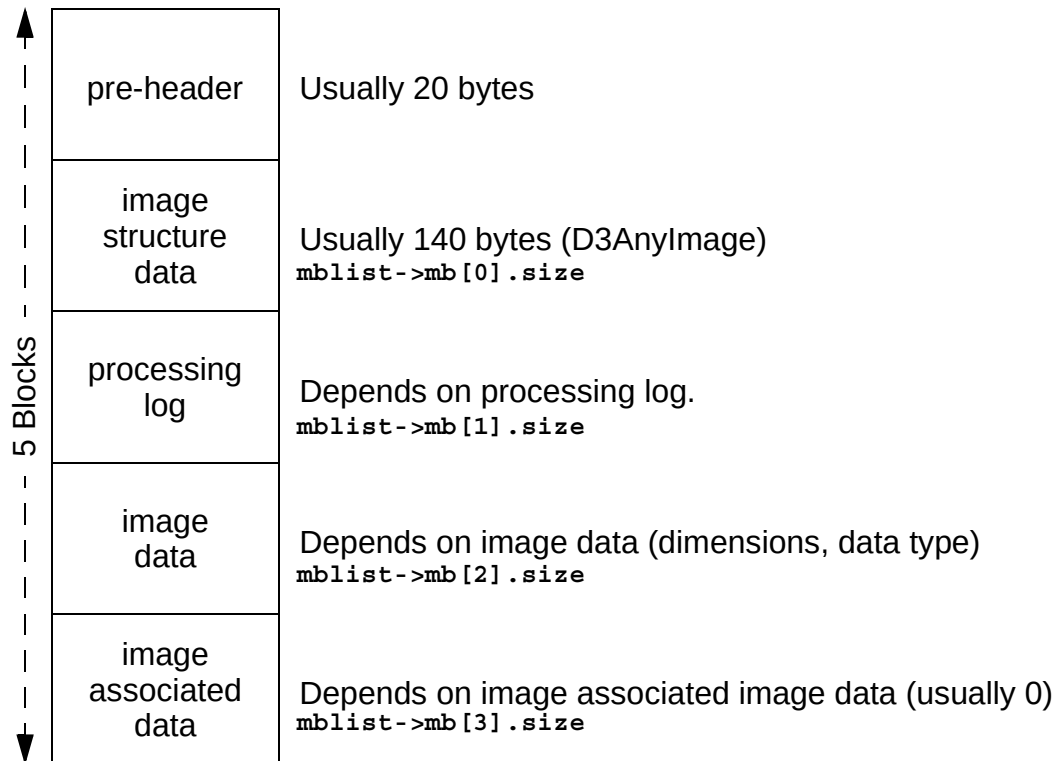
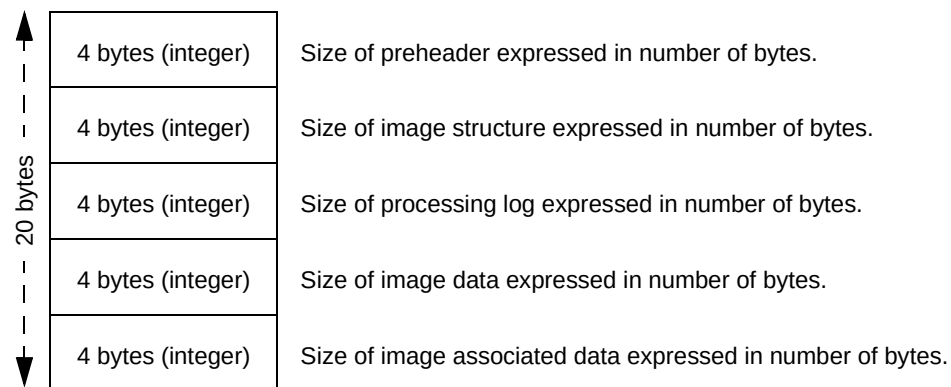


Scanco AIM file format

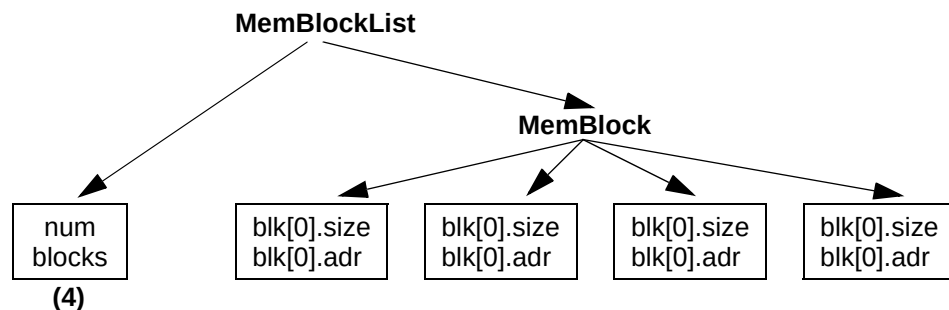
File structure



0 Pre-header structure



Memory block



1 Image Structure Data: D3AnyImage

```
typedef struct {
    char        version;           /* Version 2.0: 020          */
    char        *proc_log;
    void        *dat;
    int         id;
    int         ref;
    int         type;              /* type of data              */
    D3int       pos;              /* in pixels (absolut)       */
    D3int       dim;              /* in pixels                 */
    D3int       off;              /* in pixels                 */
    D3int       supdim;           /* in subdims ( = 0 -> no sup image) */
    D3int       suppos;           /* in pixels (relative)      */
    D3int       subdim;           /* in pixels                 */
    D3int       testoff;          /* in pixels                 */
    D3float     el_size_mm;
    D1AnyData   assoc;           /* associated data           */
} D3AnyImage, D3aim, D3AnyImage020;
```

2 Processing Log

A log of all the procedures applied to the aim.

3 Image Data

Usually type D1Tchar (1 byte) or D1Tshort (2 bytes). In the case of D1Tshort, this is always stored in little endian format (see notes on big and little endian below) in the AIM file regardless of the platform used to write the aim. When reading or writing an AIM file on a big endian machine (SUN or SGI), care must be taken to swap the bytes. See the procedures named AimpackRmSwapByteOrder and AimpackRmConvertFloatVMS2Sun.

4 Image Associated Data

This data block is rarely used, but offers potential for storing data that can be associated with the image data in block[3].

Notes on Big Endian/Little Endian

The ordering of bytes differs between the "big endian" and "little endian" platforms. These colloquial terms are used to describe byte ordering for IBM mainframes (big endian) and for Intel-based platforms (little endian).

Big Endian platforms: AIX, HP-UX, IBM mainframe, Macintosh, and Solaris.

Little Endian platforms: AXP/VMS, Digital UNIX, Intel ABI, OS/2, VAX/VMS, and Windows.

On **big endian** platforms, the value 1 is stored in binary and is represented in this example as hexadecimal notation. One byte is stored as 01, two bytes as 00 01, and four bytes as 00 00 00 01. On **little endian** platforms, the value 1 is stored in one byte as 01 (the same as big endian), in two bytes as 01 00, and in four bytes as 01 00 00 00.