Network Working Group
Internet Draft: draft-faltstrom-macmime1-01
Expiration 4/94

Patrik Fältström
Dave Crocker
Erik E. Fair
February 14, 1994

MIME Encapsulation of Macintosh files - MacMIME

1. Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents at any timne. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress". Please check the `1idabstracts.txt` listing contained in the internet-drafts Shadow Directories on `nic.ddn.mil`, `nnsc.nsf.net`, `nic.nordu.net`, `ftp.nisc.sri.com`, or `munnari.oz.au` to learn the current status of any Internet Draft.

2. Abstract

This memo describes the format to use when sending Apple Macintosh files via MIME [BORE93]. The format is compatible with existing mechanisms for distributing Macintosh files, while allowing non-Macintosh systems access to data in standardized formats.

3. Introduction

Files on the Macintosh consists of two parts, called forks:

Data fork: The actual data included in the file. The Data fork is typically the only meaningful part of a Macintosh file on a non-Macintosh computer system. For example, if a Macintosh user wants to send a file of data to a user on an IBM-PC, she would only send the Data fork.

Resource fork: Contains a collection of arbitrary attribute/value pairs, including program segments, icon bitmaps, and parametric values.

Additional information regarding Macintosh files is stored by the Finder in a hidden file, called the “Desktop Database”.

Because of the complications in storing different parts of a Macintosh file in a non-Macintosh filesystem that only handles consecutive data in one part, it is common to convert the Macintosh file into some other format before transferring it over the network.

The two styles of use are [APPL90]:

AppleSingle: Apple’s standard format for encoding Macintosh files as one byte stream.
AppleDouble: Similar to AppleSingle except that the Data fork is separated from the Macintosh-specific parts by the AppleDouble encoding.

AppleDouble is the preferred format for a Macintosh file that is to be included in an Internet mail message, because it provides recipients with Macintosh computers the entire document, including Icons and other Macintosh specific information, while other users easily can extract the Data fork (the actual data) as it is separated from the AppleDouble encoding.

4. MIME format for Apple/Macintosh-specific file information

4a. APPLICATION/APPLEFILE

MIME type-name:	APPLICATION
MIME subtype name:	APPLEFILE
Required parameters:	none
Optional parameters:	NAME, which must be a "`value`" as defined in RFC-1521 [BORE93].
Encoding considerations:	The presence of binary data will typically require use of Content-Transfer-Encoding: BASE64
Security considerations:	See separate section in the document
Published specification:	Apple-single & Apple-double [APPL90]
Rationale:	Permits MIME-based transmission of data with Apple/Macintosh specific information, while allowing general access to non-specific user data.

4b. MULTIPART/APPLEDOUBLE

MIME type-name:	MULTIPART
MIME subtype name:	APPLEDOUBLE
Required parameters:	none
Optional parameters:	NAME, which must be a "`value`" as defined in RFC-1521 [BORE93].
Encoding considerations:	none
Security considerations:	See separate section in the document
Published specification:	Apple-single & Apple-double [APPL90]
Rationale:	Permits MIME-based transmission of data with Apple/Macintosh specific information, while allowing general access to non-specific user data.

4c. Detail specific to MIME-based usage

Macintosh documents do not always need to be sent in a special format. Those documents with well-known MIME types and non-existent or trivial resource forks can be sent as regular MIME body parts, without use of AppleSingle or AppleDouble.

Documents which lack a data fork should always be sent as AppleSingle.

All other documents should normally be sent as AppleDouble. This includes documents with non-trivial resource forks, and documents without corresponding well-known MIME types.

It may be valuable in some cases to allow the user to choose one format over another, either because he disagrees with the implementor's definition of "trivial" resource forks, or for reasons of his own.

5. AppleSingle

An AppleSingle, version 2 file, is sent as one consecutive stream of bytes. The format is described in [APPL90] with a brief summary in Appendix A. The one and only part of the file is sent in an **application/applefile** message.

The first four bytes of an AppleSingle header are, in hexadecimal: 00, 05, 16, 00.

The AppleSingle file is binary data. Hence, it may be necessary to perform a Content-Transfer-Encoding for transmission. The safest encoding is Base64, since it permits transfer over the most restricted channels.

