

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)*