What's in a binary?

Simon Kågström

Consultant
https://github.com/SimonKagstrom/emilpro

February 20, 2025

Demo


?
KRASCH
Jag förstår ingenting!... Det är sabotage!... Nej, min herre, jag säger ju att det är sabotage!... Någon har saboterat farkosten!...

Outline

- Part I: Motivation
- Part II: What does the disassembly writer need?
- Part III: Why is this easier now than 10 years ago?

Part I: Motivation

Motivation

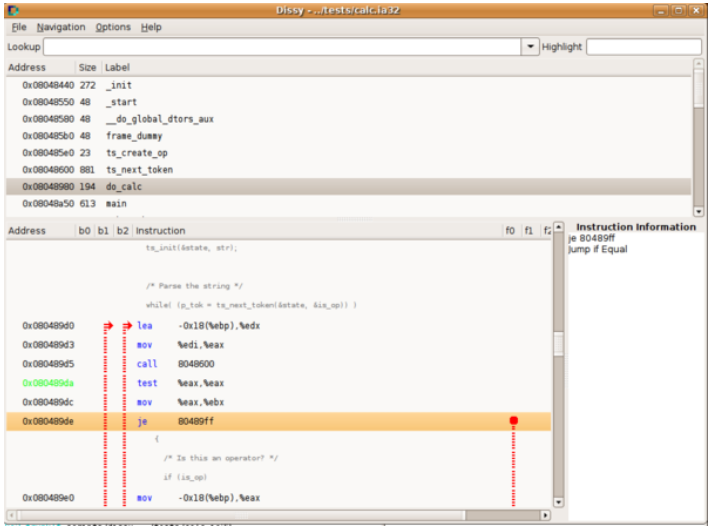
- C/C++ with inline assembly
- Systems and programming environments where a debugger wasn't available

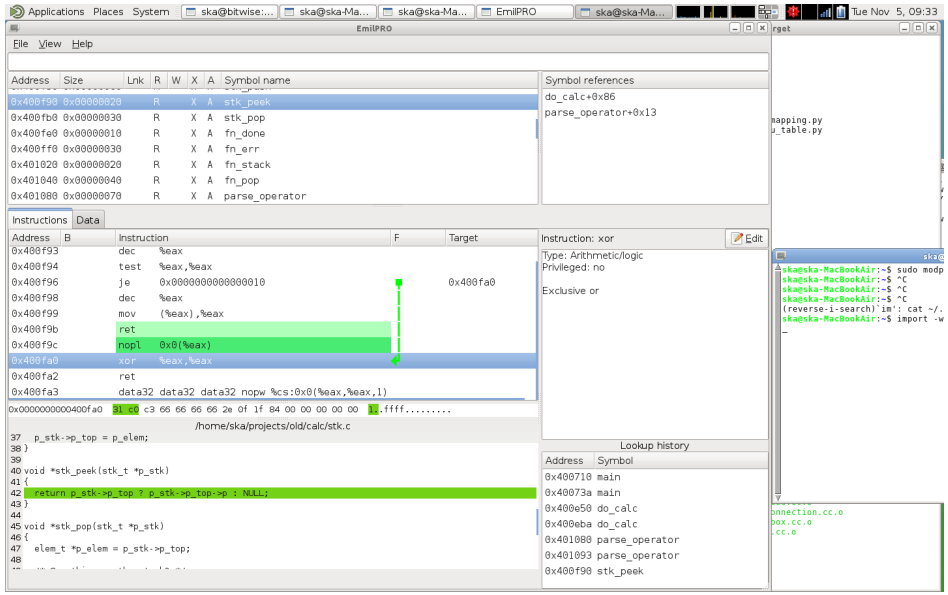
Example

```
#define _syscall1(type,name,atype,a) type name(atype a) {
        unsigned long out;
        __asm__ volatile (
        ".set  push\n.set  noreorder\n"
        ".short 0xfefe\n"
        ".short %1\n"
        ".pushsection .cibylstrtab, \"aS\"\n"
        "1: .asciz \"" #name "\"\n"
        ".popsection\n"
        ".long 1b\n"
        ".set\tpop\n"
        "move %[out], $2\n
        : [out]"=d" (out)
        : "r"(a)
        : "memory", "$2"
        );
        return (type) out;
}
```

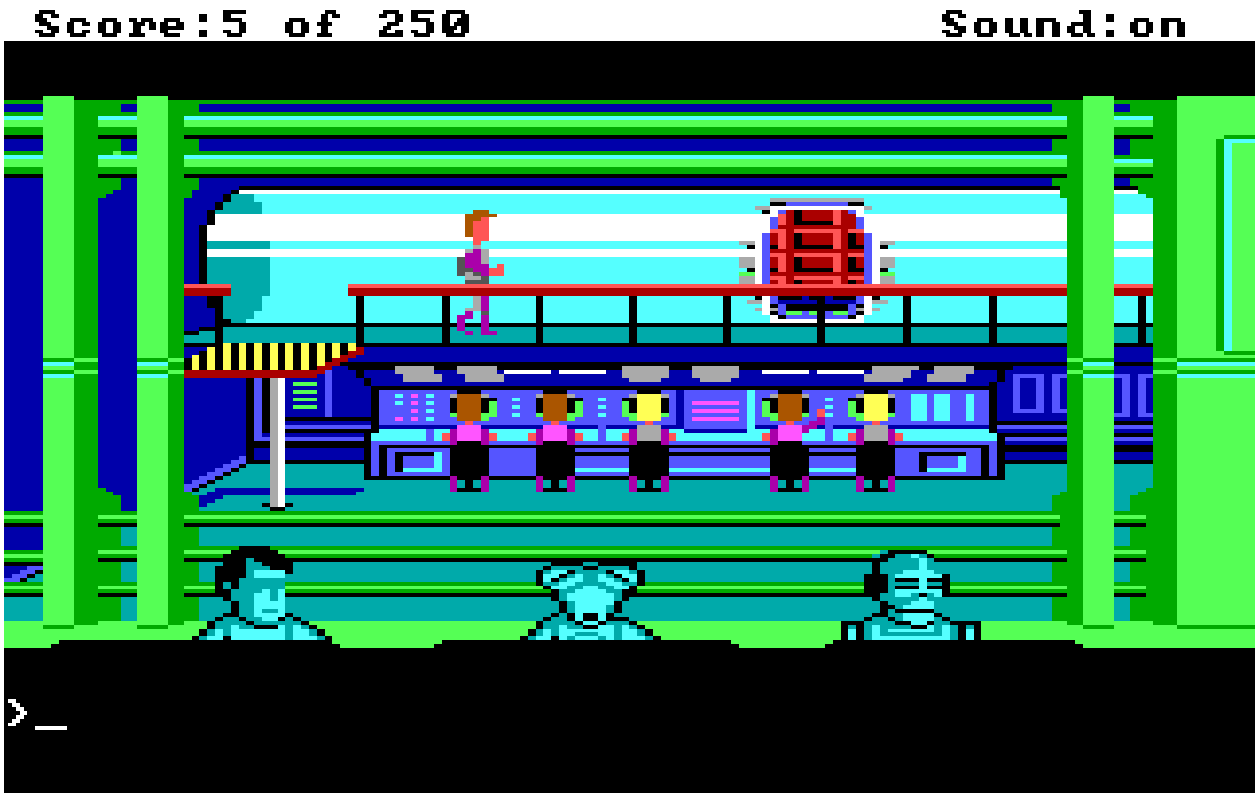
Backstory

- Objdump output is cumbersome to navigate through
- I wanted a graphical application that allows easier navigation





Part II: What is needed for a disassembler?



