

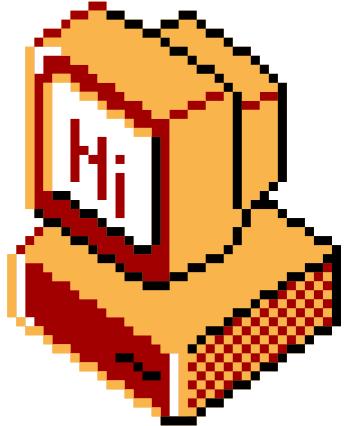
PERTH SOCIALWARE

0x03:

Reverse Engineering Workshop

Part 2

\$ ~/: groups "socialware"

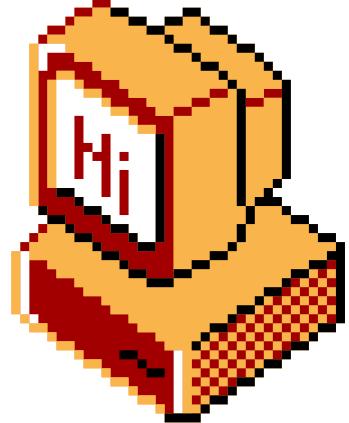


Welcome!

About & Aims

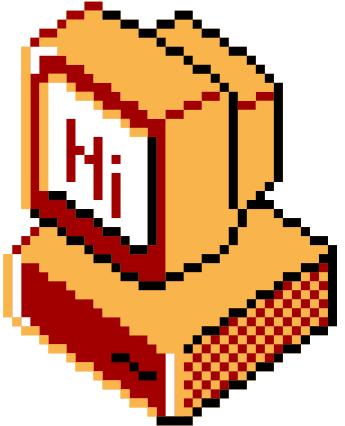
Enjoy!

\$ n/: groups "Socialware"



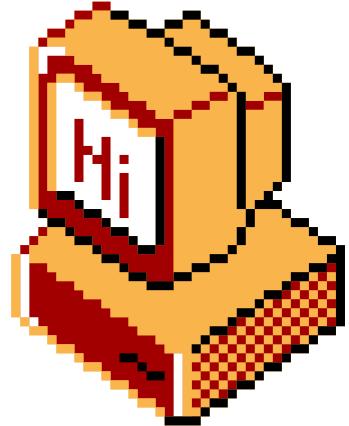
Thanks to UWA UISS for the venue!

```
$ ~/: cat ./housekeeping
```



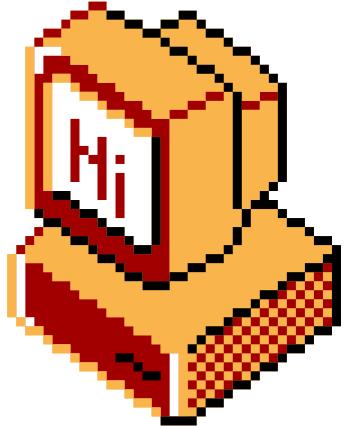
- Don't break stuff
- If you break stuff tell us
- Be respectful
- Have fun.

```
$ ./ cat ./housekeeping
```



- Also, we don't have access to a guest Wi-Fi network here - trying to connect to them is **out of scope**
- Please use your own hotspots, or ask one of us to lend you our hotspot

\$ n/: groups "socialware"



Acknowledgement of Country

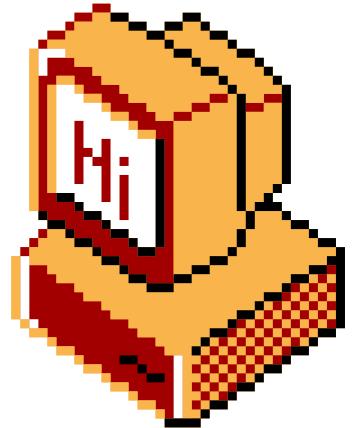
\$ n/: whoami

Emu Exploit

- We are a competitive hacking team current rank #1 in Australia on CTFtime.org
- Founded in 2021, the team consists of many highschoolers as well as industry professionals