Even though an AppleSingle file includes the original Macintosh filename, it is recommended that a name parameter be included on the Content-Type header to give the recipient a hint as to what file is attached. The value of the name parameter must be a “value” as defined by RFC-1521 [BORE93]. Note that this restricts the value to seven-bit US-ASCII characters.

5a. AppleSingle example

```
Content-Type: application/applefile; name="Computers-1/2-93"

    [The AppleSingle file goes here]
```

6. AppleDouble

An AppleDouble, version 2, file is divided in two parts:

Header:	including the Macintosh resource fork and desktop information and
Data fork:	containing the Macintosh data fork.

The AppleDouble format is described in [APPL90] with a brief summary in Appendix B.

The AppleDouble file itself is sent as a **multipart/appledouble** MIME body-part, which may have only two sub-parts. The header is sent as **application/applefile** and the data fork as whatever best describes it. For example, is the data for is actually a GIF image, it should be sent as **image/gif**. If no appropriate Content-Type has been registered for the data type, it should be sent as an **application/octet-stream**.

The first four bytes of an AppleDouble header are, in hexadecimal: 00, 05, 16, 07.

The AppleDouble header is binary data. Hence, it may be necessary to perform a Content-Transfer-Encoding for transmission. The safest encoding is Base64, since it permits transfer over the most restrictive channels.

Even though an AppleDouble file includes the original Macintosh filename, it is recommended that a name parameter be included on the Content-Type header of both the header and data parts of the AppleDouble file to give the recipient a hint as to what file is attached. The value of the name parameter must be a “value” as defined by RFC-1521 [BORE93]. Note that this restricts the value to seven-bit US-ASCII characters.

6a. AppleDouble example

```
Content-Type: multipart/appledouble; boundary=mac-part

--mac-part
Content-Type: application/applefile; name="My-new-car"

    [The AppleDouble header goes here]

--mac-part
Content-Type: image/gif;

    [The data fork goes here]

--mac-part--
```

7. References

BORE93 Borenstein N., and N. Freed, MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC 1521, Bellcore, Innosoft, September 1993.

APPL90 AppleSingle/AppleDouble Formats for Foreign Files Developer's Note, Apple Computer, Inc., 1990

8. Security Considerations

To the extent that **application/applefile** facilitates the transmission of operating-system sensitive data, it may open a door for easier relaxation of security rules than is intended either by the sender of the administrator of the sender's system.

9. Acknowledgements

Thanks to all of the people on the ietf-822 list who have provided much meaningful input for this document. Some of them must though be remembered by name, because they have almost crushed my mailbox with a very nice and interesting debate:

> Johan Berglund, Steve Dorner, David Gelhar, David Herron, Lee Jones, Raymond Lau, Jamey Maze, John B. Melby, Jan Michael Rynning, Rens Troost and Peter Svanberg

10. Authors Addresses

Patrik Fältström
Department of Numerical Analysis and Computing Science
Royal Institute of Technology
S-100 44 Stockholm
Sweden

Email: paf@nada.kth.se

Dave Crocker
675 Spruce. Dr.
Sunnyvale, CA 94086
USA

Email: dcrocker@mordor.stanford.edu

Erik E. Fair
Engineering Computer Operations
Apple Computer Inc.

Email: fair@apple.com

Appendix A. The AppleSingle format

In the AppleSingle format, a file's contents and attributes are stored in a single file in the foreign file system. For example, both forks of a Macintosh file, the Finder information, and an associated comment are arranged in a single file with a simple structure.

An AppleSingle file consists of a header followed by one or more data entries. The header consists of several fixed fields and a list of entry descriptors, each pointing to a data entry. Each entry is optional and may or may not appear in the file.

AppleSingle file header:

Field	Length
Magic number	4 bytes
Version number	4 bytes
Filler	16 bytes
Number of entries	2 bytes

Entry descriptor for each entry:

Entry ID	4 bytes
Offset	4 bytes
Length	4 bytes

Byte ordering in the file fields follows MC68000 conventions, most significant byte first. The fields in the header file follow the conventions described in the following sections.

Magic number

This field, modelled after the UNIX magic number feature, specifies the file's format. Apple has defined the magic number for the AppleSingle format as $00051600 or 0x00051600.

