.

ARCHIVE\_ENTRY\_ACL\_STYLE\_MARK\_DEFAULT

Prefix each default ACL entry with the word "default:".

ARCHIVE\_ENTRY\_ACL\_STYLE\_SOLARIS

The mask and other ACLs don not contain a double colon.

The following flags are effective only on NFSv4 ACL:

ARCHIVE\_ENTRY\_ACL\_STYLE\_COMPACT

Do not output minus characters for unset permissions and flags in NFSv4 ACL permission and flag fields.

The following flags are effective on both POSIX.1e and NFSv4 ACL:

ARCHIVE\_ENTRY\_ACL\_STYLE\_EXTRA\_ID

Add an additional colon-separated field containing the user or group id.

ARCHIVE\_ENTRY\_ACL\_STYLE\_SEPARATOR\_COMMA

Separate ACL entries with comma instead of newline.

If the archive entry contains NFSv4 ACLs, all types of NFSv4 ACLs are returned. If the entry contains POSIX.1e ACLs and none of the flags ARCHIVE\_ENTRY\_ACL\_TYPE\_ACCESS or ARCHIVE\_ENTRY\_ACL\_TYPE\_DEFAULT are specified, both access and default entries are returned and default entries are prefixed with "default:".

**archive\_entry\_acl\_types()** get ACL entry types contained in an archive entry's ACL. As POSIX.1e and NFSv4 ACL entries cannot be mixed, this function is a very efficient way to detect if an ACL already contains POSIX.1e or NFSv4 ACL entries.

## RETURN VALUES

**archive\_entry\_acl\_count()** and **archive\_entry\_acl\_reset()** returns the number of ACL entries that match the given type mask. For POSIX.1e ACLs if the type mask includes ARCHIVE\_ENTRY\_ACL\_TYPE\_ACCESS and at least one extended ACL entry exists, the three classic Unix permissions are counted.

**archive\_entry\_acl\_from\_text()** and **archive\_entry\_acl\_from\_text\_w()** return ARCHIVE\_OK if all entries were successfully parsed and ARCHIVE\_WARN if one or more entries were invalid or non-parseable.

**archive\_entry\_acl\_next()** returns ARCHIVE\_OK on success, ARCHIVE\_EOF if no more ACL entries exist and ARCHIVE\_WARN if **archive\_entry\_acl\_reset()** has not been called first.

**archive\_entry\_acl\_to\_text()** returns a string representing the ACL entries matching the given type and flags on success or NULL on error.

**archive\_entry\_acl\_to\_text\_w()** returns a wide string representing the ACL entries matching the given type and flags on success or NULL on error.

**archive\_entry\_acl\_types()** returns a bitmask of ACL entry types or 0 if archive entry has no ACL entries.

## SEE ALSO

archive\_entry(3), libarchive(3)