Session 3

From ELF to PID

Security Summer School
30th of June 2014

ACS/Ixia/Hexcellents

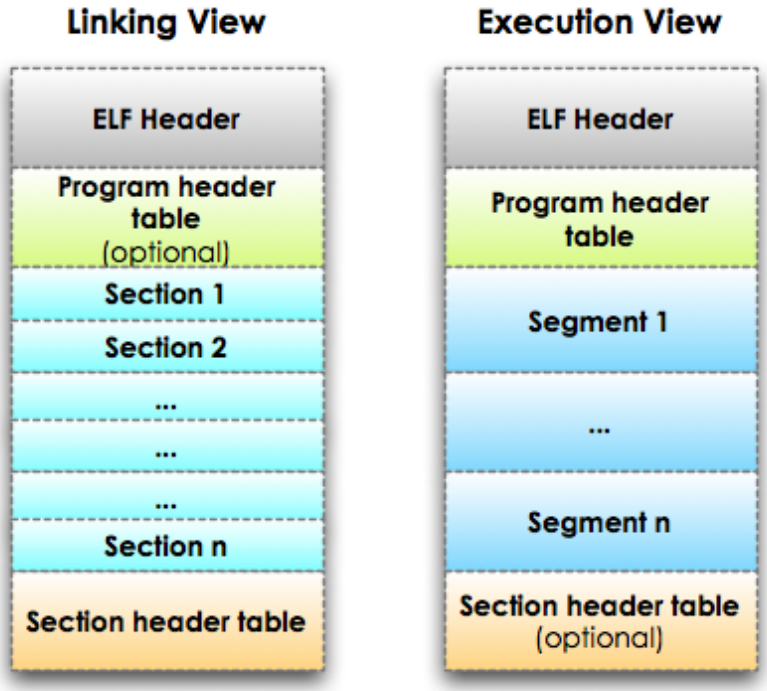
Outline

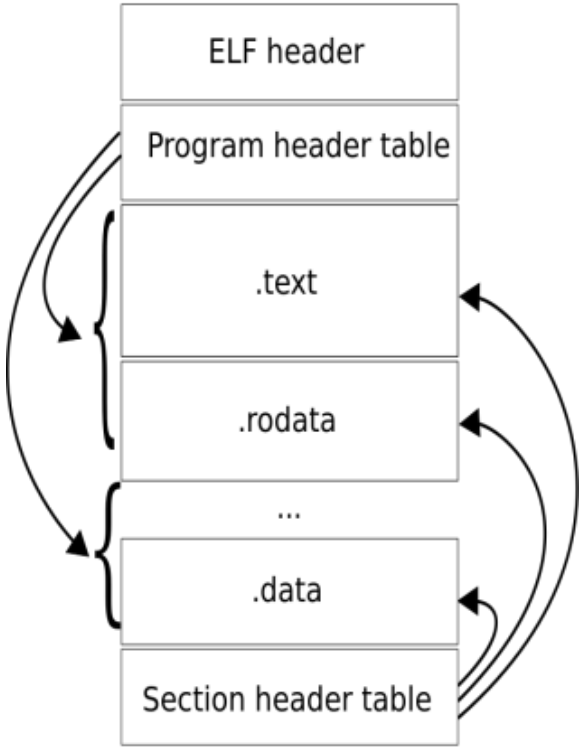
- Elf Basics
- Linking
- Elf Types
- Elf Structure
- Relocations
- Memory Mapping
- Memory Layout
- Detailed Layout
- ASLR