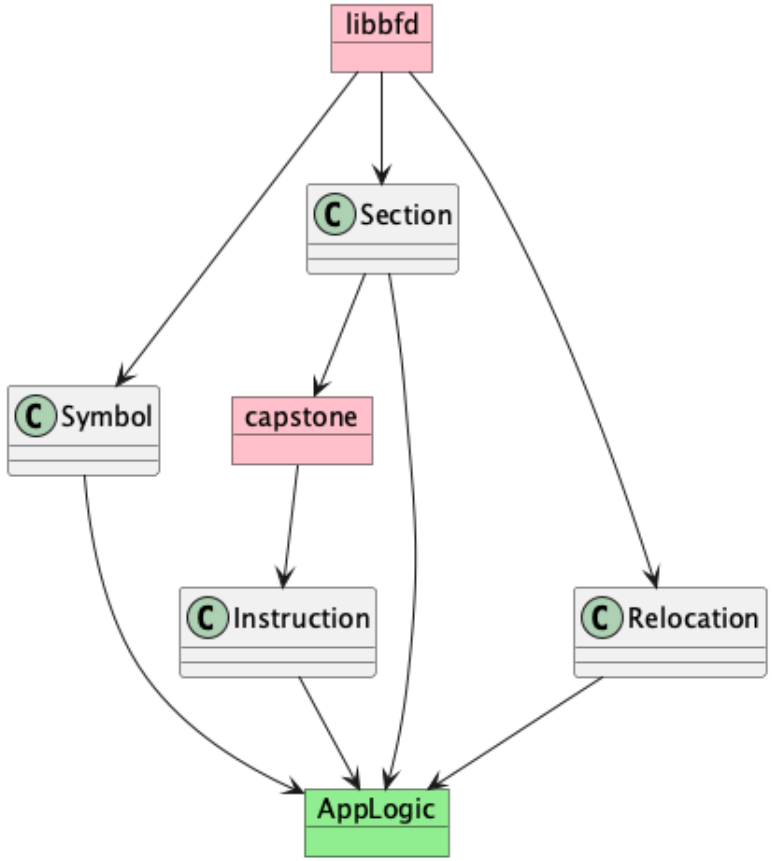
Binary formats

- **Linux/FreeBSD** etc: ELF
- **MacOS**: Mach-O
- **Windows**: PE

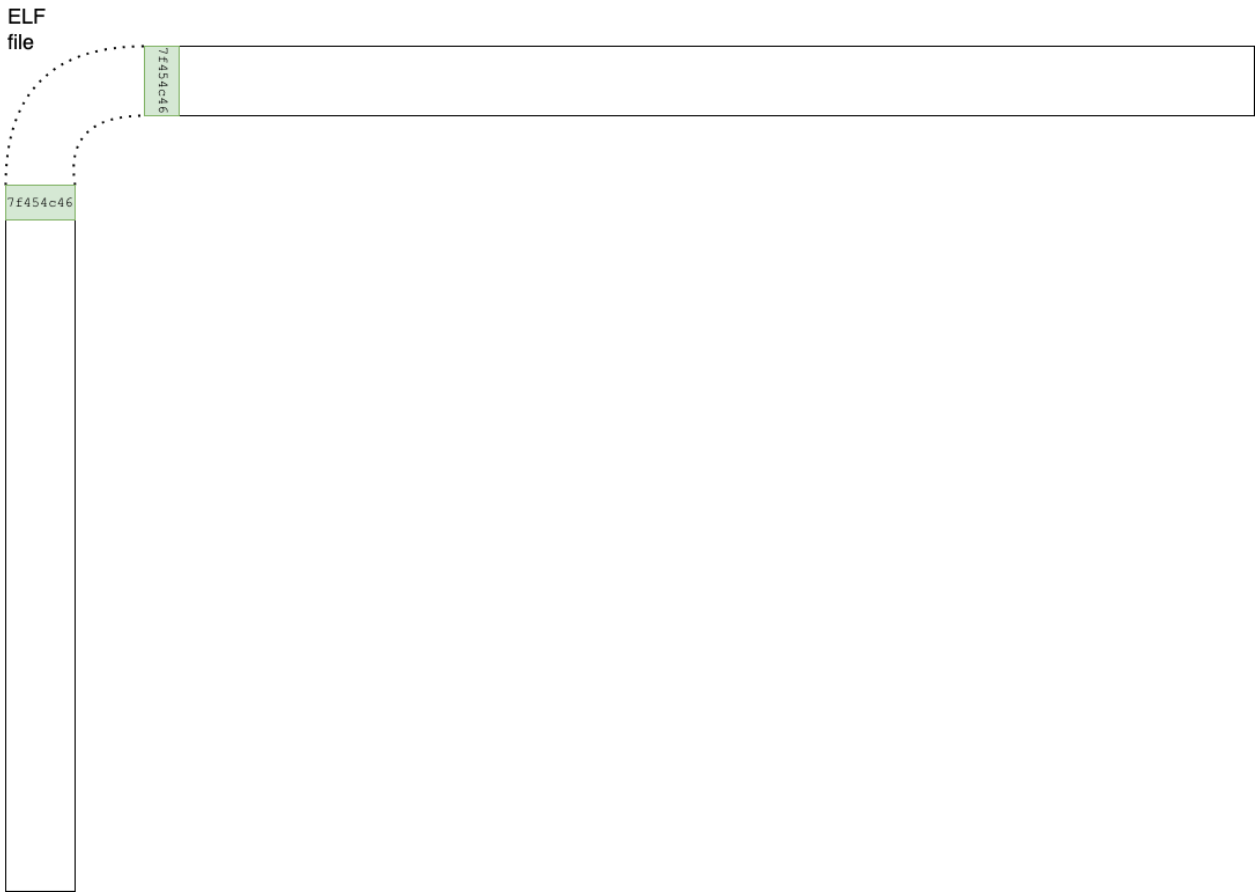
How to write a dissassembler?

