SVR4.2-ES-MP/USR/SRC/COMMON/UTS/PROC/OBJ/ELFTYPES.H

```
 1: #ifndef _PROC_OBJ_ELFTYPES_H    /* wrapper symbol for kernel use */
 2: #define _PROC_OBJ_ELFTYPES_H    /* subject to change without notice */
 3:
 4: #ident    "@(#)kern:proc/obj/elftypes.h    1.3"
 5: #ident    "$Header: $"
 6:
 7: #if defined(__cplusplus)
 8: extern "C" {
 9: #endif
10:
11: typedef unsigned long    Elf32_Addr;
12: typedef unsigned short    Elf32_Half;
13: typedef unsigned long    Elf32_Off;
14: typedef long        Elf32_Sword;
15: typedef unsigned long    Elf32_Word;
16:
17: #if defined(__cplusplus)
18:     }
19: #endif
20:
21: #endif /* _PROC_OBJ_ELFTYPES_H  */
```

SVR4.2-ES-MP/USR/SRC/COMMON/UTS/PROC/OBJ/ELF.H

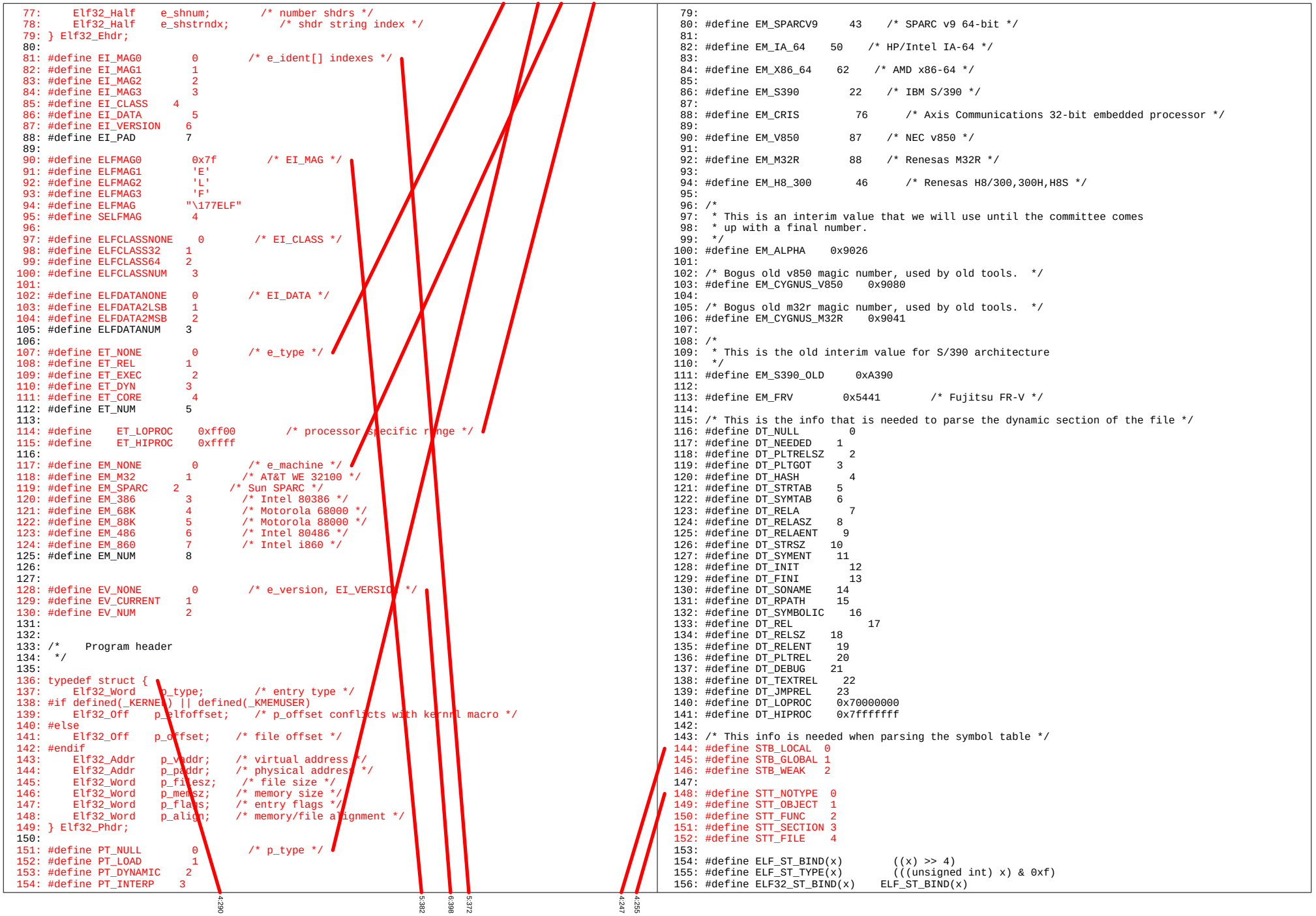
```
22: #ifndef _PROC_OBJ_ELF_H    /* wrapper symbol for kernel use */
23: #define _PROC_OBJ_ELF_H    /* subject to change without notice */
24:
25: #ident    "@(#)kern:proc/obj/elf.h    1.7"
26: #ident    "$Header: $"
27:
28: #if defined(__cplusplus)
29: extern "C" {
30: #endif
31:
32: #ifdef _KERNEL_HEADERS
33:
34: #include <proc/obj/elftypes.h> /* REQUIRED */
35:
36: #elif defined(_KERNEL) || defined(_KMEMUSER)
37:
38: #include <sys/elftypes.h> /* REQUIRED */
39:
40: #else
41:
42: #include <sys/elftypes.h> /* SVR4.0COMPAT */
43:
44: #endif /* _KERNEL_HEADERS */
45:
46: #define ELF32_FSZ_ADDR    4
47: #define ELF32_FSZ_HALF    2
48: #define ELF32_FSZ_OFF    4
49: #define ELF32_FSZ_SWORD    4
50: #define ELF32_FSZ_WORD    4
51:
52:
53: /*    "Enumerations" below use ...NUM as the number of
54:  *    values in the list.  It should be 1 greater than the
55:  *    highest "real" value.