ELF Basics - History

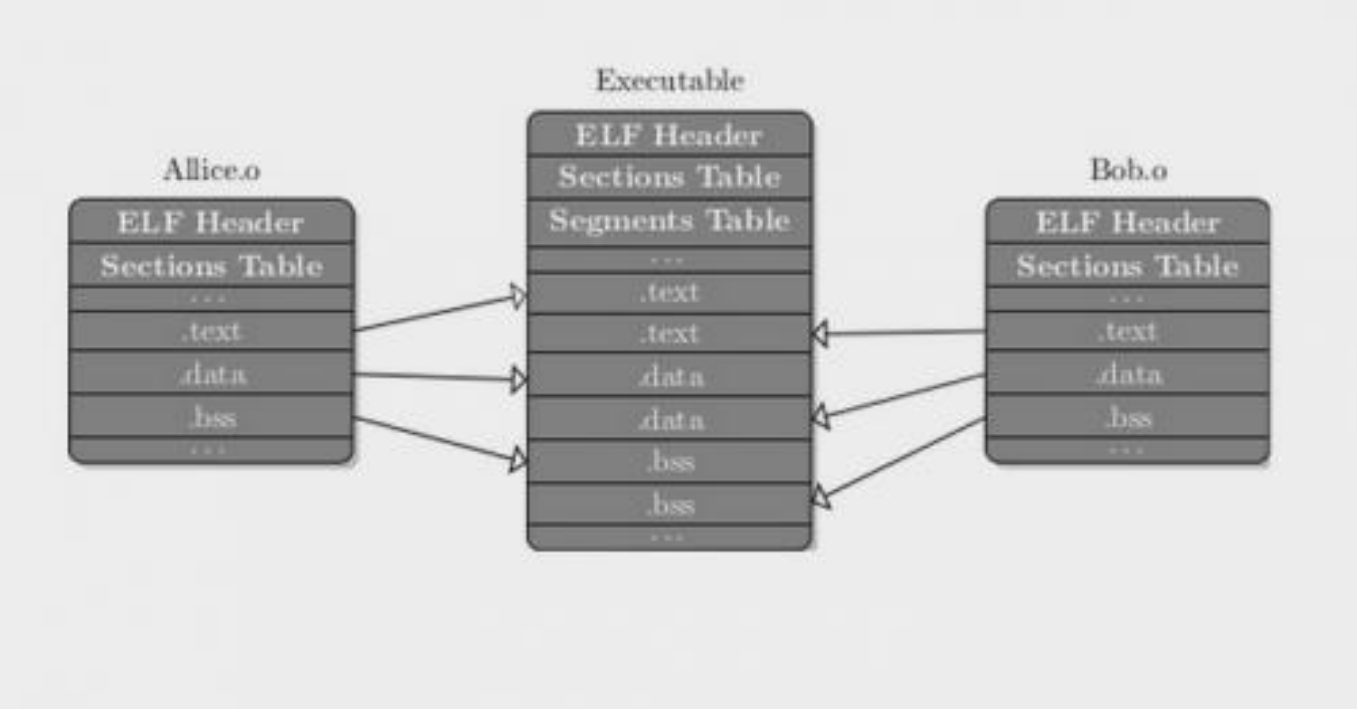
- Created by SUN Microsystems and introduced to UNIX in the late 1990s
- ELF is published in the ABI specification and becomes the standard for *NIX and BSD
- Common specification
  - ELF-32
  - ELF-64
  - ELF-ARM

