# The vi program

Timothy Daly

June 15, 2007

# Contents


**1 Linking [6] 5**

1.1 ELF sections . . . 5
1.2 Symbols and Symbol Resolution . . . 6
1.3 Linking with Static Libraries . . . 6
1.4 Relocation . . . 7
1.5 Dynamic Linking: Shared Libraries . . . 8
1.6 Loading Shared Libraries from Applications . . . 9
1.7 Static Libraries vs. Shared Libraries . . . 9
1.8 Position Independent Code (or, Win32 DLLs versus ".SO") . . . 10
1.9 Dynamic Linking in Linux . . . 10
1.9.1 The ELF data structures . . . 10
1.9.2 Global offset Table (GOT) . . . 10
1.9.3 Procedure Linkage Table (PLT) . . . 11
1.9.4 Symbol Hashtable . . . 11
1.9.5 How things work . . . 11
1.9.6 A Walk through through the Linux lazy linking procedure 12
1.9.7 Peeking into the dynamic linker ld.so . . . 15
1.10 Windows Portable Executable File Format (PE) data structures 18
1.11 Exports section (.edata) . . . 18
1.12 Imports Section(.idata) . . . 18
1.13 How things work . . . 19
1.14 Delay loading in Windows . . . 19
1.15 A walk through through the Windows lazy linking procedure . . 19
1.16 Inside the Delay Load Helper . . . 20
1.17 Explicit Dynamic Linking . . . 22
1.18 Speeding things up – Static Linked Shared Libraries . . . 22
1.19 Prelinking in Linux . . . 23
1.20 Rebasing and binding in Windows . . . 24

**2 readelf -a ex 25**

2.1 ELF Header . . . 25
2.2 Section Headers . . . 25
2.3 Program Headers . . . 32
2.4 Section to Segment mapping . . . 32

2.5 Dynamic segment . . . . . 32
2.6 Relocation section '.rel.dyn' . . . . . 33
2.7 Relocation section '.rel.plt' . . . . . 33
2.8 unwind sections . . . . . 35
2.9 Symbol table '.dynsym' . . . . . 35
2.10 Symbol table '.symtab' . . . . . 37
2.11 Histogram . . . . . 63
2.12 Version symbols section '.gnu.version' . . . . . 63
2.13 Version needs section '.gnu.version_r' . . . . . 64
2.14 Procedure for fetching symbols . . . . . 64
2.15 The Entry Point . . . . . 66
2.16 export LD_DEBUG=all . . . . . 67

# Chapter 1

# Linking [6]

## 1.1 ELF sections

The ELF file has several sections. The most common ones are:

- .text, the machine code of the compiled program
- .rodata, the read-only data, such as the format strings in printf statements
- .data, initialized global variables
- .bss, unitialized global variables, BSS stands for block storage start, and this section actually occupies no space in the object file; it is merely a place holder.
- .symtab, a symbol table with information about functions and global variables defined and referenced in the program. This table does not contain any entries for local variables; those are maintained on the stack.
- .rel.text, a list of locations in the .text section that need to be modified when the linker combines this object file with other object files.
- .rel.data, relocation information for global variables referenced but not defined in the current module
- .debug, a debugging symbol table with entries for local and global variables. This section is present only if the compiler is invoked with a -g option
- .line, a mapping between line numbers in the original C source program and machine code instructions in the .text section This information is used by debugger programs.
- .strtab, a string table for the symbol tables in the .symtab and .debug sections.

## 1.2 Symbols and Symbol Resolution

Every relocatable object file has a symbol table and associated symbols. In the context of a linker, the following kinds of symbols are present:

- Global symbols defined by the module and referenced by other modules. All non-static functions and global variables fall in this category.
- Global symbols referenced by the input module but defined elsewhere. All functions and variables with extern declaration fall in this category.
- Local symbols defined and referenced exclusively by the input module. All static functions and static variables fall here.

The linker resolves symbol references by associating each reference with exactly one symbol definition from the symbol tables of its input relocatable object files. Resolution of local symbols to a module is straightforward, as a module cannot have multiple definitions of local symbols. Resolving references to global symbols is trickier, however. At compile time, the compiler exports each global symbol as either strong or weak. Functions and initialized global variables get strong weight, while global uninitialized variables are weak. Now, the linker resolves the symbols using the following rules:

- Multiple strong symbols are not allowed.
- Given a single strong symbol and multiple weak symbols, choose the strong symbol.
- Given multiple weak symbols, choose any of the weak symbols.

## 1.3 Linking with Static Libraries

A static library is a collection of concatenated object files of similar type. These libraries are stored on disk in an archive. An archive also contains some directory information that makes it faster to search for something. Each ELF archive starts with the magic eight character string !¡arch¿
n, where
n is a newline.

Static libraries are passed as arguments to compiler tools (linker), which copy only the object modules referenced by the program. On UNIX systems, libc.a contains all the C library functions, including printf and fopen, that are used by most of the programs.

During the process of symbol resolution using static libraries, linker scans the relocatable object files and archives from left to right as input on the command line. During this scan, linker maintains a set of O, relocatable object files that go into the executable; a set U, unresolved symbols; and a set of D, symbols defined in previous input modules. Initially, all three sets are empty.

- For each input argument on the command line, linker determines if input is an object file or an archive. If input is a relocatable object file, linker adds it to set O, updates U and D and proceeds to next input file.
- If input is an archive, it scans through the list of member modules that constitute the archive to match any unresolved symbols present in U. If some archive member defines any unresolved symbol that archive member is added to the list O, and U and D are updated per symbols found in the archive member. This process is iterated for all member object files.
- After all the input arguments are processed through the above two steps, if U is found to be not empty, linker prints an error report and terminates. Otherwise, it merges and relocates the object files in O to build the output executable file.

This also explains why static libraries are placed at the end of the linker command. Special care must be taken in cases of cyclic dependencies between libraries. Input libraries must be ordered so each symbol is referenced by a member of an archive and at least one definition of a symbol is followed by a reference to it on the command line. Also, if an unresolved symbol is defined in more than one static library modules, the definition is picked from the first library found in the command line.

## 1.4 Relocation

Once the linker has resolved all the symbols, each symbol reference has exactly one definition. At this point, linker starts the process of relocation, which involves the following two steps:

- Relocating sections and symbol definitions. Linker merges all the sections of the same type into a new single section. For example, linker merges all the .data sections of all the input relocatable object files into a single .data section for the final executable. A similar process is carried out for the .code section. The linker then assigns runtime memory addresses to new aggregate sections, to each section defined by the input module and also to each symbol. After the completion of this step, every instruction and global variable in the program has a unique loadtime address.
- Relocating symbol reference within sections. In this step, linker modifies every symbol reference in the code and data sections so they point to the correct loadtime addresses.

Whenever assembler encounters an unresolved symbol, it generates a relocation entry for that object and places it in the .relo.text/.relo.data sections. A relocation entry contains information about how to resolve the reference. A typical ELF relocation entry contains the following members:

- Offset, a section offset of the reference that needs to be relocated. For a relocatable file, this value is the byte offset from the beginning of the section to the storage unit affected by relocation.
- Symbol, a symbol the modified reference should point to. It is the symbol table index with respect to which the relocation must be made.
- Type, the relocation type, normally R_386_PC32, that signifies PC-relative addressing. R_386_32 signifies absolute addressing.

The linker iterates over all the relocation entries present in the relocatable object modules and relocates the unresolved symbols depending on the type. For R_386_PC32, the relocating address is calculated as S + A - P; for R_386_32 type, the address is calculated as S + A. In these calculations, S denotes the value of the symbol from the relocation entry, P denotes the section offset or address of the storage unit being relocated (computed using the value of offset from relocation entry) and A is the address needed to compute the value of the relocatable field.

## 1.5 Dynamic Linking: Shared Libraries

Static libraries above have some significant disadvantages; for example, consider standard functions such as printf and scanf. They are used almost by every application. Now, if a system is running 50-100 processes, each process has its own copy of executable code for printf and scanf. This takes up significant space in the memory. Shared libraries, on the other hand, address the disadvantages of static libraries. A shared library is an object module that can be loaded at run time at an arbitrary memory address, and it can be linked to by a program in memory. Shared libraries often are called as shared objects. On most UNIX systems they are denoted with a .so suffix; HP-UX uses a .sl suffix and Microsoft refer to them as DLLs (dynamic link libraries).

To build a shared object, the compiler driver is invoked with a special option:

```
gcc -shared -fPIC -o libfoo.so a.o b.o
```

The above command tells the compiler driver to generate a shared library, libfoo.so, comprised of the object modules a.o and b.o. The -fPIC option tells the compiler to generate position independent code (PIC).

Now, suppose the main object module is bar.o, which has dependencies on a.o and b.o. In this case, the linker is invoked with:

```
gcc bar.o ./libfoo.so
```

This command creates an executable file, a.out, in a form that can be linked to libfoo.so at load time. Here a.out does not contain the object modules a.o and b.o, which would have been included had we created a static library instead of a shared library. The executable simply contains some relocation and symbol table information that allow references to code and data in libfoo.so to be resolved

at run time. Thus, a.out here is a partially executable file that still has its dependency in libfoo.so. The executable also contains a .interp section that contains the name of the dynamic linker, which itself is a shared object on Linux systems (ld-linux.so). So, when the executable is loaded into memory, the loader passes control to the dynamic linker. The dynamic linker contains some start-up code that maps the shared libraries to the program's address space. It then does the following:

- relocates the text and data of libfoo.so into memory segment; and
- relocates any references in a.out to symbols defined by libfoo.so.

Finally, the dynamic linker passes control to the application. From this point on, location of shared object is fixed in the memory.

## 1.6 Loading Shared Libraries from Applications

Shared libraries can be loaded from applications even in the middle of their executions. An application can request a dynamic linker to load and link shared libraries, even without linking those shared libraries to the executable. Linux, Solaris and other systems provides a series of function calls that can be used to dynamically load a shared object. Linux provides system calls, such as dlopen, dlsym and dlclose, that can be used to load a shared object, to look up a symbol in that shared object and to close the shared object, respectively. On Windows, LoadLibrary and GetProcAddress functions replace dlopen and dlsym, respectively.

## 1.7 Static Libraries vs. Shared Libraries

A library is a collection of sub-programs which allow code to be shared and changed in a modular fashion. Executables and libraries make references to each other through a process known as linking which is done by a linker.

In the most basic sense, libraries can be divided into two categories: static libraries and shared libraries.

Static libraries are a collection of object files, and conventionally they end with a ".a" suffix in UNIX variants, and ".lib" in Windows. When a program is linked against a static library, the machine code from the object files for any external functions used by the program is copied from the library into the final executable.

In contrast to static libraries, with shared libraries the library code is not bound to the executable at link time. Depending on when and how the address bindings are done, the linking process can be categorized into prelinking, load time linking, implicit run-time linking and explicit run-time linking.

## 1.8 Position Independent Code (or, Win32 DLLs versus ".SO")

Position independent code can be copied to any memory location without modification and then executed, unlike relocatable code which requires special processing by a linker to make it suitable for execution at a given location.

Win32 DLLs are not position independent. They need to be relocated during loading, unless the fixed base it was built with happens to be unused in the loading process. Relocations to the same address can be shared, but if different processes have conflicting memory layouts, the loader needs to generate multiple copies of the DLL in memory. When the Windows loader maps a DLL into memory, it opens the file and tries to map the file into memory at its preferred base address. As these mapped pages are touched, the paging system will see whether the pages are already present in memory. If they are, it just remaps the pages to the new process since the relocation has already been done by the loader at the preferred base address. Otherwise the pages are fetched from the disk.

If the preferred address range for the DLL is not available, the loader maps the pages into a free location in the process address space. In this case, it marks the code page as COW (copy-on-write) which previously was marked read+execute, since the linker will have to perform code fix ups at the time of relocation, necessitating the page to be backed up by a paging file.

Linux solves this issue by the use of PIC (Position Independent Code). Shared objects in Linux usually contain PIC which avoid the need to relocate the library at load time. All code pages can be shared amongst all processes using the same library and can be paged to/from the file system. In x86, there is no simple way to address data relative to the current location since all jumps and calls are instruction-pointer relative. Hence, all references to the static globals were directed through a table known as Global Offset Table (GOT).

## 1.9 Dynamic Linking in Linux

### 1.9.1 The ELF data structures

As this is not an article specifically on the ELF format, we will discuss in brief only those data structures which are relevant to our discussion. For dynamic linking, the ELF linker primarily uses two processor-specific tables, the Global Offset Table (GOT) and the Procedure Linkage Table (PLT).

### 1.9.2 Global offset Table (GOT)

ELF linkers support PIC Code through the GOT in each shared library. The GOT contains absolute addresses to all of the static data referenced in the program. The address of the GOT is normally stored in a register (EBX) which is a relative address from the code that references it.

### 1.9.3 Procedure Linkage Table (PLT)

Both the executables that use the shared libraries and the shared library itself has a PLT. Similar to how the GOT redirects any position-independent address calculations to absolute locations, the PLT redirects position-independent function calls to absolute locations.

Apart from these two tables, the linker also refers to .dynsym, which contains all of the file's imported and exported symbols, .dynstr, which contains name strings for the symbols, .hash which contains the hash table which the runtime linker can use to lookup symbols quickly, and .dynamic, which is a list of tagged values and pointers.

In the .dynamic section, the important tag types are:

- DT_NEEDED: This element holds the string table offset of a null-terminated string, giving the name of a needed library. The offset is an index into the table recorded in the DT_STRTAB entry.
- DT_HASH: This element holds the address of the symbol hash table which refers to the symbol table referenced by the DT_SYMTAB element.
- DT_STRTAB: This element holds the address of the string table.
- DT_SYMTAB: This element holds the address of the symbol table.

### 1.9.4 Symbol Hashtable

```
nBuckets //no of bucket entries
nChains   //no of chain entries
bucket[]
chain[]
```

Both the bucket and chain array contain symbol table indexes. For the symbol to be searched, a hash of the symbol is computed and hash%nBuckets is used as an index into the bucket[] array. The bucket element gives an index, symindx, into the chain array as well as to the symbol table. If the symbol table entry does not match, it looks up the next symbol table entry with the same hash value using the index retrieved from Chain [symindx].

### 1.9.5 How things work

In Linux, the dynamic linker ld.so is itself an ELF shared library. At program startup, the system maps the ld.so to a part of the address space and runs its bootstrap code. The main entry point for the loader is defined in dl_main(elf/rtld.c). The linker relocates and resolves references to its own routines which are needed to load everything else.

The dynamic segment (pointed to by the program header) in the ELF file contains a pointer to the file's string table (DT_STRTAB) as well as to the DT_NEEDED entries, each of which contains the offset in the string table for

the name of a required library. The dynamic linker creates a scope list for the executable, consisting of libraries to be loaded.

We have two ways to specify objects to preload, either through the environment variable LD_PRELOAD or via the file /etc/ld.so.preload. The latter can be used when security prevents the use of an environment variable. The loader adds the DT_NEEDED entries of the executable as well as the scope, after the preload entries.

For each of the entries in the scope, the linker searches for the file containing the library. Once the file is found, the linker reads the ELF Header to find the program header, which points to the dynamic segment .The linker maps the library to the process address space. From the dynamic segment, it adds the library's symbol table to the chain of symbol tables - and if the libraries has further dependencies, it adds those libraries to the list to be loaded and the process is continued. For clarification, note that in fact it actually creates a struct link_map for each of the library and adds it into a global linked list.

The Linker keeps in memory a linked list of the tables (of type struct link_map, referenced by the dl_loaded parameter in struct rtld_global) in each file. The linker utilizes the hashtable present in the ELF file (referred to by the DT_HASH) to speed symbol lookup.

Once the loader finishes constructing the linked list of tables of all the dependencies, it revisits each library and handles the library's relocation entries, filling in the library's GOT and performing the relocations needed.

The LD_BIND_NOW variable determines the dynamic linking behavior. If its set, the dynamic linker evaluates the PLT entries, which is all entries of type R_386_JMP_SLOT, at the load time itself. Otherwise, the dynamically linker does lazy linking of procedure addresses and hence the addresses are not bound unless the routines are called.

### 1.9.6 A Walk through through the Linux lazy linking procedure

In this section, we will trace how a function defined in the shared library libtest.so gets resolved at runtime. The executable, disassembled using gdb below, was created by linking with a PIC library, libtest.so.

```
objdump -d a.out | grep -A 10 main
08048464 <main>:
 8048464: 55               push %ebp
 8048465: 89 E5            move $esp,%ebp
 8048467: 83 EC 08         sub  $0x8,%esp
 804846A: 83 E4 F0         and  $0xFFFFFFF0,$esp
 804846D: B8 00 00 00 00   mov  $0x0,%eax
 8048472: 29 C4            sub  $eax,$esp
 8048474: E8 2B FF FF FF   call 80483A4 <_init+0x48>
```

Figure 1. Executable disassembled using gdb.

Let's start tracing from the call instruction shown in Figure 1.

The address in the call instruction (0x80483a4) is an entry in the PLT . The first 4 entries in the PLT (whereby the 3rd and 4th entries are reserved) are common to all function calls. The rest of the entries are grouped into blocks of three, one block for each function. This is shown in Figure 2.

```
08048374 <.plt>
 8048374: FF 35 3C 96 04 08  pushl 0x804962C (GOT+4)
 804837A: FF 25 30 96 04 08  jmp   *0x8049630 (GOT+8)
 8048380: 00 00              add   %al,(%eax)
 8048382: 00 00              add   %al,($eax)
 8048384: FF 25 34 96 04 08  jmp   *0x8049634
 804838A: 68 00 00 00 00     push  $0x0
 804838F: E9 E0 FF FF FF     jmp   8049374 <_init+0x18>
 8048394: FF 25 38 96 04 08  jmp   *0x8049638
 804839A: 68 08 00 00 00     push  $0x8
 804839F: E9 D0 FF FF FF     jmp   8048374 <_init+0x18>
 80483A4: FF 25 3C 96 04 08  jmp   *0x804963C
 80483AA: 68 10 00 00 00     push  $0x10
 80483AF: E9 C0 FF FF FF     jmp   8048374 <_init+0x18>
```

Figure 2. First few entries in the PLT.

```
objdump -s -a.out | grep -A 2 got

 8049628 44950408 00000000 00000000 8A830408 D...............
 8049638 9A830408 AA830408 00000000           ............
```

Figure 3. The GOT as shown on disk.

The instruction consists of a jump to the address in the GOT entry *(GOT + 0x14), which points back to the next entry in the PLT viz 0x80483aa (as was shown in Figure 2).

The next instructions are for resolving the address using the dynamic linker. The jump instruction pushes an offset (0x10). This is an offset into the file's relocation table, which points to the desired symbol in the symbol table, and the address points to the GOT entry (0x804963c).

```
0x00000011 (REL)          0x804833C
0x00000013 (RELSZ)        8 (bytes)
0x00000013 (RELENT)       8 (bytes)
0x6FFFFFFE (VERNEED)      0x804831C
0x6FFFFFFF (VERNEEDNUM)   1
0x6FFFFFF0 (VERSYM)       0x80482FE
0x00000000 (NULL)         0x0

Relocation section '.rel.dyn' at offset 0x33C contains 1 entries:
 Offset     Info     Type         Sym.Value Sym. Name
```

```
08049640  00000D06 R_386_GLOB_DAT  00000000 __gmon_start__

Relocation section '.rel.dyn' at offset 0x33C contains 1 entries:
 Offset     Info     Type         Sym.Value Sym. Name
08049634  00000107 R_386_JUMP_SLOT 08048384 n
08049638  00000507 R_386_JUMP_SLOT 08048394 __libc_start_main
0804963C  00000B07 R_386_JUMP_SLOT 080483A4 m
```

Figure 4. Relocation entry (RELSZ) of 8 bytes .

As seen in Figure 4, the size of a relocation entry (RELSZ) is 8 bytes. The offset 0x10 gives us the 3rd relocation entry in the .rel.plt table, which is the relocation entry for m. The offset entry inside the table gives the corresponding GOT address which has to be updated.

The code then jumps into the first entry in the PLT, which is the common part.

```
Breakpoint 1, main () at main.c:3
3       m();
(gdb) x/3i 0x80483A4
0x80483A4 <m>: jmp *0x804963C
0x80483AA <m+6>: push $0x10
0x80483AF <m+11>: jmp 0x8048374 <_init+24>
(gdb) x/1x 0x804963C
0x804963C <__JCR_LIST__+24>: 0x080483AA
(gdb) x/3i 0x8048374
0x8048374 <_init+24>: pushl 0x804962C
0x804837A <_init+30>: jmp *0x8049630
0x8048380 <_init+36>: add %al,(%eax)
(gdb) x/1x 0x8049630
0x8049630 <__JCR_LIST__+12>: 0x4000BCB0
(gdb) n
4       n();
(gdb) x/1x 0x804963C
0x804963C <__JCR_LIST__+24>: 0x400177DB
(gdb) x/3i 0x400177DB
0x400177DB <m>: push %ebp
0x400177DC <m+1>: mov %esp,%ebp
0x400177DE <m+3>: push %ebx
```

Figure 5. Breakpoint 1.

The entry in the PLT again does a jump to the address in GOT + 8.The loader at load time has updated the entries in GOT +4 and GOT +8 (which were 0x000000 earlier, as was seen in Figure 3) . Now GOT + 8 (0x 4000bcb0) points to an address which is mapped with ld-2.3.2.0 (the runtime linker), as can be seen below in Figure 6.

```
> ps -ef | grep a.out
```

```
root 2912 2911 0 12:31 pts/0 00:00:00 /home/a.out
> cat /proc/2912/maps
08048000-08049000 r-xp 00000000 03:08 213129 /home/a.out
08049000-0804A000 rw-p 00000000 03:08 213129 /home/a.out
40000000-40015000 r-xp 00000000 03:0A 32718  /lib/ld-2.3.2.so
40015000-40016000 rw-p 00014000 03:0A 32718  /lib/ld-2.3.2.so
```

Figure 6. Address mapped with the runtime linker.

Once the dynamic linker routine looks up the symbol value using the concatenated run time table, and stores the routines address (0x400177db) into the GOT entry (0x 804963c) as seen in Figure 5, the subsequent calls jump directly to the routine itself.

### 1.9.7 Peeking into the dynamic linker ld.so

Let's go back to the GOT entries. As we have seen, the GOT + 8 contains the address of the linker's symbol resolution routine. The GOT + 4 was filled by the loader with the address of a structure struct link_map, defined in include/ link.h. The GOT gets filled by the routine elf_machine_runtime_setup defined in dl-machine.h.

Let's look at this further.

```
Struct link_map
{
  ElfW(Addr) l_addr; /* Base address shared object is loaded at.  */
  char *l_name; /* Absolute file name object was found in.  */
  ElfW(Dyn) *l_ld; /* Dynamic section of the shared object.  */
  struct link_map *l_next, *l_prev; /* Chain of loaded objects.  */

 /* All following members are internal to the dynamic linker.
     They may change without notice.  */

/* Indexed pointers to dynamic section.*/
  ElfW(Dyn) *l_info[DT_NUM + DT_THISPROCNUM + DT_VERSIONTAGNUM
     + DT_EXTRANUM + DT_VALNUM + DT_ADDRNUM];
  const ElfW(Phdr) *l_phdr; /* Pointer to program header table in core.  */
  ElfW(Addr) l_entry; /* Entry point location.  */
  ElfW(Half) l_phnum; /* Number of program header entries.  */
  ElfW(Half) l_ldnum; /* Number of dynamic segment entries.  */

  /* Array of DT_NEEDED dependencies and their dependencies, in
     dependency order for symbol lookup (with and without
     duplicates).  There is no entry before the dependencies have
     been loaded.  */
  struct r_scope_elem l_searchlist;
```

```
 /* Symbol hash table.  */
  Elf_Symndx l_nbuckets;
  const Elf_Symndx *l_buckets, *l_chain;
.
.
```

At the time the execution jumps into the symbol resolution routine, we have the address of the link map and the relocation offset in the stack. The relocation offset as discussed above gives the index in the symbol table for the symbol name and the corresponding GOT address, where the resolved address should be written.The symbol resolution address (GOT +8) points to a trampoline ELF_MACHINE_RUNTIME_TRAMPOLINE.

Let's look into the ELF_MACHINE_RUNTIME_TRAMPOLINE defined in dl-machine.h. The code preserves the registers and does a call to the fixup() function:

```
movl 16(%esp), %edx # Copy args pushed by PLT in register.  Note
movl 12(%esp), %eax # that 'fixup' takes its parameters in regs.
call fixup # Call resolver.
```

The fixup function is defined in dl-runtime.c.

The l_info array inside the struct link_map contains indexed pointers to the dynamic section:

```
const ElfW(Sym) *const symtab
    = (const void *) D_PTR (l, l_info[DT_SYMTAB]);
const char *strtab = (const void *) D_PTR (l, l_info[DT_STRTAB]);

const PLTREL *const reloc
    = (const void *) (D_PTR (l, l_info[DT_JMPREL]) + reloc_offset);
```

From the l_info, the code retrieves the pointers to the symbol table and relocation table. It calculates the relocation entry for the symbol by adding the relocation offset to the relocation base:

```
  const ElfW(Sym) *sym = &symtab[ELFW(R_SYM) (reloc->r_info)];
  void *const rel_addr = (void *)(l->l_addr + reloc->r_offset);
```

From reloc-¿r_info, it gets the symbol index which is used to index into the symbol table to get the symbol table info as well as the address to be updated from reloc-¿r_offset +l-¿l_addr.

The fixup function calls the _dl_lookup_symbol() for doing the lookup job with the above information.

The dl_lookup_symbol() in turn calls do_lookup() for each entry in the scope array. The scope array contains elements of type struct r_scope_SSHT_elem for the libraries, which forms part of the global search scope. This structure gets filled at loadtime.

```
struct r_scope_elem
{
  /* Array of maps for the scope.  */
  struct link_map **r_list;
  /* Number of entries in the scope.  */
  unsigned int r_nlist;
};
```

The do_lookup is defined to FCT in do-lookup.h. Let's take a look at this now from a logical perspective in plain English, to make it it easier to follow:

```
Do_lookup algorithm()
{
For each of the link_map structures in scope->r_list

do{

Get the symtable address from link_map->l_info
Get the strtable address from link_map->l_info

Search the appropriate hash bucket in the objects symbol table
  using the hash produced from the symbol name in _dl_lookup_symbol().

Using the index entries in the hash chain, link_map ->l_chain,

Do{

Lookup the symbol table entry using the index.
Compare the symbol name with (strtab + sym->st_name).
If found, return the symbol table entry with the link_map structure;
}
}
```

Now let's go back to fixup() :

```
/*link_map ->l_addr points to the base load address */
value = link_map->l_addr + sym->st_value

/* Finally, fix up the plt itself.  */
return elf_machine_fixup_plt (l, result, reloc, rel_addr, value);
```

Back in dl-machine.h, it pops the saved registers:

```
xchgl %eax, (%esp) # Get %eax contents and store function address.
ret $8 # Jump to function address.
```

If you remember, except for the call from the main function, all the other code paths were through jumps. This unwinds the stack and does a jump to the resolved function address.

## 1.10 Windows Portable Executable File Format (PE) data structures

We know that a section is a chunk of code or data that logically belongs together, and that the data for an executable's import tables are in a section. In this part of the article, we look at some of the sections found in Windows PE files.

## 1.11 Exports section (.edata)

The .edata section begins with the export directory structure IMAGE_EXPORT_DIRECTORY. The export directory contains RVAs (relative virtual addresses) of the: Export Address Table: This contains address of exported entry points, exported data and absolutes. An ordinal number is used to index the address table. The ORDINAL BASE must be subtracted from the ordinal number before indexing into the table.

Export Name Table Pointers: This array contains address in the Export name table. The pointers are relative to the image base and are ordered lexically to enable binary search. The export name table contains ASCII names for exported entries in the image.

Export Ordinal Table: The export name table pointers and Export ordinal table form two parallel arrays .The export ordinal table array contains the ordinal associated with the exported name referenced by the Export name table pointers. The ordinal will serve as the index into the EAT.

## 1.12 Imports Section(.idata)

The .idata section does the converse of what the .edata section, described above, does. It maps symbols/ordinals back into RVAs. The .idata begins with a import directory table IMAGE_IMPORT_DIRECTORY .The import directory table consists of an array of IMAGE_IMPORT_DESCRIPTOR structures, one for each imported executable. The IMAGE_IMPORT_DESCRIPTOR contains RVAs of :

Import Lookup Table: This is an array of IMAGE_THUNK_DATA structures .The structure contains ordinal or hint/name RVAs for each imported function. The table identifies the symbols to import, with the entries in the import lookup table being parallel to those in the Import Address Table (IAT). If the high bit of an entry is set , the lower bits are the ordinal. Otherwise the entry is a RVA of an entry in the hint-name table.

Import Address Table: This is also an array of IMAGE_THUNK_DATA structures. Initially both the Import Lookup table and the IAT contain similar entries. The loader fills in the addresses of each of the imported routines in this table, while the entries in the Import Lookup Table retains the original data as before. We will see why the linker maintains the original information later when we discuss binding.

Hint-Name Table: The table consists of a 4-byte hint followed by the null terminated symbol name. The hint value is used to index the Export Name Table pointers array, allowing faster by-name imports. The hint will be right if the DLL hasn't changed or at least its list of export symbols hasn't changed. If hint is incorrect, then binary search is performed on the Export Name Pointer table.

## 1.13 How things work

Loading a Windows executable and DLL is similar to loading a dynamically linked ELF program in Linux. The difference is that here the linker is a part of the kernel itself. First the kernel maps in the executable guided by the PE headers. The loader looks at the IAT of the module and determines whether the DLL depends on additional DLLs, and if so the loader maps them also. This process continues until all of the dependent modules have been mapped into memory.

An imported function can be listed by name or it can be listed by ordinal. The ordinal represents its position in the DLL Export Address table. If listed by name, the loader does a binary search of the Export Name Pointers table of the corresponding DLL to lookup the index at which the symbol is found. It then uses that index as an index into the Export Ordinal table to get the ordinal which, in turn, is used as an index into the Export Address table. Adding the RVA of the symbol found from the EAT to the load address of the corresponding DLL will yield the absolute address which the loader writes into the corresponding entry in the IAT.

## 1.14 Delay loading in Windows

A delay loaded DLL has a structure ImgDelayDescr similar to the .idata data import directory structure but it is not in the .idata section. The ImgDelayDescr contains the addresses of an IAT and an INT for the DLL. These tables are identical in format to the normal import ones, but they are written to and read by the runtime library code rather than by the operating system. When you call an API from a delay loaded DLL for the first time, the runtime library loads the DLL (if needed), gets the address, and stores it in the delayload IAT so that future calls go directly to the API.

## 1.15 A walk through through the Windows lazy linking procedure

In this section we will trace how the linker resolves the function address defined in a delay loaded DLL, as well as the semantics of making function calls where the function is defined in a DLL.

```
__declspec(dllimport) int fndll1();
//__declspec(dllimport) int fndll2();
extern int fndll2;

int main()
{
  fndll1();
  fndll2();
  return 1;
}
```

Figure 7. Linker resolving the function address in a delay loaded DLL.

In the above case, fndll1() is defined inside d111(delay load) and fndll2 is defined inside d112.If you notice the declarations, fndll1() has been explicitly declared as __declspec(dllimport). The _declspec(dllimport) function modifier tells the compiler that the function resides in another DLL. The compiler takes the clue and generates code CALL DWORD PTR [xxxx], where xxx is an entry within the IAT.

In the case of fndll2(), the compiler emits a call instruction of the form CALL xxxxx, where xxxx points to a stub. This results in an extra jump instruction, taking a longer time to execute.

The fndll1() call results in a CALL DWORD PTR[0x412760] . Then 0x412760 is the address of the first entry in the Delayimport address table.

This entry points to a helper routine that finds and loads the DLL, and then replaces the content of the address table with the actual address.

The address 0x412760 which earlier pointed to 0x40104b, which was the helper routine address, got overwritten by the loader with 0x351070, the address of fndll1.

## 1.16 Inside the Delay Load Helper

Windows allows you to add your own delay load helper routine. We will now trace what typically happens inside the helper routine. The reader should notice the marked similarities the lazy linking procedure has with its Linux counter part.

Let's start by looking again at Figure 9. If you notice, you can see that linker inserts two types of stubs. One is of the type __imp_load_(function name) and other is __tailmerge_(dllname). As seen from the naming convention, the first type of stub is generated per API for the DLL and second type per DLL.

The instruction does a jump to the __imp_load_(function name) stub through the Delay import address table. In the stub, the first instruction is:

```
Mov     eax,offset __imp_fndll1
```

This moves the address of the zeroth entry of the Delay IAT. (Note that this is the address where the helper routine has to update with the resolved

address of the routine). The next instruction is a jump to the DLL specific stub _tailmerge_(dllname). In the _tailmerge_stub, after preserving the registers ecx and edx, it does a push of the eax.The next instruction is:

```
Push offset __DELAY_IMPORT_DESCRIPTOR_Dll1
```

This pushes the address of ImgDelayDescr struct of the DLL1.The data structure is defined in DELAYIMP.h

```
typedef struct ImgDelayDescr {
 DWORD           grAttrs;        // attributes
 LPCSTR          szName;         // pointer to dll name
 HMODULE *       phmod;          // address of module handle
 PImgThunkData   pIAT;           // address of the IAT
 PCImgThunkData  pINT;           // address of the INT
 PCImgThunkData  pBoundIAT;      // address of the optional bound IAT
 PCImgThunkData  pUnloadIAT;     // address of optional copy of original IAT
 DWORD           dwTimeStamp;    // 0 if not bound,
             // O.W. date/time stamp of DLL bound to (Old BIND)
 } ImgDelayDescr, * PImgDelayDescr;

typedef const ImgDelayDescr *   PCImgDelayDescr;
```

Now the function does a jump to the helper routine with these values in the stack as arguments. From now onwards, we will base our discussion based on the helper code defined in DELAYHLP.CPP:

```
__delayLoadHelper(PCImgDelayDescr pidd, FARPROC * ppfnIATEntry)
```

The delayLoadHelper first tries to get the module handle from the ImgDelayDescr.

```
//Calculate the function index, which is an index into the IAT.

iINT = IndexFromPImgThunkData(PCImgThunkData(ppfnIATEntry), pidd->pIAT);
```

As said before the IAT and INT are two parallel structures:

```
//Using the function index to point to the corresponding index at the INT.
 PCImgThunkData  pitd = &((pidd->pINT)[iINT]);

//Get the function name or ordinal from the INT depending on
/ whether the higher bit is set as discussed before.

if (dli.dlp.fImportByName=((pitd->u1.Ordinal & IMAGE_ORDINAL_FLAG)==0))
   {    dli.dlp.szProcName = LPCSTR(pitd->u1.AddressOfData->Name);
        }
    else {
        dli.dlp.dwOrdinal = IMAGE_ORDINAL(pitd->u1.Ordinal);
```

```
        }

If (hmodule =0) //the first time
  {
    // Load library
    // Copy handle to the global variable(Call Free library()
    // if another thread got there before us)
}
```

Now we must lookup the address of the procedure by calling GetProcAddress(), as was mentioned above in the explanation of how things work:

```
pfnRet = ::GetProcAddress(hmod, dli.dlp.szProcName);
```

We update the IAT entry with the address:

```
*ppfnIATEntry = pfnRet;
//Back in __tail_merge_dll1
```

Now the eax contains the return value which is the resolved function address. Finally, the code does a:

```
       Jmp eax // jump to the function.
```

## 1.17 Explicit Dynamic Linking

Both Linux and Windows provide routines (such as dlopen() and dlsym() in Linux, and LoadLibrary() ,GetProcAddress() in Windows) to explicitly load a library and to find the address of a routine in that library. These routines are just wrappers which in turn call the dynamic linker routines which were previously called at the time of implicit linking through the PLT or IAT.

