Milena Tshagharyan
OS - Section B - HW4 - Compilation Units
Repository link:
https://github.com/mataghinim/os_homework/tree/45dc39331eb03c2c9dd2fd548f2775910ac637dc/hw4-compilation_units

Copied what I did on the terminal, step-by-step, after creating the files. Report at the end.

—---START—--

```
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ ls
main.c  math_utils.c  math_utils.h
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ gcc -c main.c
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ gcc -c math_utils.c
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ gcc main.o math_utils.o -o square_prog
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ nm main.o
0000000000000000 T main
                 U printf
                 U square
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ nm math_utils.o
0000000000000000 T square
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ nm square_prog
0000000000003dc8 d _DYNAMIC
0000000000003fb8 d _GLOBAL_OFFSET_TABLE_
0000000000002000 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000002120 r __FRAME_END__
000000000000201c r __GNU_EH_FRAME_HDR
0000000000004010 D __TMC_END__
000000000000038c r __abi_tag
0000000000004010 B __bss_start
                 w __cxa_finalize@GLIBC_2.2.5
0000000000004000 D __data_start
0000000000001100 t __do_global_dtors_aux
0000000000003dc0 d __do_global_dtors_aux_fini_array_entry
0000000000004008 D __dso_handle
0000000000003db8 d __frame_dummy_init_array_entry
                 w __gmon_start__
                 U __libc_start_main@GLIBC_2.34
0000000000004010 D _edata
0000000000004018 B _end
00000000000011a0 T _fini
0000000000001000 T _init
0000000000001060 T _start
0000000000004010 b completed.0
0000000000004000 W data_start
0000000000001090 t deregister_tm_clones
0000000000001140 t frame_dummy
0000000000001149 T main
                 U printf@GLIBC_2.2.5
00000000000010c0 t register_tm_clones
000000000000118c T square
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ objdump -d main.o
main.o:     file format elf64-x86-64
Disassembly of section .text:
0000000000000000 <main>:
   0:        f3 0f 1e fa             endbr64
   4:        55                      push   %rbp
   5:	48 89 e5             	mov    %rsp,%rbp
   8:	48 83 ec 10          	sub    $0x10,%rsp
   c:	c7 45 f8 05 00 00 00 	movl   $0x5,-0x8(%rbp)
  13:	8b 45 f8             	mov    -0x8(%rbp),%eax
  16:	89 c7                	mov    %eax,%edi
  18:	e8 00 00 00 00       	call   1d <main+0x1d>
  1d:	89 45 fc             	mov    %eax,-0x4(%rbp)
  20:	8b 55 fc             	mov    -0x4(%rbp),%edx
  23:	8b 45 f8             	mov    -0x8(%rbp),%eax
  26:	89 c6                	mov    %eax,%esi
  28:	48 8d 05 00 00 00 00 	lea    0x0(%rip),%rax        # 2f <main+0x2f>
  2f:	48 89 c7             	mov    %rax,%rdi
  32:	b8 00 00 00 00       	mov    $0x0,%eax
  37:	e8 00 00 00 00       	call   3c <main+0x3c>
  3c:	b8 00 00 00 00       	mov    $0x0,%eax
  41:	c9                   	leave
  42:	c3                   	ret
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ objdump -d math_utils.o
math_utils.o:     file format elf64-x86-64
Disassembly of section .text:
0000000000000000 <square>:
   0:	f3 0f 1e fa          	endbr64
   4:	55                   	push   %rbp
   5:	48 89 e5             	mov    %rsp,%rbp
   8:	89 7d fc             	mov    %edi,-0x4(%rbp)
   b:	8b 45 fc             	mov    -0x4(%rbp),%eax
   e:	0f af c0             	imul   %eax,%eax
  11:	5d                   	pop    %rbp
  12:	c3                   	ret
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ objdump -d square_prog
square_prog:     file format elf64-x86-64
Disassembly of section .init:
0000000000001000 <_init>:
    1000:	f3 0f 1e fa          	endbr64
    1004:	48 83 ec 08          	sub    $0x8,%rsp
    1008:	48 8b 05 d9 2f 00 00 	mov    0x2fd9(%rip),%rax        # 3fe8 <__gmon_start__@Base>
    100f:	48 85 c0             	test   %rax,%rax
    1012:	74 02                	je     1016 <_init+0x16>
    1014:	ff d0                	call   *%rax
    1016:	48 83 c4 08          	add    $0x8,%rsp
    101a:	c3                   	ret
Disassembly of section .plt:
0000000000001020 <.plt>:
    1020:	ff 35 9a 2f 00 00    	push   0x2f9a(%rip)        # 3fc0 <_GLOBAL_OFFSET_TABLE_+0x8>
    1026:	ff 25 9c 2f 00 00    	jmp    *0x2f9c(%rip)        # 3fc8 <_GLOBAL_OFFSET_TABLE_+0x10>