I did not write everything from scratch!

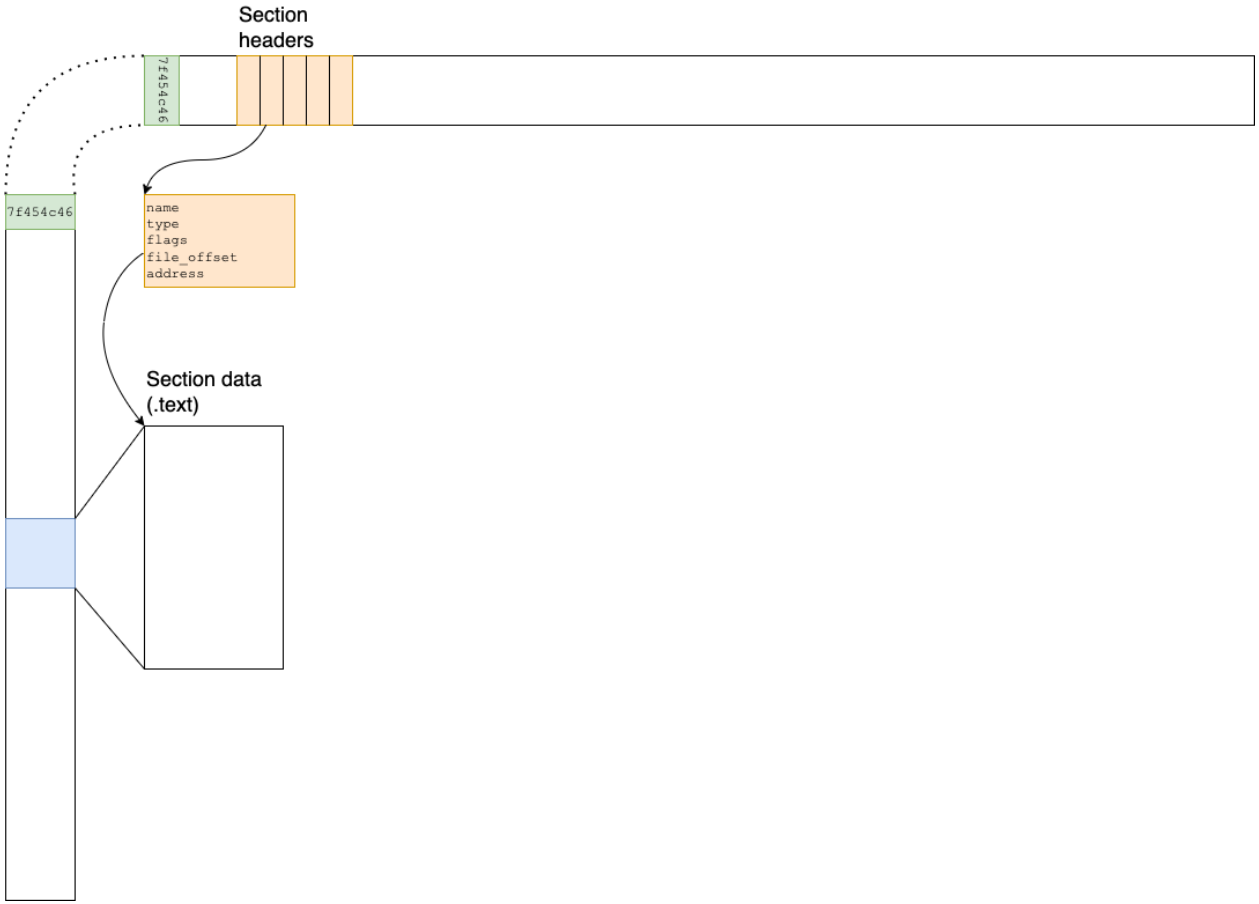
- **Qt**: GUI framework
- **libbfd**: Part of binutils, used for reading binary files
- **capstone**: Disassembler library



