NAME

0l, 1l, 2l, 5l, 6l, 7l, 8l, kl, ql, vl – loaders

SYNOPSIS

```
2l [ option ... ] [ file ... ]
etc.
```

DESCRIPTION

These commands load the named files into executable files for the corresponding architectures; see 2c(1) for the correspondence between an architecture and the character (1, 2, etc.) that specifies it. The files should be object files or libraries (archives of object files) for the appropriate architecture. Also, a name like –lext represents the library libext.a in /$objtype/lib, where objtype is one of 68000, etc. as listed in 2c(1). The libraries must have tables of contents (see ar(1)).

In practice, –l options are rarely necessary as the header files for the libraries cause their archives to be included automatically in the load (see 2c(1)). For example, any program that includes header file libc.h causes the loader to search the C library /$objtype/lib/libc.a. Also, the loader creates an undefined symbol _main (or _mainp if profiling is enabled) to force loading of the startup linkage from the C library.

The order of search to resolve undefined symbols is to load all files and libraries mentioned explicitly on the command line, and then to resolve remaining symbols by searching in topological order libraries mentioned in header files included by files already loaded. When scanning such libraries, the algorithm is to scan each library repeatedly until no new undefined symbols are picked up, then to start on the next library. Thus if library A needs B which needs A again, it may be necessary to mention A explicitly so it will be read a second time.

The loader options are:

–l (As a bare option.) Suppress the default loading of the startup linkage and libraries specified by header files.

–o out Place output in file out. Default is O.out, where O is the first letter of the loader name.

–p Insert profiling code into the executable output; no special action is needed during compilation or assembly.

–s Strip the symbol tables from the output file.

–a Print the object code in assembly language, with addresses.

–v Print debugging output that annotates the activities of the load.

–M (Kl only) Generate instructions rather than calls to emulation routines for multiply and divide.

–Esymbol
The entry point for the binary is symbol (default _main; _mainp under –p).

–x [file]
Produce an export table in the executable. The optional file restricts the exported symbols to those listed in the file. See dynld(2).

–u [file]
Produce an export table, import table and a dynamic load section in the executable. The optional file restricts the imported symbols to those listed in the file. See dynld(2).

```
     -Hn    Executable header is type n. The meaning of the types
            is architecture-dependent; typically type 1 is Plan 9
            boot format and type 2 is the regular Plan 9 format,
            the default.  These are reversed on the MIPS.  The
            Next boot format is 3.  Type 4 in vl creates a MIPS
            executable for an SGI Unix system.

     -Tt    The text segment starts at address t.

     -Dd    The data segment starts at address d.

     -Rr    The text segment is rounded to a multiple of r (if r
            is nonzero).

     The numbers in the above options can begin with `0x' or `0'
     to change the default base from decimal to hexadecimal or
     octal.  The defaults for the values depend on the compiler
     and the header type.

     The loaded image has several symbols inserted by the loader:
     etext is the address of the end of the text segment; bdata
     is the address of the beginning of the data segment; edata
     is the address of the end of the data segment; and end is
     the address of the end of the bss segment, and of the pro-
     gram.

2L(1)                                                     2L(1)

FILES
     /$objtype/lib  for -llib arguments.

SOURCE
     /sys/src/cmd/2l etc.

SEE ALSO
     2c(1), 2a(1), ar(1), nm(1), db(1), prof(1)

     Rob Pike, ``How to Use the Plan 9 C Compiler''

BUGS
     The list of loaders given above is only partial, not all
     architectures are supported on all systems, some have been
     retired and some are provided by third parties.
```