    102c:	0f 1f 40 00          	nopl   0x0(%rax)
    1030:	f3 0f 1e fa          	endbr64
    1034:	68 00 00 00 00       	push   $0x0
    1039:	e9 e2 ff ff ff       	jmp    1020 <_init+0x20>
    103e:	66 90                	xchg   %ax,%ax
Disassembly of section .plt.got:
0000000000001040 <__cxa_finalize@plt>:
    1040:	f3 0f 1e fa          	endbr64
    1044:	ff 25 ae 2f 00 00    	jmp    *0x2fae(%rip)        # 3ff8 <__cxa_finalize@GLIBC_2.2.5>
    104a:	66 0f 1f 44 00 00    	nopw   0x0(%rax,%rax,1)
Disassembly of section .plt.sec:
0000000000001050 <printf@plt>:
    1050:	f3 0f 1e fa          	endbr64
    1054:	ff 25 76 2f 00 00    	jmp    *0x2f76(%rip)        # 3fd0 <printf@GLIBC_2.2.5>
    105a:	66 0f 1f 44 00 00    	nopw   0x0(%rax,%rax,1)
Disassembly of section .text:
0000000000001060 <_start>:
    1060:  f3 0f 1e fa             endbr64
    1064:  31 ed                   xor    %ebp,%ebp
    1066:  49 89 d1                mov    %rdx,%r9
    1069:  5e                      pop    %rsi
    106a:  48 89 e2                mov    %rsp,%rdx
    106d:  48 83 e4 f0             and    $0xfffffffffffffff0,%rsp
    1071:  50                      push   %rax
    1072:  54                      push   %rsp
    1073:  45 31 c0                xor    %r8d,%r8d
    1076:  31 c9                   xor    %ecx,%ecx
    1078:  48 8d 3d ca 00 00 00             lea    0xca(%rip),%rdi        # 1149 <main>
    107f:  ff 15 53 2f 00 00       call   *0x2f53(%rip)        # 3fd8 <__libc_start_main@GLIBC_2.34>
    1085:  f4                      hlt
    1086:  66 2e 0f 1f 84 00 00             cs nopw 0x0(%rax,%rax,1)
    108d:  00 00 00
0000000000001090 <deregister_tm_clones>:
    1090:  48 8d 3d 79 2f 00 00             lea    0x2f79(%rip),%rdi        # 4010 <__TMC_END__>
    1097:  48 8d 05 72 2f 00 00             lea    0x2f72(%rip),%rax        # 4010 <__TMC_END__>
    109e:  48 39 f8                cmp    %rdi,%rax
    10a1:  74 15                   je     10b8 <deregister_tm_clones+0x28>
    10a3:  48 8b 05 36 2f 00 00             mov    0x2f36(%rip),%rax        # 3fe0 <_ITM_deregisterTMCloneTable@Base>
    10aa:  48 85 c0                test   %rax,%rax
    10ad:  74 09                   je     10b8 <deregister_tm_clones+0x28>
    10af:  ff e0                   jmp    *%rax
    10b1:  0f 1f 80 00 00 00 00             nopl   0x0(%rax)
    10b8:  c3                      ret
    10b9:  0f 1f 80 00 00 00 00             nopl   0x0(%rax)
00000000000010c0 <register_tm_clones>:
    10c0:  48 8d 3d 49 2f 00 00             lea    0x2f49(%rip),%rdi        # 4010 <__TMC_END__>
    10c7:  48 8d 35 42 2f 00 00             lea    0x2f42(%rip),%rsi        # 4010 <__TMC_END__>
    10ce:  48 29 fe                sub    %rdi,%rsi
    10d1:  48 89 f0                mov    %rsi,%rax
    10d4:  48 c1 ee 3f             shr    $0x3f,%rsi
    10d8:  48 c1 f8 03             sar    $0x3,%rax
    10dc:  48 01 c6                add    %rax,%rsi
    10df:  48 d1 fe                sar    $1,%rsi
    10e2:  74 14                   je     10f8 <register_tm_clones+0x38>
    10e4:  48 8b 05 05 2f 00 00             mov    0x2f05(%rip),%rax        # 3ff0 <_ITM_registerTMCloneTable@Base>
    10eb:  48 85 c0                test   %rax,%rax
    10ee:  74 08                   je     10f8 <register_tm_clones+0x38>
    10f0:  ff e0                   jmp    *%rax
    10f2:  66 0f 1f 44 00 00                nopw   0x0(%rax,%rax,1)
    10f8:  c3                      ret
    10f9:  0f 1f 80 00 00 00 00             nopl   0x0(%rax)
0000000000001100 <__do_global_dtors_aux>:
    1100:  f3 0f 1e fa             endbr64
    1104:  80 3d 05 2f 00 00 00             cmpb   $0x0,0x2f05(%rip)        # 4010 <__TMC_END__>
    110b:  75 2b                   jne    1138 <__do_global_dtors_aux+0x38>
    110d:  55                      push   %rbp
    110e:  48 83 3d e2 2e 00 00             cmpq   $0x0,0x2ee2(%rip)        # 3ff8 <__cxa_finalize@GLIBC_2.2.5>
    1115:  00
    1116:  48 89 e5                mov    %rsp,%rbp
    1119:  74 0c                   je     1127 <__do_global_dtors_aux+0x27>
    111b:  48 8b 3d e6 2e 00 00             mov    0x2ee6(%rip),%rdi        # 4008 <__dso_handle>
    1122:  e8 19 ff ff ff          call   1040 <__cxa_finalize@plt>
    1127:  e8 64 ff ff ff          call   1090 <deregister_tm_clones>
    112c:  c6 05 dd 2e 00 00 01             movb   $0x1,0x2edd(%rip)        # 4010 <__TMC_END__>