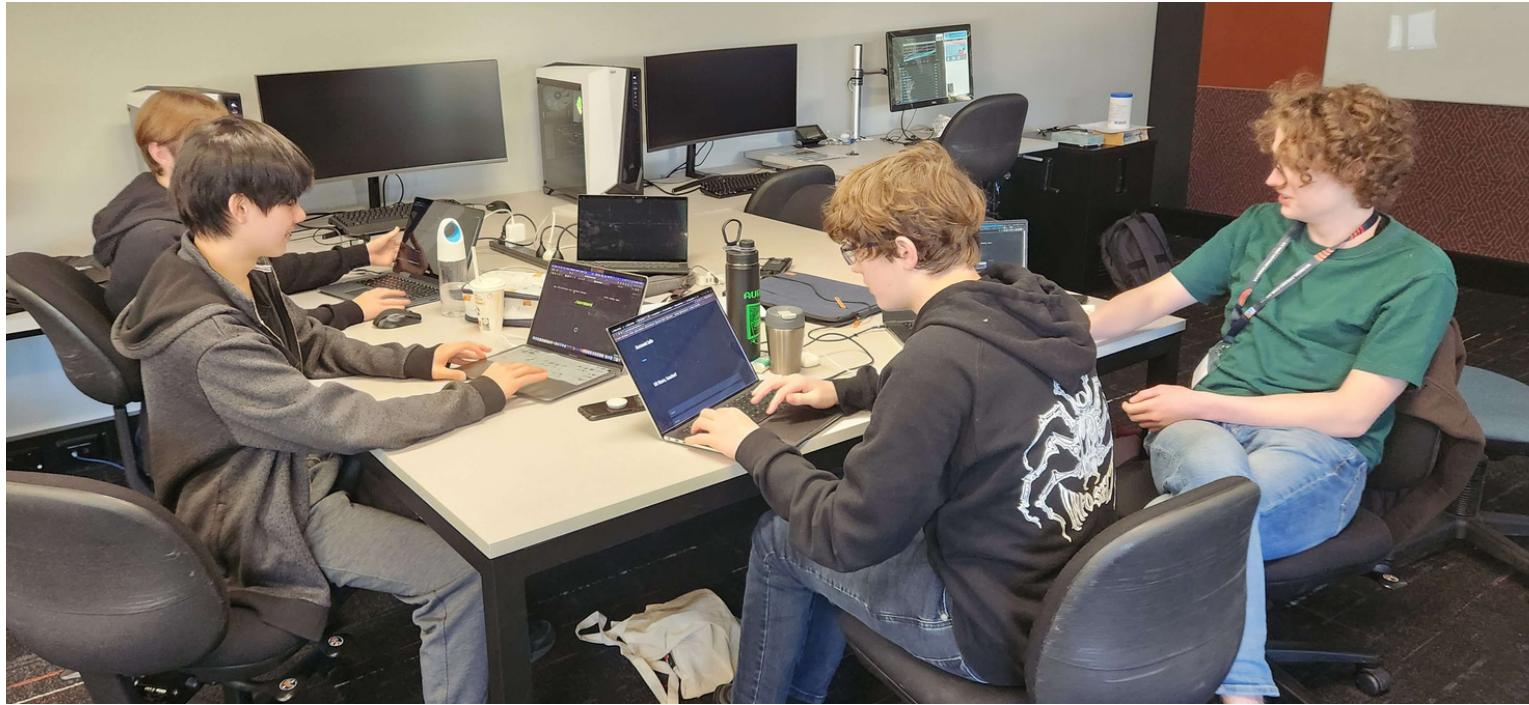
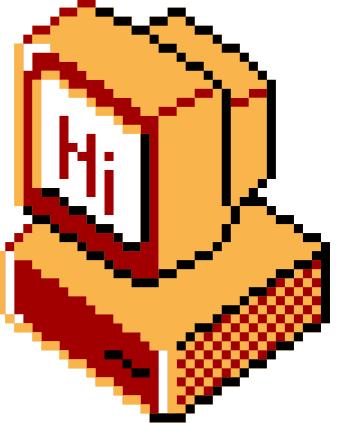
Today's Presenters

- Riley (toasterpwn) - Captain
- Rainier (teddy / TheSavageTeddy) - Vice Captain
- Torry (torry2)
- Orlando (q3stlon)
- Avery (nullableVoidPtr)