56:  */
57:
58:
59: /*    ELF header
60:  */
61:
62: #define EI_NIDENT    16
63:
64: typedef struct {
65:     unsigned char    e_ident[EI_NIDENT];    /* ident bytes */
66:     Elf32_Half    e_type;            /* file type */
67:     Elf32_Half    e_machine;        /* target machine */
68:     Elf32_Word    e_version;        /* file version */
69:     Elf32_Addr    e_entry;        /* start address */
70:     Elf32_Off    e_phoff;        /* phdr file offset */
71:     Elf32_Off    e_shoff;        /* shdr file offset */
72:     Elf32_Word    e_flags;        /* file flags */
73:     Elf32_Half    e_ehsize;        /* sizeof ehdr */
74:     Elf32_Half    e_phentsize;        /* sizeof phdr */
75:     Elf32_Half    e_phnum;        /* number phdrs */
76:     Elf32_Half    e_shentsize;        /* sizeof shdr */
```

```
 1: #ifndef _LINUX_ELF_H
 2: #define _LINUX_ELF_H
 3:
 4: #include <linux/types.h>
 5: #include <asm/elf.h>
 6:
 7: #ifndef elf_read_implies_exec
 8:   /* Executables for which elf_read_implies_exec() returns TRUE will
 9:      have the READ_IMPLIES_EXEC personality flag set automatically.
10:      Override in asm/elf.h as needed.  */
11: # define elf_read_implies_exec(ex, have_pt_gnu_stack)    0
12: #endif
13:
14: /* 32-bit ELF base types. */
15: typedef __u32    Elf32_Addr;
16: typedef __u16    Elf32_Half;
17: typedef __u32    Elf32_Off;
18: typedef __s32    Elf32_Sword;
19: typedef __u32    Elf32_Word;
20:
21: /* 64-bit ELF base types. */
22: typedef __u64    Elf64_Addr;
23: typedef __u16    Elf64_Half;
24: typedef __s16    Elf64_SHalf;
25: typedef __u64    Elf64_Off;
26: typedef __s32    Elf64_Sword;
27: typedef __u32    Elf64_Word;
28: typedef __u64    Elf64_Xword;
29: typedef __s64    Elf64_Sxword;
30:
31: /* These constants are for the segment types stored in the image headers */
32: #define PT_NULL    0
33: #define PT_LOAD    1
34: #define PT_DYNAMIC 2
35: #define PT_INTERP  3
36: #define PT_NOTE    4
37: #define PT_SHLIB   5
38: #define PT_PHDR    6
39: #define PT_TLS     7               /* Thread local storage segment */
40: #define PT_LOOS    0x60000000      /* OS-specific */
41: #define PT_HIOS    0x6fffffff      /* OS-specific */
42: #define PT_LOPROC  0x70000000
43: #define PT_HIPROC  0x7fffffff
44: #define PT_GNU_EH_FRAME        0x6474e550
45:
46: #define PT_GNU_STACK    (PT_LOOS + 0x474e551)
47:
48: /* These constants define the different elf file types */
49: #define ET_NONE   0
50: #define ET_REL    1
51: #define ET_EXEC   2
52: #define ET_DYN    3
53: #define ET_CORE   4
54: #define ET_LOPROC 0xff00
55: #define ET_HIPROC 0xffff
56:
57: /* These constants define the various ELF target machines */
58: #define EM_NONE  0
59: #define EM_M32   1
60: #define EM_SPARC 2
61: #define EM_386   3
62: #define EM_68K   4
63: #define EM_88K   5
64: #define EM_486   6   /* Perhaps disused */
65: #define EM_860   7
66:
67: #define EM_MIPS        8    /* MIPS R3000 (officially, big-endian only) */
68:
69: #define EM_MIPS_RS4_BE 10    /* MIPS R4000 big-endian */
70:
71: #define EM_PARISC      15    /* HPPA */
72:
73: #define EM_SPARC32PLUS 18    /* Sun's "v8plus" */
74:
75: #define EM_PPC          20    /* PowerPC */
76: #define EM_PPC64       21       /* PowerPC64 */
77:
78: #define EM_SH           42    /* SuperH */
```

4248

3160

(For readability, not all matches are indicated.)