    1133:  5d                      pop    %rbp
    1134:  c3                      ret
    1135:  0f 1f 00                nopl   (%rax)
    1138:  c3                      ret
    1139:  0f 1f 80 00 00 00 00             nopl   0x0(%rax)
```

```
0000000000001140 <frame_dummy>:
    1140:  f3 0f 1e fa             endbr64
    1144:  e9 77 ff ff ff          jmp    10c0 <register_tm_clones>
0000000000001149 <main>:
    1149:  f3 0f 1e fa             endbr64
    114d:  55                      push   %rbp
    114e:  48 89 e5                mov    %rsp,%rbp
    1151:  48 83 ec 10             sub    $0x10,%rsp
    1155:  c7 45 f8 05 00 00 00            movl   $0x5,-0x8(%rbp)
    115c:  8b 45 f8                mov    -0x8(%rbp),%eax
    115f:  89 c7                   mov    %eax,%edi
    1161:  e8 26 00 00 00                  call   118c <square>
    1166:  89 45 fc                mov    %eax,-0x4(%rbp)
    1169:  8b 55 fc                mov    -0x4(%rbp),%edx
    116c:  8b 45 f8                mov    -0x8(%rbp),%eax
    116f:  89 c6                   mov    %eax,%esi
    1171:  48 8d 05 8c 0e 00 00            lea    0xe8c(%rip),%rax        # 2004 <_IO_stdin_used+0x4>
    1178:  48 89 c7                mov    %rax,%rdi
    117b:  b8 00 00 00 00                  mov    $0x0,%eax
    1180:  e8 cb fe ff ff          call   1050 <printf@plt>
    1185:  b8 00 00 00 00                  mov    $0x0,%eax
    118a:  c9                      leave
    118b:  c3                      ret
000000000000118c <square>:
    118c:  f3 0f 1e fa             endbr64
    1190:  55                      push   %rbp
    1191:  48 89 e5                mov    %rsp,%rbp
    1194:  89 7d fc                mov    %edi,-0x4(%rbp)
    1197:  8b 45 fc                mov    -0x4(%rbp),%eax
    119a:  0f af c0                imul   %eax,%eax
    119d:  5d                      pop    %rbp
    119e:  c3                      ret
Disassembly of section .fini:
00000000000011a0 <_fini>:
    11a0:  f3 0f 1e fa             endbr64
    11a4:  48 83 ec 08             sub    $0x8,%rsp
    11a8:  48 83 c4 08             add    $0x8,%rsp
    11ac:  c3                      ret
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ readelf -h main.o
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          704 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           64 (bytes)
  Number of section headers:         14
  Section header string table index: 13
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ readelf -h math_utils.o
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          464 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           64 (bytes)
  Number of section headers:         12
  Section header string table index: 11
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ readelf -h square_prog
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x1060
  Start of program headers:          64 (bytes into file)
  Start of section headers:          14048 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 30
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ readelf -S main.o
There are 14 section headers, starting at offset 0x2c0:
Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .text             PROGBITS         0000000000000000  00000040
       0000000000000043  0000000000000000  AX       0     0     1
  [ 2] .rela.text        RELA             0000000000000000  000001e8
       0000000000000048  0000000000000018   I      11     1     8
  [ 3] .data             PROGBITS         0000000000000000  00000083
       0000000000000000  0000000000000000  WA       0     0     1
  [ 4] .bss              NOBITS           0000000000000000  00000083
       0000000000000000  0000000000000000  WA       0     0     1
  [ 5] .rodata           PROGBITS         0000000000000000  00000083
       0000000000000018  0000000000000000   A       0     0     1
  [ 6] .comment          PROGBITS         0000000000000000  0000009b
       000000000000002c  0000000000000001  MS       0     0     1
  [ 7] .note.GNU-stack   PROGBITS         0000000000000000  000000c7
       0000000000000000  0000000000000000           0     0     1
  [ 8] .note.gnu.pr[...] NOTE             0000000000000000  000000c8
       0000000000000020  0000000000000000   A       0     0     8
  [ 9] .eh_frame         PROGBITS         0000000000000000  000000e8