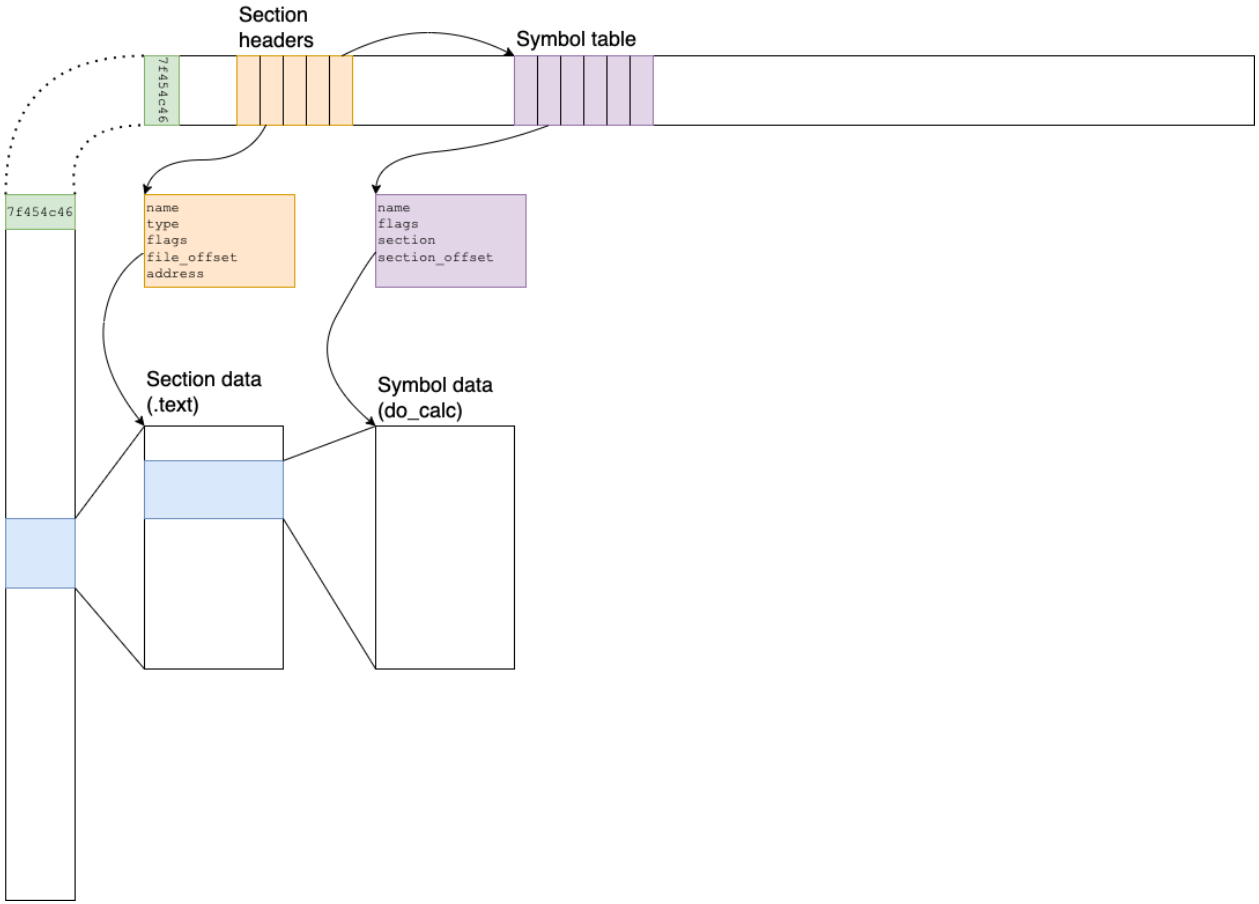
Loading a binary



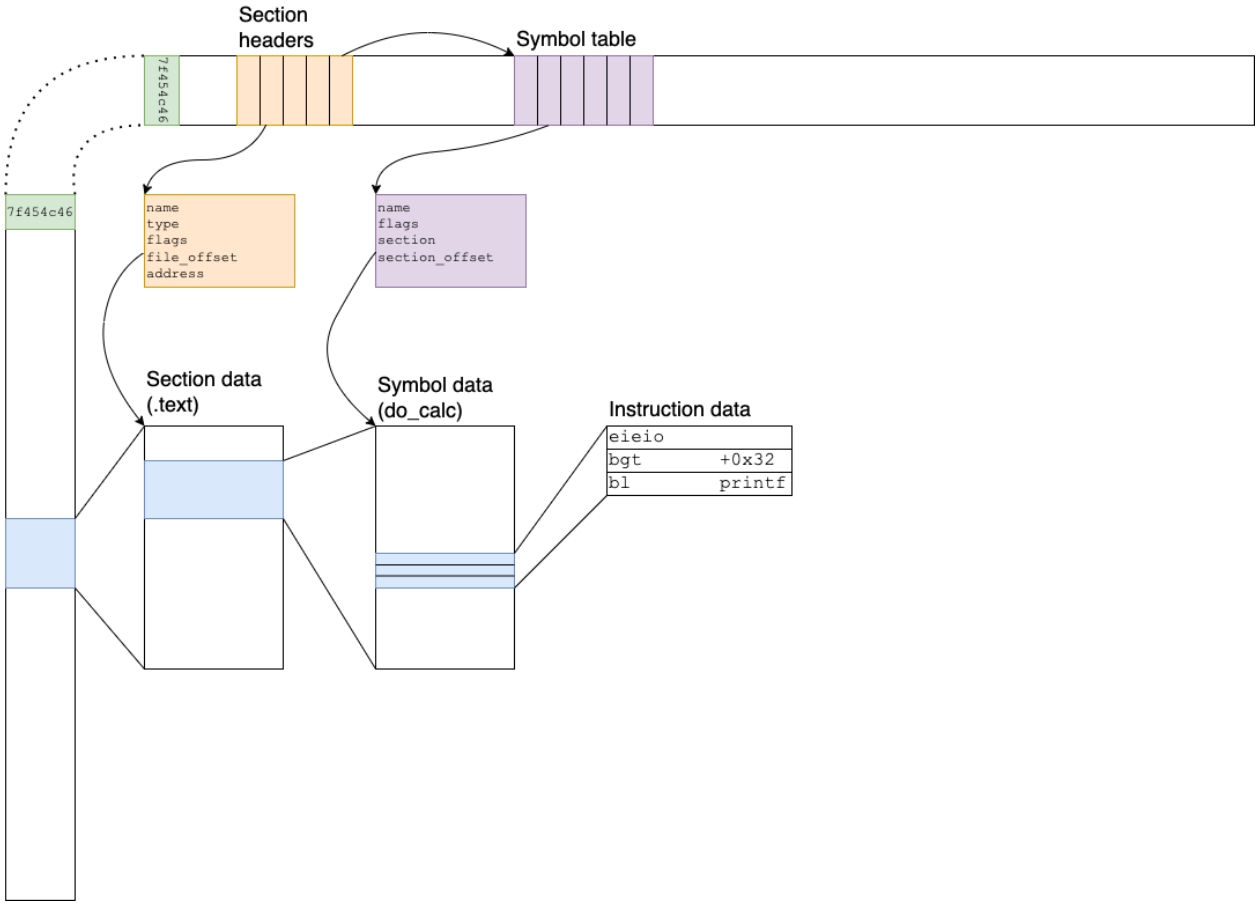
Loading a binary, sections



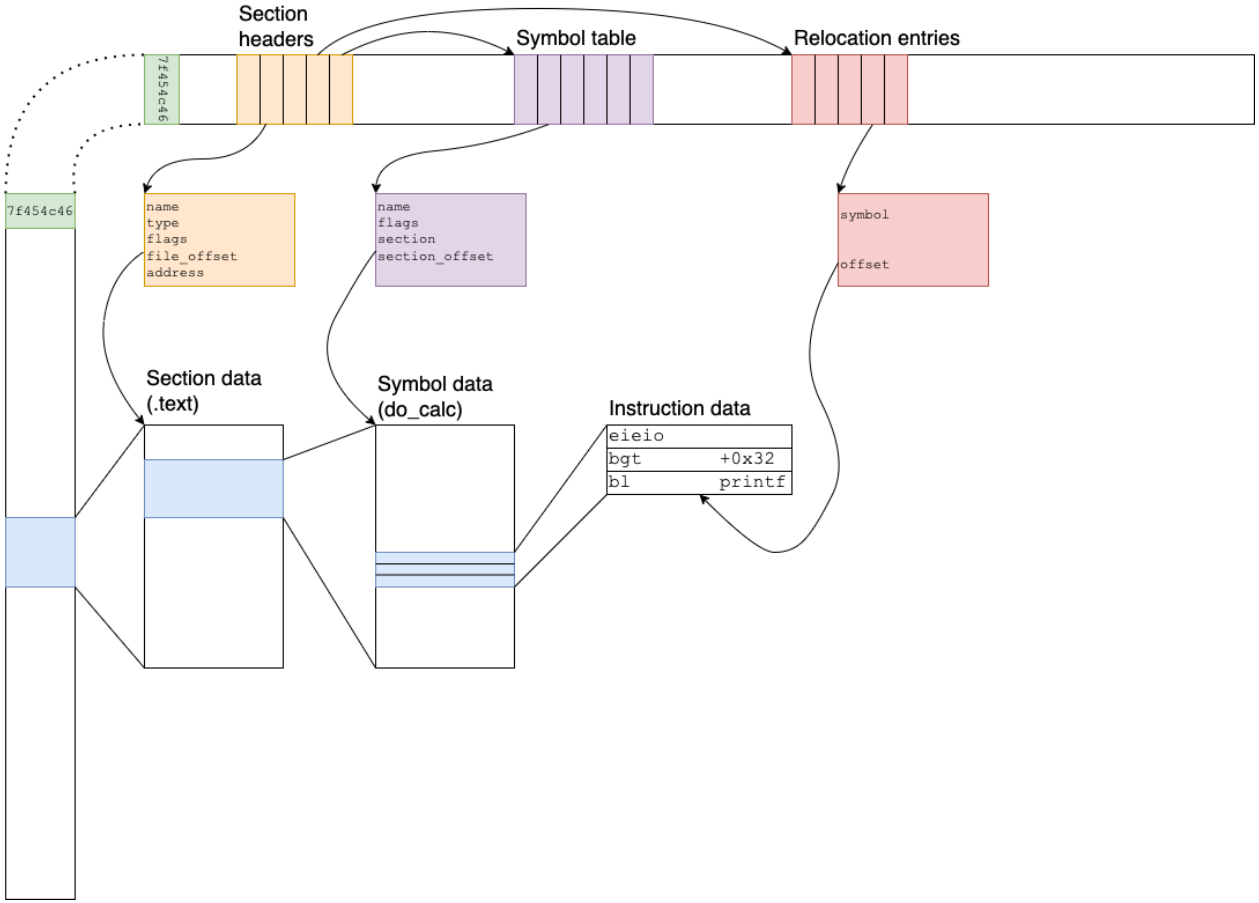