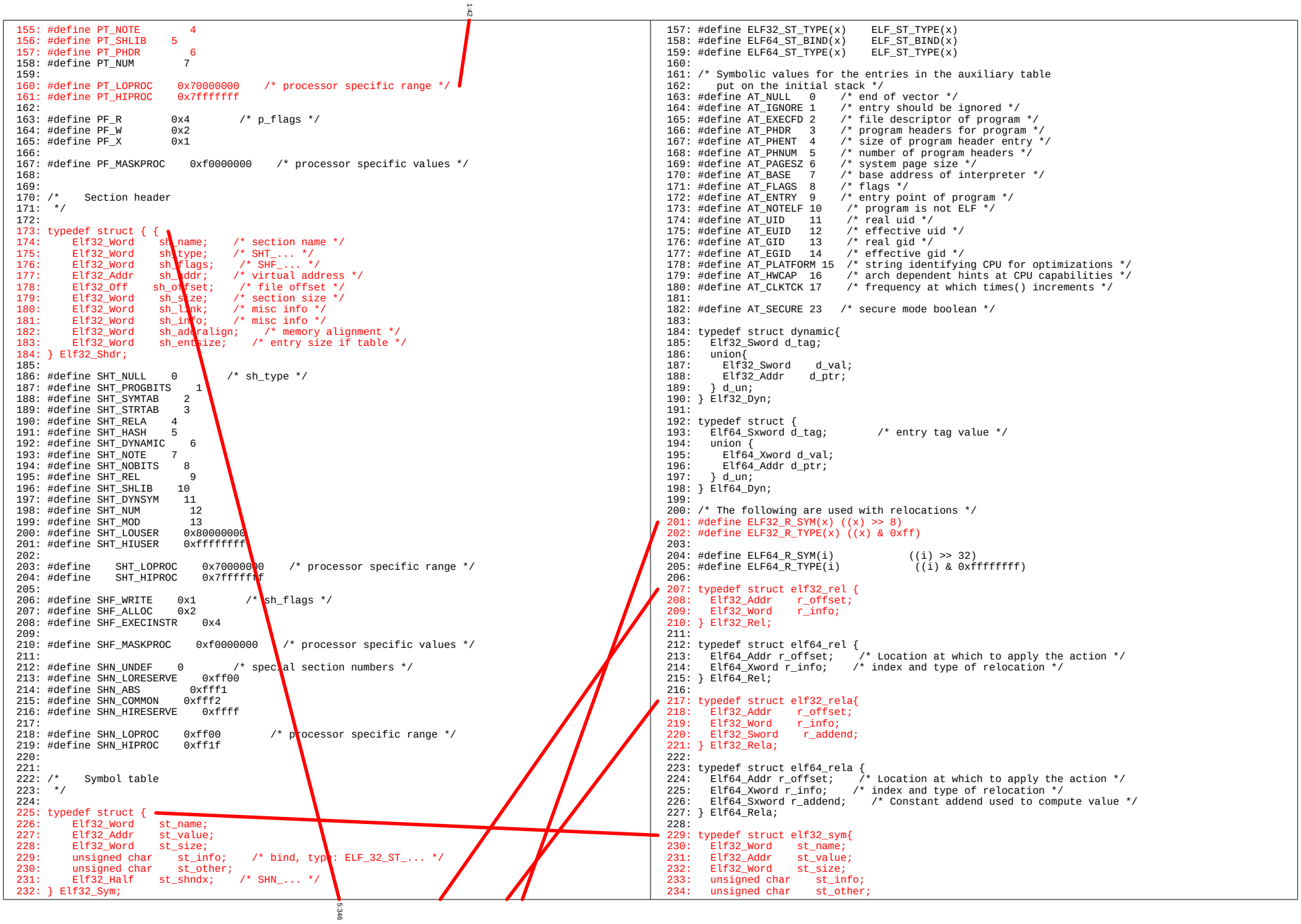


```
 77:     Elf32_Half    e_shnum;        /* number shdrs */
 78:     Elf32_Half    e_shstrndx;        /* shdr string index */
 79: } Elf32_Ehdr;
 80:
 81: #define EI_MAG0        0        /* e_ident[] indexes */
 82: #define EI_MAG1        1
 83: #define EI_MAG2        2
 84: #define EI_MAG3        3
 85: #define EI_CLASS    4
 86: #define EI_DATA        5
 87: #define EI_VERSION    6
 88: #define EI_PAD        7
 89:
 90: #define ELFMAG0        0x7f        /* EI_MAG */
 91: #define ELFMAG1        'E'
 92: #define ELFMAG2        'L'
 93: #define ELFMAG3        'F'
 94: #define ELFMAG        "\177ELF"
 95: #define SELFMAG        4
 96:
 97: #define ELFCLASSNONE    0        /* EI_CLASS */
 98: #define ELFCLASS32    1
 99: #define ELFCLASS64    2
100: #define ELFCLASSNUM    3
101:
102: #define ELFDATANONE    0        /* EI_DATA */
103: #define ELFDATA2LSB    1
104: #define ELFDATA2MSB    2
105: #define ELFDATANUM    3
106:
107: #define ET_NONE        0        /* e_type */
108: #define ET_REL        1
109: #define ET_EXEC        2
110: #define ET_DYN        3
111: #define ET_CORE        4
112: #define ET_NUM        5
113:
114: #define    ET_LOPROC    0xff00        /* processor specific range */
115: #define    ET_HIPROC    0xffff
116:
117: #define EM_NONE        0        /* e_machine */
118: #define EM_M32        1        /* AT&T WE 32100 */
119: #define EM_SPARC    2        /* Sun SPARC */
120: #define EM_386        3        /* Intel 80386 */
121: #define EM_68K        4        /* Motorola 68000 */
122: #define EM_88K        5        /* Motorola 88000 */
123: #define EM_486        6        /* Intel 80486 */
124: #define EM_860        7        /* Intel i860 */
125: #define EM_NUM        8
126:
127:
128: #define EV_NONE        0        /* e_version, EI_VERSION */
129: #define EV_CURRENT    1
130: #define EV_NUM        2
131:
132:
133: /*    Program header
134:  */
135:
136: typedef struct {
137:     Elf32_Word    p_type;        /* entry type */
138: #if defined(_KERNEL) || defined(_KMEMUSER)
139:     Elf32_Off    p_elfoffset;    /* p_offset conflicts with kernel macro */
140: #else
141:     Elf32_Off    p_offset;    /* file offset */
142: #endif
143:     Elf32_Addr    p_vaddr;    /* virtual address */
144:     Elf32_Addr    p_paddr;    /* physical address */
145:     Elf32_Word    p_filesz;    /* file size */
146:     Elf32_Word    p_memsz;    /* memory size */
147:     Elf32_Word    p_flags;    /* entry flags */
148:     Elf32_Word    p_align;    /* memory/file alignment */
149: } Elf32_Phdr;
150:
151: #define PT_NULL        0        /* p_type */
152: #define PT_LOAD        1
153: #define PT_DYNAMIC    2