Version number

This field denotes the version of AppleSingle format in the event the format evolves (more fields may be added to the header). The version described in this note is version $00020000 or 0x00020000.

Filler

This field is all zeros ($00 or 0x00).

Number of entries

This field specifies how many different entries are included in the file. It is an unsigned 16-bit number. If the number of entries is any number other than 0, then that number of entry descriptors immediately follows the number of entries field.

Entry descriptors

The entry descriptor is made up of the following three fields:

Entry ID:	an unsigned 32-bit number, defines what the entry is. Entry IDs range from 1 to $FFFFFFFF. Entry ID 0 is invalid.
Offset:	an unsigned 32-bit number, shows the offset from the beginning of the file to the beginning of the entry's data.
Length:	an unsigned 32-bit number, shows the length of the data in bytes. The length can be 0.

Predefined entry ID's

Apple has defined a set of entry IDs and their values as follows:

Data Fork	1	Data fork
Resource Fork	2	Resource fork
Real Name	3	File's name as created on home file system
Comment	4	Standard Macintosh comment
Icon, B&W	5	Standard Macintosh black and white icon
Icon, Colour	6	Macintosh colour icon
File Dates Info	8	File creation date, modification date, and so on
Finder Info	9	Standard Macintosh Finder information
Macintosh File Info	10	Macintosh file information, attributes and so on
ProDOS File Info	11	ProDOS file information, attributes and so on
MS-DOS File Info	12	MS-DOS file information, attributes and so on
Short Name	13	AFP short name
AFP File Info	14	AFP file, information, attributes and so on
Directory ID	15	AFP directory ID

Apple reserves the range of entry IDs from 1 to $7FFFFFFF. The rest of the range is available for applications to define their own entries. Apple does not arbitrate the use of the rest of the range.

Appendix B. The AppleDouble format

The AppleDouble format uses two files to store data, resources and attributes. The AppleDouble Data file contains the data fork and the AppleDouble Header file contains the resource fork.

The AppleDouble Data file contains the standard Macintosh data fork with no additional header. The AppleDouble Header file has exactly the same format as the AppleSingle file, except that it does not contain a Data fork entry. The magic number in the AppleDouble Header file differs from the magic number in the AppleSingle Header file so that an application can tell whether it needs to look in another file for the data fork. The magic number for the AppleDouble format is $00051607 or 0x00051607.

The entries in the AppleDouble Header file can appear in any order; however, since the resource fork is the entry that is most commonly extended (after the data fork), Apple recommends that the resource fork entry to be placed last in the file. The data fork is easily extended because it resides by itself in the AppleDouble Data file.

Appendix C. applefile.h

This is an example of a header file for the language C which can be used when parsing the data in either an AppleSingle file or AppleDouble header.

