7 Formats and Protocols

7.1 Introduction

This chapter describes file and data formats and protocols that are visible to applications or must be portable across POE-conformant systems. The system must provide the system programs and library routines for manipulating these file formats and implementing these protocols.

Some system file formats explicitly do not appear in this chapter.

7.2 Archive File

An archive file contains the contents of zero or more files, combined into one file. Its structure is shown in Figure 7−1.

Fixed-Length Header
Archive Body
Member Table (optional) Member Header Member Table
External Symbol Table (optional) Member Header External Symbol Table

Figure 7−1. Archive File Organization.

An archive file is composed of a fixed-length header followed by the archive body. The archive body consists of zero or more members that are either file members or free-space members. If the archive contains at least one file member, the archive body is followed by an index of the members in the archive body (a member table). If the archive body contains file members that include XCOFF files, the archive may be terminated with an external symbol table (optional).

7.2.1 The Fixed-Length Header

Each archive file begins with a fixed-length header that contains offsets to special archive members. A fixed-length header has the format shown in Figure 7−2. This structure is defined in the header file, **<ar.h>**.

```
struct fl_hdr               /* fixed - length header */
{
    char fl_magic[8];       /* Archive magic string */
    char fl_memoff[12];     /* Offset to member table */
    char fl_gstoff[12];     /* Offset to global symbol table */
    char fl_fstmoff[12];    /* Offset to first archive member */
    char fl_lstmoff[12];    /* Offset to last archive member */
    char fl_freeoff[12];    /* Offset to first member on free list */
};
```

Figure 7−2. Fixed-Length Header Format.

Field definitions:

fl_magic Identifies a file as an ar-format archive file and contains the eight ASCII characters <aiaff> followed by the ASCII new-line character. Note that this sequence of characters is not null-terminated.

Note: All of the following fields are fixed size (12 bytes), and each of their values is stored as the ASCII decimal representation of the offset left-justified with blank padding to the right. (Blank refers to the ASCII space character.) None are null-terminated.

fl_memoff Contains the offset to the member table as a byte offset relative to the beginning of the archive. For more information, see the Member Table section on page 7−7.

fl_gstoff Contains the offset to the external (global) symbol table as a byte offset relative to the beginning of the archive. If the archive has no external symbol table then this field will be set to 0 unless the archive contains no file members, in which case the value of this field is undefined. For more information, see the External Symbol Table section on page 7–8.

fl_fstmoff Contains the offset to the first file member as a byte offset relative to the beginning of the archive. If the archive file has no file members, this field will be set to 0. (First indicates the first member on the forward-linked list, not lowest offset in the archive file.) For more information, see the Member Type section on page 7–7.

fl_lstmoff Contains the offset to the last file member as a byte offset relative to the beginning of the archive. If the archive file has no file members then this field will be set to 0. (Last indicates last member on the forward-linked list, not highest offset in the archive file.) For more information, see the Member Type section on page 7–7.

fl_freeoff Contains the offset to the first member of the free space list in the archive body (see below) as a byte offset relative to the beginning of the archive. If the archive file has no free space list then this field will be set to 0. (First indicates the first member on the free-space list, not lowest offset in the archive file.)

7.2.2 Member Header

Each member of the archive (file members, free-space members, the member table, and the external symbol table) all start with a member header as described in Figure 7–3.

```
struct ar_hdr   /* Member header */
{
        char ar_size[12];        /* Member size */
        char ar_nxtmem[12];      /* Next member offset */
        char ar_prvmem[12];      /* Previous member offset */
        char ar_date[12];        /* Member date */
        char ar_uid[12];         /* Member user id (owner) */
        char ar_gid[12];         /* Member group id (group) */
        char ar_mode[12];        /* Member mode */
        char ar_namlen[4];       /* Member name length */
        union
        {
        char ar_name[2];         /* start of member name */
        char ar_fmag[2];         /* magic string */
        } _ar_name;
};
```

Figure 7–3. Member Header.

