NAME

a.out − assembler and link editor output

DESCRIPTION

A.out is the output file of the assembler *as* and the link editor *ld*. Both programs make **a.out** executable if there were no errors and no unresolved external references.

This file has four sections: a header, the program and data text, a symbol table, and relocation bits (in that order). The last two may be empty if the program was loaded with the −s option of *ld* or if the symbols and relocation have been removed by *strip*.

The structure of the entry as given in the include file is:

```
/*      @(#)a.out.h     3.3             */
struct  exec {          /* a.out header */
        int             a_magic;        /* magic number */
        unsigned        a_text;         /* size of text segment */
        unsigned        a_data;         /* size of initialized data */
        unsigned        a_bss;          /* size of unitialized data */
        unsigned        a_syms;         /* size of symbol table */
        unsigned        a_entry;        /* entry point */
        char                            a_unused;       /* not used */
        char                            a_hitext;       /* text high bits */
        char                            a_flag;         /* relocation info stripped */
        char                            a_stamp;        /* System environment stamp */
};

/* macro to calculate text size of big files */
#define TSIZE(x) x.a_text + ((long)x.a_hitext << 16)

#define     A_MAGIC1    0407        /* normal */
#define     A_MAGIC2    0410        /* read-only text */
#define     A_MAGIC3    0411        /* separated I&D */
#define     A_MAGIC4    0405        /* overlay */
#define     A_MAGIC0    0401                    /* ldp (UNIX/RT) */

/* ***** in invocation of BADMAG macro, argument should not be a function.***/
#define     BADMAG(X) X.a_magic!=A_MAGIC1 && X.a_magic!=A_MAGIC2 && X.a_magic!=A_MAGIC3 &&

struct      nlist {     /* symbol table entry */
        char        n_name[8];      /* symbol name */
        char        n_type;         /* type flag */
        char                        n_loc;                  /* text area location */
        unsigned    n_value;        /* value */
};

                        /* values for type flag */
#define     N_UNDF      0           /* undefined */
#define     N_ABS       01          /* absolute */
#define     N_TEXT      02          /* text symbol */
#define     N_DATA      03          /* data symbol */
#define     N_BSS       04          /* bss symbol */
#define     N_TYPE      037
#define     N_REG       024         /* register name */
#define     N_FN        037         /* file name symbol */
#define     N_EXT       040         /* external bit, or'ed in */
#define     FORMAT      "%06o"      /* to print a value */

                        /* values for loc flag */
#define N_SWSP0 1       /* text switchable space 0 */
#define N_SWSP1 2       /* text switchable space 1 */
#define N_SWSP2 3       /* text switchable space 2 */
#define N_SWSP3 4       /* text switchable space 3 */
```

The sizes of each segment are in bytes but are even. The size of the header is not included in any of the other sizes.

When a file produced by the assembler or loader is loaded into core for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized bss, the latter being initialized to all 0's), and a stack. The text segment begins at 0 in the core image; the header is not loaded. If the magic number (word 0) is 407, it indicates that the text segment is not to be write-protected and shared, so the data segment is immediately contiguous with the text segment. If the magic number is 410, the data segment begins at the first 0 mod 8K byte boundary following the text segment, and the text segment is not writable by the program; if other processes are executing the same file, they will share the text segment. If the magic number is 411, the text segment is again pure, write-protected, and shared, and moreover instruction and data space are separated; the text and data segment both begin at location 0. See the 11/70 handbook for restrictions which apply to this situation. The magic number 405 indicates an overlay file. On execution, the current processes' text segment is replaced with the text segment from this module.

The stack will occupy the highest possible locations in the core image: from 177776(8) and growing downward. The stack is automatically extended as required. The data segment is only extended as requested by the *break*(2) system call.

The start of the text segment in the file is 20(8); the start of the data segment is $20+S_t$ (the size of the text) the start of the relocation information is $20+S_t+S_d$; the start of the symbol table is $20+2(S_t+S_d)$ if the relocation information is present, $20+S_t+S_d$ if not.

The symbol table consists of 6-word entries. The first four words contain the ASCII name of the symbol, null-padded(n_name). The next byte is a flag indicating the type of symbol(n_type).

The next byte is a flag indicating the switchable text location for **UNIX** with switchable text areas.

The last word of a symbol table entry contains the value of the symbol.

If the symbol's type is undefined external, and the value field is non-zero, the symbol is interpreted by the loader *ld* as the name of a common region whose size is indicated by the value of the symbol.

The value of a word in the text or data portions which is not a reference to an undefined external symbol is exactly that value which will appear in core when the file is executed. If a word in the text or data portion involves a reference to an undefined external symbol, as indicated by the relocation bits for that word, then the value of the word as stored in the file is an offset from the associated external symbol. When the file is processed by the link editor and the external symbol becomes defined, the value of the symbol will be added into the word in the file.

If relocation information is present, it amounts to one word per word of program text or initialized data. There is no relocation information if the 'suppress relocation' flag in the header is on.

Bits 3-1 of a relocation word indicate the segment referred to by the text or data word associated with the relocation word:

00	indicates the reference is absolute
02	indicates the reference is to the text segment
04	indicates the reference is to initialized data
06	indicates the reference is to bss (uninitialized data)
10	indicates the reference is to an undefined external symbol.

Bit 0 of the relocation word indicates if *on* that the reference is relative to the pc (e.g. 'clr x'); if *off*, that the reference is to the actual symbol (e.g., 'clr *$x').

The remainder of the relocation word (bits 15-4) contains a symbol number in the case of external references, and is unused otherwise. The first symbol is numbered 0, the second 1, etc.

The system environment stamp (see *stamp*(1)) determines which of several possible interpretations the operating system will give to system calls from the executing process.

SEE ALSO

as(1), ld(1), nm(1), stamp(1), strip(1)