154: #define PT_INTERP    3
```

```
 79:
 80: #define EM_SPARCV9     43    /* SPARC v9 64-bit */
 81:
 82: #define EM_IA_64    50    /* HP/Intel IA-64 */
 83:
 84: #define EM_X86_64    62    /* AMD x86-64 */
 85:
 86: #define EM_S390        22    /* IBM S/390 */
 87:
 88: #define EM_CRIS         76      /* Axis Communications 32-bit embedded processor */
 89:
 90: #define EM_V850        87    /* NEC v850 */
 91:
 92: #define EM_M32R        88    /* Renesas M32R */
 93:
 94: #define EM_H8_300       46      /* Renesas H8/300,300H,H8S */
 95:
 96: /*
 97:  * This is an interim value that we will use until the committee comes
 98:  * up with a final number.
 99:  */
100: #define EM_ALPHA    0x9026
101:
102: /* Bogus old v850 magic number, used by old tools.  */
103: #define EM_CYGNUS_V850    0x9080
104:
105: /* Bogus old m32r magic number, used by old tools.  */
106: #define EM_CYGNUS_M32R    0x9041
107:
108: /*
109:  * This is the old interim value for S/390 architecture
110:  */
111: #define EM_S390_OLD     0xA390
112:
113: #define EM_FRV        0x5441        /* Fujitsu FR-V */
114:
115: /* This is the info that is needed to parse the dynamic section of the file */
116: #define DT_NULL        0
117: #define DT_NEEDED    1
118: #define DT_PLTRELSZ    2
119: #define DT_PLTGOT    3
120: #define DT_HASH        4
121: #define DT_STRTAB    5
122: #define DT_SYMTAB    6
123: #define DT_RELA        7
124: #define DT_RELASZ    8
125: #define DT_RELAENT    9
126: #define DT_STRSZ    10
127: #define DT_SYMENT    11
128: #define DT_INIT        12
129: #define DT_FINI        13
130: #define DT_SONAME    14
131: #define DT_RPATH     15
132: #define DT_SYMBOLIC    16
133: #define DT_REL            17
134: #define DT_RELSZ    18
135: #define DT_RELENT    19
136: #define DT_PLTREL    20
137: #define DT_DEBUG    21
138: #define DT_TEXTREL    22
139: #define DT_JMPREL    23
140: #define DT_LOPROC    0x70000000
141: #define DT_HIPROC    0x7fffffff
142:
143: /* This info is needed when parsing the symbol table */
144: #define STB_LOCAL  0
145: #define STB_GLOBAL 1
146: #define STB_WEAK   2
147:
148: #define STT_NOTYPE  0
149: #define STT_OBJECT  1
150: #define STT_FUNC    2
151: #define STT_SECTION 3
152: #define STT_FILE    4
153:
154: #define ELF_ST_BIND(x)        ((x) >> 4)
155: #define ELF_ST_TYPE(x)        (((unsigned int) x) & 0xf)
156: #define ELF32_ST_BIND(x)    ELF_ST_BIND(x)
```

4:290 5:382 6:398 5:372 4:247 4:255

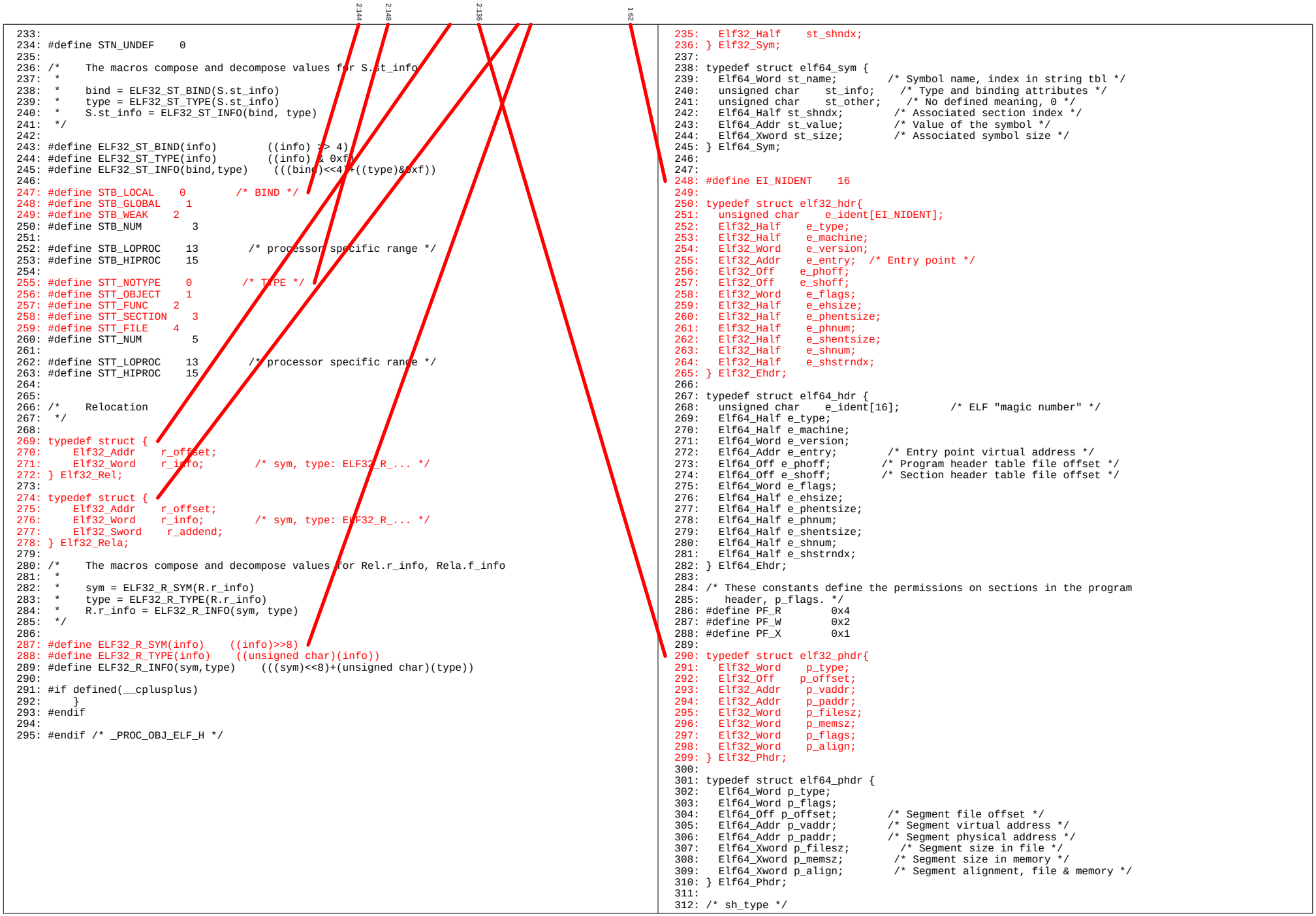
(For readability, not all matches are indicated.)



```
155: #define PT_NOTE         4
156: #define PT_SHLIB    5
157: #define PT_PHDR         6
158: #define PT_NUM         7
159:
160: #define PT_LOPROC    0x70000000    /* processor specific range */
161: #define PT_HIPROC    0x7fffffff
162:
163: #define PF_R        0x4        /* p_flags */
164: #define PF_W        0x2
165: #define PF_X        0x1
166:
167: #define PF_MASKPROC    0xf0000000    /* processor specific values */
168:
169:
170: /*    Section header
171:  */
172:
173: typedef struct { {
174:     Elf32_Word    sh_name;    /* section name */
175:     Elf32_Word    sh_type;    /* SHT_... */
176:     Elf32_Word    sh_flags;    /* SHF_... */
177:     Elf32_Addr    sh_addr;    /* virtual address */
178:     Elf32_Off    sh_offset;    /* file offset */
179:     Elf32_Word    sh_size;    /* section size */
180:     Elf32_Word    sh_link;    /* misc info */
181:     Elf32_Word    sh_info;    /* misc info */
182:     Elf32_Word    sh_addralign;    /* memory alignment */
183:     Elf32_Word    sh_entsize;    /* entry size if table */
184: } Elf32_Shdr;
185:
186: #define SHT_NULL    0        /* sh_type */
187: #define SHT_PROGBITS    1
188: #define SHT_SYMTAB    2
189: #define SHT_STRTAB    3
190: #define SHT_RELA    4
191: #define SHT_HASH    5
192: #define SHT_DYNAMIC    6
193: #define SHT_NOTE    7
194: #define SHT_NOBITS    8
195: #define SHT_REL        9
196: #define SHT_SHLIB    10
197: #define SHT_DYNSYM    11
198: #define SHT_NUM        12
199: #define SHT_MOD        13
200: #define SHT_LOUSER    0x80000000
201: #define SHT_HIUSER    0xffffffff
202:
203: #define    SHT_LOPROC    0x70000000    /* processor specific range */
204: #define    SHT_HIPROC    0x7fffffff
205:
206: #define SHF_WRITE    0x1        /* sh_flags */
207: #define SHF_ALLOC    0x2
208: #define SHF_EXECINSTR    0x4
209:
210: #define SHF_MASKPROC    0xf0000000    /* processor specific values */
211:
212: #define SHN_UNDEF    0        /* special section numbers */
213: #define SHN_LORESERVE    0xff00
214: #define SHN_ABS        0xfff1
215: #define SHN_COMMON    0xfff2
216: #define SHN_HIRESERVE    0xffff
217:
218: #define SHN_LOPROC    0xff00        /* processor specific range */
219: #define SHN_HIPROC    0xff1f
220:
221:
222: /*    Symbol table
223:  */
224:
225: typedef struct {
226:     Elf32_Word    st_name;
227:     Elf32_Addr    st_value;
228:     Elf32_Word    st_size;
229:     unsigned char    st_info;    /* bind, type: ELF_32_ST_... */
230:     unsigned char    st_other;
231:     Elf32_Half    st_shndx;    /* SHN_... */
232: } Elf32_Sym;
```

```
157: #define ELF32_ST_TYPE(x)    ELF_ST_TYPE(x)
158: #define ELF64_ST_BIND(x)    ELF_ST_BIND(x)
159: #define ELF64_ST_TYPE(x)    ELF_ST_TYPE(x)
160:
161: /* Symbolic values for the entries in the auxiliary table
162:    put on the initial stack */