       0000000000000038  0000000000000000   A       0     0     8
  [10] .rela.eh_frame    RELA             0000000000000000  00000230
       0000000000000018  0000000000000018   I      11     9     8
  [11] .symtab           SYMTAB           0000000000000000  00000120
       00000000000000a8  0000000000000018          12     4     8
  [12] .strtab           STRTAB           0000000000000000  000001c8
       000000000000001b  0000000000000000           0     0     1
  [13] .shstrtab         STRTAB           0000000000000000  00000248
       0000000000000074  0000000000000000           0     0     1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  D (mbind), l (large), p (processor specific)
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ readelf -S math_utils.o
There are 12 section headers, starting at offset 0x1d0:
Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .text             PROGBITS         0000000000000000  00000040
       0000000000000013  0000000000000000  AX       0     0     1
  [ 2] .data             PROGBITS         0000000000000000  00000053
       0000000000000000  0000000000000000  WA       0     0     1
  [ 3] .bss              NOBITS           0000000000000000  00000053
       0000000000000000  0000000000000000  WA       0     0     1
  [ 4] .comment          PROGBITS         0000000000000000  00000053
       000000000000002c  0000000000000001  MS       0     0     1
  [ 5] .note.GNU-stack   PROGBITS         0000000000000000  0000007f
       0000000000000000  0000000000000000           0     0     1
  [ 6] .note.gnu.pr[...] NOTE             0000000000000000  00000080
       0000000000000020  0000000000000000   A       0     0     8
  [ 7] .eh_frame         PROGBITS         0000000000000000  000000a0
       0000000000000038  0000000000000000   A       0     0     8
  [ 8] .rela.eh_frame    RELA             0000000000000000  00000150
       0000000000000018  0000000000000018   I       9     7     8
  [ 9] .symtab           SYMTAB           0000000000000000  000000d8
       0000000000000060  0000000000000018          10     3     8
  [10] .strtab           STRTAB           0000000000000000  00000138
       0000000000000015  0000000000000000           0     0     1
  [11] .shstrtab         STRTAB           0000000000000000  00000168
       0000000000000067  0000000000000000           0     0     1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  D (mbind), l (large), p (processor specific)
milenat@ubuntu-aua-os:~/os_homework/hw4-compilation_units$ readelf -S square_prog
There are 31 section headers, starting at offset 0x36e0:
Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .interp           PROGBITS         0000000000000318  00000318
       000000000000001c  0000000000000000   A       0     0     1
  [ 2] .note.gnu.pr[...] NOTE             0000000000000338  00000338
       0000000000000030  0000000000000000   A       0     0     8
  [ 3] .note.gnu.bu[...] NOTE             0000000000000368  00000368
       0000000000000024  0000000000000000   A       0     0     4
```

```
 [ 4] .note.ABI-tag     NOTE             000000000000038c  0000038c
      0000000000000020  0000000000000000   A       0     0     4
 [ 5] .gnu.hash         GNU_HASH         00000000000003b0  000003b0
      0000000000000024  0000000000000000   A       6     0     8
 [ 6] .dynsym           DYNSYM           00000000000003d8  000003d8
      00000000000000a8  0000000000000018   A       7     1     8
 [ 7] .dynstr           STRTAB           0000000000000480  00000480
      000000000000008f  0000000000000000   A       0     0     1
 [ 8] .gnu.version      VERSYM           0000000000000510  00000510
      000000000000000e  0000000000000002   A       6     0     2
 [ 9] .gnu.version_r    VERNEED          0000000000000520  00000520
      0000000000000030  0000000000000000   A       7     1     8
 [10] .rela.dyn         RELA             0000000000000550  00000550
      00000000000000c0  0000000000000018   A       6     0     8
 [11] .rela.plt         RELA             0000000000000610  00000610
      0000000000000018  0000000000000018  AI       6    24     8
 [12] .init             PROGBITS         0000000000001000  00001000
      000000000000001b  0000000000000000  AX       0     0     4
 [13] .plt              PROGBITS         0000000000001020  00001020