Loading a binary, symbols



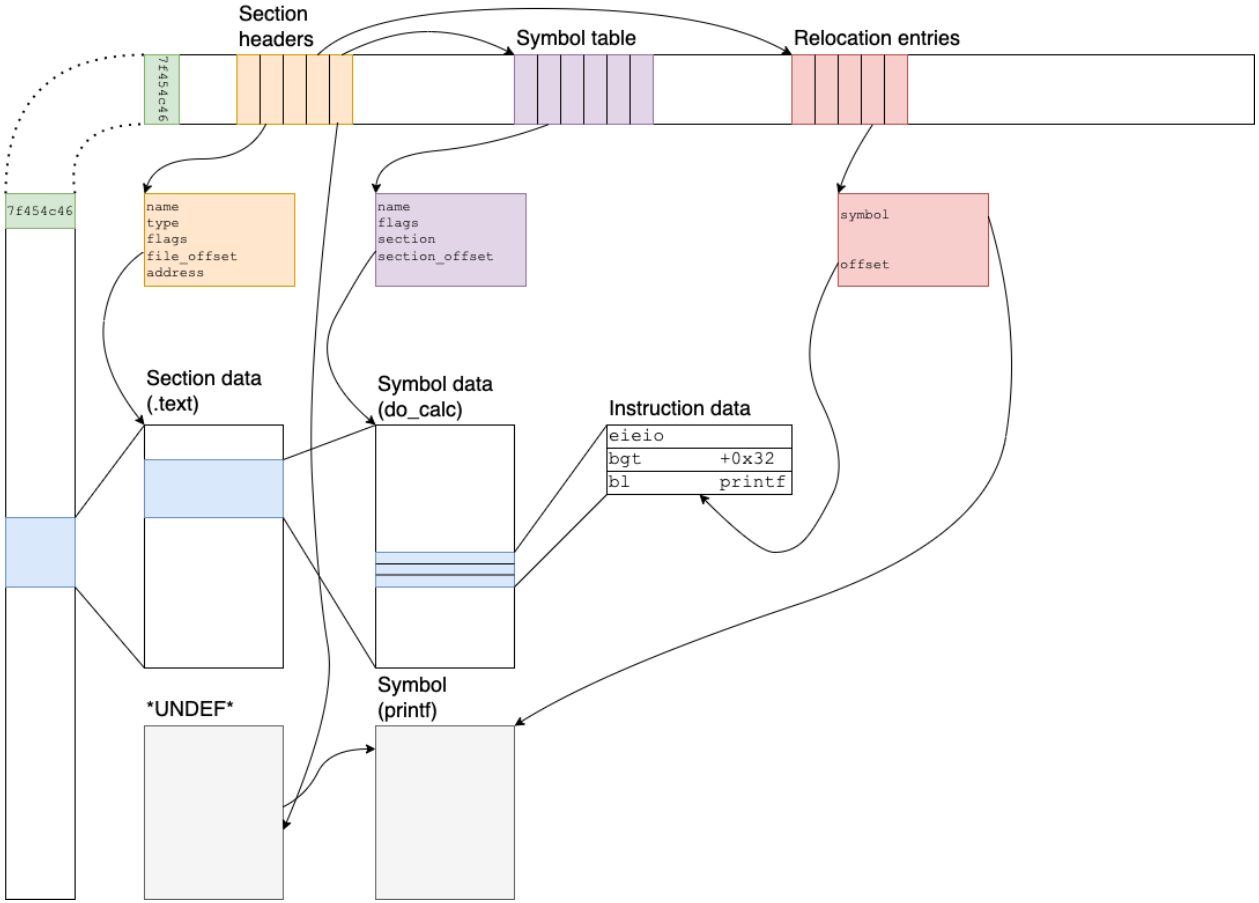
Loading a binary, instructions



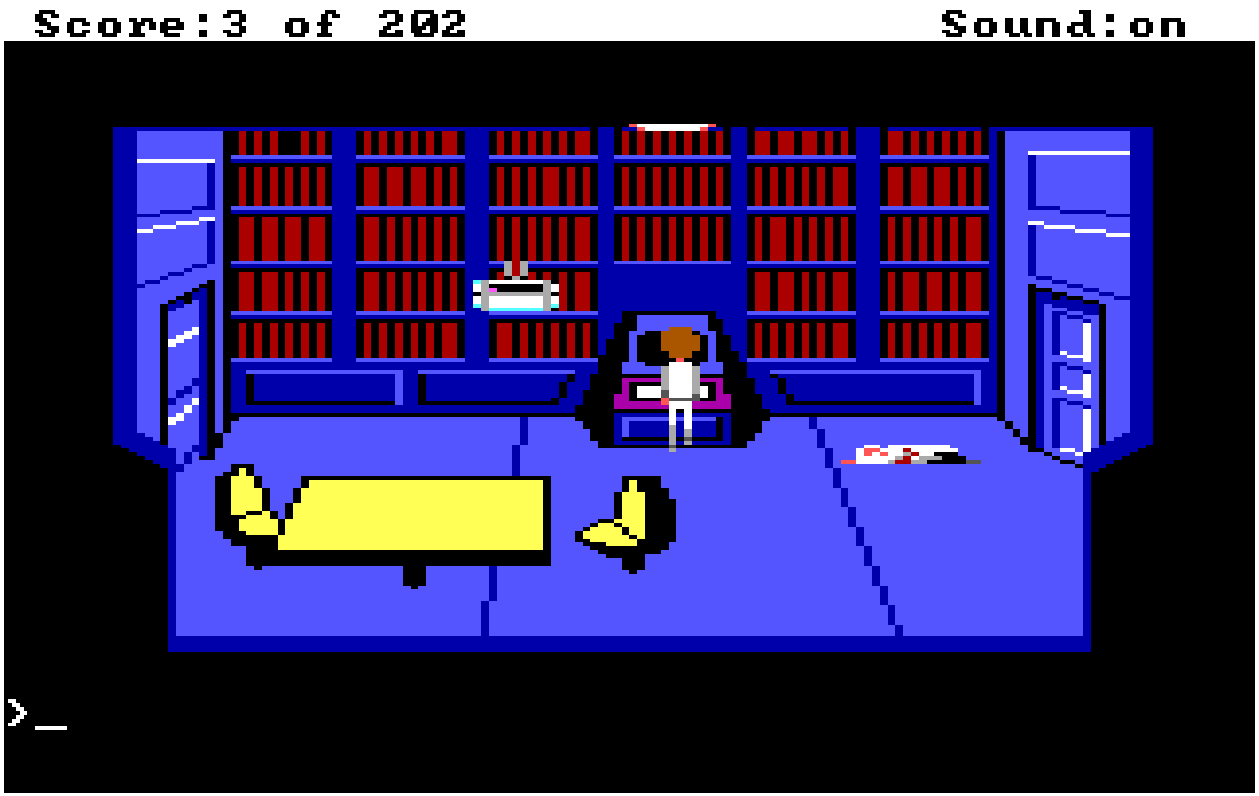