ELF Basics – Binary building blocks

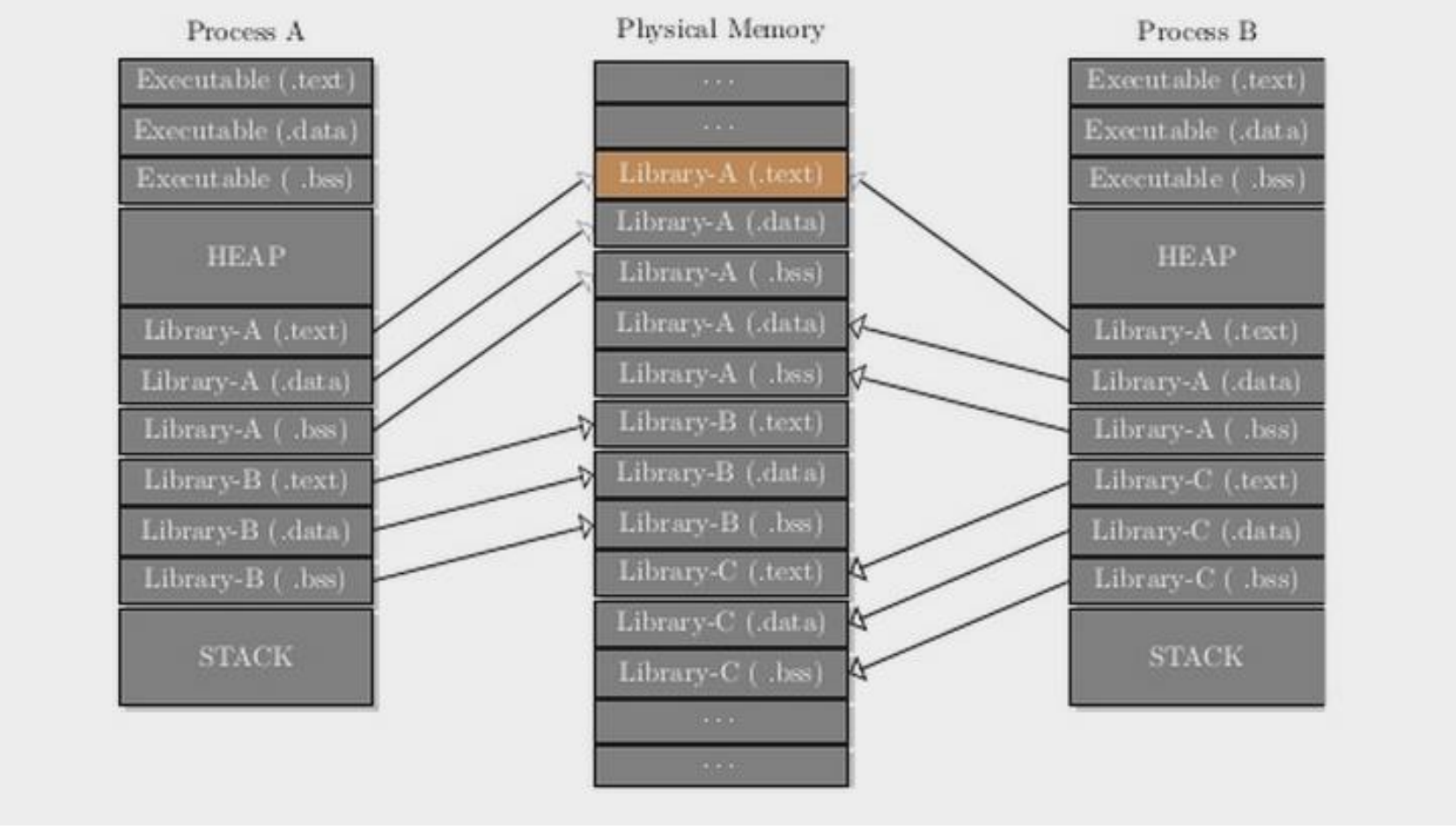




Linking – Static Linking



Linking – Dynamic Linking



ELF Structure – Tools of the trade

- **objdump** – used for section aware dumping and interpreting code
- **readelf** – tool for humanly reading and interpreting ELF files
- **ldd -** list the shared object dependencies

ELF Structure – ELF Header

```
readelf -h program

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:               0x8048330
  Start of program headers:          52 (bytes into file)
  Start of section headers:          4392 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         8
  Size of section headers:           40 (bytes)
  Number of section headers:         30
  Section header string table index: 27
```

ELF Structure – Program Headers

```
readelf -l program

Elf file type is EXEC (Executable file)
Entry point 0x8048330
There are 8 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00100 0x00100 R E 0x4
  INTERP         0x000134 0x08048134 0x08048134 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x004e4 0x004e4 R E 0x1000
  LOAD           0x000f0c 0x08049f0c 0x08049f0c 0x00108 0x00110 RW  0x1000
  DYNAMIC        0x000f20 0x08049f20 0x08049f20 0x000d0 0x000d0 RW  0x4
  NOTE           0x000148 0x08048148 0x08048148 0x00044 0x00044 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x4
  GNU_RELRO      0x000f0c 0x08049f0c 0x08049f0c 0x000f4 0x000f4 R   0x1

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .note.gnu.build-id .hash .gnu.hash .dynsym .dynstr .gnu.version
   03     .ctors .dtors .jcr .dynamic .got .got.plt .data .bss
   04     .dynamic
   05     .note.ABI-tag .note.gnu.build-id
   06
   07     .ctors .dtors .jcr .dynamic .got
```

