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** — functions for manipulating Access Control Lists in archive entry descriptions

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_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_next_w(*struct archive_entry *a*, *int type*, *int *ret_type*, *int *ret_permset*, *int *ret_tag*, *int *ret_qual*, *const wchar_t **ret_name*);

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

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

DESCRIPTION

An "Access Control List" is a generalisation of the classic Unix permission system. The ACL interface of **libarchive** is derived from the POSIX.1e draft, but restricted to simplify dealing with practical implementations in various Operating Systems and archive formats.

An ACL consists of a number of independent entries. Each entry specifies the permission set as bitmask of basic permissions. Valid permissions are:

ARCHIVE_ENTRY_ACL_EXECUTE
ARCHIVE_ENTRY_ACL_WRITE
ARCHIVE_ENTRY_ACL_READ

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 specied by the name field.
ARCHIVE_ENTRY_ACL_GROUP_OBJ	The group who owns the file.
ARCHIVE_ENTRY_ACL_MASK	The maximum permissions to be obtained via group permissions.
ARCHIVE_ENTRY_ACL_OTHER	Any principal who doesn't have a user or group entry.

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 controlls the initial access ACL for newly created directory entries.

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.

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 and ARCHIVE_ENTRY_ACL_TYPE_DEFAULT. If ARCHIVE_ENTRY_ACL_TYPE_ACCESS is included and at least one extended ACL entry is found, the three non-extened ACLs are added.

archive_entry_acl_next() and **archive_entry_acl_next_w**() 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**() or **archive_entry_acl_next_w**(). The function returns either 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_text_w() converts the ACL entries for the given type mask into a wide string. In addition to the normal type flags, ARCHIVE_ENTRY_ACL_STYLE_EXTRA_ID and ARCHIVE_ENTRY_ACL_STYLE_MARK_DEFAULT can be specified to further customize the result. The returned long string is valid until the next call to **archive_entry_acl_clear**(), **archive_entry_acl_add_entry**(), **archive_entry_acl_add_entry_w**() or **archive_entry_acl_text_w**().

RETURN VALUES

archive_entry_acl_count() and **archive_entry_acl_reset**() returns the number of ACL entries that match the given type mask. 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_next() and **archive_entry_acl_next_w**() return 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_text_w() returns a wide string representation of the ACL entrise matching the given type mask. The returned long string is valid until the next call to **archive_entry_acl_clear**(), **archive_entry_acl_add_entry**(), **archive_entry_acl_add_entry_w**() or **archive_entry_acl_text_w**().

SEE ALSO

archive(3), archive_entry(3)

BUGS

ARCHIVE_ENTRY_ACL_STYLE_EXTRA_ID and ARCHIVE_ENTRY_ACL_STYLE_MARK_DEFAULT are not documented.