Emu Exploit at Pecan CTF 2023

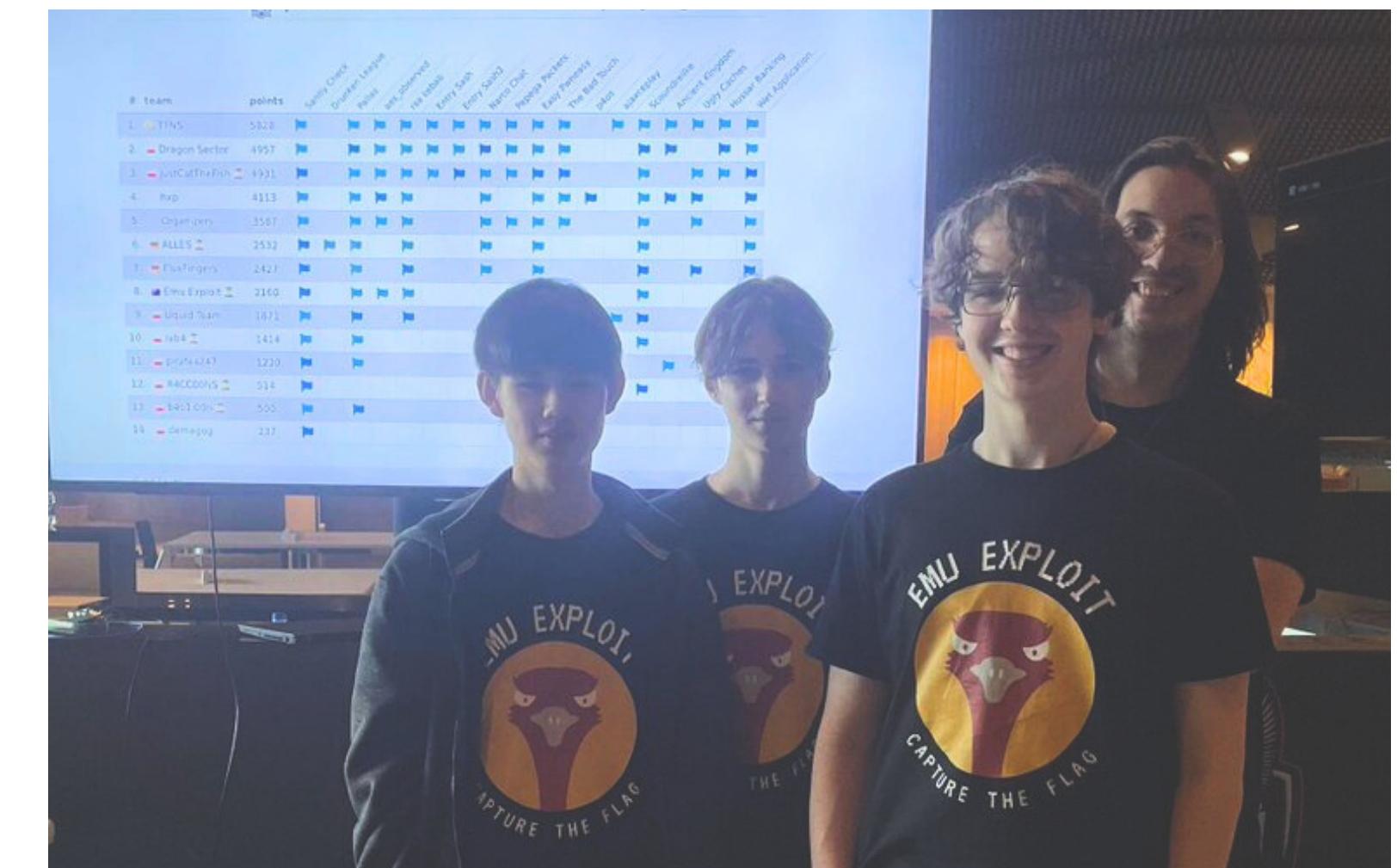
\$ \$/: whoami



Pecan CTF 2023



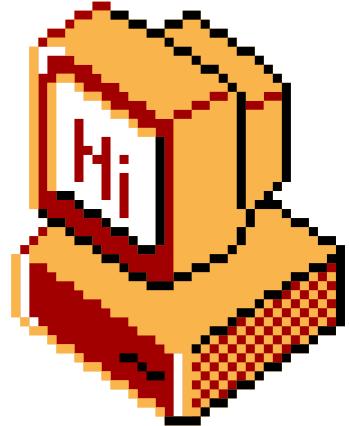
WACTF 0x05



p4CTF in Katowice, Poland

Perth Socialware 0x03

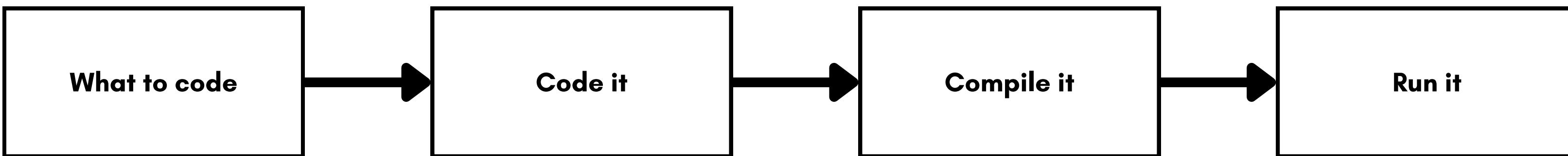
\$ n/: reverse engineering



First of all, what is **reverse engineering**?

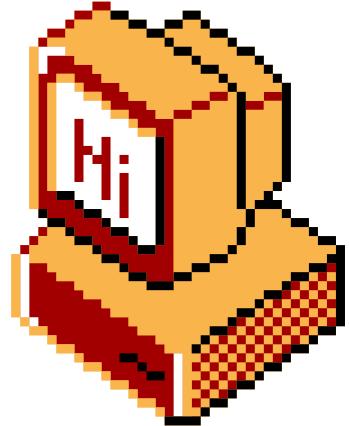
Consider the process of building a program:

- You figure out **what** you want to code