## 1.18 Speeding things up – Static Linked Shared Libraries

Shared libraries in practice can be very slow. The performance degradation in using them happens mainly because of runtime loading and address binding, indirect references to the routine addresses through intermediate tables, and machine register reservation for these tables. Today with large address spaces, its possible to bind a library to a chunk of address space at the linktime itself and also resolve the address references. If the address space is available at runtime, relocation can be avoided. Libraries where program and data addresses are bound to executables at link time are referred to as static linked shared libraries.

## 1.19 Prelinking in Linux

Glibc is the only shared library in Linux which can be statically linked. For other options, Linux instead uses a similar concept called prelinking. A prelink assigns a unique virtual address slot for each library the executable depends on, and relinks the library to that base address.

Prelinking a shared library means identifying which other shared libraries are needed by this library, and pre-patching the shared library file by assuming that the needed shared libraries will be loaded at their selected pre-relocation addresses. Prelink also stores a list of all dependent libraries together with their timestamps and checksums into the library or binary.

```
Library list section '.gnu.liblist' contains 3 entries:
   Library            Time Stamp          Checksum   Version Flags
0: libtest.so         2006-01-21T18:16:39 0x9BCCA3B5 0       0
1: libc.so.6          2006-01-21T18:16:39 0x5880D225 0       0
2: /lib/ld-linux.so.2 2006-01-21T18:16:39 0x51D61A75 0       0
```

Figure 14. List of available libraries.

If you dump a prelinked executable or shared library, one thing which you will notice is the change in relocation format. Normally IA-32 architectures use only the REL format, where the relocation addend is stored at the offset address only. The only case where you might see a RELA section in IA-32 will therefore be here.

Since the prelinked shared libraries need to be usable even in non-prelinked executables, the addend information has to be preserved. To do this, a prelink converts the .rel.dyn section to RELA format. Prelink avoids doing this in cases where the addend is zero by changing relocation type to R_386_GLOB_DAT.

```
Relocation section '.rel.dyn' at offset 0x5D4 contains 7 entries:
 Offset     Info    Type           Sym.Value Sym. Name + Addend
4114E8CC  00000008 R_386_RELATIVE                     4114E8CC
4114E8D0  00000008 R_386_RELATIVE                     4114E9A4
4114E8D4  00001E01 R_386_32       4114E9E4  k + 8
4114E9D0  00001E06 R_386_GLOB_DAT 4114E9E4  k + 0
```

Figure 15. Preserving the addend information.

The prelink utility also generates a conflict list during the prelink process and stores it inside the executable. The ELF document specifies that undefined symbols in shared libraries must be first searched in the main executable program, then searched in the needed shared libraries. Not all symbols resolve the same when looked up in a shared library's search scope (as is done when the shared library is prelinked) and when looked up in the application's global symbolsearch scope. Such symbols are called conflicts, and relocations against those symbols conflicting relocations.

The conflicting relocations are added in a separate RELA section in the executable. In this case, the Sym.name+addend will contain the actual address

of the conflicting variable (in other words, resolved with respect to the global scope of the executable).

```
Relocation section '.gnu.conflict' at offset 0x704 contains 16 entries:
 Offset     Info    Type          Sym.Value Sym. Name + Addend
41148AEC  00000001 R_386_32                                FFFFFDD8
41148AF0  00000001 R_386_32                                00000001
41148B38  00000001 R_386_32                                FFFFFFE8
```

Figure 16. Resolving conflicts.

At runtime, the dynamic linker first checks whether all dependant libraries were successfully mapped into their designated address space slots and whether they have not changed since prelinking was done. If it was, the prelinker has to do only a few adjustments which were defined by the conflict list created earlier.

## 1.20 Rebasing and binding in Windows

To look at the equivalent in Windows, Windows DLLs use the rebasing and binding concept. Every executable and DLL module has a preferred base address which identifies the address where the module should be mapped in the process address space. For an executable, the default value is 0x00400000 and for a DLL it is 0x10000000. This mean that if the executable is linked to two DLLs, one of them will have to be relocated in memory. To avoid this you can rebase your DLL by giving it a preferred address at the time of compilation. You can do this by passing the /base [address] switch in the project options.

As we have seen, win32 executables have two identical tables containing the information needed to lookup an imported function - the Import Name table and Import Lookup table. Only one copy is required by the loader. Bind takes advantage of this fact and overwrites the IAT entries with the imported function's actual address at link time. The bind adds bind information such as a timestamp to the bound executable. At loadtime, the loader verifies that the location of the symbol referenced in the DLL's exported section has not changed.

For checking the validity of the bound information, PE uses a data structure IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT which is pointed to by the data directory. This structure is a list of IMAGE_BOUND_DESCRIPTOR elements, one for each imported DLL. This structure stores the time stamp, the name of the imported DLL, and the number of forwarder references. The export forwarding concept is beyond the scope of this article. But for the sake of completion, it's good to know that Windows allows you to reference an exported API in one DLL which has been forwarded by that DLL to another one.

If the DLL's time stamp has not changed, it uses the address that bind has stored in the IAT. Otherwise, it uses the hint/name table information to do the normal lookup.

# Chapter 2

# readelf -a ex

## 2.1 ELF Header

```
Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
Class:                             ELF32
Data:                              2's complement, little endian
Version:                           1 (current)
OS/ABI:                            UNIX - System V
ABI Version:                       0
Type:                              EXEC (Executable file)
Machine:                           Intel 80386
Version:                           0x1
Entry point address:               0x8049610
Start of program headers:          52 (bytes into file)
Start of section headers:          204796 (bytes into file)
Flags:                             0x0
Size of this header:               52 (bytes)
Size of program headers:           32 (bytes)
Number of program headers:         6
Size of section headers:           40 (bytes)
Number of section headers:         34
Section header string table index: 31
```

## 2.2 Section Headers

```
[Nr] Name              Type            Addr     Off    Size
     ES Flg Lk Inf Al
```

This is the SHT_NULL section. This value marks the section header as inactive; it does not have an associated section. Other members of the section header have undefined values.

```
  [ 0]                   NULL            00000000 000000 000000
      00      0   0  0
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. This section contains the ASCII string tyhat is the name of the dynamic loader. For instance, "/lib/ld-linux.so.2". The sections ".hash", ".dynsym", and ".dynstr" are symbol tables used by the dynamic linker when performing relocations.

```
  [ 1] .interp           PROGBITS        080480f4 0000f4 000013
      00   A  0   0  1
```

This is the SHT_NOTE section. This section holds information that marks the file in some way. Every executable shall contain this section. It is structured as a note section. The section shall contain at least the following entry. The `name` field (namesz/name) contains the string "GNU". The `type` shall be 1. The `descsz` filed shall be at least 16, and the first 16 bytes of the `desc` field shall be as follows. The first 32-bit word of the `desc` field shall be 0 (this signifies a Linux executable). The second, third, and fourth 32-bit words of the `desc` field contain the earliest compatible kernel version. For example, if the 3 words are 2, 2, and 5, this signifies a 2.2.5 kernel.

```
  [ 2] .note.ABI-tag     NOTE            08048108 000108 000020
      00   A  0   0  4
```

This is the SHT_HASH section. This section holds a symbol hash table. Currently an object file shall have only one hash table. This is used by the dynamic linker when performing relocations. These are mapped into virtual memory (the virtual address is non-zero)

```
  [ 3] .hash             HASH            08048128 000128 000314
      04   A  4   0  4
```

This is the SHT_DYNSYM section. This section holds a minimal set of symbols adequate for dynamic linking. See also SHT_SYMTAB. Currently, an object file may have either a section of STH_SYMTAB type or a section of SHT_DYNSYM type, but not both. This is used by the dynamic linker when performing relocations. These are mapped into virtual memory (the virtual address is non-zero).

The dynsym table is a smaller version of the symtab (see section 32 below). The information found in the dynsym is also found in the symtab but the reverse is not necessarily true.

The reason there are two symbol tables with similar information is that some sections are allocable and others are non-allocable. Allocable sections are needed at runtime by the process. Non-allocable sections are used by linkers, debuggers and other tools but not by the running program. When the program is loaded ony the allocable sections are brought into memory.

```
[ 4] .dynsym           DYNSYM          0804843c 00043c 000620
     10   A  5   1  4
```

This is the SHT_SHRTAB section. This section holds a string table. An object file may have multiple string table sections. This is used by the dynamic linker when performing relocations. These are mapped into virtual memory (the virtual address is non-zero)

```
[ 5] .dynstr           STRTAB          08048a5c 000a5c 00030e
     00   A  0   0  1
```

This is the SHT_GNU_versym section. This section contains the Symbol Version Table.

```
[ 6] .gnu.version      VERSYM          08048d6a 000d6a 0000c4
     02   A  4   0  2
```

This is the SHT_GNU_verneed section. This section contains the symbol versions that are required.

```
[ 7] .gnu.version_r    VERNEED         08048e30 000e30 000030
     00   A  5   1  4
```

This is the SHT_REL section. This section holds relocation entries without explicit addends, such as type Elf32_Rel for a 32-bit class of object files or type Elf64_Rel for the 64-bit class of object files. An object file may have multiple relocation sections. This section header has a sh_link which is an index into the symbol table section. This indexes into .dynsym (which is section 4 above which is the meaning of the '4' in the line below).

```
[ 8] .rel.dyn          REL             08048e60 000e60 000008
     08   A  4   0  4
```

This is the SHT_REL section. This section holds relocation entries without explicit addends, such as type Elf32_Rel for a 32-bit class of object files or type Elf64_Rel for the 64-bit class of object files. An object file may have multiple relocation sections. PLT stands for the "Procedure Linkage Table".

This section header has a sh_link which is an index into the symbol table section. This indexes into .dynsym (which is section 4 above which is the meaning of the '4' in the line below).

This contains the jump table that is used when we call functions in the shared library. By default the .plt entries are all initialized by the linker not to point to the correct target functions, but instead to point to the dynamic loader itself. Thus, the first time you call any function the dynamic loader looks up the function and fixes the target of the .plt so taht the next time this .plt slot is used we call the correct function. After making this change, the dynamic loader calls the function itself.

This feature is known as lazy symbol binding. The idea is that if you have lots of shared libraries it could take the dynamic loader lots of time to look up

all of the functions to initialize all of the .plt slots. Thus it would be preferable to defer binding addresses to the functions until we actually need them. This is useful if you only call a small fraction of the functions. It is possible to instruct the dynamic loader to bind addresses to all of the .plt slots before transferring control to the application by setting the environment variable **LD_BIND_NOW=1** before running the program. This can help debugging.

Note that the .plt is in read-only memory. Thus the addresses used for the target of the jump are actually stored in the ".got" section. The ".got" also contains a set of pointers for all of the global variables that are used within a progra that come from a shared library.

```
[ 9] .rel.plt          REL             08048e68 000e68 000280
     08   A  4   b  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. This section contains executable code for the initialization of the program. Since elf assumes it is working in a multiprogramming environment it uses this code to save registers and other system state information. Also, any shared object file included in the program also has an opportunity to run its initialization code before the call to the main program. The header points to the first executable instruction referencing the beginning of the .init section. The main program is called from init. When control returns it calls .fini

```
[10] .init             PROGBITS        080490e8 0010e8 000017
     00  AX  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. PLT stands for the "Procedure Linkage Table"

```
[11] .plt              PROGBITS        08049100 001100 000510
     04  AX  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[12] .text             PROGBITS        08049610 001610 029458
     00  AX  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. The header points to the first executable instruction referencing the beginning of the .init section. The main program is called from init. When control returns it calls .fini

```
[13] .fini             PROGBITS        08072a68 02aa68 00001b
     00  AX  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[14] .rodata           PROGBITS        08072aa0 02aaa0 00404c
    00   A  0   0 32
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. This section contains information necessary for frame unwinding during exception handling.

```
[15] .eh_frame         PROGBITS        08076aec 02eaec 000004
    00   A  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[16] .data             PROGBITS        08077000 02f000 000d28
    00  WA  0   0 32
```

This is the SHT_DYNAMIC section. This section holds information for dynamic linking.

THis section contains shorthand notes used by the dynamic loader. This section table is not loaded into virtual memory. This section is a distilled version of the section header table that contains what the dynamic linker needs to do its job.

```
[17] .dynamic          DYNAMIC         08077d28 02fd28 0000c8
    08  WA  5   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. This section contains a list of global constructor function pointers.

```
[18] .ctors            PROGBITS        08077df0 02fdf0 000008
    00  WA  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. This section contains a list of global destructor function pointers.

```
[19] .dtors            PROGBITS        08077df8 02fdf8 000008
    00  WA  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[20] .jcr              PROGBITS        08077e00 02fe00 000004
    00  WA  0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program. This is the global offset table.

Note that the .plt is in read-only memory. Thus the addresses used for the target of the jump are actually stored in the ".got" section. The ".got" also contains a set of pointers for all of the global variables that are used within a progra that come from a shared library.

```
[21] .got              PROGBITS        08077e04 02fe04 000150
    04  WA  0   0  4
```

This is the SHT_NOBITS section. A section of this type occupies no space in the file but otherwise resembles SHT_PROGBITS. Although this section contains no bytes, the sh_offset member contains the conceptual file offset. The program may treat this data as uninitialized. However, the system shall initialize this data with zeros when the program begins to run.

```
[22] .bss              NOBITS          08077f60 02ff60 025f5c
    00  WA  0   0 32
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[23] .comment          PROGBITS        00000000 02ff60 0009c3
    00      0   0  1
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[24] .debug_aranges    PROGBITS        00000000 030928 000078
    00      0   0  8
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[25] .debug_pubnames   PROGBITS        00000000 0309a0 000025
    00      0   0  1
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[26] .debug_info       PROGBITS        00000000 0309c5 000a84
    00      0   0  1
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[27] .debug_abbrev     PROGBITS        00000000 031449 000138
    00      0   0  1
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[28] .debug_line       PROGBITS        00000000 031581 00027c
    00      0   0  1
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[29] .debug_frame      PROGBITS        00000000 031800 000014
    00      0   0  4
```

This is the SHT_PROGBITS section. This section holds information defined by the program, whose format and meaning are determined solely by the program.

```
[30] .debug_str        PROGBITS        00000000 031814 0006ba
    01  MS  0   0  1
```

This is the SHT_SHRTAB section. This section holds a string table. An object file may have multiple string table sections.

```
[31] .shstrtab         STRTAB          00000000 031ece 00012b
    00      0   0  1
```

This is the SHT_SYMTAB section. This section holds a symbol table. The symbols in this table are sharable objects.

The full symbol table contains a large amount of data needed to link or debug but is not needed at runtime. In fact, in the days before sharable libraries and dynamic linking, none of it was needed at runtime. There was a single, non-allocable symbol table. When dynamic linking was added to the system, the original designers decided to make a ".dynsym" table which contains the subset of symbols needed at runtime.

The symbols in the table are written in the following order:

1. index 0 in any symbol table is used to represent undefined symbols. The first entry (index 0) is always completely zeroed, has no type, and is unused.
2. if the file contains any local symbols, the second entry (index 1) of the symbol table will be of type FILE giving thename of the file
3. section symbols
4. register symbols
5. global symbols that have been reduced to local scope via a mapfile
6. for each input file that supplies local symbols, a FILE symbol giving the name of the input file followed by the symbols
7. the global symbols immediately follow the local symbols. Local and global symbols are always kept separate and cannot be mixed.

```
[32] .symtab           SYMTAB          00000000 03254c 004a20
    10     33  dd  4
```

This is the SHT_SHRTAB section. This section holds a string table. An object file may have multiple string table sections.

```
  [33] .strtab           STRTAB          00000000 036f6c 0025ff
       00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
```

## 2.3 Program Headers

```
  Type    Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR    0x000034 0x08048034 0x08048034 0x000c0 0x000c0 R E 0x4
  INTERP  0x0000f4 0x080480f4 0x080480f4 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD    0x000000 0x08048000 0x08048000 0x2eaf0 0x2eaf0 R E 0x1000
  LOAD    0x02f000 0x08077000 0x08077000 0x00f54 0x26ebc RW  0x1000
  DYNAMIC 0x02fd28 0x08077d28 0x08077d28 0x000c8 0x000c8 RW  0x4
  NOTE    0x000108 0x08048108 0x08048108 0x00020 0x00020 R   0x4
```

## 2.4 Section to Segment mapping

```
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .hash .dynsym .dynstr .gnu.version
          .gnu.version_r .rel.dyn .rel.plt .init .plt .text .fini
          .rodata .eh_frame
   03     .data .dynamic .ctors .dtors .jcr .got .bss
   04     .dynamic
   05     .note.ABI-tag
```

## 2.5 Dynamic segment

```
Dynamic segment at offset 0x2fd28 contains 20 entries
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x80490e8
 0x0000000d (FINI)                       0x8072a68
 0x00000004 (HASH)                       0x8048128
 0x00000005 (STRTAB)                     0x8048a5c
 0x00000006 (SYMTAB)                     0x804843c
 0x0000000a (STRSZ)                      782 (bytes)
```

```
 0x0000000b (SYMENT)                     16 (bytes)
 0x00000015 (DEBUG)                      0x0
 0x00000003 (PLTGOT)                     0x8077e04
 0x00000002 (PLTRELSZ)                   640 (bytes)
 0x00000014 (PLTREL)                     REL
 0x00000017 (JMPREL)                     0x8048e68
 0x00000011 (REL)                        0x8048e60
 0x00000012 (RELSZ)                      8 (bytes)
 0x00000013 (RELENT)                     8 (bytes)
 0x6ffffffe (VERNEED)                    0x8048e30
 0x6fffffff (VERNEEDNUM)                 1
 0x6ffffff0 (VERSYM)                     0x8048d6a
 0x00000000 (NULL)                       0x0
```

## 2.6 Relocation section '.rel.dyn'

```
Relocation section '.rel.dyn' at offset 0xe60 contains 1 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
08077f50  00006006 R_386_GLOB_DAT    00000000   __gmon_start__
```

## 2.7 Relocation section '.rel.plt'

PLT stands for the "Procedure Linkage Table"

```
Relocation section '.rel.plt' at offset 0xe68 contains 80 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
08077e10  00000207 R_386_JUMP_SLOT   08049110   iswctype
08077e14  00000307 R_386_JUMP_SLOT   08049120   ttyname
08077e18  00000507 R_386_JUMP_SLOT   08049130   towlower
08077e1c  00000607 R_386_JUMP_SLOT   08049140   mbtowc
08077e20  00000707 R_386_JUMP_SLOT   08049150   sigaction
08077e24  00000807 R_386_JUMP_SLOT   08049160   execl
08077e28  00000907 R_386_JUMP_SLOT   08049170   strchr
08077e2c  00000a07 R_386_JUMP_SLOT   08049180   getpid
08077e30  00000b07 R_386_JUMP_SLOT   08049190   siglongjmp
08077e34  00000c07 R_386_JUMP_SLOT   080491a0   write
08077e38  00000d07 R_386_JUMP_SLOT   080491b0   strcmp
08077e3c  00000e07 R_386_JUMP_SLOT   080491c0   close
08077e40  00000f07 R_386_JUMP_SLOT   080491d0   fork
08077e44  00001007 R_386_JUMP_SLOT   080491e0   getenv
08077e48  00001107 R_386_JUMP_SLOT   080491f0   unlink
08077e4c  00001407 R_386_JUMP_SLOT   08049200   strerror
08077e50  00001507 R_386_JUMP_SLOT   08049210   mmap
08077e54  00001607 R_386_JUMP_SLOT   08049220   tcsetattr
08077e58  00001707 R_386_JUMP_SLOT   08049230   setlocale
```

```
08077e5c  00001807 R_386_JUMP_SLOT   08049240   creat
08077e60  00001907 R_386_JUMP_SLOT   08049250   __errno_location
08077e64  00001a07 R_386_JUMP_SLOT   08049260   iswalnum
08077e68  00001b07 R_386_JUMP_SLOT   08049270   chmod
08077e6c  00001c07 R_386_JUMP_SLOT   08049280   tolower
08077e70  00001d07 R_386_JUMP_SLOT   08049290   access
08077e74  00001e07 R_386_JUMP_SLOT   080492a0   cfgetospeed
08077e78  00001f07 R_386_JUMP_SLOT   080492b0   pipe
08077e7c  00002107 R_386_JUMP_SLOT   080492c0   iswdigit
08077e80  00002307 R_386_JUMP_SLOT   080492d0   __sigsetjmp
08077e84  00002407 R_386_JUMP_SLOT   080492e0   __xstat
08077e88  00002507 R_386_JUMP_SLOT   080492f0   wctype
08077e8c  00002607 R_386_JUMP_SLOT   08049300   sysconf
08077e90  00002707 R_386_JUMP_SLOT   08049310   time
08077e94  00002907 R_386_JUMP_SLOT   08049320   chdir
08077e98  00002a07 R_386_JUMP_SLOT   08049330   __fxstat
08077e9c  00002b07 R_386_JUMP_SLOT   08049340   strlen
08077ea0  00002c07 R_386_JUMP_SLOT   08049350   sleep
08077ea4  00002d07 R_386_JUMP_SLOT   08049360   sbrk
08077ea8  00002e07 R_386_JUMP_SLOT   08049370   sigaddset
08077eac  00002f07 R_386_JUMP_SLOT   08049380   iswupper
08077eb0  00003107 R_386_JUMP_SLOT   08049390   mbrtowc
08077eb4  00003207 R_386_JUMP_SLOT   080493a0   iswspace
08077eb8  00003307 R_386_JUMP_SLOT   080493b0   wctomb
08077ebc  00003407 R_386_JUMP_SLOT   080493c0   sigprocmask
08077ec0  00003507 R_386_JUMP_SLOT   080493d0   strncmp
08077ec4  00003707 R_386_JUMP_SLOT   080493e0   fsync
08077ec8  00003807 R_386_JUMP_SLOT   080493f0   __libc_start_main
08077ecc  00003907 R_386_JUMP_SLOT   08049400   getpgid
08077ed0  00003c07 R_386_JUMP_SLOT   08049410   toupper
08077ed4  00003d07 R_386_JUMP_SLOT   08049420   strcat
08077ed8  00003e07 R_386_JUMP_SLOT   08049430   towupper
08077edc  00003f07 R_386_JUMP_SLOT   08049440   getuid
08077ee0  00004007 R_386_JUMP_SLOT   08049450   lseek
08077ee4  00004107 R_386_JUMP_SLOT   08049460   memcpy
08077ee8  00004207 R_386_JUMP_SLOT   08049470   strrchr
08077eec  00004307 R_386_JUMP_SLOT   08049480   btowc
08077ef0  00004607 R_386_JUMP_SLOT   08049490   open
08077ef4  00004707 R_386_JUMP_SLOT   080494a0   sigemptyset
08077ef8  00004807 R_386_JUMP_SLOT   080494b0   __ctype_get_mb_cur_max
08077efc  00004907 R_386_JUMP_SLOT   080494c0   atoi
08077f00  00004a07 R_386_JUMP_SLOT   080494d0   iswalpha
08077f04  00004b07 R_386_JUMP_SLOT   080494e0   iswprint
08077f08  00004c07 R_386_JUMP_SLOT   080494f0   ioctl
08077f0c  00004d07 R_386_JUMP_SLOT   08049500   getcwd
08077f10  00004e07 R_386_JUMP_SLOT   08049510   iswlower
```

```
08077f14  00004f07 R_386_JUMP_SLOT   08049520   isatty
08077f18  00005007 R_386_JUMP_SLOT   08049530   memset
08077f1c  00005107 R_386_JUMP_SLOT   08049540   _exit
08077f20  00005207 R_386_JUMP_SLOT   08049550   getsid
08077f24  00005307 R_386_JUMP_SLOT   08049560   strncpy
08077f28  00005407 R_386_JUMP_SLOT   08049570   dup
08077f2c  00005507 R_386_JUMP_SLOT   08049580   wcwidth
08077f30  00005707 R_386_JUMP_SLOT   08049590   fpathconf
08077f34  00005807 R_386_JUMP_SLOT   080495a0   kill
08077f38  00005a07 R_386_JUMP_SLOT   080495b0   __ctype_b_loc
08077f3c  00005b07 R_386_JUMP_SLOT   080495c0   tcgetattr
08077f40  00005c07 R_386_JUMP_SLOT   080495d0   read
08077f44  00005d07 R_386_JUMP_SLOT   080495e0   alarm
08077f48  00005e07 R_386_JUMP_SLOT   080495f0   wait
08077f4c  00006107 R_386_JUMP_SLOT   08049600   strcpy
```

## 2.8 unwind sections

```
There are no unwind sections in this file.
```

## 2.9 Symbol table '.dynsym'

```
Symbol table '.dynsym' contains 98 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 080521b8    67 FUNC    GLOBAL DEFAULT   12 putchar
     2: 08049110   103 FUNC    GLOBAL DEFAULT  UND iswctype@GLIBC_2.0 (2)
     3: 08049120   671 FUNC    GLOBAL DEFAULT  UND ttyname@GLIBC_2.0 (2)
     4: 0806a92c    37 FUNC    GLOBAL DEFAULT   12 printf
     5: 08049130   190 FUNC    GLOBAL DEFAULT  UND towlower@GLIBC_2.0 (2)
     6: 08049140   203 FUNC    GLOBAL DEFAULT  UND mbtowc@GLIBC_2.0 (2)
     7: 08049150    83 FUNC    GLOBAL DEFAULT  UND sigaction@GLIBC_2.0 (2)
     8: 08049160   319 FUNC    GLOBAL DEFAULT  UND execl@GLIBC_2.0 (2)
     9: 08049170   359 FUNC    GLOBAL DEFAULT  UND strchr@GLIBC_2.0 (2)
    10: 08049180    46 FUNC    GLOBAL DEFAULT  UND getpid@GLIBC_2.0 (2)
    11: 08049190   100 FUNC    GLOBAL DEFAULT  UND siglongjmp@GLIBC_2.0 (2)
    12: 080491a0   124 FUNC    GLOBAL DEFAULT  UND write@GLIBC_2.0 (2)
    13: 080491b0    37 FUNC    GLOBAL DEFAULT  UND strcmp@GLIBC_2.0 (2)
    14: 080491c0   113 FUNC    GLOBAL DEFAULT  UND close@GLIBC_2.0 (2)
    15: 080491d0    90 FUNC    GLOBAL DEFAULT  UND fork@GLIBC_2.0 (2)
    16: 080491e0   275 FUNC    GLOBAL DEFAULT  UND getenv@GLIBC_2.0 (2)
    17: 080491f0    54 FUNC    GLOBAL DEFAULT  UND unlink@GLIBC_2.0 (2)
    18: 0806b257  1055 FUNC    GLOBAL DEFAULT   12 malloc
    19: 0808b008     4 OBJECT  GLOBAL DEFAULT   22 loc2
```

```
20: 08049200   177 FUNC    GLOBAL DEFAULT  UND strerror@GLIBC_2.0 (2)
21: 08049210    54 FUNC    GLOBAL DEFAULT  UND mmap@GLIBC_2.0 (2)
22: 08049220   388 FUNC    GLOBAL DEFAULT  UND tcsetattr@GLIBC_2.0 (2)
23: 08049230  1631 FUNC    GLOBAL DEFAULT  UND setlocale@GLIBC_2.0 (2)
24: 08049240   120 FUNC    GLOBAL DEFAULT  UND creat@GLIBC_2.0 (2)
25: 08049250    57 FUNC    GLOBAL DEFAULT  UND __errno_location@GLIBC_2.0 (2)
26: 08049260   211 FUNC    GLOBAL DEFAULT  UND iswalnum@GLIBC_2.0 (2)
27: 08049270    58 FUNC    GLOBAL DEFAULT  UND chmod@GLIBC_2.0 (2)
28: 08049280   100 FUNC    GLOBAL DEFAULT  UND tolower@GLIBC_2.0 (2)
29: 08049290    58 FUNC    GLOBAL DEFAULT  UND access@GLIBC_2.0 (2)
30: 080492a0    16 FUNC    GLOBAL DEFAULT  UND cfgetospeed@GLIBC_2.0 (2)
31: 080492b0    54 FUNC    GLOBAL DEFAULT  UND pipe@GLIBC_2.0 (2)
32: 0806b7ea   109 FUNC    GLOBAL DEFAULT   12 calloc
33: 080492c0   211 FUNC    GLOBAL DEFAULT  UND iswdigit@GLIBC_2.0 (2)
34: 0806a951  1544 FUNC    GLOBAL DEFAULT   12 vprintf
35: 080492d0    53 FUNC    GLOBAL DEFAULT  UND __sigsetjmp@GLIBC_2.0 (2)
36: 080492e0   806 FUNC    GLOBAL DEFAULT  UND __xstat@GLIBC_2.0 (2)
37: 080492f0   241 FUNC    GLOBAL DEFAULT  UND wctype@GLIBC_2.0 (2)
38: 08049300     6 FUNC    GLOBAL DEFAULT  UND sysconf@GLIBC_2.0 (2)
39: 08049310    64 FUNC    GLOBAL DEFAULT  UND time@GLIBC_2.0 (2)
40: 0808b010     4 OBJECT  GLOBAL DEFAULT   22 loc1
41: 08049320    54 FUNC    GLOBAL DEFAULT  UND chdir@GLIBC_2.0 (2)
42: 08049330   806 FUNC    GLOBAL DEFAULT  UND __fxstat@GLIBC_2.0 (2)
43: 08049340   175 FUNC    GLOBAL DEFAULT  UND strlen@GLIBC_2.0 (2)
44: 08049350   513 FUNC    GLOBAL DEFAULT  UND sleep@GLIBC_2.0 (2)
45: 08049360   121 FUNC    GLOBAL DEFAULT  UND sbrk@GLIBC_2.0 (2)
46: 08049370    74 FUNC    GLOBAL DEFAULT  UND sigaddset@GLIBC_2.0 (2)
47: 08049380   211 FUNC    GLOBAL DEFAULT  UND iswupper@GLIBC_2.0 (2)
48: 0806b6a6   324 FUNC    GLOBAL DEFAULT   12 realloc
49: 08049390   379 FUNC    GLOBAL DEFAULT  UND mbrtowc@GLIBC_2.0 (2)
50: 080493a0   211 FUNC    GLOBAL DEFAULT  UND iswspace@GLIBC_2.0 (2)
51: 080493b0   157 FUNC    GLOBAL DEFAULT  UND wctomb@GLIBC_2.0 (2)
52: 080493c0    76 FUNC    GLOBAL DEFAULT  UND sigprocmask@GLIBC_2.0 (2)
53: 080493d0   179 FUNC    GLOBAL DEFAULT  UND strncmp@GLIBC_2.0 (2)
54: 0806c5ec   311 FUNC    GLOBAL DEFAULT   12 regcomp
55: 080493e0   113 FUNC    GLOBAL DEFAULT  UND fsync@GLIBC_2.0 (2)
56: 080493f0   251 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.0 (2)
57: 08049400    54 FUNC    GLOBAL DEFAULT  UND getpgid@GLIBC_2.0 (2)
58: 0804f8cb    35 FUNC    GLOBAL DEFAULT   12 getchar
59: 0806dfd4   277 FUNC    GLOBAL DEFAULT   12 regexec
60: 08049410   100 FUNC    GLOBAL DEFAULT  UND toupper@GLIBC_2.0 (2)
61: 08049420   426 FUNC    GLOBAL DEFAULT  UND strcat@GLIBC_2.0 (2)
62: 08049430   190 FUNC    GLOBAL DEFAULT  UND towupper@GLIBC_2.0 (2)
63: 08049440    67 FUNC    GLOBAL DEFAULT  UND getuid@GLIBC_2.0 (2)
64: 08049450   124 FUNC    GLOBAL DEFAULT  UND lseek@GLIBC_2.0 (2)
65: 08049460    39 FUNC    GLOBAL DEFAULT  UND memcpy@GLIBC_2.0 (2)
```

```
    66: 08049470   441 FUNC    GLOBAL DEFAULT  UND strrchr@GLIBC_2.0 (2)
    67: 08049480   381 FUNC    GLOBAL DEFAULT  UND btowc@GLIBC_2.0 (2)
    68: 0806def0   225 FUNC    GLOBAL DEFAULT   12 regerror
    69: 0804c3ef    90 FUNC    GLOBAL DEFAULT   12 error
    70: 08049490   124 FUNC    GLOBAL DEFAULT  UND open@GLIBC_2.0 (2)
    71: 080494a0    82 FUNC    GLOBAL DEFAULT  UND sigemptyset@GLIBC_2.0 (2)
    72: 080494b0    61 FUNC    GLOBAL DEFAULT  UND __ctype_get_mb_cur_max@GLIBC_2.0 (2)
    73: 080494c0    45 FUNC    GLOBAL DEFAULT  UND atoi@GLIBC_2.0 (2)
    74: 080494d0   211 FUNC    GLOBAL DEFAULT  UND iswalpha@GLIBC_2.0 (2)
    75: 080494e0   211 FUNC    GLOBAL DEFAULT  UND iswprint@GLIBC_2.0 (2)
    76: 080494f0    60 FUNC    GLOBAL DEFAULT  UND ioctl@GLIBC_2.0 (2)
    77: 08049500   218 FUNC    GLOBAL DEFAULT  UND getcwd@GLIBC_2.0 (2)
    78: 08049510   211 FUNC    GLOBAL DEFAULT  UND iswlower@GLIBC_2.0 (2)
    79: 08049520    53 FUNC    GLOBAL DEFAULT  UND isatty@GLIBC_2.0 (2)
    80: 08049530    67 FUNC    GLOBAL DEFAULT  UND memset@GLIBC_2.0 (2)
    81: 08049540    19 FUNC    GLOBAL DEFAULT  UND _exit@GLIBC_2.0 (2)
    82: 08049550    54 FUNC    GLOBAL DEFAULT  UND getsid@GLIBC_2.0 (2)
    83: 08049560   141 FUNC    GLOBAL DEFAULT  UND strncpy@GLIBC_2.0 (2)
    84: 08049570    54 FUNC    GLOBAL DEFAULT  UND dup@GLIBC_2.0 (2)
    85: 08049580   162 FUNC    GLOBAL DEFAULT  UND wcwidth@GLIBC_2.0 (2)
    86: 08072aa4     4 OBJECT  GLOBAL DEFAULT   14 _IO_stdin_used
    87: 08049590   472 FUNC    GLOBAL DEFAULT  UND fpathconf@GLIBC_2.0 (2)
    88: 080495a0    58 FUNC    GLOBAL DEFAULT  UND kill@GLIBC_2.0 (2)
    89: 0806e0ec    98 FUNC    GLOBAL DEFAULT   12 regfree
    90: 080495b0   113 FUNC    GLOBAL DEFAULT  UND __ctype_b_loc@GLIBC_2.3 (3)
    91: 080495c0   154 FUNC    GLOBAL DEFAULT  UND tcgetattr@GLIBC_2.0 (2)
    92: 080495d0   124 FUNC    GLOBAL DEFAULT  UND read@GLIBC_2.0 (2)
    93: 080495e0    54 FUNC    GLOBAL DEFAULT  UND alarm@GLIBC_2.0 (2)
    94: 080495f0   163 FUNC    GLOBAL DEFAULT  UND wait@GLIBC_2.0 (2)
    95: 0806b676    48 FUNC    GLOBAL DEFAULT   12 free
    96: 00000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
    97: 08049600    48 FUNC    GLOBAL DEFAULT  UND strcpy@GLIBC_2.0 (2)
```