ELF Structure – Section Table

```
readelf -S program
There are 30 section headers, starting at offset 0x1128:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        08048134 000134 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            08048148 000148 000020 00   A  0   0  4
  [ 3] .note.gnu.build-i NOTE            08048168 000168 000024 00   A  0   0  4
  [ 4] .hash             HASH            0804818c 00018c 000028 04   A  6   0  4
  [ 5] .gnu.hash         GNU_HASH        080481b4 0001b4 000020 04   A  6   0  4
  [ 6] .dynsym           DYNSYM          080481d4 0001d4 000050 10   A  7   1  4
  [ 7] .dynstr           STRTAB          08048224 000224 00004c 00   A  0   0  1
  [ 8] .gnu.version      VERSYM          08048270 000270 00000a 02   A  6   0  2
  [ 9] .gnu.version_r    VERNEED         0804827c 00027c 000020 00   A  7   1  4
  [10] .rel.dyn          REL             0804829c 00029c 000008 08   A  6   0  4
  [11] .rel.plt          REL             080482a4 0002a4 000018 08   A  6  13  4
  [12] .init             PROGBITS        080482bc 0002bc 000030 00  AX  0   0  4
  [13] .plt              PROGBITS        080482ec 0002ec 000040 04  AX  0   0  4
  [14] .text             PROGBITS        08048330 000330 00017c 00  AX  0   0 16
  [15] .fini             PROGBITS        080484ac 0004ac 00001c 00  AX  0   0  4
  [16] .rodata           PROGBITS        080484c8 0004c8 000015 00   A  0   0  4
  [17] .eh_frame         PROGBITS        080484e0 0004e0 000004 00   A  0   0  4
  [18] .ctors            PROGBITS        08049f0c 000f0c 000008 00  WA  0   0  4
```

ELF Structure – Symbol Table

```
readelf -s libtesting.so.1

Symbol table '.dynsym' contains 8 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00001339     1 OBJECT  GLOBAL DEFAULT   12 cPub
     2: 000001f8    10 FUNC    GLOBAL DEFAULT    7 fPub
     3: 0000020c   100 FUNC    GLOBAL DEFAULT    7 foo
     4: 00001328    16 OBJECT  GLOBAL DEFAULT   11 a
     5: 00001338     0 NOTYPE  GLOBAL DEFAULT  ABS __bss_start
     6: 0000133c     0 NOTYPE  GLOBAL DEFAULT  ABS _end
     7: 00001338     0 NOTYPE  GLOBAL DEFAULT  ABS _edata

Symbol table '.symtab' contains 27 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 000000b4     0 SECTION LOCAL  DEFAULT    1
     2: 000000e8     0 SECTION LOCAL  DEFAULT    2
     3: 00000168     0 SECTION LOCAL  DEFAULT    3
     4: 000001a8     0 SECTION LOCAL  DEFAULT    4
     5: 000001d0     0 SECTION LOCAL  DEFAULT    5
     6: 000001d8     0 SECTION LOCAL  DEFAULT    6
     7: 000001f8     0 SECTION LOCAL  DEFAULT    7
     8: 00001274     0 SECTION LOCAL  DEFAULT    8
     9: 00001314     0 SECTION LOCAL  DEFAULT    9
    10: 00001318     0 SECTION LOCAL  DEFAULT   10
    11: 00001328     0 SECTION LOCAL  DEFAULT   11
    12: 00001338     0 SECTION LOCAL  DEFAULT   12
    13: 00000000     0 SECTION LOCAL  DEFAULT   13
    14: 00000000     0 FILE    LOCAL  DEFAULT  ABS libtesting.c
```