- You **implement** it in code
- You **compile** the code
- You **run** the code



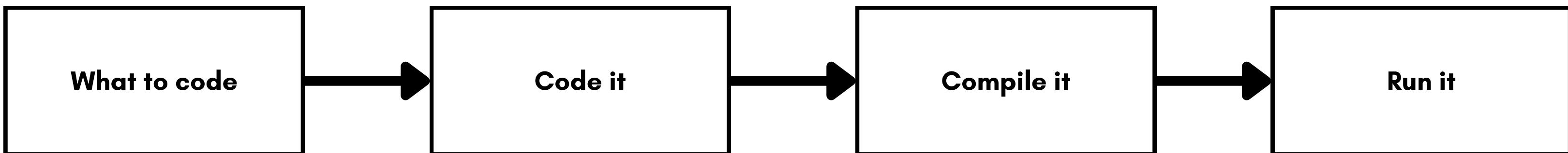
Information is lost at every stage!

\$ \wedge/: reverse engineering

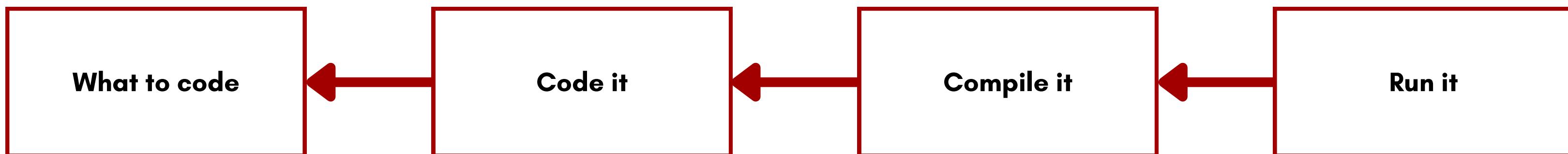


How can we get back the information that was lost?

This is what reverse engineering is!



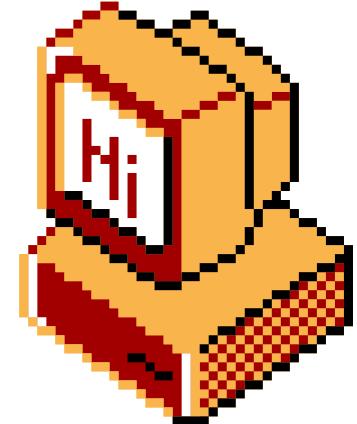
Information is lost at every stage!



How can we get it back?