## 2.10 Symbol table '.symtab'

```
Symbol table '.symtab' contains 1186 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 080480f4     0 SECTION LOCAL  DEFAULT    1
     2: 08048108     0 SECTION LOCAL  DEFAULT    2
     3: 08048128     0 SECTION LOCAL  DEFAULT    3
     4: 0804843c     0 SECTION LOCAL  DEFAULT    4
     5: 08048a5c     0 SECTION LOCAL  DEFAULT    5
     6: 08048d6a     0 SECTION LOCAL  DEFAULT    6
     7: 08048e30     0 SECTION LOCAL  DEFAULT    7
```

```
 8: 08048e60     0 SECTION LOCAL  DEFAULT    8
 9: 08048e68     0 SECTION LOCAL  DEFAULT    9
10: 080490e8     0 SECTION LOCAL  DEFAULT   10
11: 08049100     0 SECTION LOCAL  DEFAULT   11
12: 08049610     0 SECTION LOCAL  DEFAULT   12
13: 08072a68     0 SECTION LOCAL  DEFAULT   13
14: 08072aa0     0 SECTION LOCAL  DEFAULT   14
15: 08076aec     0 SECTION LOCAL  DEFAULT   15
16: 08077000     0 SECTION LOCAL  DEFAULT   16
17: 08077d28     0 SECTION LOCAL  DEFAULT   17
18: 08077df0     0 SECTION LOCAL  DEFAULT   18
19: 08077df8     0 SECTION LOCAL  DEFAULT   19
20: 08077e00     0 SECTION LOCAL  DEFAULT   20
21: 08077e04     0 SECTION LOCAL  DEFAULT   21
22: 08077f60     0 SECTION LOCAL  DEFAULT   22
23: 00000000     0 SECTION LOCAL  DEFAULT   23
24: 00000000     0 SECTION LOCAL  DEFAULT   24
25: 00000000     0 SECTION LOCAL  DEFAULT   25
26: 00000000     0 SECTION LOCAL  DEFAULT   26
27: 00000000     0 SECTION LOCAL  DEFAULT   27
28: 00000000     0 SECTION LOCAL  DEFAULT   28
29: 00000000     0 SECTION LOCAL  DEFAULT   29
30: 00000000     0 SECTION LOCAL  DEFAULT   30
31: 00000000     0 SECTION LOCAL  DEFAULT   31
32: 00000000     0 SECTION LOCAL  DEFAULT   32
33: 00000000     0 SECTION LOCAL  DEFAULT   33
34: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
35: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
36: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
37: 00000000     0 FILE    LOCAL  DEFAULT  ABS <built-in>
38: 00000000     0 FILE    LOCAL  DEFAULT  ABS abi-note.S
39: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
40: 00000000     0 FILE    LOCAL  DEFAULT  ABS abi-note.S
41: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
42: 00000000     0 FILE    LOCAL  DEFAULT  ABS abi-note.S
43: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
44: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
45: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
46: 00000000     0 FILE    LOCAL  DEFAULT  ABS <built-in>
47: 00000000     0 FILE    LOCAL  DEFAULT  ABS abi-note.S
48: 00000000     0 FILE    LOCAL  DEFAULT  ABS init.c
49: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
50: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
51: 00000000     0 FILE    LOCAL  DEFAULT  ABS initfini.c
52: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
53: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
```

```
54: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
55: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
56: 00000000     0 FILE    LOCAL  DEFAULT  ABS <built-in>
57: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
58: 08049634     0 FUNC    LOCAL  DEFAULT   12 call_gmon_start
59: 00000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
60: 08077df0     0 OBJECT  LOCAL  DEFAULT   18 __CTOR_LIST__
61: 08077df8     0 OBJECT  LOCAL  DEFAULT   19 __DTOR_LIST__
62: 08076aec     0 OBJECT  LOCAL  DEFAULT   15 __EH_FRAME_BEGIN__
63: 08077e00     0 OBJECT  LOCAL  DEFAULT   20 __JCR_LIST__
64: 08077008     0 OBJECT  LOCAL  DEFAULT   16 p.0
65: 08077f60     1 OBJECT  LOCAL  DEFAULT   22 completed.1
66: 08049658     0 FUNC    LOCAL  DEFAULT   12 __do_global_dtors_aux
67: 08049694     0 FUNC    LOCAL  DEFAULT   12 frame_dummy
68: 00000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
69: 08077df4     0 OBJECT  LOCAL  DEFAULT   18 __CTOR_END__
70: 08077dfc     0 OBJECT  LOCAL  DEFAULT   19 __DTOR_END__
71: 08076aec     0 OBJECT  LOCAL  DEFAULT   15 __FRAME_END__
72: 08077e00     0 OBJECT  LOCAL  DEFAULT   20 __JCR_END__
73: 08072a44     0 FUNC    LOCAL  DEFAULT   12 __do_global_ctors_aux
74: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
75: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
76: 00000000     0 FILE    LOCAL  DEFAULT  ABS initfini.c
77: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
78: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
79: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
80: 00000000     0 FILE    LOCAL  DEFAULT  ABS <command line>
81: 00000000     0 FILE    LOCAL  DEFAULT  ABS <built-in>
82: 00000000     0 FILE    LOCAL  DEFAULT  ABS /usr/src/build/231499-i38
83: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex.c
84: 08077f64     4 OBJECT  LOCAL  DEFAULT   22 progname
85: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_addr.c
86: 08077f68     2 OBJECT  LOCAL  DEFAULT   22 bigmove
87: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_cmds.c
88: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_cmds2.c
89: 08077f6c     2 OBJECT  LOCAL  DEFAULT   22 foocmd.0
90: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_cmdsub.c
91: 08077f70     4 OBJECT  LOCAL  DEFAULT   22 jcount
92: 0804d3e2    16 FUNC    LOCAL  DEFAULT   12 jnoop
93: 0804e473    86 FUNC    LOCAL  DEFAULT   12 splitit
94: 0804f06b   196 FUNC    LOCAL  DEFAULT   12 intmac
95: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_data.c
96: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_extern.c
97: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_get.c
98: 08077f80  2048 OBJECT  LOCAL  DEFAULT   22 in_line.0
99: 08078780     2 OBJECT  LOCAL  DEFAULT   22 junkbs
```

```
100: 08078782     2 OBJECT  LOCAL  DEFAULT   22 lastin
101: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_io.c
102: 080787a0    32 OBJECT  LOCAL  DEFAULT   22 fpatbuf.0
103: 080787c0     4 OBJECT  LOCAL  DEFAULT   22 ovro.1
104: 080787c4     4 OBJECT  LOCAL  DEFAULT   22 denied.2
105: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_put.c
106: 080787e0    82 OBJECT  LOCAL  DEFAULT   22 linb
107: 08078832     2 OBJECT  LOCAL  DEFAULT   22 phadnl
108: 08078838     4 OBJECT  LOCAL  DEFAULT   22 plodflg
109: 08078834     4 OBJECT  LOCAL  DEFAULT   22 plodcnt
110: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_re.c
111: 0807883c     4 OBJECT  LOCAL  DEFAULT   22 gsubf.0
112: 08055118   310 FUNC    LOCAL  DEFAULT   12 compile1
113: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_set.c
114: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_subr.c
115: 08078844     2 OBJECT  LOCAL  DEFAULT   22 lastsc
116: 08078840     4 OBJECT  LOCAL  DEFAULT   22 pg.0
117: 08078846     2 OBJECT  LOCAL  DEFAULT   22 vcntcol
118: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_tagio.c
119: 080779f0     4 OBJECT  LOCAL  DEFAULT   16 offset
120: 080779f4     4 OBJECT  LOCAL  DEFAULT   16 block
121: 080779f8     4 OBJECT  LOCAL  DEFAULT   16 bcnt
122: 080779fc     4 OBJECT  LOCAL  DEFAULT   16 b_size
123: 08078848     4 OBJECT  LOCAL  DEFAULT   22 ibuf
124: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_temp.c
125: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_tty.c
126: 0807884c     2 OBJECT  LOCAL  DEFAULT   22 sc.0
127: 08078850     4 OBJECT  LOCAL  DEFAULT   22 costnum
128: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_unix.c
129: 08078854     8 OBJECT  LOCAL  DEFAULT   22 pvec.0
130: 0807885c     8 OBJECT  LOCAL  DEFAULT   22 pvec.1
131: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_v.c
132: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vadj.c
133: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vget.c
134: 08078880    32 OBJECT  LOCAL  DEFAULT   22 pbuf.0
135: 080788a0     8 OBJECT  LOCAL  DEFAULT   22 pend.1
136: 080788a8     8 OBJECT  LOCAL  DEFAULT   22 pcur.2
137: 080788b0    16 OBJECT  LOCAL  DEFAULT   22 state.3
138: 080788c0     8 OBJECT  LOCAL  DEFAULT   22 incompl.4
139: 0805d55e   759 FUNC    LOCAL  DEFAULT   12 readwc
140: 0805e198   104 FUNC    LOCAL  DEFAULT   12 imacpush
141: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vmain.c
142: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_voper.c
143: 08060550   139 FUNC    LOCAL  DEFAULT   12 cblank
144: 080788c8     4 OBJECT  LOCAL  DEFAULT   22 lastFKND.0
145: 080788cc     4 OBJECT  LOCAL  DEFAULT   22 lastFCHR.1
```

```
146: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vops.c
147: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vops2.c
148: 08064ec4   134 FUNC    LOCAL  DEFAULT   12 xgappend
149: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vops3.c
150: 08066dca   489 FUNC    LOCAL  DEFAULT   12 cswitch
151: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vput.c
152: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_vwind.c
153: 00000000     0 FILE    LOCAL  DEFAULT  ABS printf.c
154: 080788d4     4 OBJECT  LOCAL  DEFAULT   22 sign
155: 080788d8     4 OBJECT  LOCAL  DEFAULT   22 fill
156: 080788d0     4 OBJECT  LOCAL  DEFAULT   22 width
157: 0806b05d   399 FUNC    LOCAL  DEFAULT   12 p_emit
158: 00000000     0 FILE    LOCAL  DEFAULT  ABS ex_version.c
159: 08077bd8     4 OBJECT  LOCAL  DEFAULT   16 versionstring
160: 00000000     0 FILE    LOCAL  DEFAULT  ABS mapmalloc.c
161: 0806b214    67 FUNC    LOCAL  DEFAULT   12 map
162: 080788dc     4 OBJECT  LOCAL  DEFAULT   22 temp.0
163: 08077bdc     4 OBJECT  LOCAL  DEFAULT   16 poolblock.1
164: 080788e0     4 OBJECT  LOCAL  DEFAULT   22 pool0
165: 00000000     0 FILE    LOCAL  DEFAULT  ABS termcap.c
166: 0806b864   184 FUNC    LOCAL  DEFAULT   12 tnamatch
167: 080788e4     4 OBJECT  LOCAL  DEFAULT   22 tbuf
168: 0806b91c   684 FUNC    LOCAL  DEFAULT   12 tnchktc
169: 0806be12    42 FUNC    LOCAL  DEFAULT   12 tskip
170: 080788e8     4 OBJECT  LOCAL  DEFAULT   22 hopcount
171: 0806c033   297 FUNC    LOCAL  DEFAULT   12 tdecode
172: 08075a80    38 OBJECT  LOCAL  DEFAULT   14 sccssl
173: 00000000     0 FILE    LOCAL  DEFAULT  ABS tgoto.c
174: 08078900    64 OBJECT  LOCAL  DEFAULT   22 result.0
175: 08078940    10 OBJECT  LOCAL  DEFAULT   22 added.1
176: 00000000     0 FILE    LOCAL  DEFAULT  ABS tputs.c
177: 08077be0    30 OBJECT  LOCAL  DEFAULT   16 tmspc10
178: 00000000     0 FILE    LOCAL  DEFAULT  ABS regcomp.c
179: 00000000     0 FILE    LOCAL  DEFAULT  ABS regdfa.c
180: 0806c724   278 FUNC    LOCAL  DEFAULT   12 copy
181: 0806c83a   953 FUNC    LOCAL  DEFAULT   12 findposn
182: 0806cbf3   374 FUNC    LOCAL  DEFAULT   12 first
183: 0806cd69   133 FUNC    LOCAL  DEFAULT   12 follow
184: 0806cdee   668 FUNC    LOCAL  DEFAULT   12 posnfoll
185: 0806d08a   682 FUNC    LOCAL  DEFAULT   12 addstate
186: 0806d97d   663 FUNC    LOCAL  DEFAULT   12 leftmost
187: 0806dc14   228 FUNC    LOCAL  DEFAULT   12 regdfaexec_opt
188: 00000000     0 FILE    LOCAL  DEFAULT  ABS regerror.c
189: 00000000     0 FILE    LOCAL  DEFAULT  ABS regexec.c
190: 00000000     0 FILE    LOCAL  DEFAULT  ABS regfree.c
191: 00000000     0 FILE    LOCAL  DEFAULT  ABS regnfa.c
```

```
192: 0806e150   167 FUNC    LOCAL  DEFAULT   12 deltolist
193: 0806e1f7    71 FUNC    LOCAL  DEFAULT   12 delgraph
194: 0806e23e   181 FUNC    LOCAL  DEFAULT   12 nopskip
195: 0806e2f3  1078 FUNC    LOCAL  DEFAULT   12 mkgraph
196: 0806e729   174 FUNC    LOCAL  DEFAULT   12 firstop
197: 0806e9e3   162 FUNC    LOCAL  DEFAULT   12 newstck
198: 0806ea85   403 FUNC    LOCAL  DEFAULT   12 mkstck
199: 0806ec18   460 FUNC    LOCAL  DEFAULT   12 newctxt
200: 0806ede4   379 FUNC    LOCAL  DEFAULT   12 casecmp
201: 00000000     0 FILE    LOCAL  DEFAULT  ABS regparse.c
202: 0806fb10  4248 FUNC    LOCAL  DEFAULT   12 lex
203: 08070ba8   649 FUNC    LOCAL  DEFAULT   12 leaf
204: 08070fcf   199 FUNC    LOCAL  DEFAULT   12 alt
205: 08070e31   235 FUNC    LOCAL  DEFAULT   12 post
206: 08070f1c   179 FUNC    LOCAL  DEFAULT   12 cat
207: 00000000     0 FILE    LOCAL  DEFAULT  ABS stubs.c
208: 08077c00   296 OBJECT  LOCAL  DEFAULT   16 curinfo.0
209: 080768a0    39 OBJECT  LOCAL  DEFAULT   14 sccsid
210: 00000000     0 FILE    LOCAL  DEFAULT  ABS bracket.c
211: 080712b0   152 FUNC    LOCAL  DEFAULT   12 addwide
212: 08071348   445 FUNC    LOCAL  DEFAULT   12 addrange
213: 08071505   137 FUNC    LOCAL  DEFAULT   12 place
214: 0807158e   299 FUNC    LOCAL  DEFAULT   12 chcls
215: 080716b9   726 FUNC    LOCAL  DEFAULT   12 mcce
216: 0807198f   562 FUNC    LOCAL  DEFAULT   12 eqcls
217: 08071bc1   179 FUNC    LOCAL  DEFAULT   12 clsym
218: 080768e0   220 OBJECT  LOCAL  DEFAULT   14 zero.0
219: 00000000     0 FILE    LOCAL  DEFAULT  ABS _collelem.c
220: 00000000     0 FILE    LOCAL  DEFAULT  ABS _collmult.c
221: 080603d4   211 FUNC    GLOBAL DEFAULT   12 str2cell
222: 08077bcc     8 OBJECT  GLOBAL DEFAULT   16 vscandir
223: 08056a91   139 FUNC    GLOBAL DEFAULT   12 morelines
224: 08057807   167 FUNC    GLOBAL DEFAULT   12 preserve
225: 08078980   512 OBJECT  GLOBAL DEFAULT   22 rhsquo
226: 080521b8    67 FUNC    GLOBAL DEFAULT   12 putchar
227: 0805c165   480 FUNC    GLOBAL DEFAULT   12 vscrap
228: 0806dcf8   501 FUNC    GLOBAL DEFAULT   12 libuxre_regdfaexec
229: 0805e37e   273 FUNC    GLOBAL DEFAULT   12 macpush
230: 080641b3   135 FUNC    GLOBAL DEFAULT   12 ateopr
231: 08078b80     4 OBJECT  GLOBAL DEFAULT   22 KD
232: 080578d6   394 FUNC    GLOBAL DEFAULT   12 onsusp
233: 08078b84     4 OBJECT  GLOBAL DEFAULT   22 CD
234: 08078b88     4 OBJECT  GLOBAL DEFAULT   22 vtube0
235: 08078ba0   512 OBJECT  GLOBAL DEFAULT   22 DEL
236: 080535b1    15 FUNC    GLOBAL DEFAULT   12 draino
237: 08049110   103 FUNC    GLOBAL DEFAULT  UND iswctype@@GLIBC_2.0
```

```
238: 08078da0     4 OBJECT  GLOBAL DEFAULT   22 vutmp
239: 08078da4     2 OBJECT  GLOBAL DEFAULT   22 outcol
240: 08088e20     4 OBJECT  GLOBAL DEFAULT   22 cntln
241: 08078da6     2 OBJECT  GLOBAL DEFAULT   22 XN
242: 08078da8     4 OBJECT  GLOBAL DEFAULT   22 DOWN_PARM
243: 08067270   131 FUNC    GLOBAL DEFAULT   12 vclear
244: 08078dac     4 OBJECT  GLOBAL DEFAULT   22 addr2
245: 080638ec   564 FUNC    GLOBAL DEFAULT   12 vfilter
246: 08078db0     2 OBJECT  GLOBAL DEFAULT   22 vundkind
247: 08078db4     4 OBJECT  GLOBAL DEFAULT   22 chng
248: 080535c0    59 FUNC    GLOBAL DEFAULT   12 flusho
249: 08049120   671 FUNC    GLOBAL DEFAULT  UND ttyname@@GLIBC_2.0
250: 0804d713   196 FUNC    GLOBAL DEFAULT   12 put
251: 080534f4    74 FUNC    GLOBAL DEFAULT   12 noteinp
252: 0805769e    46 FUNC    GLOBAL DEFAULT   12 markit
253: 08078db8     4 OBJECT  GLOBAL DEFAULT   22 msnext
254: 0805e200   382 FUNC    GLOBAL DEFAULT   12 map
255: 08078dbc     4 OBJECT  GLOBAL DEFAULT   22 vmcurs
256: 080591b5   101 FUNC    GLOBAL DEFAULT   12 rbflush
257: 08078dc0  4120 OBJECT  GLOBAL DEFAULT   22 scanre
258: 0808b160  2048 OBJECT  GLOBAL DEFAULT   22 regrbuf
259: 08057764    75 FUNC    GLOBAL DEFAULT   12 onintr
260: 08079dd8     2 OBJECT  GLOBAL DEFAULT   22 WLINES
261: 08079ddc     4 OBJECT  GLOBAL DEFAULT   22 op
262: 08067c9e   110 FUNC    GLOBAL DEFAULT   12 vprepins
263: 08050726  1047 FUNC    GLOBAL DEFAULT   12 rop
264: 080572f8    35 FUNC    GLOBAL DEFAULT   12 synced
265: 08071096   434 FUNC    GLOBAL DEFAULT   12 libuxre_regparse
266: 0806be3c   227 FUNC    GLOBAL DEFAULT   12 tgetnum
267: 0804cfc5   443 FUNC    GLOBAL DEFAULT   12 delete
268: 08079de0     2 OBJECT  GLOBAL DEFAULT   22 costCM
269: 08079de4     4 OBJECT  GLOBAL DEFAULT   22 LL
270: 0806a92c    37 FUNC    GLOBAL DEFAULT   12 printf
271: 0805d500    26 FUNC    GLOBAL DEFAULT   12 ungetkey
272: 08079de8     2 OBJECT  GLOBAL DEFAULT   22 HADUP
273: 08049130   190 FUNC    GLOBAL DEFAULT  UND towlower@@GLIBC_2.0
274: 08053737    15 FUNC    GLOBAL DEFAULT   12 setoutt
275: 08049140   203 FUNC    GLOBAL DEFAULT  UND mbtowc@@GLIBC_2.0
276: 08079dec     4 OBJECT  GLOBAL DEFAULT   22 lastcnt
277: 08079e00   256 OBJECT  GLOBAL DEFAULT   22 altfile
278: 08079f00     4 OBJECT  GLOBAL DEFAULT   22 args
279: 0805630a    30 FUNC    GLOBAL DEFAULT   12 change
280: 08079f04     4 OBJECT  GLOBAL DEFAULT   22 oldxfsz
281: 08079f08     4 OBJECT  GLOBAL DEFAULT   22 tchng
282: 08057354   155 FUNC    GLOBAL DEFAULT   12 vsmerror
283: 08079f20  2048 OBJECT  GLOBAL DEFAULT   22 linebuf
```

```
284: 08051cc0   300 FUNC    GLOBAL DEFAULT   12 listchar
285: 0805c5c8  1224 FUNC    GLOBAL DEFAULT   12 vredraw
286: 08049150    83 FUNC    GLOBAL DEFAULT  UND sigaction@@GLIBC_2.0
287: 0805b28e   105 FUNC    GLOBAL DEFAULT   12 vintr
288: 0809de5c     4 OBJECT  GLOBAL DEFAULT   22 vUA1
289: 0807a720     2 OBJECT  GLOBAL DEFAULT   22 ninbuf
290: 08049160   319 FUNC    GLOBAL DEFAULT  UND execl@@GLIBC_2.0
291: 08049170   359 FUNC    GLOBAL DEFAULT  UND strchr@@GLIBC_2.0
292: 0805e51e    46 FUNC    GLOBAL DEFAULT   12 trapalarm
293: 08059c14    47 FUNC    GLOBAL DEFAULT   12 fkey
294: 0807a722     2 OBJECT  GLOBAL DEFAULT   22 CA
295: 0807a724     4 OBJECT  GLOBAL DEFAULT   22 KL
296: 0807a728     2 OBJECT  GLOBAL DEFAULT   22 argc0
297: 0807a72c     4 OBJECT  GLOBAL DEFAULT   22 argv
298: 0805a838   634 FUNC    GLOBAL DEFAULT   12 oop
299: 0805bea1    28 FUNC    GLOBAL DEFAULT   12 vup1
300: 0808afc0     4 OBJECT  GLOBAL DEFAULT   22 stotal
301: 0806629b   421 FUNC    GLOBAL DEFAULT   12 lindent
302: 0806722f    63 FUNC    GLOBAL DEFAULT   12 wsamechar
303: 080568dc    61 FUNC    GLOBAL DEFAULT   12 markreg
304: 0804d900   424 FUNC    GLOBAL DEFAULT   12 shift
305: 08063862   138 FUNC    GLOBAL DEFAULT   12 vshftop
306: 08059300   240 FUNC    GLOBAL DEFAULT   12 gettmode
307: 0807a730     2 OBJECT  GLOBAL DEFAULT   22 nleft
308: 0807a732     2 OBJECT  GLOBAL DEFAULT   22 WCOLS
309: 0807a734     4 OBJECT  GLOBAL DEFAULT   22 pid
310: 0809de60     4 OBJECT  GLOBAL DEFAULT   22 vUD2
311: 08063cdb    27 FUNC    GLOBAL DEFAULT   12 vshift
312: 0804d180    62 FUNC    GLOBAL DEFAULT   12 deletenone
313: 08066fb3   355 FUNC    GLOBAL DEFAULT   12 vswitch
314: 0806d334   197 FUNC    GLOBAL DEFAULT   12 libuxre_regdeldfa
315: 0805883d   409 FUNC    GLOBAL DEFAULT   12 regio
316: 08057113    54 FUNC    GLOBAL DEFAULT   12 reverse
317: 08058109   123 FUNC    GLOBAL DEFAULT   12 xgetline
318: 080578ae    40 FUNC    GLOBAL DEFAULT   12 exitex
319: 080676b3    67 FUNC    GLOBAL DEFAULT   12 vcursbef
320: 0807a740  2560 OBJECT  GLOBAL DEFAULT   22 arrows
321: 0807b140     2 OBJECT  GLOBAL DEFAULT   22 costLP
322: 080677c7    62 FUNC    GLOBAL DEFAULT   12 vsetcurs
323: 08049180    46 FUNC    GLOBAL DEFAULT  UND getpid@@GLIBC_2.0
324: 08077d28     0 OBJECT  GLOBAL DEFAULT   17 _DYNAMIC
325: 0804cbab   256 FUNC    GLOBAL DEFAULT   12 vcontin
326: 08050b3d   238 FUNC    GLOBAL DEFAULT   12 rop2
327: 080506d2    84 FUNC    GLOBAL DEFAULT   12 samei
328: 0804e21c    99 FUNC    GLOBAL DEFAULT   12 yank
329: 080526e5  2730 FUNC    GLOBAL DEFAULT   12 plod
```

```
330: 0807b160   108 OBJECT  GLOBAL DEFAULT   22 names
331: 0807b1cc     4 OBJECT  GLOBAL DEFAULT   22 Peekkey
332: 0805687c    25 FUNC    GLOBAL DEFAULT   12 markDOT
333: 0804d1be   154 FUNC    GLOBAL DEFAULT   12 squish
334: 0807b1d0     4 OBJECT  GLOBAL DEFAULT   22 firstln
335: 0807b1d4     2 OBJECT  GLOBAL DEFAULT   22 dosusp
336: 0807b1d8     4 OBJECT  GLOBAL DEFAULT   22 F5
337: 08049190   100 FUNC    GLOBAL DEFAULT  UND siglongjmp@@GLIBC_2.0
338: 0804a42e  1064 FUNC    GLOBAL DEFAULT   12 address
339: 08056863    25 FUNC    GLOBAL DEFAULT   12 lineDOT
340: 08065ba3    63 FUNC    GLOBAL DEFAULT   12 vgetsplit
341: 0806c15c   745 FUNC    GLOBAL DEFAULT   12 tgoto
342: 0809de84     4 OBJECT  GLOBAL DEFAULT   22 linend
343: 0807b1e0  4120 OBJECT  GLOBAL DEFAULT   22 re
344: 0804a28c    76 FUNC    GLOBAL DEFAULT   12 setdot1
345: 0806fa2c   103 FUNC    GLOBAL DEFAULT   12 libuxre_reg1tree
346: 0805b39c   569 FUNC    GLOBAL DEFAULT   12 vopen
347: 08054cc4   270 FUNC    GLOBAL DEFAULT   12 fixcase
348: 08068679   421 FUNC    GLOBAL DEFAULT   12 vneedpos
349: 0805854d   112 FUNC    GLOBAL DEFAULT   12 tflush
350: 0807c1f8     2 OBJECT  GLOBAL DEFAULT   22 seenprompt
351: 0807c1fc     4 OBJECT  GLOBAL DEFAULT   22 F4
352: 0804c9b9    23 FUNC    GLOBAL DEFAULT   12 tailspec
353: 0807c200     4 OBJECT  GLOBAL DEFAULT   22 F9
354: 080522dd    18 FUNC    GLOBAL DEFAULT   12 flush
355: 0807c204     4 OBJECT  GLOBAL DEFAULT   22 ND
356: 08069d17    22 FUNC    GLOBAL DEFAULT   12 vputch
357: 080668e1   186 FUNC    GLOBAL DEFAULT   12 lskipa1
358: 0809dda0     2 OBJECT  GLOBAL DEFAULT   22 precbksl
359: 0806d3f9   506 FUNC    GLOBAL DEFAULT   12 regtrans
360: 080491a0   124 FUNC    GLOBAL DEFAULT  UND write@@GLIBC_2.0
361: 0807c208     4 OBJECT  GLOBAL DEFAULT   22 cntrlhm
362: 0806a8c3   104 FUNC    GLOBAL DEFAULT   12 vnline
363: 0807c20c     2 OBJECT  GLOBAL DEFAULT   22 holdupd
364: 0807c210     4 OBJECT  GLOBAL DEFAULT   22 CE
365: 0807c214     4 OBJECT  GLOBAL DEFAULT   22 peekc
366: 0804e838    40 FUNC    GLOBAL DEFAULT   12 pofix
367: 0807c218     2 OBJECT  GLOBAL DEFAULT   22 oblock
368: 0807c21c    20 OBJECT  GLOBAL DEFAULT   22 lastcmd
369: 0808afe0    36 OBJECT  GLOBAL DEFAULT   22 braelist
370: 08054f4b    99 FUNC    GLOBAL DEFAULT   12 snote
371: 0806699b   627 FUNC    GLOBAL DEFAULT   12 lnext
372: 080491b0    37 FUNC    GLOBAL DEFAULT  UND strcmp@@GLIBC_2.0
373: 0807c230     2 OBJECT  GLOBAL DEFAULT   22 OS
374: 0804e860   221 FUNC    GLOBAL DEFAULT   12 somechange
375: 0805dcb6    23 FUNC    GLOBAL DEFAULT   12 peekkey
```

```
376: 080589d6   163 FUNC    GLOBAL DEFAULT   12 REGblk
377: 0806442a  2667 FUNC    GLOBAL DEFAULT   12 vappend
378: 0805ffd2   174 FUNC    GLOBAL DEFAULT   12 vremote
379: 080496e2    48 FUNC    GLOBAL DEFAULT   12 usage
380: 080565bf   112 FUNC    GLOBAL DEFAULT   12 getn
381: 0804c8bf    58 FUNC    GLOBAL DEFAULT   12 serror
382: 080491c0   113 FUNC    GLOBAL DEFAULT  UND close@@GLIBC_2.0
383: 0807c232     2 OBJECT  GLOBAL DEFAULT   22 intty
384: 08072aa0     4 OBJECT  GLOBAL DEFAULT   14 _fp_hw
385: 0807c234     2 OBJECT  GLOBAL DEFAULT   22 wdkind
386: 0807c238     8 OBJECT  GLOBAL DEFAULT   22 pkill
387: 0807c240     2 OBJECT  GLOBAL DEFAULT   22 destline
388: 0808b960  2048 OBJECT  GLOBAL DEFAULT   22 putrbuf
389: 0806655b   468 FUNC    GLOBAL DEFAULT   12 lsmatch
390: 0804d6cb    72 FUNC    GLOBAL DEFAULT   12 getput
391: 0805655c    26 FUNC    GLOBAL DEFAULT   12 getDOT
392: 080539e1   238 FUNC    GLOBAL DEFAULT   12 tostart
393: 08077320   256 OBJECT  GLOBAL DEFAULT   16 shell
394: 0805318f   847 FUNC    GLOBAL DEFAULT   12 fgoto
395: 0805cc99   217 FUNC    GLOBAL DEFAULT   12 vadjDL
396: 0809de88     4 OBJECT  GLOBAL DEFAULT   22 insmc0
397: 0807c242     2 OBJECT  GLOBAL DEFAULT   22 vch_mac
398: 080522ef   972 FUNC    GLOBAL DEFAULT   12 flush1
399: 080491d0    90 FUNC    GLOBAL DEFAULT  UND fork@@GLIBC_2.0
400: 0807c244     2 OBJECT  GLOBAL DEFAULT   22 defwind
401: 0807c246     2 OBJECT  GLOBAL DEFAULT   22 verbose
402: 080491e0   275 FUNC    GLOBAL DEFAULT  UND getenv@@GLIBC_2.0
403: 0807c260  8192 OBJECT  GLOBAL DEFAULT   22 imapspace
404: 0807e260     4 OBJECT  GLOBAL DEFAULT   22 exitoneof
405: 0804c525   108 FUNC    GLOBAL DEFAULT   12 fixol
406: 08056c54   396 FUNC    GLOBAL DEFAULT   12 printof
407: 080604cd    41 FUNC    GLOBAL DEFAULT   12 cellcpy
408: 0807e264     2 OBJECT  GLOBAL DEFAULT   22 rubble
409: 0805bebd   317 FUNC    GLOBAL DEFAULT   12 vmoveitup
410: 08053715    34 FUNC    GLOBAL DEFAULT   12 putpad
411: 0807e268     4 OBJECT  GLOBAL DEFAULT   22 vSCROLL
412: 0807e26c     4 OBJECT  GLOBAL DEFAULT   22 SF
413: 0807226d    86 FUNC    GLOBAL DEFAULT   12 libuxre_bktfree
414: 0804c7be   105 FUNC    GLOBAL DEFAULT   12 nomore
415: 080692ec    51 FUNC    GLOBAL DEFAULT   12 endim
416: 0807e270     4 OBJECT  GLOBAL DEFAULT   22 VE
417: 0807e274     4 OBJECT  GLOBAL DEFAULT   22 F1
418: 080491f0    54 FUNC    GLOBAL DEFAULT  UND unlink@@GLIBC_2.0
419: 0807e278     4 OBJECT  GLOBAL DEFAULT   22 VS
420: 08088e24     4 OBJECT  GLOBAL DEFAULT   22 nextip
421: 08077000     0 NOTYPE  GLOBAL DEFAULT  ABS __fini_array_end
```

```
422: 0805e48f   143 FUNC    GLOBAL DEFAULT   12 vgetcnt
423: 08054039   385 FUNC    GLOBAL DEFAULT   12 substitute
424: 0808b004     4 OBJECT  GLOBAL DEFAULT   22 scount
425: 0805b2f7   104 FUNC    GLOBAL DEFAULT   12 vsetsiz
426: 08088e28     4 OBJECT  GLOBAL DEFAULT   22 oldadot
427: 0806b257  1055 FUNC    GLOBAL DEFAULT   12 malloc
428: 080572b4    31 FUNC    GLOBAL DEFAULT   12 saveall
429: 080729f4    37 FUNC    GLOBAL HIDDEN    12 __stat
430: 08057e00   637 FUNC    GLOBAL DEFAULT   12 fileinit
431: 080574f2   257 FUNC    GLOBAL DEFAULT   12 vfindcol
432: 080604f6    38 FUNC    GLOBAL DEFAULT   12 cellen
433: 080779e0     4 OBJECT  GLOBAL DEFAULT   16 Putchar
434: 0806bf1f   134 FUNC    GLOBAL DEFAULT   12 tgetflag
435: 08057416    23 FUNC    GLOBAL DEFAULT   12 strend
436: 08077a40   236 OBJECT  GLOBAL DEFAULT   16 sstrs
437: 08059c43    81 FUNC    GLOBAL DEFAULT   12 cost
438: 0804a270    28 FUNC    GLOBAL DEFAULT   12 setdot
439: 0808b008     4 OBJECT  GLOBAL DEFAULT   22 loc2
440: 080564de    50 FUNC    GLOBAL DEFAULT   12 filioerr
441: 0807e280   400 OBJECT  GLOBAL DEFAULT   22 vtube
442: 08065d5c  1071 FUNC    GLOBAL DEFAULT   12 llfind
443: 08049712    43 FUNC    GLOBAL DEFAULT   12 needarg
444: 08053797    18 FUNC    GLOBAL DEFAULT   12 putNFL
445: 0807e410     2 OBJECT  GLOBAL DEFAULT   22 tflag
446: 0807e412     2 OBJECT  GLOBAL DEFAULT   22 vcatch
447: 08061df5   169 FUNC    GLOBAL DEFAULT   12 eend
448: 0807e414     4 OBJECT  GLOBAL DEFAULT   22 input
449: 080519fe   190 FUNC    GLOBAL DEFAULT   12 checkmodeline
450: 08049200   177 FUNC    GLOBAL DEFAULT  UND strerror@@GLIBC_2.0
451: 080573ef    39 FUNC    GLOBAL DEFAULT   12 smerror
452: 08051605    72 FUNC    GLOBAL DEFAULT   12 wrerror
453: 0807e418     4 OBJECT  GLOBAL DEFAULT   22 mb_cur_max
454: 0807e41c     2 OBJECT  GLOBAL DEFAULT   22 WBOT
455: 0807e420     4 OBJECT  GLOBAL DEFAULT   22 unddol
456: 08077004     0 OBJECT  GLOBAL HIDDEN    16 __dso_handle
457: 080585bd   593 FUNC    GLOBAL DEFAULT   12 synctmp
458: 0807e424     2 OBJECT  GLOBAL DEFAULT   22 inglobal
459: 08051dec   521 FUNC    GLOBAL DEFAULT   12 normchar
460: 08058bfc    57 FUNC    GLOBAL DEFAULT   12 shread
461: 0807e440    60 OBJECT  GLOBAL DEFAULT   22 normf
462: 08049210    54 FUNC    GLOBAL DEFAULT  UND mmap@@GLIBC_2.0
463: 0805d101   807 FUNC    GLOBAL DEFAULT   12 vreplace
464: 080729c0    52 FUNC    GLOBAL DEFAULT   12 __libc_csu_fini
465: 0806a0be   430 FUNC    GLOBAL DEFAULT   12 vdown
466: 0807e47c     4 OBJECT  GLOBAL DEFAULT   22 Xcnt
467: 0809de8c     4 OBJECT  GLOBAL DEFAULT   22 tabsize
```

```
468: 0808b00c     2 OBJECT  GLOBAL DEFAULT   22 destuc
469: 0809de90     4 OBJECT  GLOBAL DEFAULT   22 inssiz
470: 08056328    37 FUNC    GLOBAL DEFAULT   12 column
471: 0804c976    67 FUNC    GLOBAL DEFAULT   12 skipend
472: 08049220   388 FUNC    GLOBAL DEFAULT  UND tcsetattr@@GLIBC_2.0
473: 0808c160     4 OBJECT  GLOBAL DEFAULT   22 rfname
474: 0807e480     4 OBJECT  GLOBAL DEFAULT   22 F7
475: 0806278c   582 FUNC    GLOBAL DEFAULT   12 vmacchng
476: 0807e484     4 OBJECT  GLOBAL DEFAULT   22 oldhup
477: 0806a5bb    77 FUNC    GLOBAL DEFAULT   12 vback
478: 0809ddc0   156 OBJECT  GLOBAL DEFAULT   22 readbuf
479: 0807e488     2 OBJECT  GLOBAL DEFAULT   22 heldech
480: 08049230  1631 FUNC    GLOBAL DEFAULT  UND setlocale@@GLIBC_2.0
481: 080543bd   709 FUNC    GLOBAL DEFAULT   12 comprhs
482: 0807e48c     4 OBJECT  GLOBAL DEFAULT   22 IM
483: 08069a3b   732 FUNC    GLOBAL DEFAULT   12 physdc
484: 0807e490     2 OBJECT  GLOBAL DEFAULT   22 costRP
485: 08077520   256 OBJECT  GLOBAL DEFAULT   16 ttylongname
486: 08049240   120 FUNC    GLOBAL DEFAULT  UND creat@@GLIBC_2.0
487: 08063cf6   495 FUNC    GLOBAL DEFAULT   12 vrep
488: 0807e494     4 OBJECT  GLOBAL DEFAULT   22 rpid
489: 0805a088   707 FUNC    GLOBAL DEFAULT   12 unixex
490: 0807e498     2 OBJECT  GLOBAL DEFAULT   22 argc
491: 0806a674   188 FUNC    GLOBAL DEFAULT   12 vroll
492: 08064133   128 FUNC    GLOBAL DEFAULT   12 takeout
493: 0804e27f   500 FUNC    GLOBAL DEFAULT   12 zop
494: 0807e4a0  1024 OBJECT  GLOBAL DEFAULT   22 tspace
495: 08049250    57 FUNC    GLOBAL DEFAULT  UND __errno_location@@GLIBC_2
496: 080538a1   128 FUNC    GLOBAL DEFAULT   12 ttcharoff
497: 0808c164     2 OBJECT  GLOBAL DEFAULT   22 rnext
498: 0807e8a0     4 OBJECT  GLOBAL DEFAULT   22 notesgn
499: 08056ba4    34 FUNC    GLOBAL DEFAULT   12 netchHAD
500: 08062170  1564 FUNC    GLOBAL DEFAULT   12 vundo
501: 08059ba6   110 FUNC    GLOBAL DEFAULT   12 gettlongname
502: 0809de94     4 OBJECT  GLOBAL DEFAULT   22 shft
503: 0809deb8     2 OBJECT  GLOBAL DEFAULT   22 ospeed
504: 0809de98     4 OBJECT  GLOBAL DEFAULT   22 tabslack
505: 08053921   192 FUNC    GLOBAL DEFAULT   12 ostart
506: 0807e8a4     2 OBJECT  GLOBAL DEFAULT   22 XB
507: 080572d3    37 FUNC    GLOBAL DEFAULT   12 span
508: 0807e8a8     4 OBJECT  GLOBAL DEFAULT   22 DL
509: 0806e7d7   140 FUNC    GLOBAL DEFAULT   12 libuxre_regdelnfa
510: 0805df27    24 FUNC    GLOBAL DEFAULT   12 setDEL
511: 0808c166     2 OBJECT  GLOBAL DEFAULT   22 stilinc
512: 0807e8ac     4 OBJECT  GLOBAL DEFAULT   22 truedol
513: 08067519   337 FUNC    GLOBAL DEFAULT   12 vclrech
```

```
514: 08049260   211 FUNC    GLOBAL DEFAULT  UND iswalnum@@GLIBC_2.0
515: 080550e5    51 FUNC    GLOBAL DEFAULT   12 resre
516: 0807e8b0     2 OBJECT  GLOBAL DEFAULT   22 pfast
517: 0809deb0     4 OBJECT  GLOBAL DEFAULT   22 BC
518: 08077420   256 OBJECT  GLOBAL DEFAULT   16 tags
519: 08061ef7   176 FUNC    GLOBAL DEFAULT   12 wordch
520: 0808b00e     2 OBJECT  GLOBAL DEFAULT   22 cflag
521: 0807e8b2     2 OBJECT  GLOBAL DEFAULT   22 ruptible
522: 08049270    58 FUNC    GLOBAL DEFAULT  UND chmod@@GLIBC_2.0
523: 0806672f    24 FUNC    GLOBAL DEFAULT   12 ltosolid
524: 08049280   100 FUNC    GLOBAL DEFAULT  UND tolower@@GLIBC_2.0
525: 08053642   211 FUNC    GLOBAL DEFAULT   12 putch
526: 08059c94    14 FUNC    GLOBAL DEFAULT   12 countnum
527: 0807e8b4     4 OBJECT  GLOBAL DEFAULT   22 vglobp
528: 0807e8b8     2 OBJECT  GLOBAL DEFAULT   22 Xhadcnt
529: 0804cea8    59 FUNC    GLOBAL DEFAULT   12 appendnone
530: 08049290    58 FUNC    GLOBAL DEFAULT  UND access@@GLIBC_2.0
531: 080563bd    38 FUNC    GLOBAL DEFAULT   12 copyw
532: 0807e8ba     2 OBJECT  GLOBAL DEFAULT   22 vcnt
533: 0805c345   643 FUNC    GLOBAL DEFAULT   12 vrepaint
534: 08051ff5   143 FUNC    GLOBAL DEFAULT   12 slobber
535: 0807e8bc     4 OBJECT  GLOBAL DEFAULT   22 globp
536: 0808c168     4 OBJECT  GLOBAL DEFAULT   22 rbuf
537: 0807e8c0     4 OBJECT  GLOBAL DEFAULT   22 lastcp
538: 080629e6   551 FUNC    GLOBAL DEFAULT   12 vmove
539: 08058c35   423 FUNC    GLOBAL DEFAULT   12 putreg
540: 080490e8     0 FUNC    GLOBAL DEFAULT   10 _init
541: 0807e8c4     2 OBJECT  GLOBAL DEFAULT   22 oprompt
542: 08057624    40 FUNC    GLOBAL DEFAULT   12 vpastwh
543: 0806051c    50 FUNC    GLOBAL DEFAULT   12 cellcat
544: 0807e8c6     2 OBJECT  GLOBAL DEFAULT   22 numberf
545: 0806423a   365 FUNC    GLOBAL DEFAULT   12 showmode
546: 0807e8c8     2 OBJECT  GLOBAL DEFAULT   22 UL
547: 0807e8cc     4 OBJECT  GLOBAL DEFAULT   22 state
548: 0807e8d0     2 OBJECT  GLOBAL DEFAULT   22 lasthad
549: 0805ab11   166 FUNC    GLOBAL DEFAULT   12 ovend
550: 08056067    81 FUNC    GLOBAL DEFAULT   12 setend
551: 080692b9    51 FUNC    GLOBAL DEFAULT   12 goim
552: 0807e8d4     4 OBJECT  GLOBAL DEFAULT   22 SC
553: 080492a0    16 FUNC    GLOBAL DEFAULT  UND cfgetospeed@@GLIBC_2.0
554: 0807e8d8     4 OBJECT  GLOBAL DEFAULT   22 HO
555: 08088e2c     2 OBJECT  GLOBAL DEFAULT   22 slevel
556: 0806a41a   331 FUNC    GLOBAL DEFAULT   12 vshow
557: 0804c8f9   125 FUNC    GLOBAL DEFAULT   12 setflav
558: 0806b1ec    40 FUNC    GLOBAL DEFAULT   12 printver
559: 0807e8dc     4 OBJECT  GLOBAL DEFAULT   22 CM
```

```
560: 0805dc3e   120 FUNC    GLOBAL DEFAULT   12 getesc
561: 080593f0  1302 FUNC    GLOBAL DEFAULT   12 setterm
562: 0807e8e0     4 OBJECT  GLOBAL DEFAULT   22 DC
563: 080629d2    20 FUNC    GLOBAL DEFAULT   12 vnoapp
564: 0807e8e4     2 OBJECT  GLOBAL DEFAULT   22 WECHO
565: 0807e8e8     4 OBJECT  GLOBAL DEFAULT   22 RIGHT_PARM
566: 0805486e    37 FUNC    GLOBAL DEFAULT   12 ugo
567: 08088e2e     2 OBJECT  GLOBAL DEFAULT   22 isalt
568: 0806618b   194 FUNC    GLOBAL DEFAULT   12 endsent
569: 0807e8ec     4 OBJECT  GLOBAL DEFAULT   22 TA
570: 0807e8f0     2 OBJECT  GLOBAL DEFAULT   22 DA
571: 0805cd72    42 FUNC    GLOBAL DEFAULT   12 vsyncCL
572: 0807e8f2     2 OBJECT  GLOBAL DEFAULT   22 HADZERO
573: 0804c4aa   123 FUNC    GLOBAL DEFAULT   12 erewind
574: 0807e8f4     2 OBJECT  GLOBAL DEFAULT   22 vreg
575: 080537a9    29 FUNC    GLOBAL DEFAULT   12 sTTY
576: 08057490    98 FUNC    GLOBAL DEFAULT   12 tabcol
577: 0804eccf   924 FUNC    GLOBAL DEFAULT   12 mapcmd
578: 0804c881    62 FUNC    GLOBAL DEFAULT   12 resetflav
579: 080492b0    54 FUNC    GLOBAL DEFAULT  UND pipe@@GLIBC_2.0
580: 0806b7ea   109 FUNC    GLOBAL DEFAULT   12 calloc
581: 080541ba   515 FUNC    GLOBAL DEFAULT   12 compsub
582: 0807e900  2048 OBJECT  GLOBAL DEFAULT   22 ibuff2
583: 0804a868    18 FUNC    GLOBAL DEFAULT   12 setNAEOL
584: 08077640   920 OBJECT  GLOBAL DEFAULT   16 options
585: 0805880e    47 FUNC    GLOBAL DEFAULT   12 TSYNC
586: 0804fdda    70 FUNC    GLOBAL DEFAULT   12 setin
587: 0807896c     4 OBJECT  GLOBAL DEFAULT   22 tad1
588: 0807f100     2 OBJECT  GLOBAL DEFAULT   22 HC
589: 0806a565    86 FUNC    GLOBAL DEFAULT   12 vreset
590: 0808c16c     4 OBJECT  GLOBAL DEFAULT   22 havetmp
591: 0807f102     2 OBJECT  GLOBAL DEFAULT   22 basWLINES
592: 0804fe20   692 FUNC    GLOBAL DEFAULT   12 filename
593: 0804f9da   101 FUNC    GLOBAL DEFAULT   12 smunch
594: 08065b62    65 FUNC    GLOBAL DEFAULT   12 vdoappend
595: 0809dd00   156 OBJECT  GLOBAL DEFAULT   22 venv