Relocations

- Provides a map to the static linker when merging multiple files
- Provides a map to the dynamic linker for fixing references to shared object subroutines
- Provide the following information
  - Where the modification needs to be done
  - The symbol that needs the fixup
  - An algorithm for doing the fixup

Relocations - GOT

- The Global Offset Table is necessary because code in memory is read-only
- The .GOT section resides in the data segment that is read/write
- The machine code that requires a symbol from a shared object points to GOT
- The dynamic linker fixes the GOT entry when the symbol is required at run time

Relocations - PLT

- The Procedure Linkage Table is required for calling subroutines from shared objects
- The machine code that requires a subroutine from a shared object points to PLT
- The PLT entry bounces of GOT in order to push the subroutine name on to the stack, and then call the dynamic loader
- After the first call to the subroutine the entry in GOT will point to the absolute address of the subroutine

Relocations

```
readelf -r libdynamic.o

Relocation section '.rel.text' at offset 0x5f8 contains 8 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
0000001d  00001402 R_386_PC32        00000000   __i686.get_pc_thunk.bx
00000023  0000150a R_386_GOTPC       00000000   _GLOBAL_OFFSET_TABLE_
00000029  00000409 R_386_GOTOFF      00000000   .bss
0000002f  00000409 R_386_GOTOFF      00000000   .bss
00000035  00000d03 R_386_GOT32       00000004   so_int_global
00000041  00000d03 R_386_GOT32       00000004   so_int_global
00000052  00000e04 R_386_PLT32       00000000   so_fpublic_global
0000005b  00000209 R_386_GOTOFF      00000000   .text

Relocation section '.rel.data.rel.local' at offset 0x638 contains 2 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
00000000  00000401 R_386_32          00000000   .bss
00000004  00000201 R_386_32          00000000   .text

Relocation section '.rel.data.rel' at offset 0x648 contains 2 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
00000000  00000d01 R_386_32          00000004   so_int_global
00000004  00000e01 R_386_32          00000000   so_fpublic_global
```