Unless otherwise noted, all of the following fields are fixed size (12 bytes), and each of their values is stored as the ASCII decimal representation of the offset left-justified with blank padding to the right. (Blank always refers to the ASCII space character.) They are not null-terminated unless otherwise noted.

ar_size Contains the length in bytes of the data associated with this member. The space occupied by the possible padding byte at the end of the member (see 7.2.3 Archive Body 7–6) is not included in the length.

ar_nxtmem Contains the offset to the next member (file member, free-space member, member table or external symbol table) as a byte offset relative to the beginning of the archive. If the current member is the last in the linked list, then this field will be set to 0.

ar_prvmem Contains the offset to the previous member (file member, free-space member, member table) as a byte offset relative to the beginning of the archive. If the current member is the first in the linked list, then this field will be set to 0.

ar_date Contains the time stamp associated with the member contents as the number of seconds elapsed since 00:00:00 GMT, January 1, 1970. For a file member this value is the time of last modification of the file. This field is set to 0 for free-space members, the member table, and the external symbol table.

ar_uid Contains the user identifier associated with the member contents. This field is set to 0 for free-space members, the member table, and the external symbol table.

ar_gid Contains the group identifier associated with the member contents. This field is set to 0 for free-space members, the member table, and the external symbol table.

ar_mode Contains the access modes associated with the member contents. Its value is stored as the ASCII octal representation of the value left-justified with blank padding to the right. This field is set to 0 for free-space members, the member table, and the external symbol table.

ar_namlen Contains the length in bytes of the name associated with the member contents. This field is fixed size (4 bytes) and the value is stored as the ASCII decimal representation of the value left-justified with blank padding to the right. This field is set to 0 for free-space members, the member table, and the external symbol table.

ar_name and **ar_fmag**
Contains the name associated with the member contents. This field is variable size (0 to 256 bytes) and the value is stored as the ASCII sequence of characters forming the name. If the name has an odd number of characters an ASCII null character will be appended to the name, padding it to a half-word boundary. The name plus possible padding will be followed by the two character sequence, ‘ (back quote) followed by a new-line character . These two characters represent the **ar_fmag** field. It has a zero length for free-space members, the member table, and the external symbol table.

Note: The **ar_name** field of the **ar_hdr** structure in **<ar.h>** is not a literal C definition of its implementation — member names are not limited to two characters.

7.2.3 Archive Body

The archive body consists of either file members or free-space members or both. File members can be found via both the forward and backward-linked lists. Free space members can be found via the free-space linked list.

The contents of a file member follow the member header. If a member body contains an odd number of bytes, an ASCII null character (ASCII value 0) pads the contents to a half-word boundary.

7.2.4 Archive Linked-List Structure

The archive file format uses a double-linked list within the archive file to order its members, so archive file members may not be sequentially ordered within the file.

7.2.5 Physical Order of Archive Members

The location of file members is indicated by the linked lists; thus, file members are not necessarily physically stored in the same order as they appear in the linked list. The member table member always follows the last archive body member. If present the external symbol table member is always the last physical member of the archive file.

7.2.5.1 Forward-Linked List

The forward-linked list starts at the absolute file byte offset that is indicated by **fl_fstmoff** in the fixed-length header. The archive member table and the optional archive external symbol table member are included in this linked list and are always the last elements of it. (If present, the external symbol table follows the member table in the linked list.) The linked list is formed using the **ar_nxtmem** fields in each archive member header, including the member header of the member table and the external symbol table, if present. Free-space members do not appear in this linked list. The field **ar_nxtmem** marks the end of the list when its value is 0. If **fl_fstmoff** is 0, there is no forward-linked list.

7.2.5.2 Backward-Linked List

The backward-linked list has three entry points, indicated by the fields **fl_lstmoff**, **fl_gstoff** and **fl_memoff**. If the external symbol table member exists, it starts the backward-linked list (**fl_gstoff** will be non-zero). This is followed by the member table and then the last file member.