The file is written by Lee Jones. Distribution is unlimited.

```
/* applefile.h - Data structures used by AppleSingle/AppleDouble
 * file format
 *
 * Written by Lee Jones, 22-Oct-1993
 *
 * For definitive information, see "AppleSingle/AppleDouble
 * Formats for Foreign Files Developer's Note"; Apple Computer
 * Inc.; (c) 1990.
 *
 * Other details were added from:
 *   Inside Macintosh [old version], volumes II to VI,
 *   Apple include files supplied with Think C 5.0.1,
 *   Microsoft MS-DOS Programmer's Reference, version 5, and
 *   Microsoft C 6.00a's dos.h include file.
 *
 * I don't have ProDOS or AFP Server documentation so related
 * entries may be a bit skimpy.
 *
 * Edit history:
 *
 * when       who  why
 * ---------  ---  ------------------------------------------
 * 22-Oct-93  LMJ  Pull together from Inside Macintosh,
 *                 Developer's Note, etc
 * 26-Oct-93  LMJ  Finish writing first version and list
 *                 references
 * 06-Feb-94  EEF  Very minor cleanup
 */

/* Following items define machine specific size (for porting). */

typedef char            xchar8;         /* 8-bit field */
typedef char            schar8;         /* signed 8-bit field */
```

```
typedef unsigned char   uchar8;         /* unsigned 8-bit field */
typedef short           xint16;         /* 16-bit field */
typedef unsigned short  uint16;         /* unsigned 16-bit field */
typedef long            xint32;         /* 32-bit field */
typedef long            sint32;         /* signed 32-bit field */
typedef unsigned long   uint32;         /* unsigned 32-bit field */

/* REMINDER: the Motorola 680x0 is a big-endian architecture! */

typedef uint32 OSType;                  /* 32 bit field */

/* In the QuickDraw coordinate plane, each coordinate is
 * -32767..32767. Each point is at the intersection of a
 * horizontal grid line and a vertical grid line.  Horizontal
 * coordinates increase from left to right. Vertical
 * coordinates increase from top to bottom. This is the way
 * both a TV screen and page of English text are scanned:
 * from top left to bottom right.
 */

struct Point /* spot in QuickDraw 2-D grid */
{
    xint16 v; /* vertical coordinate */
    xint16 h; /* horizontal coordinate */
}; /* Point */

typedef struct Point Point;

/* See older Inside Macintosh, Volume II page 84 or Volume IV
 * page 104.
 */

struct FInfo /* Finder information */
{
    OSType fdType; /* File type, 4 ASCII chars */
    OSType fdCreator; /* File's creator, 4 ASCII chars */
    uint16 fdFlags; /* Finder flag bits */
    Point  fdLocation; /* file's location in folder */
    xint16 fdFldr; /* file 's folder (aka window) */
}; /* FInfo */
```

```
typedef struct FInfo FInfo;

/*
 * Masks for finder flag bits (field fdFlags in struct
 * FInfo).
 */

#define F_fOnDesk        0x0001 /* file is on desktop (HFS only) */
#define F_maskColor      0x000E /* color coding (3 bits) */
/*                       0x0010 /* reserved (System 7) */
#define F_fSwitchLaunch 0x0020 /* reserved (System 7) */
#define F_fShared        0x0040 /* appl available to multiple users */
#define F_fNoINITs       0x0080 /* file contains no INIT resources */
#define F_fBeenInited    0x0100 /* Finder has loaded bundle res. */
/*                       0x0200  /* reserved (System 7) */
#define F_fCustomIcom    0x0400 /* file contains custom icon */
#define F_fStationary    0x0800 /* file is a stationary pad */
#define F_fNameLocked    0x1000 /* file can't be renamed by Finder */
#define F_fHasBundle     0x2000 /* file has a bundle */
#define F_fInvisible     0x4000 /* file's icon is invisible */
#define F_fAlias         0x8000 /* file is an alias file (System 7) */

/* See older Inside Macintosh, Volume IV, page 105.
 */

struct FXInfo /* Extended finder information */
{
    xint16 fdIconID; /* icon ID number */
    xint16 fdUnused[3]; /* spare */
    schar8 fdScript; /* scrip flag and code */
    schar8 fdXFlags; /* reserved */
    xint16 fdComment; /* comment ID number */
    xint32 fdPutAway; /* home directory ID */
}; /* FXInfo */

typedef struct FXInfo FXInfo;

/* Pieces used by AppleSingle & AppleDouble (defined later). */
```

```
struct ASHeader /* header portion of AppleSingle */
{