596: 0804f12f  1172 FUNC    GLOBAL DEFAULT   12 addmac1
597: 0807f104     4 OBJECT  GLOBAL DEFAULT   22 CL
598: 08050635   157 FUNC    GLOBAL DEFAULT   12 getone
599: 0807f108     2 OBJECT  GLOBAL DEFAULT   22 CDCNT
600: 08062042    47 FUNC    GLOBAL DEFAULT   12 margin
601: 080492c0   211 FUNC    GLOBAL DEFAULT  UND iswdigit@@GLIBC_2.0
602: 0805143d   456 FUNC    GLOBAL DEFAULT   12 putfile
603: 08065be2   377 FUNC    GLOBAL DEFAULT   12 vmaxrep
604: 0807f10c     4 OBJECT  GLOBAL DEFAULT   22 FO
605: 0807f110     2 OBJECT  GLOBAL DEFAULT   22 io
```

```
606: 080561f0   158 FUNC    GLOBAL DEFAULT   12 propt
607: 08053746    50 FUNC    GLOBAL DEFAULT   12 vlprintf
608: 0804c9d0    26 FUNC    GLOBAL DEFAULT   12 tail
609: 08056b1c    42 FUNC    GLOBAL DEFAULT   12 nonzero
610: 08051c4c    58 FUNC    GLOBAL DEFAULT   12 setlist
611: 080537c6   132 FUNC    GLOBAL DEFAULT   12 pstart
612: 0806ef5f  2602 FUNC    GLOBAL DEFAULT   12 libuxre_regnfaexec
613: 08072a1c    37 FUNC    WEAK   HIDDEN   12 fstat
614: 0804cee3   226 FUNC    GLOBAL DEFAULT   12 pargs
615: 080534de    22 FUNC    GLOBAL DEFAULT   12 tab
616: 0809de9c     4 OBJECT  GLOBAL DEFAULT   22 slakused
617: 0807f120   256 OBJECT  GLOBAL DEFAULT   22 file
618: 0809dea0     4 OBJECT  GLOBAL DEFAULT   22 inscol
619: 0807f220     4 OBJECT  GLOBAL DEFAULT   22 fendcore
620: 0805731b    57 FUNC    GLOBAL DEFAULT   12 skipwh
621: 08056407    35 FUNC    GLOBAL DEFAULT   12 ctlof
622: 0809de64     4 OBJECT  GLOBAL DEFAULT   22 vUD1
623: 0805921a   229 FUNC    GLOBAL DEFAULT   12 regbuf
624: 08056777   192 FUNC    GLOBAL DEFAULT   12 killcnt
625: 0807f240  2560 OBJECT  GLOBAL DEFAULT   22 abbrevs
626: 08053acf    79 FUNC    GLOBAL DEFAULT   12 ostop
627: 08066861   104 FUNC    GLOBAL DEFAULT   12 lskipbal
628: 08056290    56 FUNC    GLOBAL DEFAULT   12 any
629: 0805471a   340 FUNC    GLOBAL DEFAULT   12 confirmed
630: 0806a951  1544 FUNC    GLOBAL DEFAULT   12 vprintf
631: 0807fc40     2 OBJECT  GLOBAL DEFAULT   22 ichang2
632: 08071248    10 FUNC    GLOBAL DEFAULT   12 libuxre_lc_collate
633: 0806624d    78 FUNC    GLOBAL DEFAULT   12 endPS
634: 0805662f    43 FUNC    GLOBAL DEFAULT   12 ignnEOF
635: 0807fc42     2 OBJECT  GLOBAL DEFAULT   22 WTOP
636: 08078956    20 OBJECT  GLOBAL DEFAULT   22 tcommand
637: 08056e61    33 FUNC    GLOBAL DEFAULT   12 plural
638: 08050d31   362 FUNC    GLOBAL DEFAULT   12 rop3
639: 0807fc44     4 OBJECT  GLOBAL DEFAULT   22 DO
640: 08069d30    50 FUNC    GLOBAL DEFAULT   12 vmoveto
641: 0807fc48     2 OBJECT  GLOBAL DEFAULT   22 costDP
642: 0807fc4a     2 OBJECT  GLOBAL DEFAULT   22 GT
643: 0805bc7c   168 FUNC    GLOBAL DEFAULT   12 vopenup
644: 08077a04     4 OBJECT  GLOBAL DEFAULT   16 rfile
645: 0808c170     4 OBJECT  GLOBAL DEFAULT   22 rbufcp
646: 080492d0    53 FUNC    GLOBAL DEFAULT  UND __sigsetjmp@@GLIBC_2.0
647: 08057448    72 FUNC    GLOBAL DEFAULT   12 syserror
648: 0805353e   115 FUNC    GLOBAL DEFAULT   12 termreset
649: 080492e0   806 FUNC    GLOBAL DEFAULT  UND __xstat@@GLIBC_2.0
650: 080678dd   885 FUNC    GLOBAL DEFAULT   12 vgoto
651: 080492f0   241 FUNC    GLOBAL DEFAULT  UND wctype@@GLIBC_2.0
```

```
652: 08049300     6 FUNC    GLOBAL DEFAULT  UND sysconf@@GLIBC_2.0
653: 080502d7    76 FUNC    GLOBAL DEFAULT   12 gscan
654: 0805764c    82 FUNC    GLOBAL DEFAULT   12 whitecnt
655: 08056b46    57 FUNC    GLOBAL DEFAULT   12 notable
656: 08066747   282 FUNC    GLOBAL DEFAULT   12 ltosol1
657: 0804d7d7   297 FUNC    GLOBAL DEFAULT   12 pragged
658: 0807fc4c     4 OBJECT  GLOBAL DEFAULT   22 vUNDcurs
659: 0804d5df   171 FUNC    GLOBAL DEFAULT   12 move
660: 08069dbb   101 FUNC    GLOBAL DEFAULT   12 vupdown
661: 0807fc50     2 OBJECT  GLOBAL DEFAULT   22 laste
662: 080550b4    49 FUNC    GLOBAL DEFAULT   12 savere
663: 0807fc52     2 OBJECT  GLOBAL DEFAULT   22 recov
664: 080560b8   172 FUNC    GLOBAL DEFAULT   12 prall
665: 0807fc54     2 OBJECT  GLOBAL DEFAULT   22 XT
666: 0807fc58     4 OBJECT  GLOBAL DEFAULT   22 DM
667: 0805164d   901 FUNC    GLOBAL DEFAULT   12 source
668: 0805bffa   363 FUNC    GLOBAL DEFAULT   12 vscroll
669: 0807fc5c     2 OBJECT  GLOBAL DEFAULT   22 holdcm
670: 0806a7ed    68 FUNC    GLOBAL DEFAULT   12 vcookit
671: 0806a831   146 FUNC    GLOBAL DEFAULT   12 vdepth
672: 0807fc60  2048 OBJECT  GLOBAL DEFAULT   22 obuff
673: 0804f6c0    13 FUNC    GLOBAL DEFAULT   12 ignchar
674: 08080460     2 OBJECT  GLOBAL DEFAULT   22 costAL
675: 08058e16   187 FUNC    GLOBAL DEFAULT   12 getREG
676: 080590ae    83 FUNC    GLOBAL DEFAULT   12 kshift
677: 08080480   156 OBJECT  GLOBAL DEFAULT   22 resetlab
678: 0804a856    18 FUNC    GLOBAL DEFAULT   12 setCNL
679: 0809dea4     4 OBJECT  GLOBAL DEFAULT   22 tabstart
680: 0808051c     4 OBJECT  GLOBAL DEFAULT   22 maphopcnt
681: 08080520     2 OBJECT  GLOBAL DEFAULT   22 inopen
682: 08049310    64 FUNC    GLOBAL DEFAULT  UND time@@GLIBC_2.0
683: 08080522     2 OBJECT  GLOBAL DEFAULT   22 gobblebl
684: 08080524     2 OBJECT  GLOBAL DEFAULT   22 aiflag
685: 0805b8f9   899 FUNC    GLOBAL DEFAULT   12 vinslin
686: 0804c827    90 FUNC    GLOBAL DEFAULT   12 quickly
687: 08058b03   249 FUNC    GLOBAL DEFAULT   12 KILLreg
688: 08049610     0 FUNC    GLOBAL DEFAULT   12 _start
689: 080500d4   515 FUNC    GLOBAL DEFAULT   12 getargs
690: 0805dccd   437 FUNC    GLOBAL DEFAULT   12 readecho
691: 08080528     4 OBJECT  GLOBAL DEFAULT   22 cursor
692: 08061e9e    89 FUNC    GLOBAL DEFAULT   12 wordof
693: 08077120   256 OBJECT  GLOBAL DEFAULT   16 paragraphs
694: 0806c448   418 FUNC    GLOBAL DEFAULT   12 tputs
695: 08056465   121 FUNC    GLOBAL DEFAULT   12 fixindent
696: 0808052c     2 OBJECT  GLOBAL DEFAULT   22 insmode
697: 0806fa93   125 FUNC    GLOBAL DEFAULT   12 libuxre_reg2tree
```

```
698: 0808b010     4 OBJECT  GLOBAL DEFAULT   22 loc1
699: 0805e608  6389 FUNC    GLOBAL DEFAULT   12 vmain
700: 08056576    73 FUNC    GLOBAL DEFAULT   12 getmark
701: 08066440   283 FUNC    GLOBAL DEFAULT   12 lmatchp
702: 0805e54c   187 FUNC    GLOBAL DEFAULT   12 fastpeekkey
703: 080577af    88 FUNC    GLOBAL DEFAULT   12 setrupt
704: 08080530     4 OBJECT  GLOBAL DEFAULT   22 TRIM
705: 08080540   512 OBJECT  GLOBAL DEFAULT   22 INS
706: 08056164   140 FUNC    GLOBAL DEFAULT   12 propts
707: 08056e02    95 FUNC    GLOBAL DEFAULT   12 putmk1
708: 0805a802    51 FUNC    GLOBAL DEFAULT   12 revocer
709: 08080740     4 OBJECT  GLOBAL DEFAULT   22 KS
710: 08072724   490 FUNC    GLOBAL DEFAULT   12 libuxre_collelem
711: 08049320    54 FUNC    GLOBAL DEFAULT  UND chdir@@GLIBC_2.0
712: 0808b014     4 OBJECT  GLOBAL DEFAULT   22 slines
713: 08049330   806 FUNC    GLOBAL DEFAULT  UND __fxstat@@GLIBC_2.0
714: 08077220   256 OBJECT  GLOBAL DEFAULT   16 sections
715: 08080760  8192 OBJECT  GLOBAL DEFAULT   22 genbuf
716: 08082760     4 OBJECT  GLOBAL DEFAULT   22 BT
717: 0804f99a    32 FUNC    GLOBAL DEFAULT   12 peekchar
718: 08082764     2 OBJECT  GLOBAL DEFAULT   22 basWTOP
719: 08077020   256 OBJECT  GLOBAL DEFAULT   16 direct
720: 08072910   127 FUNC    GLOBAL DEFAULT   12 libuxre_collmult
721: 08082768     4 OBJECT  GLOBAL DEFAULT   22 SR
722: 0805a3e1   636 FUNC    GLOBAL DEFAULT   12 filter
723: 0809dce4     4 OBJECT  GLOBAL DEFAULT   22 xPC
724: 08054fae    29 FUNC    GLOBAL DEFAULT   12 cerror
725: 08056bc6   142 FUNC    GLOBAL DEFAULT   12 netchange
726: 0808276c     2 OBJECT  GLOBAL DEFAULT   22 destcol
727: 08049340   175 FUNC    GLOBAL DEFAULT  UND strlen@@GLIBC_2.0
728: 0808c180 67584 OBJECT  GLOBAL DEFAULT   22 incorb
729: 08049350   513 FUNC    GLOBAL DEFAULT  UND sleep@@GLIBC_2.0
730: 08060080   247 FUNC    GLOBAL DEFAULT   12 vsave
731: 0808276e     2 OBJECT  GLOBAL DEFAULT   22 hold
732: 0804fa3f   923 FUNC    GLOBAL DEFAULT   12 gettty
733: 08051c86    58 FUNC    GLOBAL DEFAULT   12 setnumb
734: 08069203    94 FUNC    GLOBAL DEFAULT   12 godm
735: 08049360   121 FUNC    GLOBAL DEFAULT  UND sbrk@@GLIBC_2.0
736: 08062e6d  1767 FUNC    GLOBAL DEFAULT   12 vchange
737: 08049370    74 FUNC    GLOBAL DEFAULT  UND sigaddset@@GLIBC_2.0
738: 08082770     4 OBJECT  GLOBAL DEFAULT   22 SE
739: 08088e40  8544 OBJECT  GLOBAL DEFAULT   22 G
740: 0807894c     2 OBJECT  GLOBAL DEFAULT   22 nflag
741: 08049380   211 FUNC    GLOBAL DEFAULT  UND iswupper@@GLIBC_2.0
742: 08061957   320 FUNC    GLOBAL DEFAULT   12 find
743: 08082774     4 OBJECT  GLOBAL DEFAULT   22 wdot
```

```
744: 08056895    71 FUNC    GLOBAL DEFAULT   12 markpr
745: 08082778     2 OBJECT  GLOBAL DEFAULT   22 AM
746: 0808277c     4 OBJECT  GLOBAL DEFAULT   22 IC
747: 0806b6a6   324 FUNC    GLOBAL DEFAULT   12 realloc
748: 0805ca90   521 FUNC    GLOBAL DEFAULT   12 vdellin
749: 0804c449    97 FUNC    GLOBAL DEFAULT   12 verror
750: 08056ffa   281 FUNC    GLOBAL DEFAULT   12 qcount
751: 080779ec     4 OBJECT  GLOBAL DEFAULT   16 obp
752: 08082780     2 OBJECT  GLOBAL DEFAULT   22 writing
753: 0804daa8  1908 FUNC    GLOBAL DEFAULT   12 tagfind
754: 0809de78     4 OBJECT  GLOBAL DEFAULT   22 llimit
755: 08078970     2 OBJECT  GLOBAL DEFAULT   22 zhadpr
756: 08082782     2 OBJECT  GLOBAL DEFAULT   22 undkind
757: 08049390   379 FUNC    GLOBAL DEFAULT  UND mbrtowc@@GLIBC_2.0
758: 0809c980  2048 OBJECT  GLOBAL DEFAULT   22 YANKrbuf
759: 08078972     2 OBJECT  GLOBAL DEFAULT   22 znoclear
760: 08082784     2 OBJECT  GLOBAL DEFAULT   22 anymarks
761: 08053c64   822 FUNC    GLOBAL DEFAULT   12 global
762: 08082788     4 OBJECT  GLOBAL DEFAULT   22 initev
763: 080569ba   176 FUNC    GLOBAL DEFAULT   12 vmerror
764: 0808278c     4 OBJECT  GLOBAL DEFAULT   22 imsnext
765: 08082790     4 OBJECT  GLOBAL DEFAULT   22 TI
766: 0809deba     1 OBJECT  GLOBAL DEFAULT   22 PC
767: 080493a0   211 FUNC    GLOBAL DEFAULT  UND iswspace@@GLIBC_2.0
768: 08063ff4    63 FUNC    GLOBAL DEFAULT   12 bleep
769: 0806a26c   372 FUNC    GLOBAL DEFAULT   12 vcontext
770: 080827a0  2048 OBJECT  GLOBAL DEFAULT   22 mapspace
771: 0808b018     4 OBJECT  GLOBAL DEFAULT   22 casecnt
772: 08082fa0   128 OBJECT  GLOBAL DEFAULT   22 vmacbuf
773: 0805b5d5   405 FUNC    GLOBAL DEFAULT   12 vreopen
774: 0804981c   136 FUNC    GLOBAL DEFAULT   12 setsig
775: 08083020     4 OBJECT  GLOBAL DEFAULT   22 xchng
776: 080493b0   157 FUNC    GLOBAL DEFAULT  UND wctomb@@GLIBC_2.0
777: 080498a4   127 FUNC    GLOBAL DEFAULT   12 init
778: 08062074   252 FUNC    GLOBAL DEFAULT   12 vUndo
779: 08057c0a   163 FUNC    GLOBAL DEFAULT   12 tseek
780: 0804a2d8   148 FUNC    GLOBAL DEFAULT   12 setcount
781: 08067e2a  2014 FUNC    GLOBAL DEFAULT   12 vinschar
782: 0809de7c     4 OBJECT  GLOBAL DEFAULT   22 lf
783: 080519d2    44 FUNC    GLOBAL DEFAULT   12 clrstats
784: 08083024     4 OBJECT  GLOBAL DEFAULT   22 notenam
785: 08059ca4   996 FUNC    GLOBAL DEFAULT   12 unix0
786: 0804f6cd   510 FUNC    GLOBAL DEFAULT   12 getach
787: 08056a6a    39 FUNC    GLOBAL DEFAULT   12 merror
788: 08083028     2 OBJECT  GLOBAL DEFAULT   22 outline
789: 08083040   612 OBJECT  GLOBAL DEFAULT   22 vlinfo
```

```
790: 080832a4     4 OBJECT  GLOBAL DEFAULT   22 vcolbp
791: 0805a65d   287 FUNC    GLOBAL DEFAULT   12 recover
792: 080832a8     2 OBJECT  GLOBAL DEFAULT   22 anyabbrs
793: 08050f17  1045 FUNC    GLOBAL DEFAULT   12 wop
794: 080526bb    42 FUNC    GLOBAL DEFAULT   12 plodput
795: 080832aa     2 OBJECT  GLOBAL DEFAULT   22 xHZ
796: 08077a20     4 OBJECT  GLOBAL DEFAULT   16 ATTN
797: 08058df5    33 FUNC    GLOBAL DEFAULT   12 notpart
798: 080832ac     2 OBJECT  GLOBAL DEFAULT   22 morargc
799: 080493c0    76 FUNC    GLOBAL DEFAULT  UND sigprocmask@@GLIBC_2.0
800: 080832b0     4 OBJECT  GLOBAL DEFAULT   22 LEFT_PARM
801: 08061a97   862 FUNC    GLOBAL DEFAULT   12 xword
802: 0805699f    27 FUNC    GLOBAL DEFAULT   12 merror1
803: 08053c15    27 FUNC    GLOBAL DEFAULT   12 gTTY
804: 080493d0   179 FUNC    GLOBAL DEFAULT  UND strncmp@@GLIBC_2.0
805: 08053c30    50 FUNC    GLOBAL DEFAULT   12 noonl
806: 080832b4     2 OBJECT  GLOBAL DEFAULT   22 ichanged
807: 0804c17d    38 FUNC    GLOBAL DEFAULT   12 eol
808: 08053b1e    83 FUNC    GLOBAL DEFAULT   12 tostop
809: 08054682    66 FUNC    GLOBAL DEFAULT   12 getsub
810: 08057295    31 FUNC    GLOBAL DEFAULT   12 save12
811: 08077000     0 NOTYPE  GLOBAL DEFAULT  ABS __fini_array_start
812: 0804a3c0    67 FUNC    GLOBAL DEFAULT   12 setall
813: 080832c0   156 OBJECT  GLOBAL DEFAULT   22 vreslab
814: 08078950     4 OBJECT  GLOBAL DEFAULT   22 poffset
815: 08067314   129 FUNC    GLOBAL DEFAULT   12 vclrlin
816: 08056919   134 FUNC    GLOBAL DEFAULT   12 mesg
817: 0805ae54   105 FUNC    GLOBAL DEFAULT   12 fixzero
818: 08072990    48 FUNC    GLOBAL DEFAULT   12 __libc_csu_init
819: 08057149   332 FUNC    GLOBAL DEFAULT   12 save
820: 0806778e    32 FUNC    GLOBAL DEFAULT   12 vcursaft
821: 0806c5ec   311 FUNC    GLOBAL DEFAULT   12 regcomp
822: 08083360  2048 OBJECT  GLOBAL DEFAULT   22 obuf
823: 080546c4    86 FUNC    GLOBAL DEFAULT   12 dosubcon
824: 080604a7    38 FUNC    GLOBAL DEFAULT   12 cell2str
825: 08071c74  1529 FUNC    GLOBAL DEFAULT   12 libuxre_bktmbcomp
826: 08077f54     0 NOTYPE  GLOBAL DEFAULT  ABS __bss_start
827: 080672f3    33 FUNC    GLOBAL DEFAULT   12 vclrcell
828: 080493e0   113 FUNC    GLOBAL DEFAULT  UND fsync@@GLIBC_2.0
829: 08067c52    76 FUNC    GLOBAL DEFAULT   12 vgotab
830: 08083b60     2 OBJECT  GLOBAL DEFAULT   22 vcline
831: 08083b64     4 OBJECT  GLOBAL DEFAULT   22 SO
832: 080779e8     4 OBJECT  GLOBAL DEFAULT   16 linp
833: 08049923  2379 FUNC    GLOBAL DEFAULT   12 main
834: 0804c11f    94 FUNC    GLOBAL DEFAULT   12 endcmd
835: 0805807d   140 FUNC    GLOBAL DEFAULT   12 cleanup
```

```
836: 08083b68     4 OBJECT  GLOBAL DEFAULT   22 lastvgk
837: 08083b6c     4 OBJECT  GLOBAL DEFAULT   22 bastate
838: 0805a34b   150 FUNC    GLOBAL DEFAULT   12 unixwt
839: 08078974     2 OBJECT  GLOBAL DEFAULT   22 zweight
840: 08083b70     4 OBJECT  GLOBAL DEFAULT   22 IP
841: 08083b74     4 OBJECT  GLOBAL DEFAULT   22 vUNDdot
842: 0804c2a6   329 FUNC    GLOBAL DEFAULT   12 error1
843: 0805aebd    54 FUNC    GLOBAL DEFAULT   12 savevis
844: 08056399    36 FUNC    GLOBAL DEFAULT   12 Copy
845: 08083b78     4 OBJECT  GLOBAL DEFAULT   22 dirtcnt
846: 0809d180     4 OBJECT  GLOBAL DEFAULT   22 tfname
847: 08069e20   670 FUNC    GLOBAL DEFAULT   12 vup
848: 08077bc8     4 OBJECT  GLOBAL DEFAULT   16 doingread
849: 0806e863   384 FUNC    GLOBAL DEFAULT   12 libuxre_regnfacomp
850: 0809d1a0   288 OBJECT  GLOBAL DEFAULT   22 strregs
851: 08053610    50 FUNC    GLOBAL DEFAULT   12 putS
852: 080493f0   251 FUNC    GLOBAL DEFAULT  UND __libc_start_main@@GLIBC_
853: 08083b7c     4 OBJECT  GLOBAL DEFAULT   22 F8
854: 0805d4ac    82 FUNC    GLOBAL DEFAULT   12 vdirty
855: 08064f4a  3096 FUNC    GLOBAL DEFAULT   12 vgetline
856: 08049400    54 FUNC    GLOBAL DEFAULT  UND getpgid@@GLIBC_2.0
857: 08083b80  2068 OBJECT  GLOBAL DEFAULT   22 H
858: 080843a0   108 OBJECT  GLOBAL DEFAULT   22 ncols
859: 0804f8cb    35 FUNC    GLOBAL DEFAULT   12 getchar
860: 0805be0a   151 FUNC    GLOBAL DEFAULT   12 vrollup
861: 0806bbc8   586 FUNC    GLOBAL DEFAULT   12 tgetent
862: 08077000     0 NOTYPE  GLOBAL DEFAULT  ABS __init_array_end
863: 0808440c     2 OBJECT  GLOBAL DEFAULT   22 costSR
864: 0808440e     2 OBJECT  GLOBAL DEFAULT   22 listf
865: 08084410     2 OBJECT  GLOBAL DEFAULT   22 BS
866: 0806dfd4   277 FUNC    GLOBAL DEFAULT   12 regexec
867: 0808afa0     2 OBJECT  GLOBAL DEFAULT   22 wasalt
868: 0805b35f    60 FUNC    GLOBAL DEFAULT   12 onwinch
869: 08084420   512 OBJECT  GLOBAL DEFAULT   22 rhsbuf
870: 0806788e    79 FUNC    GLOBAL DEFAULT   12 vigotoCL
871: 08084620     4 OBJECT  GLOBAL DEFAULT   22 linebp
872: 08084624    20 OBJECT  GLOBAL DEFAULT   22 workcmd
873: 08084638     4 OBJECT  GLOBAL DEFAULT   22 UP_PARM
874: 0805665a    79 FUNC    GLOBAL DEFAULT   12 is_white
875: 08049410   100 FUNC    GLOBAL DEFAULT  UND toupper@@GLIBC_2.0
876: 0808463c     4 OBJECT  GLOBAL DEFAULT   22 bsize
877: 08066d4f   123 FUNC    GLOBAL DEFAULT   12 isa
878: 0804d3f2   493 FUNC    GLOBAL DEFAULT   12 move1
879: 0805dff1   357 FUNC    GLOBAL DEFAULT   12 noteit
880: 08077bd4     2 OBJECT  GLOBAL DEFAULT   16 vshnam
881: 0804d68a    65 FUNC    GLOBAL DEFAULT   12 getcopy
```

```
882: 08084640     2 OBJECT  GLOBAL DEFAULT   22 fixedzero
883: 0804d258   394 FUNC    GLOBAL DEFAULT   12 xjoin
884: 08053778    31 FUNC    GLOBAL DEFAULT   12 lprintf
885: 08084660  2048 OBJECT  GLOBAL DEFAULT   22 ibuff
886: 08084e60     4 OBJECT  GLOBAL DEFAULT   22 args0
887: 08055a7a   412 FUNC    GLOBAL DEFAULT   12 execute
888: 08084e64     4 OBJECT  GLOBAL DEFAULT   22 undadot
889: 08049797   133 FUNC    GLOBAL DEFAULT   12 iownit
890: 08049420   426 FUNC    GLOBAL DEFAULT  UND strcat@@GLIBC_2.0
891: 08084e68     2 OBJECT  GLOBAL DEFAULT   22 dir
892: 08084e6a     2 OBJECT  GLOBAL DEFAULT   22 splitw
893: 08084e6c     2 OBJECT  GLOBAL DEFAULT   22 iblock
894: 080779e4     4 OBJECT  GLOBAL DEFAULT   16 Pline
895: 08067d0c   286 FUNC    GLOBAL DEFAULT   12 vmaktop
896: 0805384a    87 FUNC    GLOBAL DEFAULT   12 pstop
897: 08056de0    34 FUNC    GLOBAL DEFAULT   12 putmark
898: 08066c0e   321 FUNC    GLOBAL DEFAULT   12 lbrack
899: 08053b71    62 FUNC    GLOBAL DEFAULT   12 normal
900: 08084e70     4 OBJECT  GLOBAL DEFAULT   22 wcursor
901: 08067805    23 FUNC    GLOBAL DEFAULT   12 vigoto
902: 08059906   544 FUNC    GLOBAL DEFAULT   12 setsize
903: 08062c0d   608 FUNC    GLOBAL DEFAULT   12 vdelete
904: 08084e74     2 OBJECT  GLOBAL DEFAULT   22 XX
905: 08084e78     4 OBJECT  GLOBAL DEFAULT   22 xCR
906: 08077000     0 NOTYPE  WEAK   DEFAULT   16 data_start
907: 0806af59   260 FUNC    GLOBAL DEFAULT   12 p_dconv
908: 0804f8ee    54 FUNC    GLOBAL DEFAULT   12 checkjunk
909: 08049430   190 FUNC    GLOBAL DEFAULT  UND towupper@@GLIBC_2.0
910: 0809d2c0  2048 OBJECT  GLOBAL DEFAULT   22 KILLrbuf
911: 08084e80   130 OBJECT  GLOBAL DEFAULT   22 uxb
912: 0805b0cc   450 FUNC    GLOBAL DEFAULT   12 vok
913: 08084f02     8 OBJECT  GLOBAL DEFAULT   22 winsz
914: 08084f0c     4 OBJECT  GLOBAL DEFAULT   22 vmacp
915: 08069d62    89 FUNC    GLOBAL DEFAULT   12 vjumpto
916: 08049440    67 FUNC    GLOBAL DEFAULT  UND getuid@@GLIBC_2.0
917: 08072a68     0 FUNC    GLOBAL DEFAULT   13 _fini
918: 08049450   124 FUNC    GLOBAL DEFAULT  UND lseek@@GLIBC_2.0
919: 08049460    39 FUNC    GLOBAL DEFAULT  UND memcpy@@GLIBC_2.0
920: 0809dac0     4 OBJECT  GLOBAL DEFAULT   22 strp
921: 08063b20   443 FUNC    GLOBAL DEFAULT   12 xdw
922: 08084f10     2 OBJECT  GLOBAL DEFAULT   22 doomed
923: 0804c1a3   259 FUNC    GLOBAL DEFAULT   12 error0
924: 080643a7   131 FUNC    GLOBAL DEFAULT   12 addc
925: 08084f14     4 OBJECT  GLOBAL DEFAULT   22 failed
926: 08068c3a  1481 FUNC    GLOBAL DEFAULT   12 viin
927: 08084f18     2 OBJECT  GLOBAL DEFAULT   22 iblock2
```

```
928: 08084f1c     4 OBJECT  GLOBAL DEFAULT   22 xNL
929: 08084f20     4 OBJECT  GLOBAL DEFAULT   22 firstpat
930: 080522c6    23 FUNC    GLOBAL DEFAULT   12 flush2
931: 08049470   441 FUNC    GLOBAL DEFAULT  UND strrchr@@GLIBC_2.0
932: 08084f24     4 OBJECT  GLOBAL DEFAULT   22 zero
933: 08084f40  8544 OBJECT  GLOBAL DEFAULT   22 frob
934: 080870a0     2 OBJECT  GLOBAL DEFAULT   22 EO
935: 0809de68     4 OBJECT  GLOBAL DEFAULT   22 vUA2
936: 080668c9    24 FUNC    GLOBAL DEFAULT   12 lskipatom
937: 08056752    37 FUNC    GLOBAL DEFAULT   12 killed
938: 08049480   381 FUNC    GLOBAL DEFAULT  UND btowc@@GLIBC_2.0
939: 0806def0   225 FUNC    GLOBAL DEFAULT   12 regerror
940: 080563e3    36 FUNC    GLOBAL DEFAULT   12 copywR
941: 080870a4     4 OBJECT  GLOBAL DEFAULT   22 KE
942: 080870a8     2 OBJECT  GLOBAL DEFAULT   22 lastreg
943: 080870ac     4 OBJECT  GLOBAL DEFAULT   22 dol
944: 080870b0     4 OBJECT  GLOBAL DEFAULT   22 KH
945: 080870b4     2 OBJECT  GLOBAL DEFAULT   22 hitin2
946: 0805aef3   154 FUNC    GLOBAL DEFAULT   12 undvis
947: 0806783f    79 FUNC    GLOBAL DEFAULT   12 vgotoCL
948: 0804c6d7   231 FUNC    GLOBAL DEFAULT   12 newline
949: 0804c3ef    90 FUNC    GLOBAL DEFAULT   12 error
950: 0809de80     2 OBJECT  GLOBAL DEFAULT   22 wasend
951: 08053baf   102 FUNC    GLOBAL DEFAULT   12 setty
952: 080870b6     2 OBJECT  GLOBAL DEFAULT   22 inappend
953: 080870b8     4 OBJECT  GLOBAL DEFAULT   22 endcore
954: 0806a608   108 FUNC    GLOBAL DEFAULT   12 vfit
955: 0809de6c     2 OBJECT  GLOBAL DEFAULT   22 gobbled
956: 0804c5fa   221 FUNC    GLOBAL DEFAULT   12 next
957: 080520c3   245 FUNC    GLOBAL DEFAULT   12 normline
958: 080870bc     4 OBJECT  GLOBAL DEFAULT   22 one
959: 0805d51a    37 FUNC    GLOBAL DEFAULT   12 getkey
960: 080870c0     2 OBJECT  GLOBAL DEFAULT   22 DB
961: 08061fa7   155 FUNC    GLOBAL DEFAULT   12 edge
962: 080870c4     4 OBJECT  GLOBAL DEFAULT   22 unddel
963: 08063ee5   220 FUNC    GLOBAL DEFAULT   12 vyankit
964: 08049490   124 FUNC    GLOBAL DEFAULT  UND open@@GLIBC_2.0
965: 080870c8     4 OBJECT  GLOBAL DEFAULT   22 F3
966: 080870cc     4 OBJECT  GLOBAL DEFAULT   22 dot
967: 0809dae0   512 OBJECT  GLOBAL DEFAULT   22 rused
968: 080870d0     2 OBJECT  GLOBAL DEFAULT   22 NC
969: 080521fb   203 FUNC    GLOBAL DEFAULT   12 termchar
970: 08063554   782 FUNC    GLOBAL DEFAULT   12 voOpen
971: 0808afa4     4 OBJECT  GLOBAL DEFAULT   22 cntnull
972: 0804f9ba    32 FUNC    GLOBAL DEFAULT   12 peekcd
973: 08054dd2   377 FUNC    GLOBAL DEFAULT   12 place
```

```
 974: 080870d2     2 OBJECT  GLOBAL DEFAULT   22 MI
 975: 080494a0    82 FUNC    GLOBAL DEFAULT  UND sigemptyset@@GLIBC_2.0
 976: 08051b29    60 FUNC    GLOBAL DEFAULT   12 widthok
 977: 0805636e    43 FUNC    GLOBAL DEFAULT   12 comment
 978: 080494b0    61 FUNC    GLOBAL DEFAULT  UND __ctype_get_mb_cur_max@@G
 979: 08078954     2 OBJECT  GLOBAL DEFAULT   22 pflag
 980: 0804c0d8    71 FUNC    GLOBAL DEFAULT   12 cmdreg
 981: 080870d4     2 OBJECT  GLOBAL DEFAULT   22 normtty
 982: 0809de6e     2 OBJECT  GLOBAL DEFAULT   22 vaifirst
 983: 080870d8     4 OBJECT  GLOBAL DEFAULT   22 KU
 984: 080870dc     4 OBJECT  GLOBAL DEFAULT   22 otchng
 985: 080870e0     4 OBJECT  GLOBAL DEFAULT   22 F6
 986: 0805dede    73 FUNC    GLOBAL DEFAULT   12 addtext
 987: 0805b76a   399 FUNC    GLOBAL DEFAULT   12 vglitchup
 988: 0806a730   189 FUNC    GLOBAL DEFAULT   12 vrollR
 989: 0809de70     4 OBJECT  GLOBAL DEFAULT   22 ogcursor
 990: 0805bd24   230 FUNC    GLOBAL DEFAULT   12 vadjAL
 991: 08059101   180 FUNC    GLOBAL DEFAULT   12 YANKline
 992: 080779d8     4 OBJECT  GLOBAL DEFAULT   16 lastc
 993: 080870e4     4 OBJECT  GLOBAL DEFAULT   22 DL_PARM
 994: 0804f5c3   252 FUNC    GLOBAL DEFAULT   12 cmdmac
 995: 080494c0    45 FUNC    GLOBAL DEFAULT  UND atoi@@GLIBC_2.0
 996: 080494d0   211 FUNC    GLOBAL DEFAULT  UND iswalpha@@GLIBC_2.0
 997: 0808b060   256 OBJECT  GLOBAL DEFAULT   22 optname
 998: 0804e7b8   128 FUNC    GLOBAL DEFAULT   12 plines
 999: 0809dea8     4 OBJECT  GLOBAL DEFAULT   22 tabend
1000: 08052084    63 FUNC    GLOBAL DEFAULT   12 numbline
1001: 080870e8     4 OBJECT  GLOBAL DEFAULT   22 tline
1002: 0804e4c9   751 FUNC    GLOBAL DEFAULT   12 zop2
1003: 080870ec     2 OBJECT  GLOBAL DEFAULT   22 hadcnt
1004: 080729f4    37 FUNC    WEAK   HIDDEN   12 stat
1005: 0809deac     4 OBJECT  GLOBAL DEFAULT   22 insmc1
1006: 08058441   243 FUNC    GLOBAL DEFAULT   12 blkio
1007: 08064033   256 FUNC    GLOBAL DEFAULT   12 vdcMID
1008: 0808afa8     4 OBJECT  GLOBAL DEFAULT   22 cntch
1009: 0805df76   123 FUNC    GLOBAL DEFAULT   12 addto
1010: 0805d855  1001 FUNC    GLOBAL DEFAULT   12 getbr
1011: 0809dce0     2 OBJECT  GLOBAL DEFAULT   22 rnleft
1012: 08051b65   228 FUNC    GLOBAL DEFAULT   12 GETWC
1013: 0808b020    36 OBJECT  GLOBAL DEFAULT   22 braslist
1014: 080870f0     4 OBJECT  GLOBAL DEFAULT   22 EI
1015: 080870f4     4 OBJECT  GLOBAL DEFAULT   22 AL
1016: 080870f8     2 OBJECT  GLOBAL DEFAULT   22 NS
1017: 080494e0   211 FUNC    GLOBAL DEFAULT  UND iswprint@@GLIBC_2.0
1018: 08077b40    68 OBJECT  GLOBAL DEFAULT   16 sflags
1019: 080870fa     2 OBJECT  GLOBAL DEFAULT   22 UPPERCASE
```

```
1020: 0804c5bb    63 FUNC    GLOBAL DEFAULT   12 makargs
1021: 08049768    47 FUNC    GLOBAL DEFAULT   12 tailpath
1022: 0805742d    27 FUNC    GLOBAL DEFAULT   12 strcLIN
1023: 08077f54     0 NOTYPE  GLOBAL DEFAULT  ABS _edata
1024: 080677ae    25 FUNC    GLOBAL DEFAULT   12 vfixcurs
1025: 0804a36c    84 FUNC    GLOBAL DEFAULT   12 getnum
1026: 08077e04     0 OBJECT  GLOBAL DEFAULT   21 _GLOBAL_OFFSET_TABLE_
1027: 08087100   255 OBJECT  GLOBAL DEFAULT   22 ttynbuf
1028: 0809debc     0 NOTYPE  GLOBAL DEFAULT  ABS _end
1029: 08087200     2 OBJECT  GLOBAL DEFAULT   22 vmovcol
1030: 080494f0    60 FUNC    GLOBAL DEFAULT  UND ioctl@@GLIBC_2.0
1031: 0805de82    92 FUNC    GLOBAL DEFAULT   12 setLAST
1032: 08087220    60 OBJECT  GLOBAL DEFAULT   22 tty
1033: 0808725c     4 OBJECT  GLOBAL DEFAULT   22 KR
1034: 08071252    93 FUNC    GLOBAL DEFAULT   12 libuxre_mb2wc
1035: 0806781c    35 FUNC    GLOBAL DEFAULT   12 vcsync
1036: 080576cc   152 FUNC    GLOBAL DEFAULT   12 onhup
1037: 080722c3  1119 FUNC    GLOBAL DEFAULT   12 libuxre_bktmbexec
1038: 08059b26   128 FUNC    GLOBAL DEFAULT   12 zap
1039: 08049500   218 FUNC    GLOBAL DEFAULT  UND getcwd@@GLIBC_2.0
1040: 0806f98c   160 FUNC    GLOBAL DEFAULT   12 libuxre_regdeltree
1041: 08049510   211 FUNC    GLOBAL DEFAULT  UND iswlower@@GLIBC_2.0
1042: 0804c591    42 FUNC    GLOBAL DEFAULT   12 exclam
1043: 08058ed1   477 FUNC    GLOBAL DEFAULT   12 YANKreg
1044: 0805d0b8    73 FUNC    GLOBAL DEFAULT   12 vcloseup
1045: 0806d5f3   906 FUNC    GLOBAL DEFAULT   12 libuxre_regdfacomp
1046: 08054fcb   233 FUNC    GLOBAL DEFAULT   12 refree
1047: 08049520    53 FUNC    GLOBAL DEFAULT  UND isatty@@GLIBC_2.0
1048: 0805634d    33 FUNC    GLOBAL DEFAULT   12 lcolumn
1049: 08087260     1 OBJECT  GLOBAL DEFAULT   22 lastmac
1050: 08087264     4 OBJECT  GLOBAL DEFAULT   22 Vlines
1051: 080671ad   130 FUNC    GLOBAL DEFAULT   12 wskipright
1052: 08049530    67 FUNC    GLOBAL DEFAULT  UND memset@@GLIBC_2.0
1053: 08077a00     4 OBJECT  GLOBAL DEFAULT   16 tfile
1054: 08049540    19 FUNC    GLOBAL DEFAULT  UND _exit@@GLIBC_2.0
1055: 08087268     2 OBJECT  GLOBAL DEFAULT   22 hush
1056: 080562c8    66 FUNC    GLOBAL DEFAULT   12 backtab
1057: 08057af2   132 FUNC    GLOBAL DEFAULT   12 safecat
1058: 0804ccab   137 FUNC    GLOBAL DEFAULT   12 vnfl
1059: 0807700c     2 OBJECT  GLOBAL DEFAULT   16 endline
1060: 0805a77c   134 FUNC    GLOBAL DEFAULT   12 waitfor
1061: 0805132c   273 FUNC    GLOBAL DEFAULT   12 getfile
1062: 0805d53f    31 FUNC    GLOBAL DEFAULT   12 peekbr
1063: 08049550    54 FUNC    GLOBAL DEFAULT  UND getsid@@GLIBC_2.0
1064: 0804cd34   372 FUNC    GLOBAL DEFAULT   12 append
1065: 0805aab2    95 FUNC    GLOBAL DEFAULT   12 ovbeg
```

```
1066: 0808726c     4 OBJECT  GLOBAL DEFAULT   22 ppid
1067: 0804f924   118 FUNC    GLOBAL DEFAULT   12 getcd
1068: 0806894c   750 FUNC    GLOBAL DEFAULT   12 vishft
1069: 0809dce2     2 OBJECT  GLOBAL DEFAULT   22 rblock
1070: 08087270     2 OBJECT  GLOBAL DEFAULT   22 shudclob
1071: 08054893  1073 FUNC    GLOBAL DEFAULT   12 dosub
1072: 08087274     4 OBJECT  GLOBAL DEFAULT   22 addr1
1073: 0804a87c  6234 FUNC    GLOBAL DEFAULT   12 commands
1074: 0809de74     4 OBJECT  GLOBAL DEFAULT   22 vsplitpt
1075: 08049560   141 FUNC    GLOBAL DEFAULT  UND strncpy@@GLIBC_2.0
1076: 08087278     2 OBJECT  GLOBAL DEFAULT   22 vmoving
1077: 08053f9a   159 FUNC    GLOBAL DEFAULT   12 gdelete
1078: 08057cad   287 FUNC    GLOBAL DEFAULT   12 tgets
1079: 0806881e   302 FUNC    GLOBAL DEFAULT   12 vnpins
1080: 0806a3e0    58 FUNC    GLOBAL DEFAULT   12 vclean
1081: 0808727c     4 OBJECT  GLOBAL DEFAULT   22 RC
1082: 08087280     4 OBJECT  GLOBAL DEFAULT   22 F2
1083: 0808afac     2 OBJECT  GLOBAL DEFAULT   22 ttyindes
1084: 080575f3    49 FUNC    GLOBAL DEFAULT   12 vskipwh
1085: 0804e93d   914 FUNC    GLOBAL DEFAULT   12 undo
1086: 08087284     2 OBJECT  GLOBAL DEFAULT   22 edited
1087: 08049570    54 FUNC    GLOBAL DEFAULT  UND dup@@GLIBC_2.0
1088: 080605db  4988 FUNC    GLOBAL DEFAULT   12 operate
1089: 08049580   162 FUNC    GLOBAL DEFAULT  UND wcwidth@@GLIBC_2.0
1090: 0808afb0     4 OBJECT  GLOBAL DEFAULT   22 altdot
1091: 08057a60    41 FUNC    GLOBAL DEFAULT   12 movestr
1092: 0805cdcc   748 FUNC    GLOBAL DEFAULT   12 vsync1
1093: 08058ddc    25 FUNC    GLOBAL DEFAULT   12 partreg
1094: 08077000     0 NOTYPE  GLOBAL DEFAULT  ABS __init_array_start
1095: 08072a1c    37 FUNC    GLOBAL HIDDEN    12 __fstat
1096: 08051abc   109 FUNC    GLOBAL DEFAULT   12 mbtowi
1097: 08058a79   138 FUNC    GLOBAL DEFAULT   12 mapreg
1098: 08087288     4 OBJECT  GLOBAL DEFAULT   22 undap2
1099: 08050c2b   262 FUNC    GLOBAL DEFAULT   12 iostats
1100: 08067395   388 FUNC    GLOBAL DEFAULT   12 vclreol
1101: 08072aa4     4 OBJECT  GLOBAL DEFAULT   14 _IO_stdin_used
1102: 08056b7f    37 FUNC    GLOBAL DEFAULT   12 notempty
1103: 0804c9ea    26 FUNC    GLOBAL DEFAULT   12 tail2of
1104: 08058534    25 FUNC    GLOBAL DEFAULT   12 tlaste
1105: 08049590   472 FUNC    GLOBAL DEFAULT  UND fpathconf@@GLIBC_2.0
1106: 08057a89   105 FUNC    GLOBAL DEFAULT   12 safecp
1107: 0809deb4     4 OBJECT  GLOBAL DEFAULT   22 UP
1108: 080495a0    58 FUNC    GLOBAL DEFAULT  UND kill@@GLIBC_2.0
1109: 0808728c     4 OBJECT  GLOBAL DEFAULT   22 notecnt
1110: 08087290     4 OBJECT  GLOBAL DEFAULT   22 TE
1111: 080496c0    34 FUNC    GLOBAL DEFAULT   12 erropen
```

```
1112: 08063fc1    34 FUNC    GLOBAL DEFAULT   12 setpk
1113: 0806931f  1820 FUNC    GLOBAL DEFAULT   12 vputchar
1114: 08067116   151 FUNC    GLOBAL DEFAULT   12 wskipleft
1115: 08064e95    47 FUNC    GLOBAL DEFAULT   12 back1
1116: 080872a0  2560 OBJECT  GLOBAL DEFAULT   22 immacs
1117: 0805d428   132 FUNC    GLOBAL DEFAULT   12 sethard
1118: 0805cd9c    48 FUNC    GLOBAL DEFAULT   12 vsync
1119: 08087ca0     4 OBJECT  GLOBAL DEFAULT   22 ttymesg
1120: 08055c18  1103 FUNC    GLOBAL DEFAULT   12 set
1121: 0806b857    10 FUNC    GLOBAL DEFAULT   12 poolsbrk
1122: 0805e156    66 FUNC    GLOBAL DEFAULT   12 obeep
1123: 0806e0ec    98 FUNC    GLOBAL DEFAULT   12 regfree
1124: 0805abb7   669 FUNC    GLOBAL DEFAULT   12 vop
1125: 08087cc0   256 OBJECT  GLOBAL DEFAULT   22 lasttag
1126: 08056e82   376 FUNC    GLOBAL DEFAULT   12 qcolumn
1127: 08087dc0     2 OBJECT  GLOBAL DEFAULT   22 NONL
1128: 08058184   209 FUNC    GLOBAL DEFAULT   12 putline
1129: 08077000     0 NOTYPE  GLOBAL DEFAULT   16 __data_start
1130: 08077620     2 OBJECT  GLOBAL DEFAULT   16 TCOLUMNS
1131: 08057b78   146 FUNC    GLOBAL DEFAULT   12 topen
1132: 00000000     0 NOTYPE  WEAK   DEFAULT  UND _Jv_RegisterClasses
1133: 08087dc4     4 OBJECT  GLOBAL DEFAULT   22 vUNDsav
1134: 080495b0   113 FUNC    GLOBAL DEFAULT  UND __ctype_b_loc@@GLIBC_2.3
1135: 08087dc8     4 OBJECT  GLOBAL DEFAULT   22 AL_PARM
1136: 080779dc     4 OBJECT  GLOBAL DEFAULT   16 Outchar
1137: 08087dcc     2 OBJECT  GLOBAL DEFAULT   22 OCOLUMNS
1138: 0805df3f    55 FUNC    GLOBAL DEFAULT   12 setBUF
1139: 080535fb    21 FUNC    GLOBAL DEFAULT   12 putnl
1140: 08077622     2 OBJECT  GLOBAL DEFAULT   16 TLINES
1141: 08087dd0     4 OBJECT  GLOBAL DEFAULT   22 Command
1142: 08087dd4     2 OBJECT  GLOBAL DEFAULT   22 IN
1143: 08087dd8     4 OBJECT  GLOBAL DEFAULT   22 QUOTE
1144: 0805fefd   178 FUNC    GLOBAL DEFAULT   12 grabtag
1145: 08056510    76 FUNC    GLOBAL DEFAULT   12 genindent
1146: 08087ddc     2 OBJECT  GLOBAL DEFAULT   22 ZERO
1147: 08060177   605 FUNC    GLOBAL DEFAULT   12 vzop
1148: 08087de0     4 OBJECT  GLOBAL DEFAULT   22 undap1
1149: 0805ffaf    35 FUNC    GLOBAL DEFAULT   12 prepapp
1150: 08087de4     4 OBJECT  GLOBAL DEFAULT   22 xCS
1151: 08050e9b   124 FUNC    GLOBAL DEFAULT   12 edfile
1152: 08087de8     4 OBJECT  GLOBAL DEFAULT   22 ED
1153: 08063fe3    15 FUNC    GLOBAL DEFAULT   12 vkillDEL
1154: 0805524e  2092 FUNC    GLOBAL DEFAULT   12 compile
1155: 08057dcc    52 FUNC    GLOBAL DEFAULT   12 tclose
1156: 0806bfa5   142 FUNC    GLOBAL DEFAULT   12 tgetstr
1157: 08087dec     4 OBJECT  GLOBAL DEFAULT   22 oldquit
```

```
  1158: 080676f6   152 FUNC    GLOBAL DEFAULT   12 vcursat
  1159: 0805684a    25 FUNC    GLOBAL DEFAULT   12 lineDOL
  1160: 08087df0     4 OBJECT  GLOBAL DEFAULT   22 status
  1161: 08087df4     4 OBJECT  GLOBAL DEFAULT   22 aoftspace
  1162: 08068608   113 FUNC    GLOBAL DEFAULT   12 vrigid
  1163: 080495c0   154 FUNC    GLOBAL DEFAULT  UND tcgetattr@@GLIBC_2.0
  1164: 080495d0   124 FUNC    GLOBAL DEFAULT  UND read@@GLIBC_2.0
  1165: 0804a403    43 FUNC    GLOBAL DEFAULT   12 setnoaddr
  1166: 08077ba0    40 OBJECT  GLOBAL DEFAULT   16 fkeys
  1167: 080495e0    54 FUNC    GLOBAL DEFAULT  UND alarm@@GLIBC_2.0
  1168: 0806766a    73 FUNC    GLOBAL DEFAULT   12 fixech
  1169: 080566a9   169 FUNC    GLOBAL DEFAULT   12 junk
  1170: 0804ca04   423 FUNC    GLOBAL DEFAULT   12 tailprim
  1171: 08050323   786 FUNC    GLOBAL DEFAULT   12 gglob
  1172: 0804973d    43 FUNC    GLOBAL DEFAULT   12 invopt
  1173: 080495f0   163 FUNC    GLOBAL DEFAULT  UND wait@@GLIBC_2.0
  1174: 08087df8     4 OBJECT  GLOBAL DEFAULT   22 VB
  1175: 0805642a    59 FUNC    GLOBAL DEFAULT   12 dingdong
  1176: 0805af8d   319 FUNC    GLOBAL DEFAULT   12 setwind
  1177: 0806b676    48 FUNC    GLOBAL DEFAULT   12 free
  1178: 08069261    88 FUNC    GLOBAL DEFAULT   12 enddm
  1179: 08058255   492 FUNC    GLOBAL DEFAULT   12 getblock
  1180: 08056837    19 FUNC    GLOBAL DEFAULT   12 lineno
  1181: 00000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
  1182: 08087dfc     4 OBJECT  GLOBAL DEFAULT   22 argv0
  1183: 08049600    48 FUNC    GLOBAL DEFAULT  UND strcpy@@GLIBC_2.0
  1184: 08087e00  4120 OBJECT  GLOBAL DEFAULT   22 subre
  1185: 08088e18     4 OBJECT  GLOBAL DEFAULT   22 ldisc
```