Loading a binary, relocations



Loading a binary, relocations

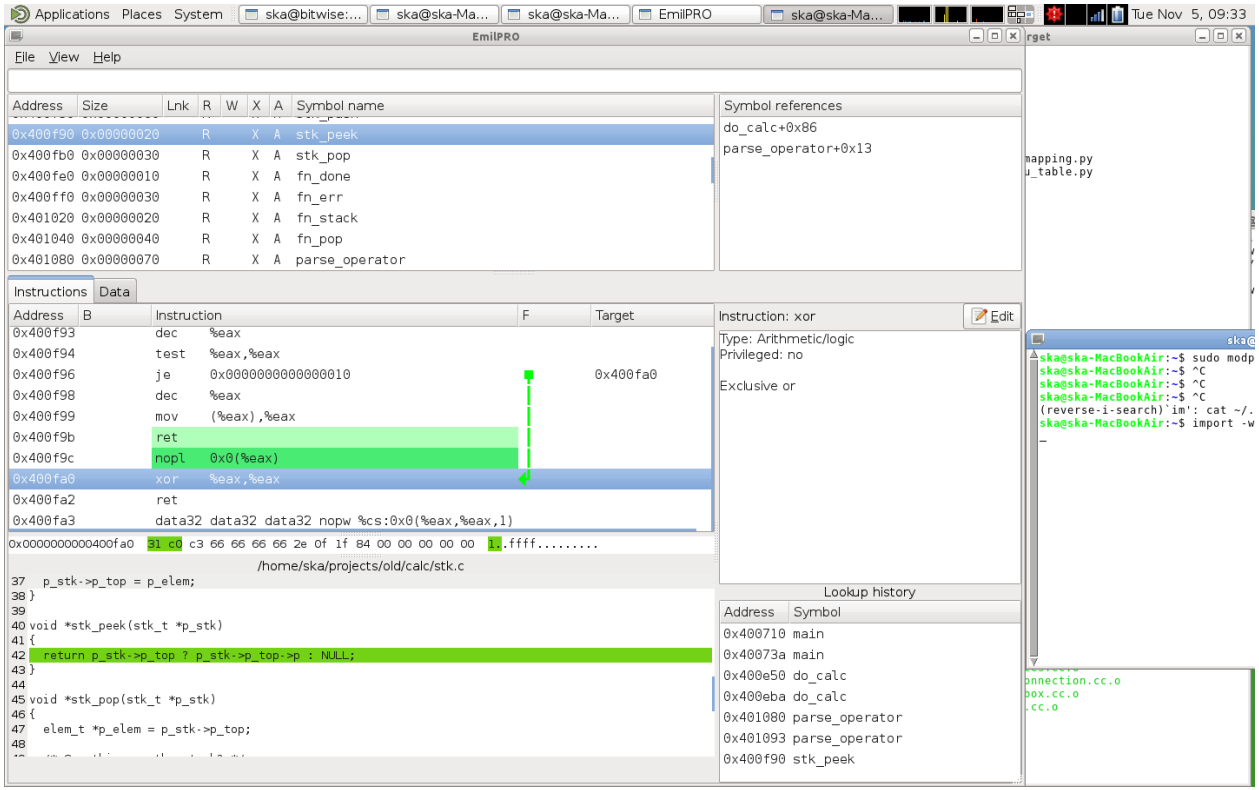


Part III: Why is this easier now than 10 years ago?