            /* AppleSingle = 0x00051600; AppleDouble = 0x00051607 */
    uint32 magicNum; /* internal file type tag */
    uint32 versionNum; /* format version: 2 = 0x00020000 */
    uchar8 filler[16]; /* filler, currently all bits 0 */
    uint16 numEntries; /* number of entries which follow */
}; /* ASHeader */

typedef struct ASHeader ASHeader;

struct ASEntry /* one AppleSingle entry descriptor */
{
    uint32 entryID; /* entry type: see list, 0 invalid */
    uint32 entryOffset; /* offset, in octets, from beginning */
                                /* of file to this entry's data */
    uint32 entryLength; /* length of data in octets */
}; /* ASEntry */

typedef struct ASEntry ASEntry;

/* Apple reserves the range of entry IDs from 1 to 0x7FFFFFFF.
 * Entry ID 0 is invalid.  The rest of the range is available
 * for applications to define their own entry types.  "Apple does
 * not arbitrate the use of the rest of the range."
 */

#define AS_DATA         1 /* data fork */
#define AS_RESOURCE     2 /* resource fork */
#define AS_REALNAME     3 /* File's name on home file system */
#define AS_COMMENT      4 /* standard Mac comment */
#define AS_ICONBW       5 /* Mac black & white icon */
#define AS_ICONCOLOR    6 /* Mac color icon */
        /*              7       /* not used */
#define AS_FILEDATES    8 /* file dates; create, modify, etc */
#define AS_FINDERINFO   9 /* Mac Finder info & extended info */
#define AS_MACINFO      10 /* Mac file info, attributes, etc */
#define AS_PRODOSINFO   11 /* Pro-DOS file info, attrib., etc */
#define AS_MSDOSINFO    12 /* MS-DOS file info, attributes, etc */
#define AS_AFPNAME      13 /* Short name on AFP server */
```

```
#define AS_AFPINFO      14 /* AFP file info, attrib., etc */
#define AS_AFPDIRID     15 /* AFP directory ID */

/* matrix of entry types and their usage:
 *
 *                   Macintosh    Pro-DOS    MS-DOS    AFP server
 *                   ---------    -------    ------    ----------
 *  1   AS_DATA         xxx          xxx       xxx         xxx
 *  2   AS_RESOURCE     xxx          xxx
 *  3   AS_REALNAME     xxx          xxx       xxx         xxx
 *
 *  4   AS_COMMENT      xxx
 *  5   AS_ICONBW       xxx
 *  6   AS_ICONCOLOR    xxx
 *
 *  8   AS_FILEDATES    xxx          xxx       xxx         xxx
 *  9   AS_FINDERINFO   xxx
 * 10   AS_MACINFO      xxx
 *
 * 11   AS_PRODOSINFO                xxx
 * 12   AS_MSDOSINFO                           xxx
 *
 * 13   AS_AFPNAME                                         xxx
 * 14   AS_AFPINFO                                         xxx
 * 15   AS_AFPDIRID                                        xxx
 */

/* entry ID 1, data fork of file - arbitrary length octet string */

/* entry ID 2, resource fork - arbitrary length opaque octet string;
 *              as created and managed by Mac O.S. resoure manager
 */

/* entry ID 3, file's name as created on home file system - arbitrary
 *              length octet string; usually short, printable ASCII
 */

/* entry ID 4, standard Macintosh comment - arbitrary length octet
 *              string; printable ASCII, claimed 200 chars or less
 */
```

```
/* This is probably a simple duplicate of the 128 octet bitmap
 * stored as the 'ICON' resource or the icon element from an 'ICN#'
 * resource.
 */

struct ASIconBW /* entry ID 5, standard Mac black and white icon */
{
    uint32 bitrow[32]; /* 32 rows of 32 1-bit pixels */
}; /* ASIconBW */

typedef struct ASIconBW ASIconBW;

/* entry ID 6, "standard" Macintosh color icon - several competing
 *             color icons are defined.  Given the copyright dates
 * of the Inside Macintosh volumes, the 'cicn' resource predominated
 * when the AppleSingle Developer's Note was written (most probable
 * candidate).  See Inside Macintosh, Volume V, pages 64 & 80-81 for
 * a description of 'cicn' resources.
 *
 * With System 7, Apple introduced icon families.  They consist of:
 *      large (32x32) B&W icon, 1-bit/pixel,    type 'ICN#',
 *      small (16x16) B&W icon, 1-bit/pixel,    type 'ics#',
 *      large (32x32) color icon, 4-bits/pixel, type 'icl4',
 *      small (16x16) color icon, 4-bits/pixel, type 'ics4',
 *      large (32x32) color icon, 8-bits/pixel, type 'icl8', and
 *      small (16x16) color icon, 8-bits/pixel, type 'ics8'.
