Map for Finding Sections and Names

```
Elf64_Ehdr
  e_shoff
  e_shstrndx

AT_SEC(ehdr, shdrs + i)
  name is
  strs + shdrs[i].sh_name

strs = AT_SEC(ehdr, shdrs + ehdr->e_shstrndx)
  .rodata .text .data...

shdrs = (void*)ehdr + ehdr->e_shoff
  shdrs[i].sh_addr
  shdrs[i].sh_offset
  shdrs[i].sh_name
  Elf64_Shdr[]
```

0

ehdr
Map for Finding All Function Names and Code

```
0

Elf64_Ehdr

e_shoff

Elf64_Shdr[

shdrs

(shdrs[i].sh_addr - shdrs[j].sh_addr)

strs

strs + sym[m].st_name

Elf64_Sym[]

syms

syms[m].st_name = sym[m].st_shndx

sym[m].st_value = Elf64_Sym[]

my_array

sections

.code = AT_SEC(ehdr, shdrs + j) + (sym[m].st_value - shdrs[j].sh_addr)

shdrs = (void*)ehdr + ehdr->e_shoff

.shdrs[i].sh_addr

.shdrs[i].sh_offset

Elf64_Shdr[]
```

---

**Map for Finding All Function Names and Code**

The diagram represents the process of finding all function names and code in an ELF (Executable and Linkable Format) file. It involves navigating through various sections of the ELF file, such as the ELF header (Elf64_Ehdr), symbol table (Elf64_Sym[]), sections (.text, .rodata, .data), and symbol offsets. The code snippet provided illustrates the logic for calculating the code address based on the symbol table and section headers.

- **0** represents the starting point, the ELF header (Elf64_Ehdr).
- **ehdr** is the ELF header pointer.
- **syms** is the symbol table, where `sym[m].st_name = sym[m].st_shndx` and `sym[m].st_value = Elf64_Sym[]` are used.
- **strs** is the string table, used for accessing symbol names.
- **shdrs** is the section header table, where `shdrs = (void*)ehdr + ehdr->e_shoff` and `shdrs[i].sh_addr` and `shdrs[i].sh_offset` are accessed.

The code snippet shows how to calculate the code address using the symbol and section header information:

```
code = AT_SEC(ehdr, shdrs + j) + (sym[m].st_value - shdrs[j].sh_addr)
```
Map for Finding Accessed-Variable Names

```
m = ELF64_R_SYM(relas[k].r_info)
relas[k].r_offset
Elf64_Rela[]
strs + syms[m].st_name
my_array[i].
section[i].sh_addr
section[i].sh_offset
Elf64_Sym[]
syms
Elf64_Sym
ELF header
e_shoff
.dynsym
.text
.relas
.strs
.my_array[i]
[dynstr]
[dynsym]
[p]
find k by matching
p to relas[k].r_offset
```

```c
m = ELF64_R_SYM(relas[k].r_info)
relas[k].r_offset
Elf64_Rela[]
strs + syms[m].st_name
my_array[i].
section[i].sh_addr
section[i].sh_offset
Elf64_Sym[]
syms
Elf64_Sym
ELF header
e_shoff
.dynsym
.text
.relas
.strs
.my_array[i]
[dynstr]
[dynsym]
[p]
find k by matching
p to relas[k].r_offset
```
Map for Finding Called-Function Names

```
m = ELF64_R_SYM(relas[k].r_info)
relas[k].r_offset
Elf64_Rela[]
jmp q
[i].sh_addr
[i].sh_offset
```

find k by matching q to \texttt{relas[k].r_offset}
use m to get name from \texttt{.dynsym} and \texttt{.dynstr}
in the same way as for variables