163: #define AT_NULL    0    /* end of vector */
164: #define AT_IGNORE 1    /* entry should be ignored */
165: #define AT_EXECFD 2    /* file descriptor of program */
166: #define AT_PHDR   3    /* program headers for program */
167: #define AT_PHENT  4    /* size of program header entry */
168: #define AT_PHNUM  5    /* number of program headers */
169: #define AT_PAGESZ 6    /* system page size */
170: #define AT_BASE   7    /* base address of interpreter */
171: #define AT_FLAGS  8    /* flags */
172: #define AT_ENTRY  9    /* entry point of program */
173: #define AT_NOTELF 10    /* program is not ELF */
174: #define AT_UID    11    /* real uid */
175: #define AT_EUID   12    /* effective uid */
176: #define AT_GID    13    /* real gid */
177: #define AT_EGID   14    /* effective gid */
178: #define AT_PLATFORM 15  /* string identifying CPU for optimizations */
179: #define AT_HWCAP  16    /* arch dependent hints at CPU capabilities */
180: #define AT_CLKTCK 17    /* frequency at which times() increments */
181:
182: #define AT_SECURE 23   /* secure mode boolean */
183:
184: typedef struct dynamic{
185:   Elf32_Sword d_tag;
186:   union{
187:     Elf32_Sword    d_val;
188:     Elf32_Addr    d_ptr;
189:   } d_un;
190: } Elf32_Dyn;
191:
192: typedef struct {
193:   Elf64_Sxword d_tag;        /* entry tag value */
194:   union {
195:     Elf64_Xword d_val;
196:     Elf64_Addr d_ptr;
197:   } d_un;
198: } Elf64_Dyn;
199:
200: /* The following are used with relocations */
201: #define ELF32_R_SYM(x) ((x) >> 8)
202: #define ELF32_R_TYPE(x) ((x) & 0xff)
203:
204: #define ELF64_R_SYM(i)            ((i) >> 32)
205: #define ELF64_R_TYPE(i)            ((i) & 0xffffffff)
206:
207: typedef struct elf32_rel {
208:   Elf32_Addr    r_offset;
209:   Elf32_Word    r_info;
210: } Elf32_Rel;
211:
212: typedef struct elf64_rel {
213:   Elf64_Addr r_offset;    /* Location at which to apply the action */
214:   Elf64_Xword r_info;    /* index and type of relocation */
215: } Elf64_Rel;
216:
217: typedef struct elf32_rela{
218:   Elf32_Addr    r_offset;
219:   Elf32_Word    r_info;
220:   Elf32_Sword    r_addend;
221: } Elf32_Rela;
222:
223: typedef struct elf64_rela {
224:   Elf64_Addr r_offset;    /* Location at which to apply the action */
225:   Elf64_Xword r_info;    /* index and type of relocation */
226:   Elf64_Sxword r_addend;    /* Constant addend used to compute value */
227: } Elf64_Rela;
228:
229: typedef struct elf32_sym{
230:   Elf32_Word    st_name;
231:   Elf32_Addr    st_value;
232:   Elf32_Word    st_size;
233:   unsigned char    st_info;
234:   unsigned char    st_other;
```


(For readability, not all matches are indicated.)