 * If entry ID 6 is one of these, take your pick.  See Inside
 * Macintosh, Volume VI, pages 2-18 to 2-22 and 9-9 to 9-13, for
 * descriptions.
 */

/* entry ID 7, not used */

/* Times are stored as a "signed number of seconds before of after
 * 12:00 a.m. (midnight), January 1, 2000 Greenwich Mean Time (GMT).
 * Applications must convert to their native date and time
 * conventions." Any unknown entries are set to 0x80000000
 * (earliest reasonable time).
 */
```

```
struct ASFileDates      /* entry ID 8, file dates info */
{
    sint32 create; /* file creation date/time */
    sint32 modify; /* last modification date/time */
    sint32 backup; /* last backup date/time */
    sint32 access; /* last access date/time */
}; /* ASFileDates */

typedef struct ASFileDates ASFileDates;

/* See older Inside Macintosh, Volume II, page 115 for
 * PBGetFileInfo(), and Volume IV, page 155, for PBGetCatInfo().
 */

/* entry ID 9, Macintosh Finder info & extended info */
struct ASFinderInfo
{
    FInfo ioFlFndrInfo; /* PBGetFileInfo() or PBGetCatInfo() */
    FXInfo ioFlXFndrInfo; /* PBGetCatInfo() (HFS only) */
}; /* ASFinderInfo */

typedef struct ASFinderInfo ASFinderInfo;

struct ASMacInfo        /* entry ID 10, Macintosh file information */
{
    uchar8 filler[3]; /* filler, currently all bits 0 */
    uchar8 ioFlAttrib; /* PBGetFileInfo() or PBGetCatInfo() */
}; /* ASMacInfo */

typedef struct ASMacInfo ASMacInfo;

#define AS_PROTECTED    0x0002 /* protected bit */
#define AS_LOCKED       0x0001 /* locked bit */

/* NOTE: ProDOS-16 and GS/OS use entire fields.  ProDOS-8 uses low
 * order half of each item (low byte in access & filetype, low word
 * in auxtype); remainder of each field should be zero filled.
 */
```

```
struct ASProdosInfo     /* entry ID 11, ProDOS file information */
{
    uint16 access; /* access word */
    uint16 filetype; /* file type of original file */
    uint32 auxtype; /* auxiliary type of the orig file */
}; /* ASProDosInfo */

typedef struct ASProdosInfo ASProdosInfo;

/* MS-DOS file attributes occupy 1 octet; since the Developer Note
 * is unspecific, I've placed them in the low order portion of the
 * field (based on example of other ASMacInfo & ASProdosInfo).
 */

struct ASMsdosInfo      /* entry ID 12, MS-DOS file information */
{
    uchar8 filler; /* filler, currently all bits 0 */
    uchar8 attr; /* _dos_getfileattr(), MS-DOS */
                                 /* interrupt 21h function 4300h */
}; /* ASMsdosInfo */

typedef struct ASMsdosInfo ASMsdosInfo;

#define AS_DOS_NORMAL   0x00 /* normal file (all bits clear) */
#define AS_DOS_READONLY 0x01 /* file is read-only */
#define AS_DOS_HIDDEN   0x02 /* hidden file (not shown by DIR) */
#define AS_DOS_SYSTEM   0x04 /* system file (not shown by DIR) */
#define AS_DOS_VOLID    0x08 /* volume label (only in root dir) */
#define AS_DOS_SUBDIR   0x10 /* file is a subdirectory */
#define AS_DOS_ARCHIVE  0x20 /* new or modified (needs backup) */

/* entry ID 13, short file name on AFP server - arbitrary length
 *              octet string; usualy printable ASCII starting with
 *              '!' (0x21)
 */

struct ASAfpInfo   /* entry ID 12, AFP server file information */
{
    uchar8 filler[3]; /* filler, currently all bits 0 */
    uchar8 attr; /* file attributes */
```

```
}; /* ASAfpInfo */

typedef struct ASAfpInfo ASAfpInfo;

#define AS_AFP_Invisible    0x01 /* file is invisible */
#define AS_AFP_MultiUser    0x02 /* simultaneous access allowed */
#define AS_AFP_System       0x04 /* system file */
#define AS_AFP_BackupNeeded 0x40 /* new or modified (needs backup) */

struct ASAfpDirId       /* entry ID 15, AFP server directory ID */
{
    uint32 dirid; /* file's directory ID on AFP server */
}; /* ASAfpDirId */

typedef struct ASAfpDirId ASAfpDirId;

/*
 * The format of an AppleSingle/AppleDouble header
 */
struct AppleSingle /* format of disk file */
{
    ASHeader header; /* AppleSingle header part */
    ASEntry  entry[1]; /* array of entry descriptors */
/* uchar8  filedata[];          /* followed by rest of file */
}; /* AppleSingle */

typedef struct AppleSingle AppleSingle;

/*
 * FINAL REMINDER: the Motorola 680x0 is a big-endian architecture!
 */

/* End of applefile.h */
```