## 2.11 Histogram

```
Histogram for bucket list length (total of 97 buckets):
 Length  Number     % of total  Coverage
      0  36         ( 37.1%)
      1  36         ( 37.1%)     37.1%
      2  16         ( 16.5%)     70.1%
      3  7          (  7.2%)     91.8%
      4  2          (  2.1%)    100.0%
```

## 2.12 Version symbols section '.gnu.version'

```
Version symbols section '.gnu.version' contains 98 entries:
 Addr: 0000000008048d6a  Offset: 0x000d6a  Link: 4 (.dynsym)
000: 0 (*local*)    1 (*global*)   2 (GLIBC_2.0)  2 (GLIBC_2.0)
```

```
004: 1 (*global*)   2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
008: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
00c: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
010: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  1 (*global*)   1 (*global*)
014: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
018: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
01c: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
020: 1 (*global*)   2 (GLIBC_2.0)  1 (*global*)   2 (GLIBC_2.0)
024: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
028: 1 (*global*)   2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
02c: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
030: 1 (*global*)   2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
034: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  1 (*global*)   2 (GLIBC_2.0)
038: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  1 (*global*)   1 (*global*)
03c: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
040: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
044: 1 (*global*)   1 (*global*)   2 (GLIBC_2.0)  2 (GLIBC_2.0)
048: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
04c: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
050: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)
054: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  1 (*global*)   2 (GLIBC_2.0)
058: 2 (GLIBC_2.0)  1 (*global*)   3 (GLIBC_2.3)  2 (GLIBC_2.0)
05c: 2 (GLIBC_2.0)  2 (GLIBC_2.0)  2 (GLIBC_2.0)  1 (*global*)
060: 0 (*local*)    2 (GLIBC_2.0)
```

## 2.13 Version needs section '.gnu.version_r'

```
Version needs section '.gnu.version_r' contains 1 entries:
 Addr: 0x0000000008048e30  Offset: 0x000e30  Link to section: 5 (.dynstr)
  000000: Version: 1  File: libc.so.6  Cnt: 2
  0x0010: Name: GLIBC_2.3  Flags: none  Version: 3
  0x0020: Name: GLIBC_2.0  Flags: none  Version: 2
```

## 2.14 Procedure for fetching symbols

1. scan the section header table for a section of type SHT_REL
2. grab its sh_link member value
3. iterate thru the reloc table entries and for each entry
4. compute NUM:=((r_info) ¿¿ 8)
5. NUM corresponds to an entry in the .dynsym symbol table.
6. parse that entry to get the symbol name

7. compute TYPE:=((r_info) & 0xff)
8. look in elf.h
9. note that function are found in .rel.plt section

## 2.15 The Entry Point

The entry point routine can be found by looking at the e_entry in the header data structure, that is (header-e_entry *header*).