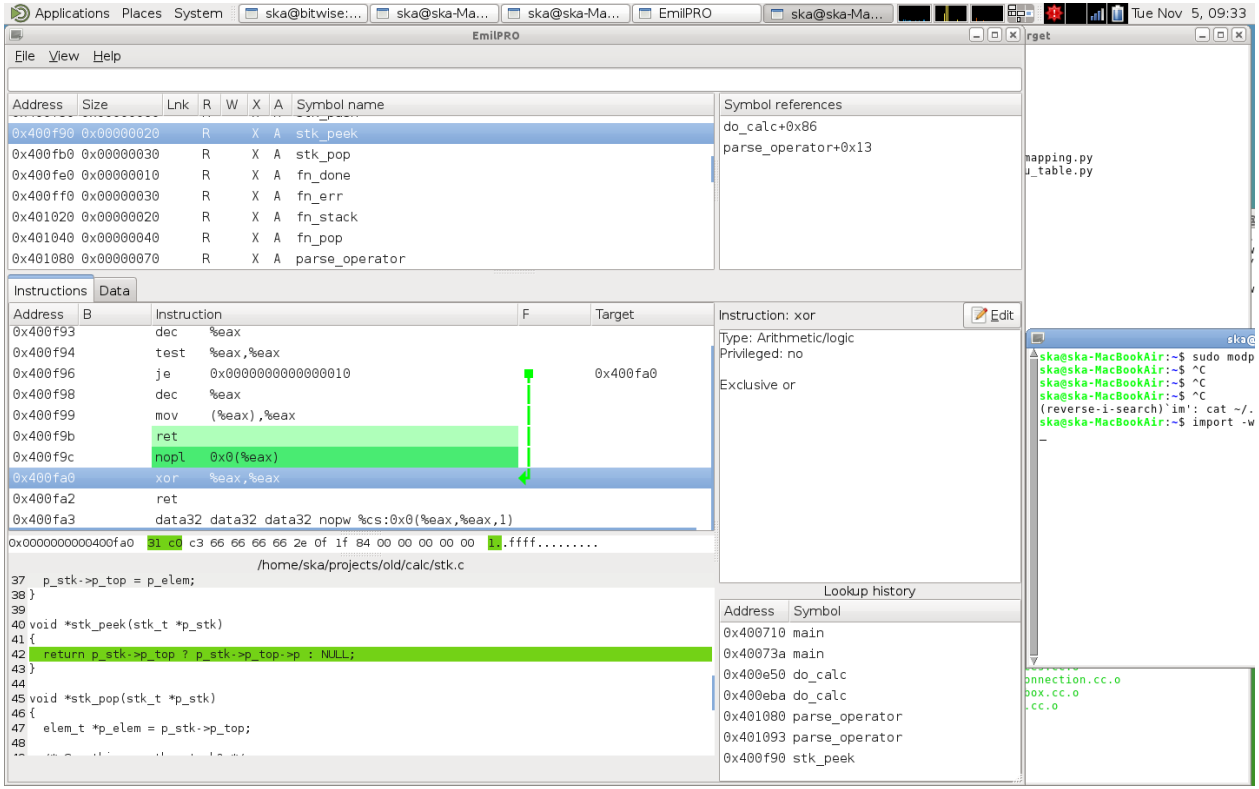
Bad design decisions in the original project

- feature creep: core functionality sketchy, work on irrelevant features



Bad design decisions in the original project

- feature creep: core functionality sketchy, work on irrelevant features
- libbfd for disassembly: the multiarch-dev issue



C++20 and onwards

- The previous implementation was done just around the C++11 introduction, but used C++03
- Now C++23, so much better!

C++03

```
for (InstructionList_t::const_iterator it = instructions.begin();
     it != instructions.end();
     ++it) {
```

C++23

```
for (const auto& insn : m_instructions)
{
```

C++ infrastructure

- The conan package manager
- The addres sanitizer

Conan

```
[requires]
doctest/2.4.11
trompeloeil/48
fmt/11.0.2
capstone/5.0.1
etl/20.39.4
libiberty/9.1.0

[generators]
CMakeDeps
CMakeToolchain

[layout]
cmake_layout
```