      0000000000000020  0000000000000010  AX       0     0     16
 [14] .plt.got          PROGBITS         0000000000001040  00001040
      0000000000000010  0000000000000010  AX       0     0     16
 [15] .plt.sec          PROGBITS         0000000000001050  00001050
      0000000000000010  0000000000000010  AX       0     0     16
 [16] .text             PROGBITS         0000000000001060  00001060
      000000000000013f  0000000000000000  AX       0     0     16
 [17] .fini             PROGBITS         00000000000011a0  000011a0
      000000000000000d  0000000000000000  AX       0     0     4
 [18] .rodata           PROGBITS         0000000000002000  00002000
      000000000000001c  0000000000000000   A       0     0     4
 [19] .eh_frame_hdr     PROGBITS         000000000000201c  0000201c
      000000000000003c  0000000000000000   A       0     0     4
 [20] .eh_frame         PROGBITS         0000000000002058  00002058
      00000000000000cc  0000000000000000   A       0     0     8
 [21] .init_array       INIT_ARRAY       0000000000003db8  00002db8
      0000000000000008  0000000000000008  WA       0     0     8
 [22] .fini_array       FINI_ARRAY       0000000000003dc0  00002dc0
      0000000000000008  0000000000000008  WA       0     0     8
 [23] .dynamic          DYNAMIC          0000000000003dc8  00002dc8
      00000000000001f0  0000000000000010  WA       7     0     8
 [24] .got              PROGBITS         0000000000003fb8  00002fb8
      0000000000000048  0000000000000008  WA       0     0     8
 [25] .data             PROGBITS         0000000000004000  00003000
      0000000000000010  0000000000000000  WA       0     0     8
 [26] .bss              NOBITS           0000000000004010  00003010
      0000000000000008  0000000000000000  WA       0     0     1
 [27] .comment          PROGBITS         0000000000000000  00003010
      000000000000002b  0000000000000001  MS       0     0     1
 [28] .symtab           SYMTAB           0000000000000000  00003040
      0000000000000390  0000000000000018          29    19     8
 [29] .strtab           STRTAB           0000000000000000  000033d0
      00000000000001f0  0000000000000000           0     0     1
 [30] .shstrtab         STRTAB           0000000000000000  000035c0
      000000000000011a  0000000000000000           0     0     1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  D (mbind), l (large), p (processor specific)
```

—---THE END—--

Compiling C program with multiple files, the compiler first creates object files. These object files, like main.o and math_utils.o, are still unfinished, they contain the code, but they are not ready to run.
The nm command shows us that in the main.o object file, the main function is defined, but it has two undefined parts, which are **printf** and **square**. It knows it needs these functions, but it doesn't know where to find them yet, they are still undefined (I highlighted them with this color in the terminal). On the other hand, the math_utils.o file contains the missing **square** function.

If we look at the assembly code inside these object files with objdump command, we see the problem. The call instructions that should jump to the **square** and **printf** functions have placeholder addresses set to zero (I highlighted them with this color in the terminal). The object file is basically saying, "I need to call this function here, but I don't know its address yet."

The readelf command confirms that these object files are of the REL type, which stands for relocatable (I highlighted them with this color in the terminal), btw square_prog is not, it is DYN (Position-Independent Executable file). This means relocatables have no starting point and cannot be executed by the os. They are just chunks of code and data.

Then running the linker, it takes all the object files and combines them into a single executable file, which we called square_prog. First, it finds all the undefined symbols. It sees that main.o needs square and finds it in math_utils.o. It then sees that both files need printf and finds it in the C library on the system. Next, it assigns real memory addresses to everything. It decides where in the program's memory the main function will be and where the square function will be for example. It then goes back and replaces all those placeholder zeros in the call instructions with the correct addresses.