In a standalone command we can find it with:

```
#!/bin/sh
fil=${1:-/bin/bash}
entry_point=$( od -j24 -An -td4 -N4 ${fil} )
gdb ${fil} -q <<EOT | sed -n -e '/:/p' -e '/ret *$/q' -e '/hlt *$/q'
    break *$entry_point
    run
    set disassembly-flavor intel
    disassemble
EOT
```

which if we name it ENTRYPOINT.sh we can run on the `ex` program thus:

```
./ENTRYPOINT.sh ex
```

which gives output that looks like:

```
(gdb) Breakpoint 1 at 0x8049610: file ../sysdeps/i386/elf/start.S, line 47.
(gdb) Starting program: /mnt/hgfs/vmshare/vic/ex
Current language:  auto; currently asm
(gdb) (gdb) Dump of assembler code for function _start:
0x08049610 <_start+0>: xor    ebp,ebp
0x08049612 <_start+2>: pop    esi
0x08049613 <_start+3>: mov    ecx,esp
0x08049615 <_start+5>: and    esp,0xfffffff0
0x08049618 <_start+8>: push   eax
0x08049619 <_start+9>: push   esp
0x0804961a <_start+10>: push   edx
0x0804961b <_start+11>: push   0x80729c0
0x08049620 <_start+16>: push   0x8072990
0x08049625 <_start+21>: push   ecx
0x08049626 <_start+22>: push   esi
0x08049627 <_start+23>: push   0x8049923
0x0804962c <_start+28>: call   0x80493f0 <__libc_start_main>
0x08049631 <_start+33>: hlt
```

Notice that all this routine does is set up the arguments to a call to `__libc_start_main`. By convention C programs push the arguments on the stack in reverse order.

So where does the __libc_start_main symbol come from? We can find out with the `ldd` command:

```
ldd ex
```

which gives the output:

```
libc.so.6 => /lib/tls/libc.so.6 (0x42000000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

and we can find tha that in libc with the `nm` command:

```
nm -D /lib/tls/libc.so.6 --line-numbers --no-sort | grep __libc_start_main
```

which, in this instance gives:

```
42015490 T __libc_start_main
```

If we look at the signature of `__libc_start_main` we find the source code in:

```
glibc-2.3.5/sysdeps/generic/libc-start.c
```

and the signature looks like:

```
int LIBC_START_MAIN (int (*main) (int, char **, char **)
             int argc,
     char *__unbounded *__unbounded ubp_av,
     void (*init) (void),
     void (*fini) (void),
     void (*rtld_fini) (void),
     void *__unbounded stack_end)
```

## 2.16 export LD_DEBUG=all

If we set the shell variable `LD_DEBUG` we can see the loader perform several operations including the display of:

1. library search paths (libs)
2. relocation processing (reloc)
3. progress for input file (files)
4. symbol table processing (symbols)
5. information about symbol bindings (bindings)
6. version dependencies (versions)

For `vi` the results are:

```
export LD_DEBUG=all
[root@dhcppc1 vic]# ex
```

Once we start the program the loader is dynamically called to find symbols. The loader has a cache of libraries.

libtermcap is the library for accessing the termcap database. It is necessary to be installed for a system to be able to do much of anything.

```
file=libtermcap.so.2;  needed by ex
find library=libtermcap.so.2; searching
 search cache=/etc/ld.so.cache
  trying file=/lib/libtermcap.so.2

file=libtermcap.so.2;  generating link map
  dynamic: 0x40025814  base: 0x40022000   size: 0x00003a08
    entry: 0x40022de0  phdr: 0x40022034  phnum:          3
```

Here libtermcap was loaded at a base address The link map requires 3A08 bytes so we round up to 4000 bytes.

libacl contains the POSIX 1003.1e draft standard 17 functions for manipulating access control lists.

```
file=libacl.so.1;  needed by ex
find library=libacl.so.1; searching
 search cache=/etc/ld.so.cache
  trying file=/lib/libacl.so.1

file=libacl.so.1;  generating link map
  dynamic: 0x4002b52c  base: 0x40026000   size: 0x000056d0
    entry: 0x40027440  phdr: 0x40026034  phnum:          3
```

libdl is a dynamic linking library that is generally of use only for porting applications. libdl is a wrapper to the add-on functions with the semantics of the dynamic linking library. This is used for finding shared objects at run time. If the items are not found the program can continue to run.

```
file=libdl.so.2;  needed by ex
find library=libdl.so.2; searching
 search cache=/etc/ld.so.cache
  trying file=/lib/libdl.so.2

file=libdl.so.2;  generating link map
  dynamic: 0x4002e008  base: 0x4002c000   size: 0x000021ac
    entry: 0x4002d6f0  phdr: 0x4002c034  phnum:          6
```

This is the linux C library version 6, the run-time libraries.

```
file=libc.so.6;  needed by ex
find library=libc.so.6; searching
 search cache=/etc/ld.so.cache
  trying file=/lib/tls/libc.so.6

file=libc.so.6;  generating link map
  dynamic: 0x42130920  base: 0x00000000   size: 0x00132f08
    entry: 0x42015660  phdr: 0x42000034  phnum:          8
```

libattr contains the extended attribute system calls and library functions.

```
file=libattr.so.1;  needed by /lib/libacl.so.1
find library=libattr.so.1; searching
 search cache=/etc/ld.so.cache
  trying file=/lib/libattr.so.1

file=libattr.so.1;  generating link map
  dynamic: 0x4003163c  base: 0x4002f000   size: 0x0000278c
    entry: 0x4002fad0  phdr: 0x4002f034  phnum:          3

checking for version ‘GLIBC_2.3’
 in file /lib/tls/libc.so.6 required by file ex
checking for version ‘GLIBC_2.2’
 in file /lib/tls/libc.so.6 required by file ex
checking for version ‘GLIBC_2.1’
 in file /lib/tls/libc.so.6 required by file ex
checking for version ‘GLIBC_2.0’
 in file /lib/tls/libc.so.6 required by file ex
checking for version ‘GLIBC_2.1.3’
 in file /lib/tls/libc.so.6 required by file /lib/libtermcap.so.2
checking for version ‘GLIBC_2.1’
 in file /lib/tls/libc.so.6 required by file /lib/libtermcap.so.2
checking for version ‘GLIBC_2.0’
 in file /lib/tls/libc.so.6 required by file /lib/libtermcap.so.2
checking for version ‘GLIBC_2.1.3’
 in file /lib/tls/libc.so.6 required by file /lib/libacl.so.1
checking for version ‘GLIBC_2.2’
 in file /lib/tls/libc.so.6 required by file /lib/libacl.so.1
checking for version ‘GLIBC_2.0’
 in file /lib/tls/libc.so.6 required by file /lib/libacl.so.1
checking for version ‘GLIBC_PRIVATE’
 in file /lib/ld-linux.so.2 required by file /lib/libdl.so.2
checking for version ‘GLIBC_2.3’
 in file /lib/tls/libc.so.6 required by file /lib/libdl.so.2
checking for version ‘GLIBC_2.1.3’
 in file /lib/tls/libc.so.6 required by file /lib/libdl.so.2
checking for version ‘GLIBC_2.1’
 in file /lib/tls/libc.so.6 required by file /lib/libdl.so.2
checking for version ‘GLIBC_PRIVATE’
 in file /lib/tls/libc.so.6 required by file /lib/libdl.so.2
checking for version ‘GLIBC_2.0’
 in file /lib/tls/libc.so.6 required by file /lib/libdl.so.2
checking for version ‘GLIBC_2.3’
 in file /lib/ld-linux.so.2 required by file /lib/tls/libc.so.6
checking for version ‘GLIBC_2.1’
```

```
 in file /lib/ld-linux.so.2 required by file /lib/tls/libc.so.6
checking for version 'GLIBC_2.0'
 in file /lib/ld-linux.so.2 required by file /lib/tls/libc.so.6
checking for version 'GLIBC_PRIVATE'
 in file /lib/ld-linux.so.2 required by file /lib/tls/libc.so.6
checking for version 'GLIBC_2.1.3'
 in file /lib/tls/libc.so.6 required by file /lib/libattr.so.1
checking for version 'GLIBC_2.0'
 in file /lib/tls/libc.so.6 required by file /lib/libattr.so.1
```

Here we start binding referenced symbols. The symbol `__cxa_finalize` was found in libc. The symbol is in the text (code) section. The symbol is global.

```
relocation processing: /lib/libattr.so.1 (lazy)
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libattr.so.1 to /lib/tls/libc.so.6:
   normal symbol '__cxa_finalize' [GLIBC_2.1.3]
```

The symbol `_Jv_RegisterClasses` and `__gmon_start__` are not found.

```
symbol=_Jv_RegisterClasses;  lookup in file=ex
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libtermcap.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libacl.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libdl.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/tls/libc.so.6
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libattr.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/ld-linux.so.2
symbol=__gmon_start__;  lookup in file=ex
symbol=__gmon_start__;  lookup in file=/lib/libtermcap.so.2
symbol=__gmon_start__;  lookup in file=/lib/libacl.so.1
symbol=__gmon_start__;  lookup in file=/lib/libdl.so.2
symbol=__gmon_start__;  lookup in file=/lib/tls/libc.so.6
symbol=__gmon_start__;  lookup in file=/lib/libattr.so.1
symbol=__gmon_start__;  lookup in file=/lib/ld-linux.so.2
```

The symbol `_IO_file_close` comes from the fileops routine in libc. The symbol is in the text (code) section. The symbol is global.

```
relocation processing: /lib/tls/libc.so.6 (lazy)
symbol=_IO_file_close;  lookup in file=ex
symbol=_IO_file_close;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_file_close;  lookup in file=/lib/libacl.so.1
symbol=_IO_file_close;  lookup in file=/lib/libdl.so.2
symbol=_IO_file_close;  lookup in file=/lib/tls/libc.so.6
```

```
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_IO_file_close' [GLIBC_2.0]
```

The symbol _IO_2_1_stdin_ comes from stdfiles in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=_IO_2_1_stdin_;  lookup in file=ex
symbol=_IO_2_1_stdin_;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_2_1_stdin_;  lookup in file=/lib/libacl.so.1
symbol=_IO_2_1_stdin_;  lookup in file=/lib/libdl.so.2
symbol=_IO_2_1_stdin_;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_IO_2_1_stdin_' [GLIBC_2.1]
```

The symbol _IO_2_1_stdout_ comes from stdfiles in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=_IO_2_1_stdout_;  lookup in file=ex
symbol=_IO_2_1_stdout_;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_2_1_stdout_;  lookup in file=/lib/libacl.so.1
symbol=_IO_2_1_stdout_;  lookup in file=/lib/libdl.so.2
symbol=_IO_2_1_stdout_;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_IO_2_1_stdout_' [GLIBC_2.1]
```

The symbol _IO_2_1_stderr_ comes from stdfiles in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=_IO_2_1_stderr_;  lookup in file=ex
symbol=_IO_2_1_stderr_;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_2_1_stderr_;  lookup in file=/lib/libacl.so.1
symbol=_IO_2_1_stderr_;  lookup in file=/lib/libdl.so.2
symbol=_IO_2_1_stderr_;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_IO_2_1_stderr_' [GLIBC_2.1]
```

The symbol _IO_stdin_ comes from unknown in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=_IO_stdin_;  lookup in file=ex
symbol=_IO_stdin_;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_stdin_;  lookup in file=/lib/libacl.so.1
symbol=_IO_stdin_;  lookup in file=/lib/libdl.so.2
symbol=_IO_stdin_;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_IO_stdin_' [GLIBC_2.0]
```

The symbol _IO_stdout_ comes from unknown in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=_IO_stdout_;  lookup in file=ex
symbol=_IO_stdout_;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_stdout_;  lookup in file=/lib/libacl.so.1
symbol=_IO_stdout_;  lookup in file=/lib/libdl.so.2
symbol=_IO_stdout_;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `_IO_stdout_' [GLIBC_2.0]
```

The symbol `__morecore` comes from malloc in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__morecore;  lookup in file=ex
symbol=__morecore;  lookup in file=/lib/libtermcap.so.2
symbol=__morecore;  lookup in file=/lib/libacl.so.1
symbol=__morecore;  lookup in file=/lib/libdl.so.2
symbol=__morecore;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__morecore' [GLIBC_2.0]
```

The symbol `__daylight` comes from tzset in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__daylight;  lookup in file=ex
symbol=__daylight;  lookup in file=/lib/libtermcap.so.2
symbol=__daylight;  lookup in file=/lib/libacl.so.1
symbol=__daylight;  lookup in file=/lib/libdl.so.2
symbol=__daylight;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__daylight' [GLIBC_2.0]
```

The symbol `__malloc_hook` comes from malloc in libc. The symbol is global. The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error.

```
symbol=__malloc_hook;  lookup in file=ex
symbol=__malloc_hook;  lookup in file=/lib/libtermcap.so.2
symbol=__malloc_hook;  lookup in file=/lib/libacl.so.1
symbol=__malloc_hook;  lookup in file=/lib/libdl.so.2
symbol=__malloc_hook;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__malloc_hook' [GLIBC_2.0]
```

The symbol `h_nerr` comes from herror in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=h_nerr;  lookup in file=ex
symbol=h_nerr;  lookup in file=/lib/libtermcap.so.2
```

```
symbol=h_nerr;  lookup in file=/lib/libacl.so.1
symbol=h_nerr;  lookup in file=/lib/libdl.so.2
symbol=h_nerr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'h_nerr' [GLIBC_2.0]
```

The symbol __malloc_initialize_hook comes from malloc in libc. The symbol is a weak object. The symbol is global.

```
symbol=__malloc_initialize_hook;  lookup in file=ex
symbol=__malloc_initialize_hook;  lookup in file=/lib/libtermcap.so.2
symbol=__malloc_initialize_hook;  lookup in file=/lib/libacl.so.1
symbol=__malloc_initialize_hook;  lookup in file=/lib/libdl.so.2
symbol=__malloc_initialize_hook;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '__malloc_initialize_hook' [GLIBC_2.0]
```

The symbol _dl_starting_up comes from dl-support in libc. The symbol is a weak object. The symbol is local.

```
symbol=_dl_starting_up;  lookup in file=ex
symbol=_dl_starting_up;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_starting_up;  lookup in file=/lib/libacl.so.1
symbol=_dl_starting_up;  lookup in file=/lib/libdl.so.2
symbol=_dl_starting_up;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_starting_up;  lookup in file=/lib/libattr.so.1
symbol=_dl_starting_up;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol '_dl_starting_up' [GLIBC_PRIVATE]
```

The symbol _r_debug comes from dl-debug in ld-linux.so.2. The symbol is global. The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references.

```
symbol=_r_debug;  lookup in file=ex
symbol=_r_debug;  lookup in file=/lib/libtermcap.so.2
symbol=_r_debug;  lookup in file=/lib/libacl.so.1
symbol=_r_debug;  lookup in file=/lib/libdl.so.2
symbol=_r_debug;  lookup in file=/lib/tls/libc.so.6
symbol=_r_debug;  lookup in file=/lib/libattr.so.1
symbol=_r_debug;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol '_r_debug' [GLIBC_2.0]
```

The symbol stdout comes from /lib/tls/libc.so.6. The symbol is in the initialized data section. The symbol is global.

```
symbol=stdout;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex:
  normal symbol 'stdout' [GLIBC_2.0]
```

The symbol __pthread_mutex_lock comes from unknown in ld-linux.so.2 The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_mutex_lock;  lookup in file=ex
symbol=__pthread_mutex_lock;  lookup in file=/lib/libtermcap.so.2
symbol=__pthread_mutex_lock;  lookup in file=/lib/libacl.so.1
symbol=__pthread_mutex_lock;  lookup in file=/lib/libdl.so.2
symbol=__pthread_mutex_lock;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_mutex_lock;  lookup in file=/lib/libattr.so.1
symbol=__pthread_mutex_lock;  lookup in file=/lib/ld-linux.so.2
```

The symbol __rcmd_errstr comes from rcmd in libc. The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=__rcmd_errstr;  lookup in file=ex
symbol=__rcmd_errstr;  lookup in file=/lib/libtermcap.so.2
symbol=__rcmd_errstr;  lookup in file=/lib/libacl.so.1
symbol=__rcmd_errstr;  lookup in file=/lib/libdl.so.2
symbol=__rcmd_errstr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '__rcmd_errstr' [GLIBC_2.0]
```

The symbol _nl_domain_bindings comes from in libc. The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=_nl_domain_bindings;  lookup in file=ex
symbol=_nl_domain_bindings;  lookup in file=/lib/libtermcap.so.2
symbol=_nl_domain_bindings;  lookup in file=/lib/libacl.so.1
symbol=_nl_domain_bindings;  lookup in file=/lib/libdl.so.2
symbol=_nl_domain_bindings;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_nl_domain_bindings' [GLIBC_2.0]
```

The symbol re_syntax_options comes from regex in libc. The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common

symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=re_syntax_options;  lookup in file=ex
symbol=re_syntax_options;  lookup in file=/lib/libtermcap.so.2
symbol=re_syntax_options;  lookup in file=/lib/libacl.so.1
symbol=re_syntax_options;  lookup in file=/lib/libdl.so.2
symbol=re_syntax_options;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘re_syntax_options’ [GLIBC_2.0]
```

The symbol `argp_program_bug_address` is unknown in libc. The symbol is in the uninitialized data section (known as BSS).

```
symbol=argp_program_bug_address;  lookup in file=ex
symbol=argp_program_bug_address;  lookup in file=/lib/libtermcap.so.2
symbol=argp_program_bug_address;  lookup in file=/lib/libacl.so.1
symbol=argp_program_bug_address;  lookup in file=/lib/libdl.so.2
symbol=argp_program_bug_address;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘argp_program_bug_address’ [GLIBC_2.1]
```

The symbol `__tzname` comes from tzset in libc. The symbol is global. The symbol is in the initialized data section.

```
symbol=__tzname;  lookup in file=ex
symbol=__tzname;  lookup in file=/lib/libtermcap.so.2
symbol=__tzname;  lookup in file=/lib/libacl.so.1
symbol=__tzname;  lookup in file=/lib/libdl.so.2
symbol=__tzname;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘__tzname’ [GLIBC_2.0]
```

The symbol `_IO_funlockfile` comes from funlockfile in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_IO_funlockfile;  lookup in file=ex
symbol=_IO_funlockfile;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_funlockfile;  lookup in file=/lib/libacl.so.1
symbol=_IO_funlockfile;  lookup in file=/lib/libdl.so.2
symbol=_IO_funlockfile;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘_IO_funlockfile’ [GLIBC_2.0]
```

The symbol `__realloc_hook` comes from malloc in libc. The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=__realloc_hook;  lookup in file=ex
symbol=__realloc_hook;  lookup in file=/lib/libtermcap.so.2
symbol=__realloc_hook;  lookup in file=/lib/libacl.so.1
symbol=__realloc_hook;  lookup in file=/lib/libdl.so.2
symbol=__realloc_hook;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '__realloc_hook' [GLIBC_2.0]
```

The symbol `_IO_stderr_` comes from unknown in /lib/tls/libc.so.6 The symbol is in the initialized data section.

```
symbol=_IO_stderr_;  lookup in file=ex
symbol=_IO_stderr_;  lookup in file=/lib/libtermcap.so.2
symbol=_IO_stderr_;  lookup in file=/lib/libacl.so.1
symbol=_IO_stderr_;  lookup in file=/lib/libdl.so.2
symbol=_IO_stderr_;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_IO_stderr_' [GLIBC_2.0]
```

The symbol `malloc` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=malloc;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex:
  normal symbol 'malloc' [GLIBC_2.0]
```

The symbol `_nl_msg_cat_cntr` comes from /lib/tls/libc.so.6 The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=_nl_msg_cat_cntr;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex:
  normal symbol '_nl_msg_cat_cntr' [GLIBC_2.0]
```

The symbol `optarg` comes from /lib/tls/libc.so.6 The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=optarg;  lookup in file=ex
symbol=optarg;  lookup in file=/lib/libtermcap.so.2
symbol=optarg;  lookup in file=/lib/libacl.so.1
symbol=optarg;  lookup in file=/lib/libdl.so.2
symbol=optarg;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'optarg' [GLIBC_2.0]
```

The symbol `loc2` comes from /lib/tls/libc.so.6 The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=loc2;  lookup in file=ex
symbol=loc2;  lookup in file=/lib/libtermcap.so.2
symbol=loc2;  lookup in file=/lib/libacl.so.1
symbol=loc2;  lookup in file=/lib/libdl.so.2
symbol=loc2;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'loc2' [GLIBC_2.0]
```

The symbol `h_errlist` comes from herror in /lib/tls/libc The symbol is in the initialized data section. The symbol is global.

```
symbol=h_errlist;  lookup in file=ex
symbol=h_errlist;  lookup in file=/lib/libtermcap.so.2
symbol=h_errlist;  lookup in file=/lib/libacl.so.1
symbol=h_errlist;  lookup in file=/lib/libdl.so.2
symbol=h_errlist;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'h_errlist' [GLIBC_2.0]
```

The symbol `opterr` comes from getopt in /lib/tls/libc.so.6 The symbol is in the initialized data section. The symbol is global.

```
symbol=opterr;  lookup in file=ex
symbol=opterr;  lookup in file=/lib/libtermcap.so.2
symbol=opterr;  lookup in file=/lib/libacl.so.1
symbol=opterr;  lookup in file=/lib/libdl.so.2
symbol=opterr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'opterr' [GLIBC_2.0]
```

The symbol `error_message_count` comes from error in /lib/tls/libc.so.6 The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=error_message_count;  lookup in file=ex
symbol=error_message_count;  lookup in file=/lib/libtermcap.so.2
symbol=error_message_count;  lookup in file=/lib/libacl.so.1
symbol=error_message_count;  lookup in file=/lib/libdl.so.2
symbol=error_message_count;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'error_message_count' [GLIBC_2.0]
```

The symbol _environ comes from environ in /lib/tls/libc.so.6 The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=_environ;  lookup in file=ex
symbol=_environ;  lookup in file=/lib/libtermcap.so.2
symbol=_environ;  lookup in file=/lib/libacl.so.1
symbol=_environ;  lookup in file=/lib/libdl.so.2
symbol=_environ;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '_environ' [GLIBC_2.0]
```

The symbol getdate_err comes from in /lib/tls/libc.so.6 The symbol is in the uninitialized data section (known as BSS). The symbol is common. The symbol is global.

```
symbol=getdate_err;  lookup in file=ex
symbol=getdate_err;  lookup in file=/lib/libtermcap.so.2
symbol=getdate_err;  lookup in file=/lib/libacl.so.1
symbol=getdate_err;  lookup in file=/lib/libdl.so.2
symbol=getdate_err;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'getdate_err' [GLIBC_2.1]
```

The symbol __environ come from environ in /lib/tls/libc.so.6 The symbol is in the initialized data section. The symbol is global.

```
symbol=__environ;  lookup in file=ex
symbol=__environ;  lookup in file=/lib/libtermcap.so.2
symbol=__environ;  lookup in file=/lib/libacl.so.1
symbol=__environ;  lookup in file=/lib/libdl.so.2
symbol=__environ;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '__environ' [GLIBC_2.0]
```

The symbol obstack_exit_failure comes from obstack in libc.so.6 The symbol is in the initialized data section. The symbol is global.

```
symbol=obstack_exit_failure;  lookup in file=ex
symbol=obstack_exit_failure;  lookup in file=/lib/libtermcap.so.2
symbol=obstack_exit_failure;  lookup in file=/lib/libacl.so.1
symbol=obstack_exit_failure;  lookup in file=/lib/libdl.so.2
symbol=obstack_exit_failure;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'obstack_exit_failure' [GLIBC_2.0]
```

The symbol _rtld_global comes from ld-linux.so.2 The symbol is in the initialized data section. The symbol is global.

```
symbol=_rtld_global;  lookup in file=ex
symbol=_rtld_global;  lookup in file=/lib/libtermcap.so.2
symbol=_rtld_global;  lookup in file=/lib/libacl.so.1
symbol=_rtld_global;  lookup in file=/lib/libdl.so.2
symbol=_rtld_global;  lookup in file=/lib/tls/libc.so.6
symbol=_rtld_global;  lookup in file=/lib/libattr.so.1
symbol=_rtld_global;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol `_rtld_global' [GLIBC_PRIVATE]
```

The symbol `error_print_progname` comes from error in /lib/tls/libc.so.6 The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=error_print_progname;  lookup in file=ex
symbol=error_print_progname;  lookup in file=/lib/libtermcap.so.2
symbol=error_print_progname;  lookup in file=/lib/libacl.so.1
symbol=error_print_progname;  lookup in file=/lib/libdl.so.2
symbol=error_print_progname;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `error_print_progname' [GLIBC_2.0]
```

The symbol `__after_morecore_hook` comes from malloc. The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=__after_morecore_hook;  lookup in file=ex
symbol=__after_morecore_hook;  lookup in file=/lib/libtermcap.so.2
symbol=__after_morecore_hook;  lookup in file=/lib/libacl.so.1
symbol=__after_morecore_hook;  lookup in file=/lib/libdl.so.2
symbol=__after_morecore_hook;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__after_morecore_hook' [GLIBC_2.0]
```

The symbol `__pthread_mutex_unlock` is undefined. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_mutex_unlock;  lookup in file=ex
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libtermcap.so.2
```

```
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libacl.so.1
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libdl.so.2
symbol=__pthread_mutex_unlock;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libattr.so.1
symbol=__pthread_mutex_unlock;  lookup in file=/lib/ld-linux.so.2
```

The symbol `__ctype32_toupper` occurs in /lib/tls/libc.so.6 The symbol is in the initialized data section. The symbol is global.

```
symbol=__ctype32_toupper;  lookup in file=ex
symbol=__ctype32_toupper;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype32_toupper;  lookup in file=/lib/libacl.so.1
symbol=__ctype32_toupper;  lookup in file=/lib/libdl.so.2
symbol=__ctype32_toupper;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__ctype32_toupper' [GLIBC_2.2]
```

The symbol `__key_encryptsession_pk_LOCAL` occurs in /lib/tls/libc.so.6 The symbol is in the initialized data section. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=__key_encryptsession_pk_LOCAL;  lookup in file=ex
symbol=__key_encryptsession_pk_LOCAL;  lookup in file=/lib/libtermcap.so.2
symbol=__key_encryptsession_pk_LOCAL;  lookup in file=/lib/libacl.so.1
symbol=__key_encryptsession_pk_LOCAL;  lookup in file=/lib/libdl.so.2
symbol=__key_encryptsession_pk_LOCAL;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__key_encryptsession_pk_LOCAL' [GLIBC_2.1]
```

The symbol `argp_program_version` occurs in argp-parse in /lib/tls/libc.so.6. The symbol is in the uninitialized data section (known as BSS). The symbol is undefined. The symbol is global.

```
symbol=argp_program_version;  lookup in file=ex
symbol=argp_program_version;  lookup in file=/lib/libtermcap.so.2
symbol=argp_program_version;  lookup in file=/lib/libacl.so.1
symbol=argp_program_version;  lookup in file=/lib/libdl.so.2
symbol=argp_program_version;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `argp_program_version' [GLIBC_2.1]
```

The symbol `__ctype_toupper` in ctype-info in /lib/tls/libc.so.6. The symbol is in the text (code) section. The symbol is global.

```
symbol=__ctype_toupper;  lookup in file=ex
symbol=__ctype_toupper;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype_toupper;  lookup in file=/lib/libacl.so.1
```

```
symbol=__ctype_toupper;  lookup in file=/lib/libdl.so.2
symbol=__ctype_toupper;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__ctype_toupper' [GLIBC_2.0]
```

The symbol __fpu_control in fpu_control in /lib/tls/libc.so.6. The symbol is in the initialized data section. The symbol is global.

```
symbol=__fpu_control;  lookup in file=ex
symbol=__fpu_control;  lookup in file=/lib/libtermcap.so.2
symbol=__fpu_control;  lookup in file=/lib/libacl.so.1
symbol=__fpu_control;  lookup in file=/lib/libdl.so.2
symbol=__fpu_control;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__fpu_control' [GLIBC_2.0]
```

The symbol optind is in getopt in /lib/tls/libc.so.6 The symbol is in the initialized data section. The symbol is global.

```
symbol=optind;  lookup in file=ex
symbol=optind;  lookup in file=/lib/libtermcap.so.2
symbol=optind;  lookup in file=/lib/libacl.so.1
symbol=optind;  lookup in file=/lib/libdl.so.2
symbol=optind;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `optind' [GLIBC_2.0]
```

The symbol _res_hconf is in res_hconf in /lib/tls/libc.so.6 The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=_res_hconf;  lookup in file=ex
symbol=_res_hconf;  lookup in file=/lib/libtermcap.so.2
symbol=_res_hconf;  lookup in file=/lib/libacl.so.1
symbol=_res_hconf;  lookup in file=/lib/libdl.so.2
symbol=_res_hconf;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `_res_hconf' [GLIBC_2.2]
```

The symbol stdin comes from /lib/tls/libc.so.6. The symbol is in the initialized data section. The symbol is global.

```
symbol=stdin;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex:
  normal symbol `stdin' [GLIBC_2.0]
```

The symbol __progname comes from init-misc in /lib/tls/libc.so.6 The symbol is in the initialized data section. The symbol is global.

```
symbol=__progname;  lookup in file=ex
symbol=__progname;  lookup in file=/lib/libtermcap.so.2
symbol=__progname;  lookup in file=/lib/libacl.so.1
symbol=__progname;  lookup in file=/lib/libdl.so.2
symbol=__progname;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘__progname’ [GLIBC_2.0]
```

The symbol `loc1` comes from regexp in /lib/tls/libc.so.6 The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=loc1;  lookup in file=ex
symbol=loc1;  lookup in file=/lib/libtermcap.so.2
symbol=loc1;  lookup in file=/lib/libacl.so.1
symbol=loc1;  lookup in file=/lib/libdl.so.2
symbol=loc1;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘loc1’ [GLIBC_2.0]
```

The symbol `program_invocation_short_name` comes from init-misc in libc. The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=program_invocation_short_name;  lookup in file=ex
symbol=program_invocation_short_name;  lookup in file=/lib/libtermcap.so.2
symbol=program_invocation_short_name;  lookup in file=/lib/libacl.so.1
symbol=program_invocation_short_name;  lookup in file=/lib/libdl.so.2
symbol=program_invocation_short_name;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘program_invocation_short_name’ [GLIBC_2.0]
```

The symbol `argp_err_exit_status` comes from libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=argp_err_exit_status;  lookup in file=ex
symbol=argp_err_exit_status;  lookup in file=/lib/libtermcap.so.2
symbol=argp_err_exit_status;  lookup in file=/lib/libacl.so.1
symbol=argp_err_exit_status;  lookup in file=/lib/libdl.so.2
symbol=argp_err_exit_status;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘argp_err_exit_status’ [GLIBC_2.1]
```

The symbol `__ctype_tolower` is in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__ctype_tolower;  lookup in file=ex
symbol=__ctype_tolower;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype_tolower;  lookup in file=/lib/libacl.so.1
symbol=__ctype_tolower;  lookup in file=/lib/libdl.so.2
symbol=__ctype_tolower;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘__ctype_tolower’ [GLIBC_2.0]
```

The symbol `strcmp` comes from strcmp in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strcmp;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex: normal symbol ‘strcmp’ [GLIBC_2.0]
```

The symbol `__check_rhosts_file` comes from rcmd in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__check_rhosts_file;  lookup in file=ex
symbol=__check_rhosts_file;  lookup in file=/lib/libtermcap.so.2
symbol=__check_rhosts_file;  lookup in file=/lib/libacl.so.1
symbol=__check_rhosts_file;  lookup in file=/lib/libdl.so.2
symbol=__check_rhosts_file;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘__check_rhosts_file’ [GLIBC_2.0]
```

The symbol `obstack_alloc_failed_handler` comes from obstack in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=obstack_alloc_failed_handler;  lookup in file=ex
symbol=obstack_alloc_failed_handler;  lookup in file=/lib/libtermcap.so.2
symbol=obstack_alloc_failed_handler;  lookup in file=/lib/libacl.so.1
symbol=obstack_alloc_failed_handler;  lookup in file=/lib/libdl.so.2
symbol=obstack_alloc_failed_handler;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘obstack_alloc_failed_handler’ [GLIBC_2.0]
```

The symbol `stderr` comes from stdio in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=stderr;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex: normal symbol ‘stderr’ [GLIBC_2.0]
```

The symbol `__key_gendes_LOCAL` comes from key_call in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=__key_gendes_LOCAL;  lookup in file=ex
symbol=__key_gendes_LOCAL;  lookup in file=/lib/libtermcap.so.2
symbol=__key_gendes_LOCAL;  lookup in file=/lib/libacl.so.1
symbol=__key_gendes_LOCAL;  lookup in file=/lib/libdl.so.2
symbol=__key_gendes_LOCAL;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__key_gendes_LOCAL' [GLIBC_2.1]
```

The symbol `optopt` comes from getopt in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=optopt;  lookup in file=ex
symbol=optopt;  lookup in file=/lib/libtermcap.so.2
symbol=optopt;  lookup in file=/lib/libacl.so.1
symbol=optopt;  lookup in file=/lib/libdl.so.2
symbol=optopt;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `optopt' [GLIBC_2.0]
```

The symbol `__timezone` comes from tzset in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__timezone;  lookup in file=ex
symbol=__timezone;  lookup in file=/lib/libtermcap.so.2
symbol=__timezone;  lookup in file=/lib/libacl.so.1
symbol=__timezone;  lookup in file=/lib/libdl.so.2
symbol=__timezone;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__timezone' [GLIBC_2.0]
```

The symbol `svcauthdes_stats` comes from svcauth_des in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=svcauthdes_stats;  lookup in file=ex
symbol=svcauthdes_stats;  lookup in file=/lib/libtermcap.so.2
symbol=svcauthdes_stats;  lookup in file=/lib/libacl.so.1
symbol=svcauthdes_stats;  lookup in file=/lib/libdl.so.2
symbol=svcauthdes_stats;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `svcauthdes_stats' [GLIBC_2.0]
```