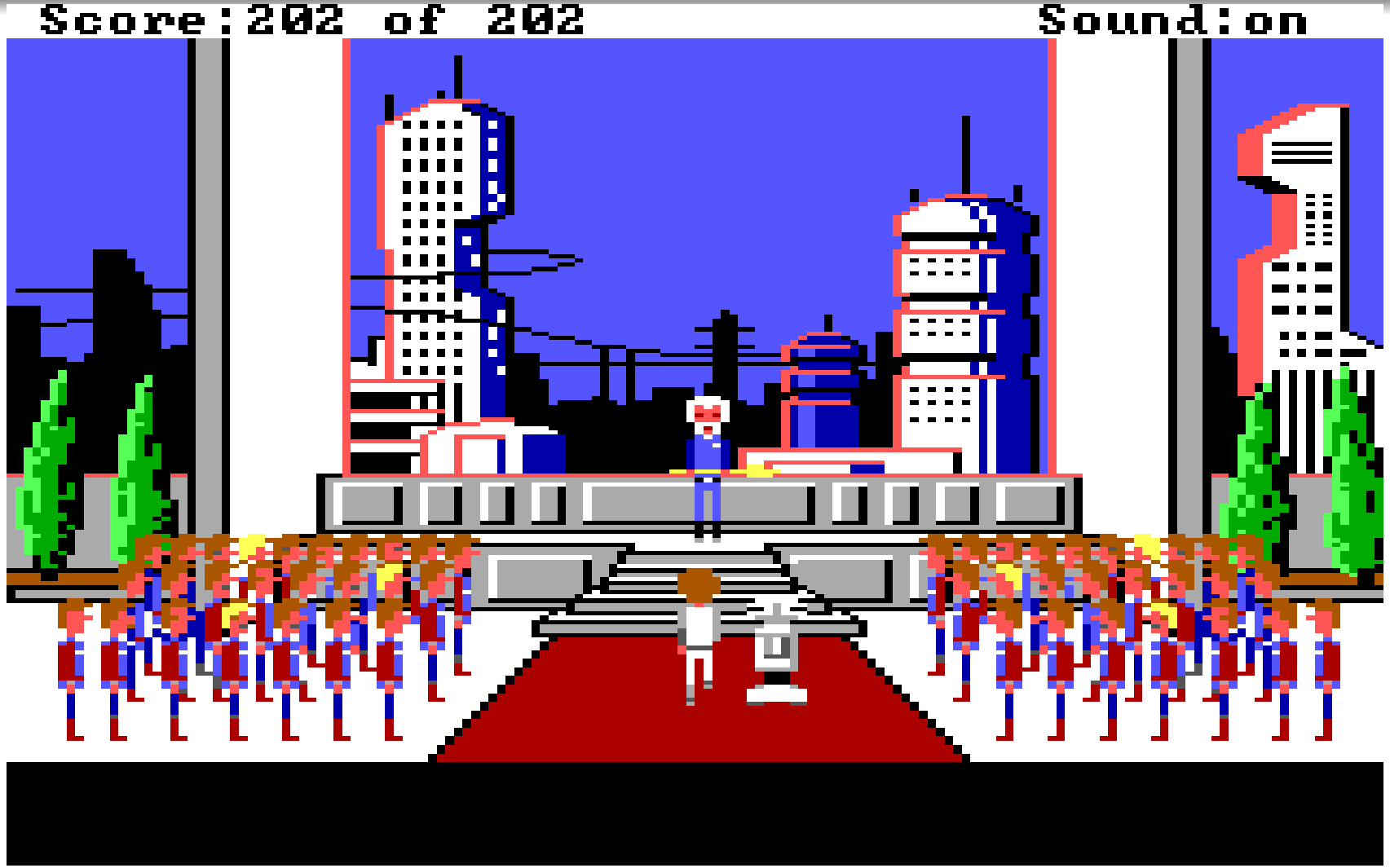
AI

- I use Github copilot
- Very helpful with Qt development

AI

```
Accept Tab Accept Word ⌘ → ···
\begin{frame}{Static executables}      You, 37 seconds ago • Uncommitted changes
  % Sections, symbols, relocations
  % Example from embedded system
  % Load sections into memory
  % Entry point
  % Disassembler, compiler, linker, loader
  % Relocation entries
  % Why are they needed?
  % - when a function is called, the linker doesn't know where it is
  % - the relocation entry tells the linker to fix the call site
  % - the linker will then fix the call site to point to the function
  % - the function is in the shared library
  % - the linker will also fix the function to point to the shared library
  % - the shared library is loaded into memory
  % - the function is called
  % - the function is executed
  % - the function returns
  % - the function is called again
  % - the function is executed again
  % - the function returns again
  % - the function is called a third time
  % - the function is executed a third time
  % - the function returns a third time
  % - the function is called a fourth time
  % - the function is executed a fourth time
  % - the function returns a fourth time
  % - the function is called a fifth time
  % - the function is executed a fifth time
  % - the function returns a fifth time
  % - the function is called a sixth time
  % - the function is executed a sixth time
  % - the function returns a sixth time
  % - the function is called a seventh time
  % - the function is executed a seventh time
  % - the function returns a seventh time
  % - the function is called an eighth time
  % - the function is executed an eighth time
  % - the function returns an eighth time
  % - the function is called a ninth time
  % - the function is executed a ninth time
  % - the function returns a ninth time
  % - the function is called a tenth time
  % - the function is executed a tenth time
  % - the function returns a tenth time
  % - the function is called an eleventh time
  % - the function is executed an eleventh time
```

Questions and comments!



Images from
`http://www.falselogic.net/LetsPlay/SpaceQuest.html`
Ian Lance Taylors linker series is the source of parts of this talk

Actors and objects

Actors

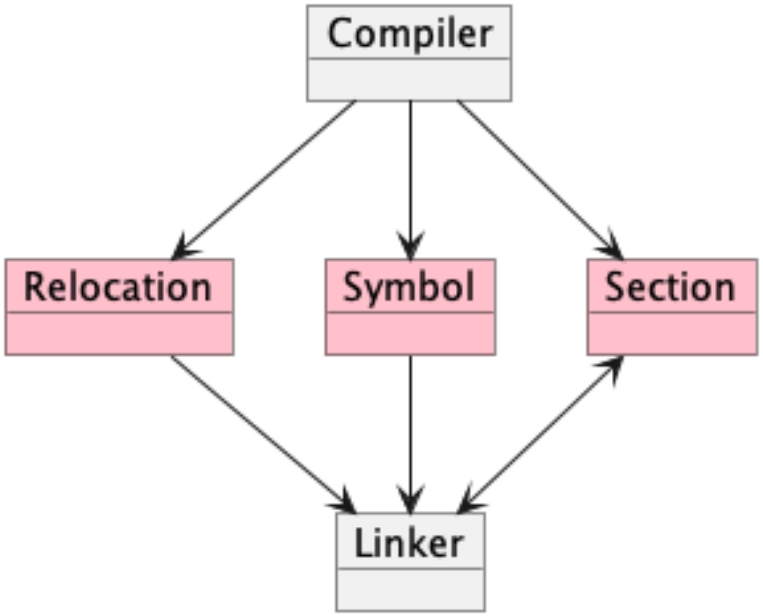
- **Compiler**
- **Linker**
- **Loader**
- **Disassembler**

Objects

- **Instructions**: The actual code
- **Sections**: Text, data, debug info etc
- **Symbols**: Functions/methods, variables, ...
- **Relocations**: Call sites for later resolving

Producing a binary

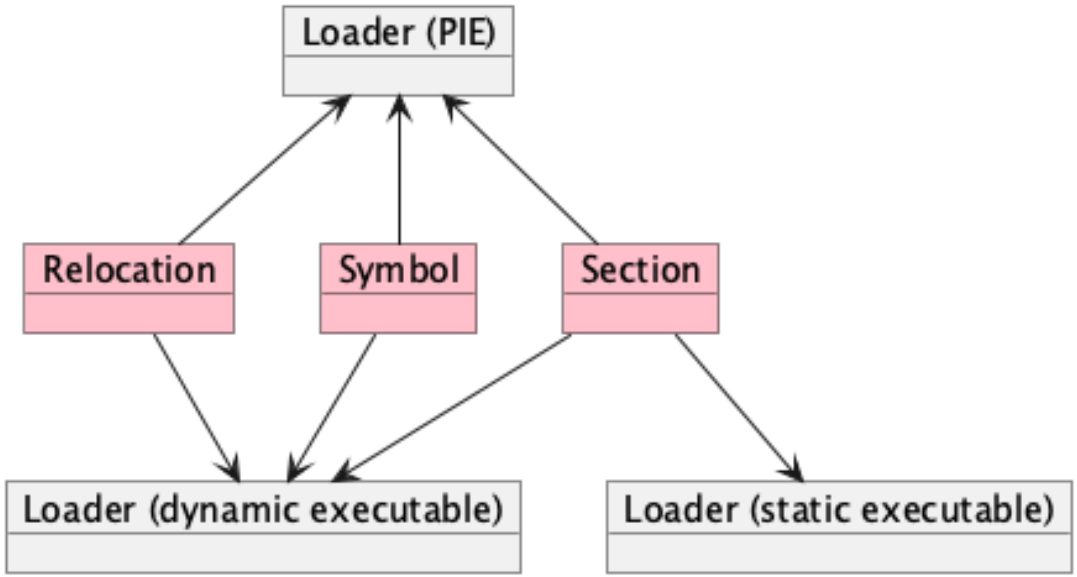
The compiler produces symbols, relocations plus data and text sections



Loading a binary

Different categories of binaries are handled differently:

- Execute from direct-mapped flash (embedded systems)
- Static executables
- Dynamic executables
- PIEs (Position-Independent Executables)



Static executables

Dynamic executables

PIEs

Relocations

- If a non-local function is called, an undefined symbol is added
- The compiler adds a relocation entry for the call site
- When linking, the linker will resolve these symbols
- Different types depending on instruction

libbfd