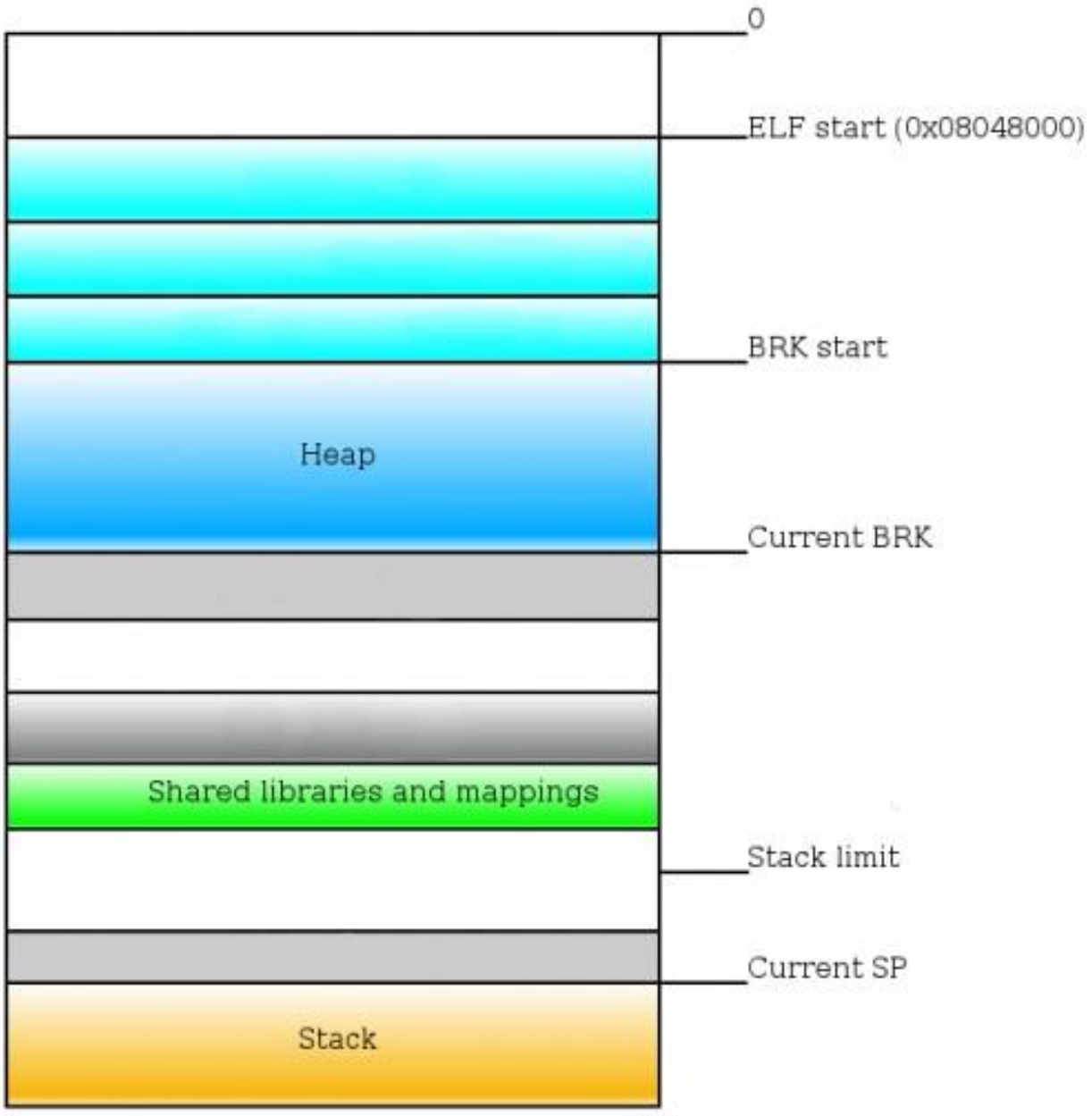
Memory Mapping – Loader flow

1) Load the main binary
2) Check and load dependencies
3) Load the symbol resolution map
4) Fix data relocations (.GOT)
5) Fix function relocation (GOT.PLT)
6) Call library initializers (.init)
7) Start the program