```
233:
234: #define STN_UNDEF    0
235:
236: /*    The macros compose and decompose values for S.st_info
237:  *
238:  *    bind = ELF32_ST_BIND(S.st_info)
239:  *    type = ELF32_ST_TYPE(S.st_info)
240:  *    S.st_info = ELF32_ST_INFO(bind, type)
241:  */
242:
243: #define ELF32_ST_BIND(info)        ((info) >> 4)
244: #define ELF32_ST_TYPE(info)        ((info) & 0xf)
245: #define ELF32_ST_INFO(bind,type)    (((bind)<<4)+((type)&0xf))
246:
247: #define STB_LOCAL    0        /* BIND */
248: #define STB_GLOBAL    1
249: #define STB_WEAK    2
250: #define STB_NUM        3
251:
252: #define STB_LOPROC    13        /* processor specific range */
253: #define STB_HIPROC    15
254:
255: #define STT_NOTYPE    0        /* TYPE */
256: #define STT_OBJECT    1
257: #define STT_FUNC    2
258: #define STT_SECTION    3
259: #define STT_FILE    4
260: #define STT_NUM        5
261:
262: #define STT_LOPROC    13        /* processor specific range */
263: #define STT_HIPROC    15
264:
265:
266: /*    Relocation
267:  */
268:
269: typedef struct {
270:     Elf32_Addr    r_offset;
271:     Elf32_Word    r_info;        /* sym, type: ELF32_R_... */
272: } Elf32_Rel;
273:
274: typedef struct {
275:     Elf32_Addr    r_offset;
276:     Elf32_Word    r_info;        /* sym, type: ELF32_R_... */
277:     Elf32_Sword    r_addend;
278: } Elf32_Rela;
279:
280: /*    The macros compose and decompose values for Rel.r_info, Rela.f_info
281:  *
282:  *    sym = ELF32_R_SYM(R.r_info)
283:  *    type = ELF32_R_TYPE(R.r_info)
284:  *    R.r_info = ELF32_R_INFO(sym, type)
285:  */
286:
287: #define ELF32_R_SYM(info)    ((info)>>8)
288: #define ELF32_R_TYPE(info)    ((unsigned char)(info))
289: #define ELF32_R_INFO(sym,type)    (((sym)<<8)+(unsigned char)(type))
290:
291: #if defined(__cplusplus)
292:     }
293: #endif
294:
295: #endif /* _PROC_OBJ_ELF_H */
```

```
235:   Elf32_Half    st_shndx;
236: } Elf32_Sym;
237:
238: typedef struct elf64_sym {
239:   Elf64_Word st_name;        /* Symbol name, index in string tbl */
240:   unsigned char    st_info;    /* Type and binding attributes */
241:   unsigned char    st_other;    /* No defined meaning, 0 */
242:   Elf64_Half st_shndx;        /* Associated section index */
243:   Elf64_Addr st_value;        /* Value of the symbol */
244:   Elf64_Xword st_size;        /* Associated symbol size */
245: } Elf64_Sym;
246:
247:
248: #define EI_NIDENT    16
249:
250: typedef struct elf32_hdr{
251:   unsigned char    e_ident[EI_NIDENT];
252:   Elf32_Half    e_type;
253:   Elf32_Half    e_machine;
254:   Elf32_Word    e_version;
255:   Elf32_Addr    e_entry;  /* Entry point */
256:   Elf32_Off    e_phoff;
257:   Elf32_Off    e_shoff;
258:   Elf32_Word    e_flags;
259:   Elf32_Half    e_ehsize;
260:   Elf32_Half    e_phentsize;
261:   Elf32_Half    e_phnum;
262:   Elf32_Half    e_shentsize;
263:   Elf32_Half    e_shnum;
264:   Elf32_Half    e_shstrndx;
265: } Elf32_Ehdr;
266:
267: typedef struct elf64_hdr {
268:   unsigned char    e_ident[16];        /* ELF "magic number" */
269:   Elf64_Half e_type;
270:   Elf64_Half e_machine;
271:   Elf64_Word e_version;
272:   Elf64_Addr e_entry;        /* Entry point virtual address */
273:   Elf64_Off e_phoff;        /* Program header table file offset */
274:   Elf64_Off e_shoff;        /* Section header table file offset */
275:   Elf64_Word e_flags;
276:   Elf64_Half e_ehsize;
277:   Elf64_Half e_phentsize;
278:   Elf64_Half e_phnum;
279:   Elf64_Half e_shentsize;
280:   Elf64_Half e_shnum;
281:   Elf64_Half e_shstrndx;
282: } Elf64_Ehdr;
283:
284: /* These constants define the permissions on sections in the program
285:    header, p_flags. */
286: #define PF_R        0x4
287: #define PF_W        0x2
288: #define PF_X        0x1
289:
290: typedef struct elf32_phdr{
291:   Elf32_Word    p_type;
292:   Elf32_Off    p_offset;
293:   Elf32_Addr    p_vaddr;
294:   Elf32_Addr    p_paddr;
295:   Elf32_Word    p_filesz;
296:   Elf32_Word    p_memsz;
297:   Elf32_Word    p_flags;
298:   Elf32_Word    p_align;
299: } Elf32_Phdr;
300:
301: typedef struct elf64_phdr {
302:   Elf64_Word p_type;
303:   Elf64_Word p_flags;
304:   Elf64_Off p_offset;        /* Segment file offset */
305:   Elf64_Addr p_vaddr;        /* Segment virtual address */
306:   Elf64_Addr p_paddr;        /* Segment physical address */
307:   Elf64_Xword p_filesz;        /* Segment size in file */
308:   Elf64_Xword p_memsz;        /* Segment size in memory */
309:   Elf64_Xword p_align;        /* Segment alignment, file & memory */
310: } Elf64_Phdr;
311:
312: /* sh_type */
```


(For readability, not all matches are indicated.)