The symbol `mallwatch` comes from mtrace in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=mallwatch;  lookup in file=ex
symbol=mallwatch;  lookup in file=/lib/libtermcap.so.2
symbol=mallwatch;  lookup in file=/lib/libacl.so.1
symbol=mallwatch;  lookup in file=/lib/libdl.so.2
symbol=mallwatch;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘mallwatch’ [GLIBC_2.0]
```

The symbol `svc_fdset` comes from rcp_common in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=svc_fdset;  lookup in file=ex
symbol=svc_fdset;  lookup in file=/lib/libtermcap.so.2
symbol=svc_fdset;  lookup in file=/lib/libacl.so.1
symbol=svc_fdset;  lookup in file=/lib/libdl.so.2
symbol=svc_fdset;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘svc_fdset’ [GLIBC_2.0]
```

The symbol `__curbrk` comes from brk in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__curbrk;  lookup in file=ex
symbol=__curbrk;  lookup in file=/lib/libtermcap.so.2
symbol=__curbrk;  lookup in file=/lib/libacl.so.1
symbol=__curbrk;  lookup in file=/lib/libdl.so.2
symbol=__curbrk;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘__curbrk’ [GLIBC_2.0]
```

The symbol `__libc_enable_secure` comes from enbl-secure in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__libc_enable_secure;  lookup in file=ex
symbol=__libc_enable_secure;  lookup in file=/lib/libtermcap.so.2
symbol=__libc_enable_secure;  lookup in file=/lib/libacl.so.1
symbol=__libc_enable_secure;  lookup in file=/lib/libdl.so.2
symbol=__libc_enable_secure;  lookup in file=/lib/tls/libc.so.6
symbol=__libc_enable_secure;  lookup in file=/lib/libattr.so.1
symbol=__libc_enable_secure;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol ‘__libc_enable_secure’ [GLIBC_PRIVATE]
```

The symbol `__ctype32_tolower` comes from libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__ctype32_tolower;  lookup in file=ex
symbol=__ctype32_tolower;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype32_tolower;  lookup in file=/lib/libacl.so.1
symbol=__ctype32_tolower;  lookup in file=/lib/libdl.so.2
symbol=__ctype32_tolower;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__ctype32_tolower' [GLIBC_2.2]
```

The symbol `__free_hook` comes from malloc in libc. The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=__free_hook;  lookup in file=ex
symbol=__free_hook;  lookup in file=/lib/libtermcap.so.2
symbol=__free_hook;  lookup in file=/lib/libacl.so.1
symbol=__free_hook;  lookup in file=/lib/libdl.so.2
symbol=__free_hook;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__free_hook' [GLIBC_2.0]
```

The symbol `_null_auth` comes from rpc_common in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=_null_auth;  lookup in file=ex
symbol=_null_auth;  lookup in file=/lib/libtermcap.so.2
symbol=_null_auth;  lookup in file=/lib/libacl.so.1
symbol=_null_auth;  lookup in file=/lib/libdl.so.2
symbol=_null_auth;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `_null_auth' [GLIBC_2.0]
```

The symbol `error_one_per_line` comes from error in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=error_one_per_line;  lookup in file=ex
symbol=error_one_per_line;  lookup in file=/lib/libtermcap.so.2
symbol=error_one_per_line;  lookup in file=/lib/libacl.so.1
symbol=error_one_per_line;  lookup in file=/lib/libdl.so.2
symbol=error_one_per_line;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `error_one_per_line' [GLIBC_2.0]
```

The symbol _dl_argv comes from dl-support in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=_dl_argv;  lookup in file=ex
symbol=_dl_argv;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_argv;  lookup in file=/lib/libacl.so.1
symbol=_dl_argv;  lookup in file=/lib/libdl.so.2
symbol=_dl_argv;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_argv;  lookup in file=/lib/libattr.so.1
symbol=_dl_argv;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol `_dl_argv' [GLIBC_PRIVATE]
```

The symbol _IO_stdin_used comes from libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=_IO_stdin_used;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex:
  normal symbol `_IO_stdin_used'
```

The symbol program_invocation_name comes from init-misc in libc. The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=program_invocation_name;  lookup in file=ex
symbol=program_invocation_name;  lookup in file=/lib/libtermcap.so.2
symbol=program_invocation_name;  lookup in file=/lib/libacl.so.1
symbol=program_invocation_name;  lookup in file=/lib/libdl.so.2
symbol=program_invocation_name;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `program_invocation_name' [GLIBC_2.0]
```

The symbol __ctype_b comes from ctype-info in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__ctype_b;  lookup in file=ex
symbol=__ctype_b;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype_b;  lookup in file=/lib/libacl.so.1
symbol=__ctype_b;  lookup in file=/lib/libdl.so.2
symbol=__ctype_b;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__ctype_b' [GLIBC_2.0]
```

The symbol `svc_max_pollfd` comes from rpc_common in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=svc_max_pollfd;  lookup in file=ex
symbol=svc_max_pollfd;  lookup in file=/lib/libtermcap.so.2
symbol=svc_max_pollfd;  lookup in file=/lib/libacl.so.1
symbol=svc_max_pollfd;  lookup in file=/lib/libdl.so.2
symbol=svc_max_pollfd;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘svc_max_pollfd’ [GLIBC_2.2]
```

The symbol `rpc_createerr` comes from rpc_common in libc The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=rpc_createerr;  lookup in file=ex
symbol=rpc_createerr;  lookup in file=/lib/libtermcap.so.2
symbol=rpc_createerr;  lookup in file=/lib/libacl.so.1
symbol=rpc_createerr;  lookup in file=/lib/libdl.so.2
symbol=rpc_createerr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘rpc_createerr’ [GLIBC_2.0]
```

The symbol `argp_program_version_hook` comes from libc. The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=argp_program_version_hook;  lookup in file=ex
symbol=argp_program_version_hook;  lookup in file=/lib/libtermcap.so.2
symbol=argp_program_version_hook;  lookup in file=/lib/libacl.so.1
symbol=argp_program_version_hook;  lookup in file=/lib/libdl.so.2
symbol=argp_program_version_hook;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘argp_program_version_hook’ [GLIBC_2.1]
```

The symbol `svc_pollfd` comes from rpc_common in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=svc_pollfd;  lookup in file=ex
symbol=svc_pollfd;  lookup in file=/lib/libtermcap.so.2
symbol=svc_pollfd;  lookup in file=/lib/libacl.so.1
```

```
symbol=svc_pollfd;  lookup in file=/lib/libdl.so.2
symbol=svc_pollfd;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'svc_pollfd' [GLIBC_2.2]
```

The symbol _dl_out_of_memory is in libc. The symbol is undefined. The symbol is global.

```
symbol=_dl_out_of_memory;  lookup in file=ex
symbol=_dl_out_of_memory;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_out_of_memory;  lookup in file=/lib/libacl.so.1
symbol=_dl_out_of_memory;  lookup in file=/lib/libdl.so.2
symbol=_dl_out_of_memory;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_out_of_memory;  lookup in file=/lib/libattr.so.1
symbol=_dl_out_of_memory;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol '_dl_out_of_memory' [GLIBC_PRIVATE]
```

The symbol __key_decryptsession_pk_LOCAL comes from key_call in libc. The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=__key_decryptsession_pk_LOCAL;  lookup in file=ex
symbol=__key_decryptsession_pk_LOCAL;  lookup in file=/lib/libtermcap.so.2
symbol=__key_decryptsession_pk_LOCAL;  lookup in file=/lib/libacl.so.1
symbol=__key_decryptsession_pk_LOCAL;  lookup in file=/lib/libdl.so.2
symbol=__key_decryptsession_pk_LOCAL;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '__key_decryptsession_pk_LOCAL' [GLIBC_2.1]
```

The symbol __ctype32_b is in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__ctype32_b;  lookup in file=ex
symbol=__ctype32_b;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype32_b;  lookup in file=/lib/libacl.so.1
symbol=__ctype32_b;  lookup in file=/lib/libdl.so.2
symbol=__ctype32_b;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol '__ctype32_b' [GLIBC_2.0]
```

The symbol __memalign_hook comes from malloc in libc. The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=__memalign_hook;  lookup in file=ex
symbol=__memalign_hook;  lookup in file=/lib/libtermcap.so.2
symbol=__memalign_hook;  lookup in file=/lib/libacl.so.1
symbol=__memalign_hook;  lookup in file=/lib/libdl.so.2
symbol=__memalign_hook;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__memalign_hook' [GLIBC_2.0]
```

The symbol `__progname_full` comes from init-misc in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=__progname_full;  lookup in file=ex
symbol=__progname_full;  lookup in file=/lib/libtermcap.so.2
symbol=__progname_full;  lookup in file=/lib/libacl.so.1
symbol=__progname_full;  lookup in file=/lib/libdl.so.2
symbol=__progname_full;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol `__progname_full' [GLIBC_2.0]
```

The symbol `__libc_stack_end` comes from dl-support in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=__libc_stack_end;  lookup in file=ex
symbol=__libc_stack_end;  lookup in file=/lib/libtermcap.so.2
symbol=__libc_stack_end;  lookup in file=/lib/libacl.so.1
symbol=__libc_stack_end;  lookup in file=/lib/libdl.so.2
symbol=__libc_stack_end;  lookup in file=/lib/tls/libc.so.6
symbol=__libc_stack_end;  lookup in file=/lib/libattr.so.1
symbol=__libc_stack_end;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol `__libc_stack_end' [GLIBC_2.1]
```

The symbol `malloc` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=free;  lookup in file=ex
binding file /lib/tls/libc.so.6 to ex: normal symbol `free' [GLIBC_2.0]
```

The symbol `__pthread_once` is undefined. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is

used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
relocation processing: /lib/libdl.so.2 (lazy)
symbol=__pthread_once;  lookup in file=ex
symbol=__pthread_once;  lookup in file=/lib/libtermcap.so.2
symbol=__pthread_once;  lookup in file=/lib/libacl.so.1
symbol=__pthread_once;  lookup in file=/lib/libdl.so.2
symbol=__pthread_once;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_once;  lookup in file=/lib/libattr.so.1
symbol=__pthread_once;  lookup in file=/lib/ld-linux.so.2
```

The symbol __pthread_key_create is undefined. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_key_create;  lookup in file=ex
symbol=__pthread_key_create;  lookup in file=/lib/libtermcap.so.2
symbol=__pthread_key_create;  lookup in file=/lib/libacl.so.1
symbol=__pthread_key_create;  lookup in file=/lib/libdl.so.2
symbol=__pthread_key_create;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_key_create;  lookup in file=/lib/libattr.so.1
symbol=__pthread_key_create;  lookup in file=/lib/ld-linux.so.2
```

The symbol _libc_intl_domainname comes from SYS_libc in libc. The symbol is in a read only data section. The symbol is global.

```
symbol=_libc_intl_domainname;  lookup in file=ex
symbol=_libc_intl_domainname;  lookup in file=/lib/libtermcap.so.2
symbol=_libc_intl_domainname;  lookup in file=/lib/libacl.so.1
symbol=_libc_intl_domainname;  lookup in file=/lib/libdl.so.2
symbol=_libc_intl_domainname;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libdl.so.2 to /lib/tls/libc.so.6:
  normal symbol '_libc_intl_domainname' [GLIBC_2.0]
```

The symbol __pthread_getspecific is undefined. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_getspecific;  lookup in file=ex
symbol=__pthread_getspecific;  lookup in file=/lib/libtermcap.so.2
```

```
symbol=__pthread_getspecific;  lookup in file=/lib/libacl.so.1
symbol=__pthread_getspecific;  lookup in file=/lib/libdl.so.2
symbol=__pthread_getspecific;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_getspecific;  lookup in file=/lib/libattr.so.1
symbol=__pthread_getspecific;  lookup in file=/lib/ld-linux.so.2
```

The symbol `__cxa_finalize` was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libdl.so.2 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

The symbol `stdin` comes from stdio in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=stdin;  lookup in file=ex
binding file /lib/libdl.so.2 to ex: normal symbol `stdin' [GLIBC_2.0]
```

The symbol `__pthread_setspecific` is not found. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_setspecific;  lookup in file=ex
symbol=__pthread_setspecific;  lookup in file=/lib/libtermcap.so.2
symbol=__pthread_setspecific;  lookup in file=/lib/libacl.so.1
symbol=__pthread_setspecific;  lookup in file=/lib/libdl.so.2
symbol=__pthread_setspecific;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_setspecific;  lookup in file=/lib/libattr.so.1
symbol=__pthread_setspecific;  lookup in file=/lib/ld-linux.so.2
```

The symbol `_Jv_RegisterClasses` is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=_Jv_RegisterClasses;  lookup in file=ex
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libtermcap.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libacl.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libdl.so.2
```

```
symbol=_Jv_RegisterClasses;  lookup in file=/lib/tls/libc.so.6
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libattr.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/ld-linux.so.2
```

The symbol __gmon_start__ is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__gmon_start__;  lookup in file=ex
symbol=__gmon_start__;  lookup in file=/lib/libtermcap.so.2
symbol=__gmon_start__;  lookup in file=/lib/libacl.so.1
symbol=__gmon_start__;  lookup in file=/lib/libdl.so.2
symbol=__gmon_start__;  lookup in file=/lib/tls/libc.so.6
symbol=__gmon_start__;  lookup in file=/lib/libattr.so.1
symbol=__gmon_start__;  lookup in file=/lib/ld-linux.so.2
```

The symbol __cxa_finalize was found in libc. The symbol is in the text (code) section. The symbol is global.

```
relocation processing: /lib/libacl.so.1 (lazy)
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libacl.so.1 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

The symbol _Jv_RegisterClasses is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=_Jv_RegisterClasses;  lookup in file=ex
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libtermcap.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libacl.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libdl.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/tls/libc.so.6
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libattr.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/ld-linux.so.2
```

The symbol __gmon_start__ is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined

symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__gmon_start__;  lookup in file=ex
symbol=__gmon_start__;  lookup in file=/lib/libtermcap.so.2
symbol=__gmon_start__;  lookup in file=/lib/libacl.so.1
symbol=__gmon_start__;  lookup in file=/lib/libdl.so.2
symbol=__gmon_start__;  lookup in file=/lib/tls/libc.so.6
symbol=__gmon_start__;  lookup in file=/lib/libattr.so.1
symbol=__gmon_start__;  lookup in file=/lib/ld-linux.so.2
```

Note that libtermcap has no symbols. The symbol `ospeed` is defined in ex. The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
relocation processing: /lib/libtermcap.so.2 (lazy)
symbol=ospeed;  lookup in file=ex
binding file /lib/libtermcap.so.2 to ex: normal symbol ‘ospeed’
```

Note that libtermcap has no symbols. The symbol `BC` is defined in ex. The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=BC;  lookup in file=ex
binding file /lib/libtermcap.so.2 to ex: normal symbol ‘BC’
```

Note that libtermcap has no symbols. The symbol `PC` is defined in ex. The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=PC;  lookup in file=ex
binding file /lib/libtermcap.so.2 to ex: normal symbol ‘PC’
```

The symbol `__cxa_finalize` was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘__cxa_finalize’ [GLIBC_2.1.3]
\begin{verbatim}

Note that libtermcap has no symbols.
The symbol {\tt UP} is defined in ex.
The symbol is in the uninitialized data section (known as BSS).
The symbol is global.
```

```
\begin{verbatim}
 symbol=UP;  lookup in file=ex
 binding file /lib/libtermcap.so.2 to ex: normal symbol `UP'
```

The symbol _Jv_RegisterClasses is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
 symbol=_Jv_RegisterClasses;  lookup in file=ex
 symbol=_Jv_RegisterClasses;  lookup in file=/lib/libtermcap.so.2
 symbol=_Jv_RegisterClasses;  lookup in file=/lib/libacl.so.1
 symbol=_Jv_RegisterClasses;  lookup in file=/lib/libdl.so.2
 symbol=_Jv_RegisterClasses;  lookup in file=/lib/tls/libc.so.6
 symbol=_Jv_RegisterClasses;  lookup in file=/lib/libattr.so.1
 symbol=_Jv_RegisterClasses;  lookup in file=/lib/ld-linux.so.2
```

The symbol __gmon_start__ is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
 symbol=__gmon_start__;  lookup in file=ex
 symbol=__gmon_start__;  lookup in file=/lib/libtermcap.so.2
 symbol=__gmon_start__;  lookup in file=/lib/libacl.so.1
 symbol=__gmon_start__;  lookup in file=/lib/libdl.so.2
 symbol=__gmon_start__;  lookup in file=/lib/tls/libc.so.6
 symbol=__gmon_start__;  lookup in file=/lib/libattr.so.1
 symbol=__gmon_start__;  lookup in file=/lib/ld-linux.so.2
```

The symbol __gmon_start__ is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
 relocation processing: ex (lazy)
 symbol=__gmon_start__;  lookup in file=ex
 symbol=__gmon_start__;  lookup in file=/lib/libtermcap.so.2
 symbol=__gmon_start__;  lookup in file=/lib/libacl.so.1
 symbol=__gmon_start__;  lookup in file=/lib/libdl.so.2
 symbol=__gmon_start__;  lookup in file=/lib/tls/libc.so.6
 symbol=__gmon_start__;  lookup in file=/lib/libattr.so.1
 symbol=__gmon_start__;  lookup in file=/lib/ld-linux.so.2
```

Note that libtermcap has no symbols. The symbol `ospeed` is defined in ex. The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=ospeed;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2: normal symbol ‘ospeed’
```

Note that libtermcap has no symbols. The symbol `BC` is defined in ex. The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=BC;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2: normal symbol ‘BC’
```

The symbol `stdout` comes from /lib/tls/libc.so.6. The symbol is in the initialized data section. The symbol is global.

```
symbol=stdout;  lookup in file=/lib/libtermcap.so.2
symbol=stdout;  lookup in file=/lib/libacl.so.1
symbol=stdout;  lookup in file=/lib/libdl.so.2
symbol=stdout;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘stdout’ [GLIBC_2.0]
```

The symbol `stderr` comes from stdio in libc. The symbol is in the initialized data section. The symbol is global.

```
symbol=stderr;  lookup in file=/lib/libtermcap.so.2
symbol=stderr;  lookup in file=/lib/libacl.so.1
symbol=stderr;  lookup in file=/lib/libdl.so.2
symbol=stderr;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘stderr’ [GLIBC_2.0]
```

Note that libtermcap has no symbols. The symbol `PC` is defined in ex. The symbol is in the uninitialized data section (known as BSS). The symbol is global.

```
symbol=PC;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2: normal symbol ‘PC’
```

The symbol `_nl_msg_cat_cntr` comes from loadmsgcat in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=_nl_msg_cat_cntr;  lookup in file=/lib/libtermcap.so.2
symbol=_nl_msg_cat_cntr;  lookup in file=/lib/libacl.so.1
symbol=_nl_msg_cat_cntr;  lookup in file=/lib/libdl.so.2
symbol=_nl_msg_cat_cntr;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘_nl_msg_cat_cntr’ [GLIBC_2.0]
```

The symbol stdin comes from stdio in libc. The symbol is in the initialized data section. The symbol is global.

```
 symbol=stdin;  lookup in file=/lib/libtermcap.so.2
 symbol=stdin;  lookup in file=/lib/libacl.so.1
 symbol=stdin;  lookup in file=/lib/libdl.so.2
 symbol=stdin;  lookup in file=/lib/tls/libc.so.6
 binding file ex to /lib/tls/libc.so.6:
   normal symbol 'stdin' [GLIBC_2.0]
\begin{verbatim}

Note that libtermcap has no symbols.
The symbol {\tt UP} is defined in ex.
The symbol is in the uninitialized data section (known as BSS).
The symbol is global.
\begin{verbatim}
 symbol=UP;  lookup in file=/lib/libtermcap.so.2
 binding file ex to /lib/libtermcap.so.2: normal symbol 'UP'
```

The symbol __pthread_mutex_lock comes from unknown in ld-linux.so.2 The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
 relocation processing: /lib/ld-linux.so.2
 symbol=__pthread_mutex_lock;  lookup in file=ex
 symbol=__pthread_mutex_lock;  lookup in file=/lib/libtermcap.so.2
 symbol=__pthread_mutex_lock;  lookup in file=/lib/libacl.so.1
 symbol=__pthread_mutex_lock;  lookup in file=/lib/libdl.so.2
 symbol=__pthread_mutex_lock;  lookup in file=/lib/tls/libc.so.6
 symbol=__pthread_mutex_lock;  lookup in file=/lib/libattr.so.1
 symbol=__pthread_mutex_lock;  lookup in file=/lib/ld-linux.so.2
```

The symbol __libc_stack_end comes from dl-support in libc. The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
 symbol=__libc_stack_end;  lookup in file=ex
 symbol=__libc_stack_end;  lookup in file=/lib/libtermcap.so.2
 symbol=__libc_stack_end;  lookup in file=/lib/libacl.so.1
 symbol=__libc_stack_end;  lookup in file=/lib/libdl.so.2
 symbol=__libc_stack_end;  lookup in file=/lib/tls/libc.so.6
 symbol=__libc_stack_end;  lookup in file=/lib/libattr.so.1
 symbol=__libc_stack_end;  lookup in file=/lib/ld-linux.so.2
```

```
binding file /lib/ld-linux.so.2 to /lib/ld-linux.so.2:
  normal symbol '__libc_stack_end' [GLIBC_2.1]
```

The symbol __pthread_mutex_unlock is undefined. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_mutex_unlock;  lookup in file=ex
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libtermcap.so.2
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libacl.so.1
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libdl.so.2
symbol=__pthread_mutex_unlock;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libattr.so.1
symbol=__pthread_mutex_unlock;  lookup in file=/lib/ld-linux.so.2
```

The symbol _r_debug comes from dl-debug in ld-linux.so.2. The symbol is global. The symbol is in the uninitialized data section (known as BSS). The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references.

```
symbol=_r_debug;  lookup in file=ex
symbol=_r_debug;  lookup in file=/lib/libtermcap.so.2
symbol=_r_debug;  lookup in file=/lib/libacl.so.1
symbol=_r_debug;  lookup in file=/lib/libdl.so.2
symbol=_r_debug;  lookup in file=/lib/tls/libc.so.6
symbol=_r_debug;  lookup in file=/lib/libattr.so.1
symbol=_r_debug;  lookup in file=/lib/ld-linux.so.2
binding file /lib/ld-linux.so.2 to /lib/ld-linux.so.2:
  normal symbol '_r_debug' [GLIBC_2.0]
```

The symbol __libc_memalign comes from malloc in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__libc_memalign;  lookup in file=ex
symbol=__libc_memalign;  lookup in file=/lib/libtermcap.so.2
symbol=__libc_memalign;  lookup in file=/lib/libacl.so.1
symbol=__libc_memalign;  lookup in file=/lib/libdl.so.2
symbol=__libc_memalign;  lookup in file=/lib/tls/libc.so.6
binding file /lib/ld-linux.so.2 to /lib/tls/libc.so.6:
  normal symbol '__libc_memalign' [GLIBC_2.0]
```

The symbol malloc comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined

symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=malloc;  lookup in file=ex
symbol=malloc;  lookup in file=/lib/libtermcap.so.2
symbol=malloc;  lookup in file=/lib/libacl.so.1
symbol=malloc;  lookup in file=/lib/libdl.so.2
symbol=malloc;  lookup in file=/lib/tls/libc.so.6
binding file /lib/ld-linux.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘malloc’ [GLIBC_2.0]
```

The symbol `calloc` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=calloc;  lookup in file=ex
symbol=calloc;  lookup in file=/lib/libtermcap.so.2
symbol=calloc;  lookup in file=/lib/libacl.so.1
symbol=calloc;  lookup in file=/lib/libdl.so.2
symbol=calloc;  lookup in file=/lib/tls/libc.so.6
binding file /lib/ld-linux.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘calloc’ [GLIBC_2.0]
```

The symbol `realloc` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=realloc;  lookup in file=ex
symbol=realloc;  lookup in file=/lib/libtermcap.so.2
symbol=realloc;  lookup in file=/lib/libacl.so.1
symbol=realloc;  lookup in file=/lib/libdl.so.2
symbol=realloc;  lookup in file=/lib/tls/libc.so.6
binding file /lib/ld-linux.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘realloc’ [GLIBC_2.0]
```

The symbol `free` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=free;  lookup in file=ex
symbol=free;  lookup in file=/lib/libtermcap.so.2
symbol=free;  lookup in file=/lib/libacl.so.1
symbol=free;  lookup in file=/lib/libdl.so.2
symbol=free;  lookup in file=/lib/tls/libc.so.6
binding file /lib/ld-linux.so.2 to /lib/tls/libc.so.6:
  normal symbol 'free' [GLIBC_2.0]
```

The symbol `init` is in ex. The symbol is in the text (code) section. The symbol is global.

```
calling init: /lib/tls/libc.so.6
```

The symbol `strrchr` comes from strrchr in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strrchr;  lookup in file=ex
symbol=strrchr;  lookup in file=/lib/libtermcap.so.2
symbol=strrchr;  lookup in file=/lib/libacl.so.1
symbol=strrchr;  lookup in file=/lib/libdl.so.2
symbol=strrchr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'strrchr' [GLIBC_2.0]
```

Note that /lib/libattr.so.1 has no symbols.

```
calling init: /lib/libattr.so.1
```

The symbol `init` is in /lib/libdl.so.2. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
calling init: /lib/libdl.so.2
```

Note that /lib/libacl.so.1 has no symbols.

```
calling init: /lib/libacl.so.1
```

Note that /lib/libtermcap.so.2 has no symbols.

```
calling init: /lib/libtermcap.so.2
```

This is the entry point of the user's program from the loader. The symbol `__libc_start_main` comes from libc-start in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__libc_start_main;  lookup in file=ex
symbol=__libc_start_main;  lookup in file=/lib/libtermcap.so.2
symbol=__libc_start_main;  lookup in file=/lib/libacl.so.1
symbol=__libc_start_main;  lookup in file=/lib/libdl.so.2
symbol=__libc_start_main;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol '__libc_start_main' [GLIBC_2.0]
```

The symbol `_dl_debug_printf` comes from dl-misc in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_debug_printf;  lookup in file=ex
symbol=_dl_debug_printf;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_debug_printf;  lookup in file=/lib/libacl.so.1
symbol=_dl_debug_printf;  lookup in file=/lib/libdl.so.2
symbol=_dl_debug_printf;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_debug_printf;  lookup in file=/lib/libattr.so.1
symbol=_dl_debug_printf;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol '_dl_debug_printf' [GLIBC_PRIVATE]

initialize program: ex


transferring control: ex
```

The symbol `getrlimit64` comes from getrlimit64 in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=getrlimit64;  lookup in file=ex
symbol=getrlimit64;  lookup in file=/lib/libtermcap.so.2
symbol=getrlimit64;  lookup in file=/lib/libacl.so.1
symbol=getrlimit64;  lookup in file=/lib/libdl.so.2
symbol=getrlimit64;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol 'getrlimit64' [GLIBC_2.2]
```

The symbol `malloc` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=malloc;  lookup in file=ex
symbol=malloc;  lookup in file=/lib/libtermcap.so.2
symbol=malloc;  lookup in file=/lib/libacl.so.1
symbol=malloc;  lookup in file=/lib/libdl.so.2
symbol=malloc;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘malloc’ [GLIBC_2.0]
```

The symbol `malloc` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=malloc;  lookup in file=ex
symbol=malloc;  lookup in file=/lib/libtermcap.so.2
symbol=malloc;  lookup in file=/lib/libacl.so.1
symbol=malloc;  lookup in file=/lib/libdl.so.2
symbol=malloc;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘malloc’ [GLIBC_2.0]
```

The symbol `__libc_malloc` comes from malloc in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__libc_malloc;  lookup in file=ex
symbol=__libc_malloc;  lookup in file=/lib/libtermcap.so.2
symbol=__libc_malloc;  lookup in file=/lib/libacl.so.1
symbol=__libc_malloc;  lookup in file=/lib/libdl.so.2
symbol=__libc_malloc;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘__libc_malloc’ [GLIBC_2.0]
```

The symbol `sigaltstack` comes from sigaltstack in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=sigaltstack;  lookup in file=ex
symbol=sigaltstack;  lookup in file=/lib/libtermcap.so.2
symbol=sigaltstack;  lookup in file=/lib/libacl.so.1
symbol=sigaltstack;  lookup in file=/lib/libdl.so.2
symbol=sigaltstack;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘sigaltstack’ [GLIBC_2.0]
```

The symbol `qsort` comes from msort in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=qsort;  lookup in file=ex
symbol=qsort;  lookup in file=/lib/libtermcap.so.2
symbol=qsort;  lookup in file=/lib/libacl.so.1
symbol=qsort;  lookup in file=/lib/libdl.so.2
symbol=qsort;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘qsort’ [GLIBC_2.0]
```

The symbol `memcpy` comes from memcpy in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=memcpy;  lookup in file=ex
symbol=memcpy;  lookup in file=/lib/libtermcap.so.2
symbol=memcpy;  lookup in file=/lib/libacl.so.1
symbol=memcpy;  lookup in file=/lib/libdl.so.2
symbol=memcpy;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘memcpy’ [GLIBC_2.0]
```

The symbol `setlocale` comes from setlocale in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=setlocale;  lookup in file=ex
symbol=setlocale;  lookup in file=/lib/libtermcap.so.2
symbol=setlocale;  lookup in file=/lib/libacl.so.1
symbol=setlocale;  lookup in file=/lib/libdl.so.2
symbol=setlocale;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘setlocale’ [GLIBC_2.0]
```

The symbol `strcmp` comes from strcmp in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strcmp;  lookup in file=ex
symbol=strcmp;  lookup in file=/lib/libtermcap.so.2
symbol=strcmp;  lookup in file=/lib/libacl.so.1
symbol=strcmp;  lookup in file=/lib/libdl.so.2
symbol=strcmp;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘strcmp’ [GLIBC_2.0]
```

The symbol `strlen` comes from strlen in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strlen;  lookup in file=ex
symbol=strlen;  lookup in file=/lib/libtermcap.so.2
```

```
symbol=strlen;  lookup in file=/lib/libacl.so.1
symbol=strlen;  lookup in file=/lib/libdl.so.2
symbol=strlen;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'strlen' [GLIBC_2.0]
```

The symbol `strncmp` comes from strncmp in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strncmp;  lookup in file=ex
symbol=strncmp;  lookup in file=/lib/libtermcap.so.2
symbol=strncmp;  lookup in file=/lib/libacl.so.1
symbol=strncmp;  lookup in file=/lib/libdl.so.2
symbol=strncmp;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'strncmp' [GLIBC_2.0]
```

The symbol `strchr` comes from strchr in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strchr;  lookup in file=ex
symbol=strchr;  lookup in file=/lib/libtermcap.so.2
symbol=strchr;  lookup in file=/lib/libacl.so.1
symbol=strchr;  lookup in file=/lib/libdl.so.2
symbol=strchr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'strchr' [GLIBC_2.0]
```

The symbol `free` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=free;  lookup in file=ex
symbol=free;  lookup in file=/lib/libtermcap.so.2
symbol=free;  lookup in file=/lib/libacl.so.1
symbol=free;  lookup in file=/lib/libdl.so.2
symbol=free;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'free' [GLIBC_2.0]
```

The symbol `isatty` comes from isatty in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=isatty;  lookup in file=ex
symbol=isatty;  lookup in file=/lib/libtermcap.so.2
symbol=isatty;  lookup in file=/lib/libacl.so.1
symbol=isatty;  lookup in file=/lib/libdl.so.2
symbol=isatty;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘isatty’ [GLIBC_2.0]
```

The symbol `memset` comes from memset in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=memset;  lookup in file=ex
symbol=memset;  lookup in file=/lib/libtermcap.so.2
symbol=memset;  lookup in file=/lib/libacl.so.1
symbol=memset;  lookup in file=/lib/libdl.so.2
symbol=memset;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘memset’ [GLIBC_2.0]
```

The symbol `getenv` comes from getenv in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=getenv;  lookup in file=ex
symbol=getenv;  lookup in file=/lib/libtermcap.so.2
symbol=getenv;  lookup in file=/lib/libacl.so.1
symbol=getenv;  lookup in file=/lib/libdl.so.2
symbol=getenv;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘getenv’ [GLIBC_2.0]
```

The symbol `getcwd` comes from getcwd in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=getcwd;  lookup in file=ex
symbol=getcwd;  lookup in file=/lib/libtermcap.so.2
symbol=getcwd;  lookup in file=/lib/libacl.so.1
symbol=getcwd;  lookup in file=/lib/libdl.so.2
symbol=getcwd;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘getcwd’ [GLIBC_2.0]
```

The symbol `chdir` comes from chdir in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined

symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=chdir;  lookup in file=ex
symbol=chdir;  lookup in file=/lib/libtermcap.so.2
symbol=chdir;  lookup in file=/lib/libacl.so.1
symbol=chdir;  lookup in file=/lib/libdl.so.2
symbol=chdir;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘chdir’ [GLIBC_2.0]
```

The symbol `strlen` comes from strlen in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strlen;  lookup in file=ex
symbol=strlen;  lookup in file=/lib/libtermcap.so.2
symbol=strlen;  lookup in file=/lib/libacl.so.1
symbol=strlen;  lookup in file=/lib/libdl.so.2
symbol=strlen;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘strlen’ [GLIBC_2.0]
```

The symbol `memmove` is in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=memmove;  lookup in file=ex
symbol=memmove;  lookup in file=/lib/libtermcap.so.2
symbol=memmove;  lookup in file=/lib/libacl.so.1
symbol=memmove;  lookup in file=/lib/libdl.so.2
symbol=memmove;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘memmove’ [GLIBC_2.0]
```

The symbol `strcmp` comes from strcmp in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strcmp;  lookup in file=ex
symbol=strcmp;  lookup in file=/lib/libtermcap.so.2
symbol=strcmp;  lookup in file=/lib/libacl.so.1
symbol=strcmp;  lookup in file=/lib/libdl.so.2
symbol=strcmp;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘strcmp’ [GLIBC_2.0]
```

The symbol `strcat` comes from strcat in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strcat;  lookup in file=ex
symbol=strcat;  lookup in file=/lib/libtermcap.so.2
symbol=strcat;  lookup in file=/lib/libacl.so.1
symbol=strcat;  lookup in file=/lib/libdl.so.2
symbol=strcat;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘strcat’ [GLIBC_2.0]
```

The symbol `free` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=free;  lookup in file=ex
symbol=free;  lookup in file=/lib/libtermcap.so.2
symbol=free;  lookup in file=/lib/libacl.so.1
symbol=free;  lookup in file=/lib/libdl.so.2
symbol=free;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘free’ [GLIBC_2.0]
```

The symbol `sysinfo` comes from sysinfo in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=sysinfo;  lookup in file=ex
symbol=sysinfo;  lookup in file=/lib/libtermcap.so.2
symbol=sysinfo;  lookup in file=/lib/libacl.so.1
symbol=sysinfo;  lookup in file=/lib/libdl.so.2
symbol=sysinfo;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘sysinfo’ [GLIBC_2.0]
```

The symbol `__ctype_b_loc` comes from ctype-info in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__ctype_b_loc;  lookup in file=ex
symbol=__ctype_b_loc;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype_b_loc;  lookup in file=/lib/libacl.so.1
symbol=__ctype_b_loc;  lookup in file=/lib/libdl.so.2
symbol=__ctype_b_loc;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘__ctype_b_loc’ [GLIBC_2.3]
```

The symbol `__strtol_internal` comes from strtol in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__strtol_internal;  lookup in file=ex
symbol=__strtol_internal;  lookup in file=/lib/libtermcap.so.2
symbol=__strtol_internal;  lookup in file=/lib/libacl.so.1
symbol=__strtol_internal;  lookup in file=/lib/libdl.so.2
symbol=__strtol_internal;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol `__strtol_internal' [GLIBC_2.0]
```

The symbol `strncmp` comes from strncmp in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strncmp;  lookup in file=ex
symbol=strncmp;  lookup in file=/lib/libtermcap.so.2
symbol=strncmp;  lookup in file=/lib/libacl.so.1
symbol=strncmp;  lookup in file=/lib/libdl.so.2
symbol=strncmp;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol `strncmp' [GLIBC_2.0]
```

The symbol `strcpy` comes from strcpy in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strcpy;  lookup in file=ex
symbol=strcpy;  lookup in file=/lib/libtermcap.so.2
symbol=strcpy;  lookup in file=/lib/libacl.so.1
symbol=strcpy;  lookup in file=/lib/libdl.so.2
symbol=strcpy;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol `strcpy' [GLIBC_2.0]
```

The symbol `strncpy` comes from strncpy in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strncpy;  lookup in file=ex
symbol=strncpy;  lookup in file=/lib/libtermcap.so.2
symbol=strncpy;  lookup in file=/lib/libacl.so.1
symbol=strncpy;  lookup in file=/lib/libdl.so.2
symbol=strncpy;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol `strncpy' [GLIBC_2.0]
```

The symbol `nl_langinfo` comes from nl_langinfo in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=nl_langinfo;  lookup in file=ex
symbol=nl_langinfo;  lookup in file=/lib/libtermcap.so.2
symbol=nl_langinfo;  lookup in file=/lib/libacl.so.1
symbol=nl_langinfo;  lookup in file=/lib/libdl.so.2
symbol=nl_langinfo;  lookup in file=/lib/tls/libc.so.6
```

```
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘nl_langinfo’ [GLIBC_2.0]
```

The symbol `bind_textdomain_codeset` comes from bindtestcom in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=bind_textdomain_codeset;  lookup in file=ex
symbol=bind_textdomain_codeset;  lookup in file=/lib/libtermcap.so.2
symbol=bind_textdomain_codeset;  lookup in file=/lib/libacl.so.1
symbol=bind_textdomain_codeset;  lookup in file=/lib/libdl.so.2
symbol=bind_textdomain_codeset;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘bind_textdomain_codeset’ [GLIBC_2.2]
```

The symbol `strncasecmp` comes from strncase in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=strncasecmp;  lookup in file=ex
symbol=strncasecmp;  lookup in file=/lib/libtermcap.so.2
symbol=strncasecmp;  lookup in file=/lib/libacl.so.1
symbol=strncasecmp;  lookup in file=/lib/libdl.so.2
symbol=strncasecmp;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘strncasecmp’ [GLIBC_2.0]
```

The symbol `strcasecmp` comes from strcasecmp in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=strcasecmp;  lookup in file=ex
symbol=strcasecmp;  lookup in file=/lib/libtermcap.so.2
symbol=strcasecmp;  lookup in file=/lib/libacl.so.1
symbol=strcasecmp;  lookup in file=/lib/libdl.so.2
symbol=strcasecmp;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘strcasecmp’ [GLIBC_2.0]
```

The symbol `sigset` comes from sigset in libc. The symbol is in the text (code) section. The symbol is global.