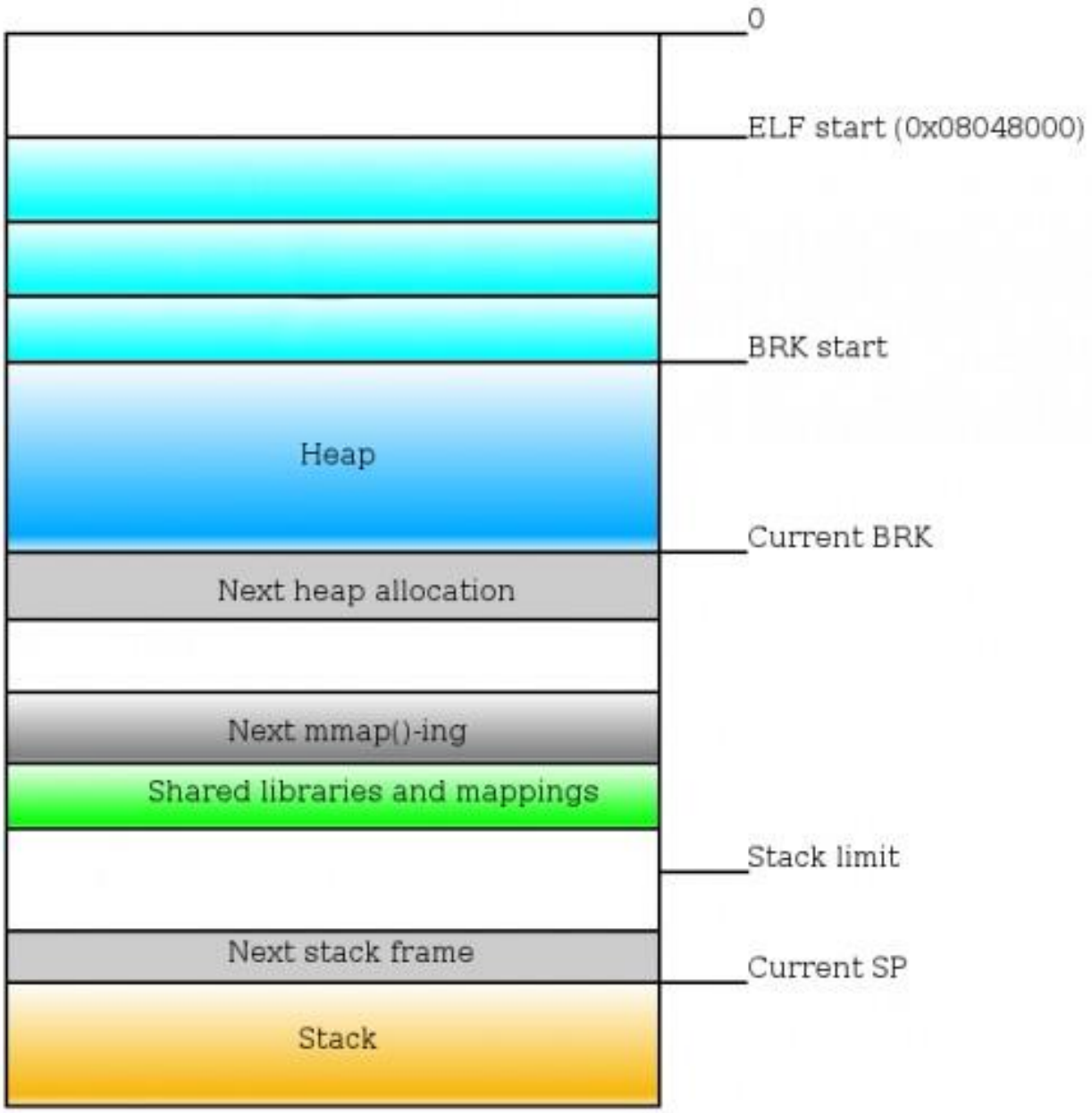
Memory Mappings – /proc/PID/maps

```
$ gcc -Wall hw.c -o hw -m32
$ ./hw  &
[1] 4771
Hello world
$ cat /proc/4771/maps
08048000-08049000 r-xp 00000000 08:06 1843771                            /tmp/hw
08049000-0804a000 r--p 00000000 08:06 1843771                            /tmp/hw
0804a000-0804b000 rw-p 00001000 08:06 1843771                            /tmp/hw
0804b000-0806e000 rw-p 00000000 00:00 0                                  [heap]
f7ded000-f7dee000 rw-p 00000000 00:00 0
f7dee000-f7f93000 r-xp 00000000 08:06 917808                             /lib32/libc-2.17.so
f7f93000-f7f95000 r--p 001a5000 08:06 917808                             /lib32/libc-2.17.so
f7f95000-f7f96000 rw-p 001a7000 08:06 917808                             /lib32/libc-2.17.so
f7f96000-f7f99000 rw-p 00000000 00:00 0
f7fd9000-f7fdb000 rw-p 00000000 00:00 0
f7fdb000-f7fdc000 r-xp 00000000 00:00 0                                  [vdso]
f7fdc000-f7ffc000 r-xp 00000000 08:06 917869                             /lib32/ld-2.17.so
f7ffc000-f7ffd000 r--p 0001f000 08:06 917869                             /lib32/ld-2.17.so
f7ffd000-f7ffe000 rw-p 00020000 08:06 917869                             /lib32/ld-2.17.so
fffdd000-ffffe000 rw-p 00000000 00:00 0                                  [stack]
```

