**NAME**

ar – archive file format

**SYNOPSIS**

**#include <ar.h>**

**DESCRIPTION**

The archive command **ar** is used to combine several files into one. Archives are used mainly as libraries to be searched by the link editor **ld**.

Each archive begins with the archive magic string.

```
#define  ARMAG   "!<arch>\n"        /* magic string */
#define  SARMAG  8                  /* length of magic string */
```

Following the archive magic string are the archive file members. Each file member is preceded by a file member header which is of the following format:

```
#define  ARFMAG   "`\n"        /* header trailer string */

struct  ar_hdr                 /* file member header */
{
   char   ar_name[16];         /* '/' terminated file member name */
   char   ar_date[12];         /* file member date */
   char   ar_uid[6];           /* file member user identification */
   char   ar_gid[6];           /* file member group identification */
   char   ar_mode[8];          /* file member mode (octal) */
   char   ar_size[10];         /* file member size */
   char   ar_fmag[2];          /* header trailer string */
};
```

All information in the file member headers is in printable ASCII. The numeric information contained in the headers is stored as decimal numbers (except for *ar_mode* which is in octal). Thus, if the archive contains printable files, the archive itself is printable.

If the file member name fits, the *ar_name* field contains the name directly, and is terminated by a slash (/) and padded with blanks on the right. If the member's name does not fit, *ar_name* contains a slash (/) followed by a decimal representation of the name's offset in the archive string table described below.

The *ar_date* field is the modification date of the file at the time of its insertion into the archive. Common format archives can be moved from system to system as long as the portable archive command **ar** is used.

Each archive file member begins on an even byte boundary; a newline is inserted between files if necessary. Nevertheless, the size given reflects the actual size of the file exclusive of padding.

Notice there is no provision for empty areas in an archive file.

Each archive that contains object files (see **a.out**(4)) includes an archive symbol table. This symbol table is used by the link editor **ld** to determine which archive members must be loaded during the link edit process. The archive symbol table (if it exists) is always the first file in the archive (but is never listed) and is automatically created and/or updated by **ar**.

The archive symbol table has a zero length name (that is, **ar_name[0]** is **'/'**), **ar_name[1]==' '**, etc.). All ''words'' in this symbol table have four bytes, using the machine-independent encoding shown below. All machines use the encoding described here for the symbol table, even if the machine's ''natural'' byte order is different.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0x01020304** | 01 | 02 | 03 | 04 |

The contents of this file are as follows:

1. The number of symbols. Length: 4 bytes.
2. The array of offsets into the archive file. Length: 4 bytes ∗ ''the number of symbols''.
3. The name string table. Length: *ar_size* – 4 bytes ∗ (''the number of symbols'' + 1).

As an example, the following symbol table defines 4 symbols. The archive member at file offset 114 defines *name* and *object*. The archive member at file offset 426 defines and a second version of *name*.

**Example Symbol Table**

| Offset | +0 | +1 | +2 | +3 | |
|---|---|---|---|---|---|
| 0 | **4** | | | | 4 offset entries |
| 4 | **114** | | | | **name** |
| 8 | **114** | | | | **object** |
| 12 | **426** | | | | **function** |
| 16 | **426** | | | | **name** |
| 20 | **n** | **a** | **m** | **e** | |
| 24 | **\0** | **o** | **b** | **j** | |
| 28 | **e** | **c** | **t** | **\0** | |
| 32 | **f** | **u** | **n** | **c** | |
| 36 | **t** | **i** | **o** | **n** | |
| 40 | **\0** | **n** | **a** | **m** | |
| 44 | **e** | **\0** | | | |

The string table contains exactly as many null terminated strings as there are elements in the offsets array. Each offset from the array is associated with the corresponding name from the string table (in order). The names in the string table are all the defined global symbols found in the common object files in the archive. Each offset is the location of the archive header for the associated symbol.

If some archive member's name is more than 15 bytes long, a special archive member contains a table of file names, each followed by a slash and a new-line. This string table member, if present, will precede all "normal" archive members. The special archive symbol table is not a "normal" member, and must be first if it exists. The **ar_name** entry of the string table's member header holds a zero length name **ar_name[0]=='/'**, followed by one trailing slash (**ar_name[1]=='/'**), followed by blanks (**ar_name[2]==' '**, etc.). Offsets into the string table begin at zero. Example *ar_name* values for short and long file names appear below.

| Offset | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | **f** | **i** | **l** | **e** | **_** | **n** | **a** | **m** | **e** | **_** |
| 10 | **s** | **a** | **m** | **p** | **l** | **e** | **/** | **\n** | **l** | **o** |
| 20 | **n** | **g** | **e** | **r** | **f** | **i** | **l** | **e** | **n** | **a** |
| 30 | **m** | **e** | **x** | **a** | **m** | **p** | **l** | **e** | **/** | **\n** |

| Member Name | ar_name | |
|---|---|---|
| **short-name** | **short-name/** | Not in string table |
| **file_name_sample** | **/0** | Offset 0 in string table |
| **longerfilenamexample** | **/18** | Offset 18 in string table |

## SEE ALSO

**ar**(1), **ld**(1), **strip**(1), **a.out**(4)

## NOTES

**strip** will remove all archive symbol entries from the header. The archive symbol entries must be restored via the **–ts** options of the **ar** command before the archive can be used with the link editor **ld**.