If there is no external symbol table, the backward-linked list starts with the member table indicated by **fl_memoff**. The following member is the last file member, given that there is at least one file member in the archive. The fixed-length header field **fl_lstmem** indicates the last file member's offset in the archive body.

The backward-linked list may be accessed by any of these three entry points and is constructed using the **ar_prvmem** field in each archive member header (including the member header of the member table and the external symbol table). The end of the linked list is indicated by a 0 value for a **ar_prvmem** field. If both **fl_gstoff** and **fl_memoff** are 0, there is no backward-linked list. Free-space members do not appear in the backward-linked list.

7.2.5.3 Free-Space Linked List

Free-space members also use the fields **ar_prvmem** and **ar_nxtmem** to maintain an independent linked list. The start of the free-space linked list is indicated by **fl_freeoff** in the fixed-length header. (The first member of the free-space linked list is not necessarily the free space member having the lowest offset in the archive.) A value of 0 for **fl_freeoff** means there is no free-space linked list.

7.2.6 Member Types

A file member is an archive member that contains a copy of a file. A file member contains a member header followed by the member contents. The member header contains both information about the file (its name, size, owner, *et cetera*) as described under the member header as well as its place in the double-linked list of the archive.

Free space members may contain unused sections of disk space that are structured as file members except that all file−related information (owner, group, mode, *et cetera*) is set to 0 and has a zero-length name field. The indicated file size gives the number of bytes available. Free-space members are not included in the double-linked list but appear in the free-space linked list described above.

7.2.7 Member Table

A member table is always present if there is at least one file member, and it can be used to locate members of the archive quickly. The **fl_memoff** field in the fixed-length header contains the offset to the member table, which has a zero-length name. A member table contains the following information in the order given:

Number of file members
This field is a fixed size (12 bytes) and the value is stored as the ASCII decimal representation of the value left-justified with blank padding to the right. (Blank refers to the ASCII space character.) The field is not null-terminated.

Array of offsets into the archive file
There is one offset value for each file member in the archive and they are ordered in the same order as the forward-linked list. Each offset is of fixed size (12 bytes) and the value is stored as the ASCII decimal representation of the value left-justified with blank padding to the right. (Blank refers to the ASCII space character.)

Member name table
There is one name for each file member in the archive and they are ordered in the same order as the forward-linked list. Each name is null-terminated with no padding.

The member table and external symbol table are not include as entries in the member table. If the length of the member table is odd, it is padded to a half-word boundary with an ASCII null byte at the end of the member table.

7.2.8 External Symbol Table

If an archive file contains XCOFF file members that are not stripped, the archive file may contain an external symbol table. This table is used to locate members that define external symbols. The **fl_gstoff** field in the fixed-length header contains the offset to the external symbol table, which has a zero-length name. An external symbol table contains the following information:

The number of external symbols
This is four bytes long and can be accessed with the **sgetl** and **sputl** functions. The format of this value is a 32 bit 2's complement representation with the most-significant byte first and the least-significant byte last.

The array of offsets into the archive file
Each offset is four bytes long and can be accessed with the **sgetl** and **sputl** functions. The format of each value is a 32 bit 2's complement representation with the most-significant byte first and the least-significant byte last.

The external symbol name string table
There is one symbol name for every entry in the array of offsets into the archive file. Each symbol is terminated by an ASCII null byte (ASCII value 0) with no padding.

If the length of the external symbol table is odd, the entry is padded to a half-word boundary by adding an ASCII null to the end of the member.

7.3 Other Archive Formats

Other PowerOpen ABI-conforming systems' archives can be created by the **cpio** and **tar** commands (see the POE API for more information). These archives are commonly used as vehicles for collecting files for storage or transmission.

The format of the archives the commands create is included in the *ISO/IEC 9945−1:1990; POSIX.1 Standard*.