```
 symbol=sigset;  lookup in file=ex
 symbol=sigset;  lookup in file=/lib/libtermcap.so.2
 symbol=sigset;  lookup in file=/lib/libacl.so.1
 symbol=sigset;  lookup in file=/lib/libdl.so.2
 symbol=sigset;  lookup in file=/lib/tls/libc.so.6
 binding file ex to /lib/tls/libc.so.6:
   normal symbol ‘sigset’ [GLIBC_2.1]
```

The symbol `sigemptyset` comes from sigempty in libc. The symbol is in the text (code) section. The symbol is global.

```
 symbol=sigemptyset;  lookup in file=ex
 symbol=sigemptyset;  lookup in file=/lib/libtermcap.so.2
 symbol=sigemptyset;  lookup in file=/lib/libacl.so.1
 symbol=sigemptyset;  lookup in file=/lib/libdl.so.2
 symbol=sigemptyset;  lookup in file=/lib/tls/libc.so.6
 binding file ex to /lib/tls/libc.so.6:
   normal symbol ‘sigemptyset’ [GLIBC_2.0]
```

The symbol `memset` comes from memset in libc. The symbol is in the text (code) section. The symbol is global.

```
 symbol=memset;  lookup in file=ex
 symbol=memset;  lookup in file=/lib/libtermcap.so.2
 symbol=memset;  lookup in file=/lib/libacl.so.1
 symbol=memset;  lookup in file=/lib/libdl.so.2
 symbol=memset;  lookup in file=/lib/tls/libc.so.6
 binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
   normal symbol ‘memset’ [GLIBC_2.0]
\begin{verbatim}

The symbol {\tt sigaction} comes from sigaction in libc.
The  symbol  is  a  weak  symbol that has not been specifically
tagged as a weak object symbol.  When a weak defined symbol  is
linked  with a normal defined symbol, the normal defined symbol
is used with no error.  When a weak undefined symbol is  linked
and  the  symbol  is  not defined, the value of the weak symbol
becomes zero with no error.
The symbol is global.
\begin{verbatim}
 symbol=sigaction;  lookup in file=ex
 symbol=sigaction;  lookup in file=/lib/libtermcap.so.2
 symbol=sigaction;  lookup in file=/lib/libacl.so.1
 symbol=sigaction;  lookup in file=/lib/libdl.so.2
 symbol=sigaction;  lookup in file=/lib/tls/libc.so.6
```

```
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘sigaction’ [GLIBC_2.0]
```

The symbol `strstr` comes from strstr in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strstr;  lookup in file=ex
symbol=strstr;  lookup in file=/lib/libtermcap.so.2
symbol=strstr;  lookup in file=/lib/libacl.so.1
symbol=strstr;  lookup in file=/lib/libdl.so.2
symbol=strstr;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘strstr’ [GLIBC_2.0]
```

Note that /lib/libtermcap.so.2 has no symbols.

```
symbol=tgetent;  lookup in file=ex
symbol=tgetent;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2:
  normal symbol ‘tgetent’
```

The symbol `getenv` comes from getenv in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=getenv;  lookup in file=ex
symbol=getenv;  lookup in file=/lib/libtermcap.so.2
symbol=getenv;  lookup in file=/lib/libacl.so.1
symbol=getenv;  lookup in file=/lib/libdl.so.2
symbol=getenv;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘getenv’ [GLIBC_2.0]
```

The symbol `strlen` comes from strlen in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strlen;  lookup in file=ex
symbol=strlen;  lookup in file=/lib/libtermcap.so.2
symbol=strlen;  lookup in file=/lib/libacl.so.1
symbol=strlen;  lookup in file=/lib/libdl.so.2
symbol=strlen;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘strlen’ [GLIBC_2.0]
```

The symbol `fopen` comes from iofopen in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fopen;  lookup in file=ex
symbol=fopen;  lookup in file=/lib/libtermcap.so.2
symbol=fopen;  lookup in file=/lib/libacl.so.1
symbol=fopen;  lookup in file=/lib/libdl.so.2
symbol=fopen;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol 'fopen' [GLIBC_2.1]
```

The symbol `strstr` comes from strstr in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strstr;  lookup in file=ex
symbol=strstr;  lookup in file=/lib/libtermcap.so.2
symbol=strstr;  lookup in file=/lib/libacl.so.1
symbol=strstr;  lookup in file=/lib/libdl.so.2
symbol=strstr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol 'strstr' [GLIBC_2.0]
```

The symbol `rewind` comes from rewind in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=rewind;  lookup in file=ex
symbol=rewind;  lookup in file=/lib/libtermcap.so.2
symbol=rewind;  lookup in file=/lib/libacl.so.1
symbol=rewind;  lookup in file=/lib/libdl.so.2
symbol=rewind;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol 'rewind' [GLIBC_2.0]
```

The symbol `fgets` comes from iofgets in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fgets;  lookup in file=ex
symbol=fgets;  lookup in file=/lib/libtermcap.so.2
symbol=fgets;  lookup in file=/lib/libacl.so.1
symbol=fgets;  lookup in file=/lib/libdl.so.2
symbol=fgets;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol 'fgets' [GLIBC_2.0]
```

The symbol `memchr` comes from memchr in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined

symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=memchr;  lookup in file=ex
symbol=memchr;  lookup in file=/lib/libtermcap.so.2
symbol=memchr;  lookup in file=/lib/libacl.so.1
symbol=memchr;  lookup in file=/lib/libdl.so.2
symbol=memchr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/tls/libc.so.6 to /lib/tls/libc.so.6:
  normal symbol ‘memchr’ [GLIBC_2.0]
```

The symbol `strncmp` comes from strncmp in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strncmp;  lookup in file=ex
symbol=strncmp;  lookup in file=/lib/libtermcap.so.2
symbol=strncmp;  lookup in file=/lib/libacl.so.1
symbol=strncmp;  lookup in file=/lib/libdl.so.2
symbol=strncmp;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘strncmp’ [GLIBC_2.0]
```

The symbol `malloc` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=malloc;  lookup in file=ex
symbol=malloc;  lookup in file=/lib/libtermcap.so.2
symbol=malloc;  lookup in file=/lib/libacl.so.1
symbol=malloc;  lookup in file=/lib/libdl.so.2
symbol=malloc;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘malloc’ [GLIBC_2.0]
```

The symbol `memcpy` comes from memcpy in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=memcpy;  lookup in file=ex
symbol=memcpy;  lookup in file=/lib/libtermcap.so.2
symbol=memcpy;  lookup in file=/lib/libacl.so.1
symbol=memcpy;  lookup in file=/lib/libdl.so.2
symbol=memcpy;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘memcpy’ [GLIBC_2.0]
```

The symbol `strchr` comes from strchr in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strchr;  lookup in file=ex
symbol=strchr;  lookup in file=/lib/libtermcap.so.2
symbol=strchr;  lookup in file=/lib/libacl.so.1
symbol=strchr;  lookup in file=/lib/libdl.so.2
symbol=strchr;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘strchr’ [GLIBC_2.0]
```

The symbol `strncpy` comes from strncpy in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=strncpy;  lookup in file=ex
symbol=strncpy;  lookup in file=/lib/libtermcap.so.2
symbol=strncpy;  lookup in file=/lib/libacl.so.1
symbol=strncpy;  lookup in file=/lib/libdl.so.2
symbol=strncpy;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘strncpy’ [GLIBC_2.0]
```

The symbol `fclose` comes from iofclose in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fclose;  lookup in file=ex
symbol=fclose;  lookup in file=/lib/libtermcap.so.2
symbol=fclose;  lookup in file=/lib/libacl.so.1
symbol=fclose;  lookup in file=/lib/libdl.so.2
symbol=fclose;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘fclose’ [GLIBC_2.1]
```

The symbol `ioctl` comes from ioctl in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=ioctl;  lookup in file=ex
symbol=ioctl;  lookup in file=/lib/libtermcap.so.2
symbol=ioctl;  lookup in file=/lib/libacl.so.1
symbol=ioctl;  lookup in file=/lib/libdl.so.2
```

```
symbol=ioctl;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘ioctl’ [GLIBC_2.0]
```

The symbol `free` comes from malloc in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=free;  lookup in file=ex
symbol=free;  lookup in file=/lib/libtermcap.so.2
symbol=free;  lookup in file=/lib/libacl.so.1
symbol=free;  lookup in file=/lib/libdl.so.2
symbol=free;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘free’ [GLIBC_2.0]
```

Note that /lib/libtermcap.so.2 has no symbols.

```
symbol=tgetstr;  lookup in file=ex
symbol=tgetstr;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2:
  normal symbol ‘tgetstr’
```

Note that /lib/libtermcap.so.2 has no symbols.

```
symbol=tgetflag;  lookup in file=ex
symbol=tgetflag;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2:
  normal symbol ‘tgetflag’
```

Note that /lib/libtermcap.so.2 has no symbols.

```
symbol=tgetnum;  lookup in file=ex
symbol=tgetnum;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2:
  normal symbol ‘tgetnum’
```

The symbol `__strtol_internal` comes from strtol in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__strtol_internal;  lookup in file=ex
symbol=__strtol_internal;  lookup in file=/lib/libtermcap.so.2
symbol=__strtol_internal;  lookup in file=/lib/libacl.so.1
symbol=__strtol_internal;  lookup in file=/lib/libdl.so.2
symbol=__strtol_internal;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol ‘__strtol_internal’ [GLIBC_2.0]
```

The symbol `tcgetattr` comes from tcgetattr in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=tcgetattr;  lookup in file=ex
symbol=tcgetattr;  lookup in file=/lib/libtermcap.so.2
symbol=tcgetattr;  lookup in file=/lib/libacl.so.1
symbol=tcgetattr;  lookup in file=/lib/libdl.so.2
symbol=tcgetattr;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘tcgetattr’ [GLIBC_2.0]
```

The symbol `ioctl` comes from ioctl in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=ioctl;  lookup in file=ex
symbol=ioctl;  lookup in file=/lib/libtermcap.so.2
symbol=ioctl;  lookup in file=/lib/libacl.so.1
symbol=ioctl;  lookup in file=/lib/libdl.so.2
symbol=ioctl;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘ioctl’ [GLIBC_2.0]
```

The symbol `open64` comes from open64 in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=open64;  lookup in file=ex
symbol=open64;  lookup in file=/lib/libtermcap.so.2
symbol=open64;  lookup in file=/lib/libacl.so.1
symbol=open64;  lookup in file=/lib/libdl.so.2
symbol=open64;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘open64’ [GLIBC_2.1]
```

The symbol `fchdir` comes from fchdir in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined

symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fchdir;  lookup in file=ex
symbol=fchdir;  lookup in file=/lib/libtermcap.so.2
symbol=fchdir;  lookup in file=/lib/libacl.so.1
symbol=fchdir;  lookup in file=/lib/libdl.so.2
symbol=fchdir;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘fchdir’ [GLIBC_2.0]
```

The symbol `close` comes from close in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=close;  lookup in file=ex
symbol=close;  lookup in file=/lib/libtermcap.so.2
symbol=close;  lookup in file=/lib/libacl.so.1
symbol=close;  lookup in file=/lib/libdl.so.2
symbol=close;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘close’ [GLIBC_2.0]
```

The symbol `__xstat64` comes from xstat64 in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__xstat64;  lookup in file=ex
symbol=__xstat64;  lookup in file=/lib/libtermcap.so.2
symbol=__xstat64;  lookup in file=/lib/libacl.so.1
symbol=__xstat64;  lookup in file=/lib/libdl.so.2
symbol=__xstat64;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘__xstat64’ [GLIBC_2.2]
```

The symbol `fopen64` comes from iofopen64 in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fopen64;  lookup in file=ex
symbol=fopen64;  lookup in file=/lib/libtermcap.so.2
symbol=fopen64;  lookup in file=/lib/libacl.so.1
```

```
symbol=fopen64;  lookup in file=/lib/libdl.so.2
symbol=fopen64;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘fopen64’ [GLIBC_2.1]
```

The symbol `fgets` comes from iofgets in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fgets;  lookup in file=ex
symbol=fgets;  lookup in file=/lib/libtermcap.so.2
symbol=fgets;  lookup in file=/lib/libacl.so.1
symbol=fgets;  lookup in file=/lib/libdl.so.2
symbol=fgets;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘fgets’ [GLIBC_2.0]
```

The symbol `fclose` comes from iofclose in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fclose;  lookup in file=ex
symbol=fclose;  lookup in file=/lib/libtermcap.so.2
symbol=fclose;  lookup in file=/lib/libacl.so.1
symbol=fclose;  lookup in file=/lib/libdl.so.2
symbol=fclose;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘fclose’ [GLIBC_2.1]
```

The symbol `getuid` comes from getuid in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=getuid;  lookup in file=ex
symbol=getuid;  lookup in file=/lib/libtermcap.so.2
symbol=getuid;  lookup in file=/lib/libacl.so.1
symbol=getuid;  lookup in file=/lib/libdl.so.2
symbol=getuid;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘getuid’ [GLIBC_2.0]
```

The symbol `getpwuid` comes from getpwuid in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=getpwuid;  lookup in file=ex
symbol=getpwuid;  lookup in file=/lib/libtermcap.so.2
symbol=getpwuid;  lookup in file=/lib/libacl.so.1
symbol=getpwuid;  lookup in file=/lib/libdl.so.2
symbol=getpwuid;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol 'getpwuid' [GLIBC_2.0]
```

The symbol `_dl_catch_error` comes from dl-error in libc? The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_catch_error;  lookup in file=ex
symbol=_dl_catch_error;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_catch_error;  lookup in file=/lib/libacl.so.1
symbol=_dl_catch_error;  lookup in file=/lib/libdl.so.2
symbol=_dl_catch_error;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_catch_error;  lookup in file=/lib/libattr.so.1
symbol=_dl_catch_error;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol '_dl_catch_error' [GLIBC_PRIVATE]
```

The symbol `_dl_map_object` comes from dl-load in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_map_object;  lookup in file=ex
symbol=_dl_map_object;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_map_object;  lookup in file=/lib/libacl.so.1
symbol=_dl_map_object;  lookup in file=/lib/libdl.so.2
symbol=_dl_map_object;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_map_object;  lookup in file=/lib/libattr.so.1
symbol=_dl_map_object;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol '_dl_map_object' [GLIBC_PRIVATE]
```

Well, this is new...

```
find library=libnss_files.so.2; searching
 search cache=/etc/ld.so.cache
  trying file=/lib/libnss_files.so.2

file=libnss_files.so.2;  generating link map
  dynamic: 0x4023e434  base: 0x40233000   size: 0x0000b7dc
    entry: 0x40234d10  phdr: 0x40233034  phnum:          6
```

The symbol `_dl_map_object_deps` comes from dl-deps in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_map_object_deps;  lookup in file=ex
symbol=_dl_map_object_deps;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_map_object_deps;  lookup in file=/lib/libacl.so.1
symbol=_dl_map_object_deps;  lookup in file=/lib/libdl.so.2
symbol=_dl_map_object_deps;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_map_object_deps;  lookup in file=/lib/libattr.so.1
symbol=_dl_map_object_deps;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol `_dl_map_object_deps' [GLIBC_PRIVATE]
```

The symbol `_dl_check_map_versions` comes from dl-version in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_check_map_versions;  lookup in file=ex
symbol=_dl_check_map_versions;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_check_map_versions;  lookup in file=/lib/libacl.so.1
symbol=_dl_check_map_versions;  lookup in file=/lib/libdl.so.2
symbol=_dl_check_map_versions;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_check_map_versions;  lookup in file=/lib/libattr.so.1
symbol=_dl_check_map_versions;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol `_dl_check_map_versions' [GLIBC_PRIVATE]

checking for version `GLIBC_2.3' in
  file /lib/tls/libc.so.6 required by file /lib/libnss_files.so.2
checking for version `GLIBC_2.1.3' in
  file /lib/tls/libc.so.6 required by file /lib/libnss_files.so.2
checking for version `GLIBC_PRIVATE' in
  file /lib/tls/libc.so.6 required by file /lib/libnss_files.so.2
checking for version `GLIBC_2.2' in
  file /lib/tls/libc.so.6 required by file /lib/libnss_files.so.2
checking for version `GLIBC_2.1' in
  file /lib/tls/libc.so.6 required by file /lib/libnss_files.so.2
checking for version `GLIBC_2.0' in
  file /lib/tls/libc.so.6 required by file /lib/libnss_files.so.2
```

The symbol `_dl_relocate_object` comes from dl-reloc in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_relocate_object;  lookup in file=ex
symbol=_dl_relocate_object;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_relocate_object;  lookup in file=/lib/libacl.so.1
symbol=_dl_relocate_object;  lookup in file=/lib/libdl.so.2
symbol=_dl_relocate_object;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_relocate_object;  lookup in file=/lib/libattr.so.1
symbol=_dl_relocate_object;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
```

```
  normal symbol '_dl_relocate_object' [GLIBC_PRIVATE]

relocation processing: /lib/libnss_files.so.2 (lazy)
```

The symbol __pthread_mutex_lock comes from unknown in ld-linux.so.2 The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_mutex_lock;  lookup in file=ex
symbol=__pthread_mutex_lock;  lookup in file=/lib/libtermcap.so.2
symbol=__pthread_mutex_lock;  lookup in file=/lib/libacl.so.1
symbol=__pthread_mutex_lock;  lookup in file=/lib/libdl.so.2
symbol=__pthread_mutex_lock;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_mutex_lock;  lookup in file=/lib/libattr.so.1
symbol=__pthread_mutex_lock;  lookup in file=/lib/ld-linux.so.2
symbol=__pthread_mutex_lock;  lookup in file=/lib/libnss_files.so.2
symbol=__pthread_mutex_lock;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_mutex_lock;  lookup in file=/lib/ld-linux.so.2
```

The symbol _res_hconf is in res_hconf in /lib/tls/libc.so.6 The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references. The symbol is global.

```
symbol=_res_hconf;  lookup in file=ex
symbol=_res_hconf;  lookup in file=/lib/libtermcap.so.2
symbol=_res_hconf;  lookup in file=/lib/libacl.so.1
symbol=_res_hconf;  lookup in file=/lib/libdl.so.2
symbol=_res_hconf;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol '_res_hconf' [GLIBC_2.2]
```

The symbol __pthread_mutex_unlock is undefined. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__pthread_mutex_unlock;  lookup in file=ex
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libtermcap.so.2
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libacl.so.1
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libdl.so.2
symbol=__pthread_mutex_unlock;  lookup in file=/lib/tls/libc.so.6
```

```
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libattr.so.1
symbol=__pthread_mutex_unlock;  lookup in file=/lib/ld-linux.so.2
symbol=__pthread_mutex_unlock;  lookup in file=/lib/libnss_files.so.2
symbol=__pthread_mutex_unlock;  lookup in file=/lib/tls/libc.so.6
symbol=__pthread_mutex_unlock;  lookup in file=/lib/ld-linux.so.2
```

The symbol `__cxa_finalize` was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

The symbol `_Jv_RegisterClasses` is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=_Jv_RegisterClasses;  lookup in file=ex
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libtermcap.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libacl.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libdl.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/tls/libc.so.6
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libattr.so.1
symbol=_Jv_RegisterClasses;  lookup in file=/lib/ld-linux.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/libnss_files.so.2
symbol=_Jv_RegisterClasses;  lookup in file=/lib/tls/libc.so.6
symbol=_Jv_RegisterClasses;  lookup in file=/lib/ld-linux.so.2
```

The symbol `__gmon_start__` is in ex. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is local.

```
symbol=__gmon_start__;  lookup in file=ex
symbol=__gmon_start__;  lookup in file=/lib/libtermcap.so.2
symbol=__gmon_start__;  lookup in file=/lib/libacl.so.1
symbol=__gmon_start__;  lookup in file=/lib/libdl.so.2
symbol=__gmon_start__;  lookup in file=/lib/tls/libc.so.6
symbol=__gmon_start__;  lookup in file=/lib/libattr.so.1
```

```
symbol=__gmon_start__;  lookup in file=/lib/ld-linux.so.2
symbol=__gmon_start__;  lookup in file=/lib/libnss_files.so.2
symbol=__gmon_start__;  lookup in file=/lib/tls/libc.so.6
symbol=__gmon_start__;  lookup in file=/lib/ld-linux.so.2
```

The symbol `_dl_init` comes from _dl_init in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_init;  lookup in file=ex
symbol=_dl_init;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_init;  lookup in file=/lib/libacl.so.1
symbol=_dl_init;  lookup in file=/lib/libdl.so.2
symbol=_dl_init;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_init;  lookup in file=/lib/libattr.so.1
symbol=_dl_init;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol `_dl_init' [GLIBC_PRIVATE]
```

The symbol `init` comes from /lib/libnss_files.so.2. The symbol is in the text (code) section. The symbol is local.

```
calling init: /lib/libnss_files.so.2

opening file=/lib/libnss_files.so.2; opencount == 1
```

The symbol `_dl_unload_cache` comes from /lib/ld-linux.so.2 The symbol is in the text (code) section. The symbol is global.

```
symbol=_dl_unload_cache;  lookup in file=ex
symbol=_dl_unload_cache;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_unload_cache;  lookup in file=/lib/libacl.so.1
symbol=_dl_unload_cache;  lookup in file=/lib/libdl.so.2
symbol=_dl_unload_cache;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_unload_cache;  lookup in file=/lib/libattr.so.1
symbol=_dl_unload_cache;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol `_dl_unload_cache' [GLIBC_PRIVATE]
```

The symbol `_dl_lookup_symbol` comes from /lib/ld-linux.so.2 The symbol is in the text (code) section. The symbol is global. Note that this symbol is also exported from libc.

```
symbol=_dl_lookup_symbol;  lookup in file=ex
symbol=_dl_lookup_symbol;  lookup in file=/lib/libtermcap.so.2
symbol=_dl_lookup_symbol;  lookup in file=/lib/libacl.so.1
symbol=_dl_lookup_symbol;  lookup in file=/lib/libdl.so.2
symbol=_dl_lookup_symbol;  lookup in file=/lib/tls/libc.so.6
symbol=_dl_lookup_symbol;  lookup in file=/lib/libattr.so.1
```

```
symbol=_dl_lookup_symbol;  lookup in file=/lib/ld-linux.so.2
binding file /lib/tls/libc.so.6 to /lib/ld-linux.so.2:
  normal symbol '_dl_lookup_symbol' [GLIBC_PRIVATE]
```

The symbol `_nss_files_getpwuid_r` comes from /lib/libnss_files.so.2 The symbol is in the text (code) section. The symbol is global.

```
symbol=_nss_files_getpwuid_r;  lookup in file=/lib/libnss_files.so.2
binding file /lib/libnss_files.so.2 to /lib/libnss_files.so.2:
  normal symbol '_nss_files_getpwuid_r'
```

The symbol `fopen` comes from iofopen in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fopen;  lookup in file=ex
symbol=fopen;  lookup in file=/lib/libtermcap.so.2
symbol=fopen;  lookup in file=/lib/libacl.so.1
symbol=fopen;  lookup in file=/lib/libdl.so.2
symbol=fopen;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol 'fopen' [GLIBC_2.1]
```

The symbol `fileno` comes from fileno in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=fileno;  lookup in file=ex
symbol=fileno;  lookup in file=/lib/libtermcap.so.2
symbol=fileno;  lookup in file=/lib/libacl.so.1
symbol=fileno;  lookup in file=/lib/libdl.so.2
symbol=fileno;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol 'fileno' [GLIBC_2.0]
```

The symbol `fcntl` comes from fcntl in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fcntl;  lookup in file=ex
symbol=fcntl;  lookup in file=/lib/libtermcap.so.2
symbol=fcntl;  lookup in file=/lib/libacl.so.1
symbol=fcntl;  lookup in file=/lib/libdl.so.2
symbol=fcntl;  lookup in file=/lib/tls/libc.so.6
```

```
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol 'fcntl' [GLIBC_2.0]
```

The symbol `fgets_unlocked` comes from iofgets_u in libc The symbol is in the text (code) section. The symbol is global.

```
symbol=fgets_unlocked;  lookup in file=ex
symbol=fgets_unlocked;  lookup in file=/lib/libtermcap.so.2
symbol=fgets_unlocked;  lookup in file=/lib/libacl.so.1
symbol=fgets_unlocked;  lookup in file=/lib/libdl.so.2
symbol=fgets_unlocked;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol 'fgets_unlocked' [GLIBC_2.1]
```

The symbol `__ctype_b_loc` comes from ctype-info in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__ctype_b_loc;  lookup in file=ex
symbol=__ctype_b_loc;  lookup in file=/lib/libtermcap.so.2
symbol=__ctype_b_loc;  lookup in file=/lib/libacl.so.1
symbol=__ctype_b_loc;  lookup in file=/lib/libdl.so.2
symbol=__ctype_b_loc;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol '__ctype_b_loc' [GLIBC_2.3]
```

The symbol `_nss_files_parse_pwent` comes from fgetpwent_r in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=_nss_files_parse_pwent;  lookup in file=ex
symbol=_nss_files_parse_pwent;  lookup in file=/lib/libtermcap.so.2
symbol=_nss_files_parse_pwent;  lookup in file=/lib/libacl.so.1
symbol=_nss_files_parse_pwent;  lookup in file=/lib/libdl.so.2
symbol=_nss_files_parse_pwent;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol '_nss_files_parse_pwent' [GLIBC_PRIVATE]
```

The symbol `fclose` comes from iofclose in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=fclose;  lookup in file=ex
symbol=fclose;  lookup in file=/lib/libtermcap.so.2
symbol=fclose;  lookup in file=/lib/libacl.so.1
symbol=fclose;  lookup in file=/lib/libdl.so.2
symbol=fclose;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol 'fclose' [GLIBC_2.1]
```

The symbol `uname` comes from uname in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=uname;  lookup in file=ex
symbol=uname;  lookup in file=/lib/libtermcap.so.2
symbol=uname;  lookup in file=/lib/libacl.so.1
symbol=uname;  lookup in file=/lib/libdl.so.2
symbol=uname;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘uname’ [GLIBC_2.0]
```

The symbol `getpid` comes from getpid in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=getpid;  lookup in file=ex
symbol=getpid;  lookup in file=/lib/libtermcap.so.2
symbol=getpid;  lookup in file=/lib/libacl.so.1
symbol=getpid;  lookup in file=/lib/libdl.so.2
symbol=getpid;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘getpid’ [GLIBC_2.0]
```

Note that /lib/libtermcap.so.2 has no symbols.

```
symbol=tgoto;  lookup in file=ex
symbol=tgoto;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2:
  normal symbol ‘tgoto’
```

Note that /lib/libtermcap.so.2 has no symbols.

```
symbol=tputs;  lookup in file=ex
symbol=tputs;  lookup in file=/lib/libtermcap.so.2
binding file ex to /lib/libtermcap.so.2:
  normal symbol ‘tputs’
```

The symbol `write` comes from write in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=write;  lookup in file=ex
symbol=write;  lookup in file=/lib/libtermcap.so.2
symbol=write;  lookup in file=/lib/libacl.so.1
symbol=write;  lookup in file=/lib/libdl.so.2
symbol=write;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘write’ [GLIBC_2.0]
```

The symbol `select` comes from select in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=select;  lookup in file=ex
symbol=select;  lookup in file=/lib/libtermcap.so.2
symbol=select;  lookup in file=/lib/libacl.so.1
symbol=select;  lookup in file=/lib/libdl.so.2
symbol=select;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘select’ [GLIBC_2.0]
```

The symbol `read` comes from read in libc. The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. The symbol is global.

```
symbol=read;  lookup in file=ex
symbol=read;  lookup in file=/lib/libtermcap.so.2
symbol=read;  lookup in file=/lib/libacl.so.1
symbol=read;  lookup in file=/lib/libdl.so.2
symbol=read;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘read’ [GLIBC_2.0]
```

The symbol `exit` comes from exit in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=exit;  lookup in file=ex
symbol=exit;  lookup in file=/lib/libtermcap.so.2
symbol=exit;  lookup in file=/lib/libacl.so.1
symbol=exit;  lookup in file=/lib/libdl.so.2
symbol=exit;  lookup in file=/lib/tls/libc.so.6
binding file ex to /lib/tls/libc.so.6:
  normal symbol ‘exit’ [GLIBC_2.0]
```

```
calling fini: /lib/libtermcap.so.2
```

The symbol __cxa_finalize was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libtermcap.so.2 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

Note that /lib/libacl.so.1 has no symbols.

```
calling fini: /lib/libacl.so.1
```

The symbol __cxa_finalize was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libacl.so.1 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

The symbol fini comes from /lib/libdl.so.2 The symbol is in the text (code) section. The symbol is local.

```
calling fini: /lib/libdl.so.2
```

The symbol __cxa_finalize was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libdl.so.2 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

Note that /lib/libattr.so.1 has no symbols.

```
calling fini: /lib/libattr.so.1
```

The symbol __cxa_finalize was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libattr.so.1 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

The symbol fini comes from /lib/libnss_files.so.2 The symbol is in the text (code) section. The symbol is local.

```
calling fini: /lib/libnss_files.so.2
```

The symbol __cxa_finalize was found in libc. The symbol is in the text (code) section. The symbol is global.

```
symbol=__cxa_finalize;  lookup in file=ex
symbol=__cxa_finalize;  lookup in file=/lib/libtermcap.so.2
symbol=__cxa_finalize;  lookup in file=/lib/libacl.so.1
symbol=__cxa_finalize;  lookup in file=/lib/libdl.so.2
symbol=__cxa_finalize;  lookup in file=/lib/tls/libc.so.6
binding file /lib/libnss_files.so.2 to /lib/tls/libc.so.6:
  normal symbol `__cxa_finalize' [GLIBC_2.1.3]
```

Note that the symbol fini does not occur in /lib/tls/libc.so.6

```
calling fini: /lib/tls/libc.so.6
```

# Bibliography


[1] Appel, Andrew, **JLex: A Lexical Analyzer Generator for Java**, http://www.cs.princeton.edu/ appel/modern/java/JLex

[2] Hudson, Scott, **LALR Parser Generator for Java**, http://www2.cs.tum.edu/projects/cup

[3] IA-32 Intel Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z Order Number 253667-016 June 2005, p182

[4] IA-32 Intel Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z Order Number 253667-016 June 2005, Appendix A pA-8

[5] IA-32 Intel Architecture Software Developer's Manual Volume 2A: Instruction Set Reference, A-M Order Number 253666-016 June 2005, Chapter 3

[6] http://www.linuxjournal.com/article/6463

[7] http://www.securityfocus.com/infocus/1872

[8] http://www.securityfocus.com/infocus/1873