```
313: #define SHT_NULL    0
314: #define SHT_PROGBITS    1
315: #define SHT_SYMTAB    2
316: #define SHT_STRTAB    3
317: #define SHT_RELA    4
318: #define SHT_HASH    5
319: #define SHT_DYNAMIC    6
320: #define SHT_NOTE    7
321: #define SHT_NOBITS    8
322: #define SHT_REL        9
323: #define SHT_SHLIB    10
324: #define SHT_DYNSYM    11
325: #define SHT_NUM        12
326: #define SHT_LOPROC    0x70000000
327: #define SHT_HIPROC    0x7fffffff
328: #define SHT_LOUSER    0x80000000
329: #define SHT_HIUSER    0xffffffff
330:
331: /* sh_flags */
332: #define SHF_WRITE    0x1
333: #define SHF_ALLOC    0x2
334: #define SHF_EXECINSTR    0x4
335: #define SHF_MASKPROC    0xf0000000
336:
337: /* special section indexes */
338: #define SHN_UNDEF    0
339: #define SHN_LORESERVE    0xff00
340: #define SHN_LOPROC    0xff00
341: #define SHN_HIPROC    0xff1f
342: #define SHN_ABS        0xfff1
343: #define SHN_COMMON    0xfff2
344: #define SHN_HIRESERVE    0xffff
345:
346: typedef struct {
347:   Elf32_Word    sh_name;
348:   Elf32_Word    sh_type;
349:   Elf32_Word    sh_flags;
350:   Elf32_Addr    sh_addr;
351:   Elf32_Off    sh_offset;
352:   Elf32_Word    sh_size;
353:   Elf32_Word    sh_link;
354:   Elf32_Word    sh_info;
355:   Elf32_Word    sh_addralign;
356:   Elf32_Word    sh_entsize;
357: } Elf32_Shdr;
358:
359: typedef struct elf64_shdr {
360:   Elf64_Word sh_name;        /* Section name, index in string tbl */
361:   Elf64_Word sh_type;        /* Type of section */
362:   Elf64_Xword sh_flags;       /* Miscellaneous section attributes */
363:   Elf64_Addr sh_addr;        /* Section virtual addr at execution */
364:   Elf64_Off sh_offset;        /* Section file offset */
365:   Elf64_Xword sh_size;        /* Size of section in bytes */
366:   Elf64_Word sh_link;        /* Index of another section */
367:   Elf64_Word sh_info;        /* Additional section information */
368:   Elf64_Xword sh_addralign;    /* Section alignment */
369:   Elf64_Xword sh_entsize;    /* Entry size if section holds table */
370: } Elf64_Shdr;
371:
372: #define    EI_MAG0        0        /* e_ident[] indexes */
373: #define    EI_MAG1        1
374: #define    EI_MAG2        2
375: #define    EI_MAG3        3
376: #define    EI_CLASS    4
377: #define    EI_DATA        5
378: #define    EI_VERSION    6
379: #define    EI_OSABI    7
380: #define    EI_PAD        8
381:
382: #define    ELFMAG0        0x7f        /* EI_MAG */
383: #define    ELFMAG1        'E'
384: #define    ELFMAG2        'L'
385: #define    ELFMAG3        'F'
386: #define    ELFMAG        "\177ELF"
387: #define    SELFMAG        4
388:
389: #define    ELFCLASSNONE    0        /* EI_CLASS */
390: #define    ELFCLASS32    1
```


(For readability, not all matches are indicated.)

2:128

```
391: #define    ELFCLASS64    2
392: #define    ELFCLASSNUM    3
393:
394: #define ELFDATANONE    0        /* e_ident[EI_DATA] */
395: #define ELFDATA2LSB    1
396: #define ELFDATA2MSB    2
397:
398: #define EV_NONE        0        /* e_version, EI_VERSION */
399: #define EV_CURRENT    1
400: #define EV_NUM        2
401:
402: #define ELFOSABI_NONE    0
403: #define ELFOSABI_LINUX    3
404:
405: #ifndef ELF_OSABI
406: #define ELF_OSABI ELFOSABI_NONE
407: #endif
408:
409: /* Notes used in ET_CORE */
410: #define NT_PRSTATUS    1
411: #define NT_PRFPREG    2
412: #define NT_PRPSINFO    3
413: #define NT_TASKSTRUCT    4
414: #define NT_AUXV        6
415: #define NT_PRXFPREG     0x46e62b7f      /* copied from gdb5.1/include/elf/common.h */
416:
417:
418: /* Note header in a PT_NOTE section */
419: typedef struct elf32_note {
420:   Elf32_Word    n_namesz;    /* Name size */
421:   Elf32_Word    n_descsz;    /* Content size */
422:   Elf32_Word    n_type;        /* Content type */
423: } Elf32_Nhdr;
424:
425: /* Note header in a PT_NOTE section */
426: typedef struct elf64_note {
427:   Elf64_Word n_namesz;    /* Name size */
428:   Elf64_Word n_descsz;    /* Content size */
429:   Elf64_Word n_type;    /* Content type */
430: } Elf64_Nhdr;
431:
432: #if ELF_CLASS == ELFCLASS32
433:
434: extern Elf32_Dyn _DYNAMIC [];
435: #define elfhdr        elf32_hdr
436: #define elf_phdr    elf32_phdr
437: #define elf_note    elf32_note
438:
439: #else
440:
441: extern Elf64_Dyn _DYNAMIC [];
442: #define elfhdr        elf64_hdr
443: #define elf_phdr    elf64_phdr
444: #define elf_note    elf64_note
445:
446: #endif
447:
448:
449: #endif /* _LINUX_ELF_H */
```


(For readability, not all matches are indicated.)