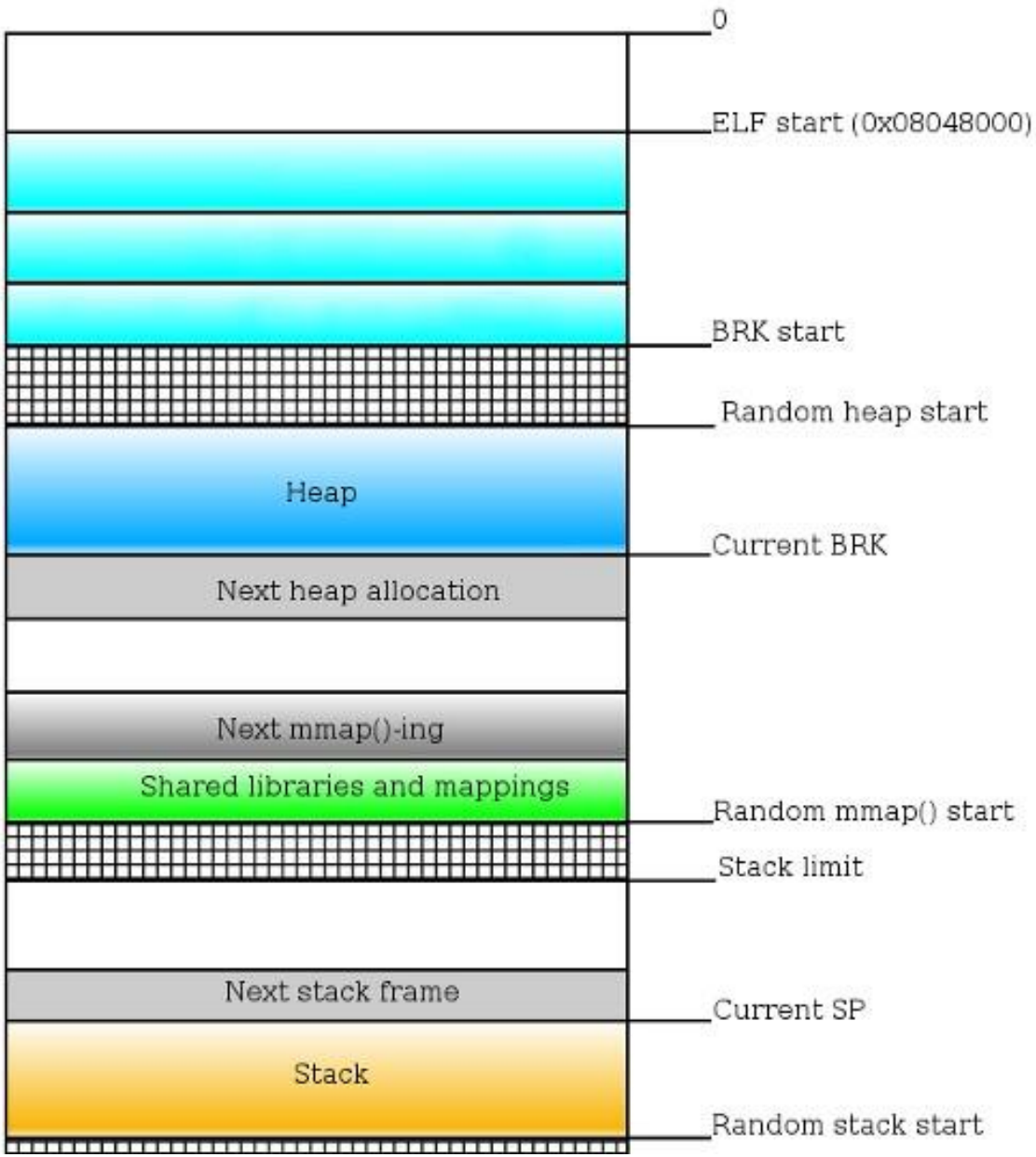
Memory Layout



Detailed Layout



ASLR


0
ELF start (0x08048000)
BRK start
Random heap start
Heap
Current BRK
Next heap allocation
Next mmap()-ing
Shared libraries and mappings
Random mmap() start
Stack limit
Next stack frame
Current SP
Stack
Random stack start