\$./cat content

Presentation [6:00] -> Workshop [6:30] -> End [8:00]



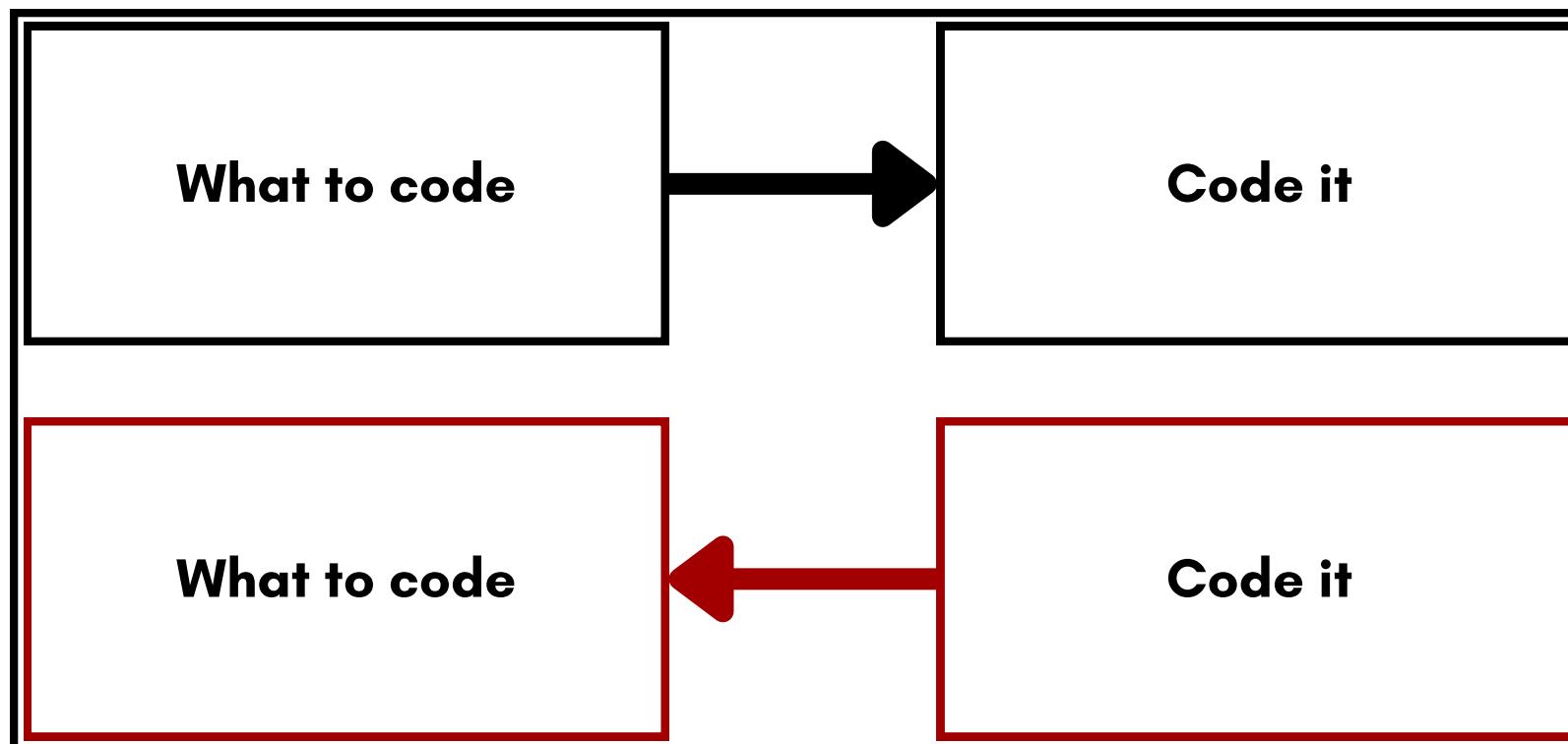
Art of Reverse Engineering

The C Programming Language

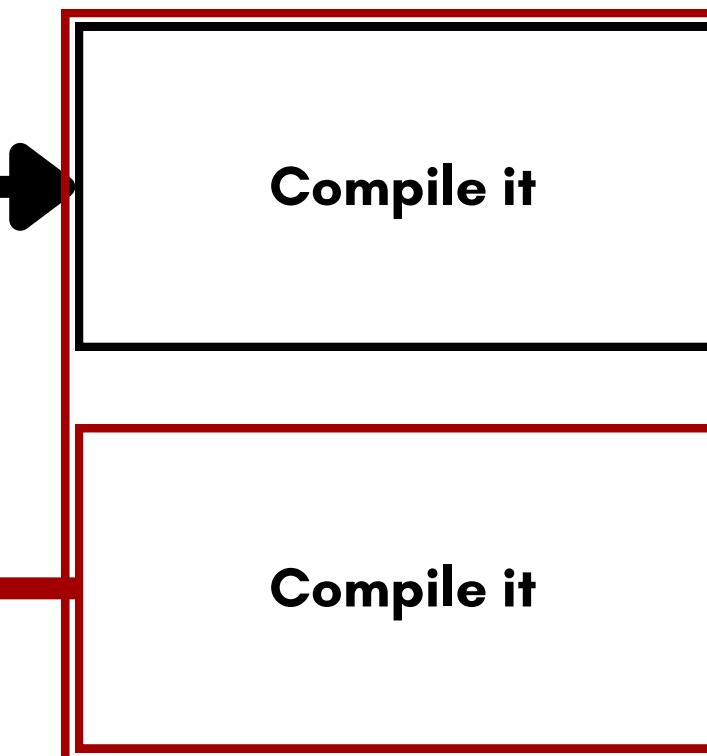
Decompilation

Static Analysis

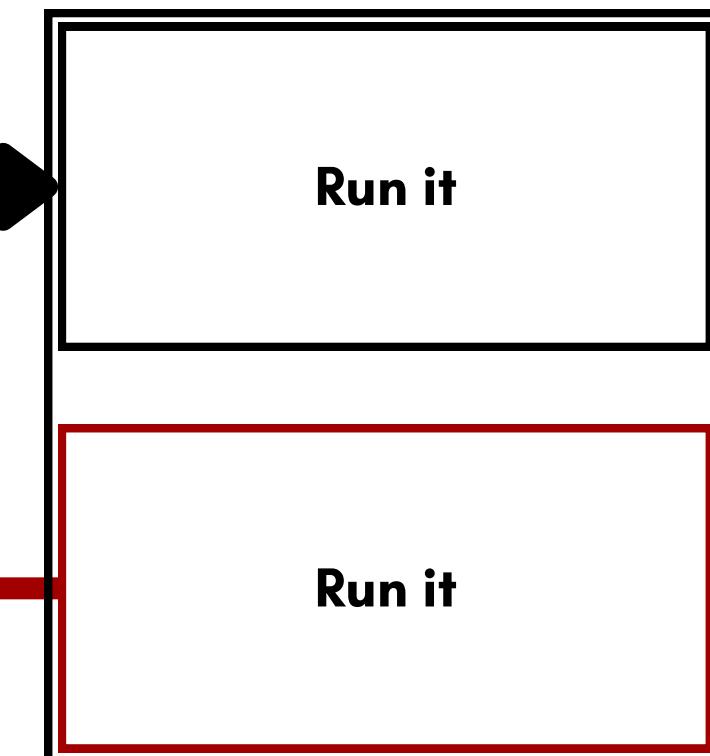
Normal Development
(you do this already)



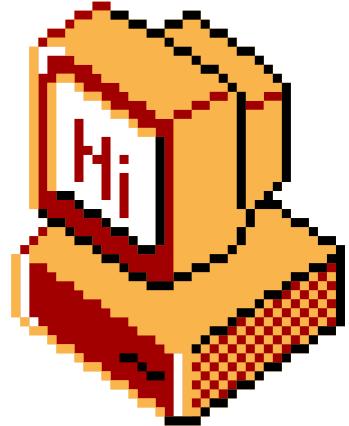
**Intro To Rev
Part 2**



**Intro To Rev
Part 1**



\$ n/: The C Programming Language



- General Purpose Programming Language
 - can be initially intimidating
- Statically Typed & Compiled
- Used for low level systems & applications
- Influential in computing and development
- E.g Used In: Operating systems, drivers and applications
- Understanding of computer memory is helpful to learn C
- Understanding of C is helpful to reverse a variety of applications



\$./c - Syntax

Syntax in C:

- Lines delimited by ";" semicolins;
- Comments are // for single line or /* multi line */

Include Statements: "#include <library>"

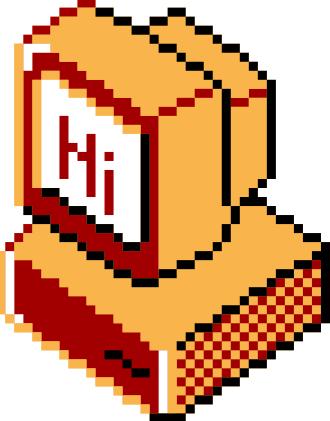
Declarations: "<datatype> <name> <operator> <value>"

Keywords: "for", "if", "const", "return"

Operators: "+", "-", "*", "/", "==", "&", "|" and more...

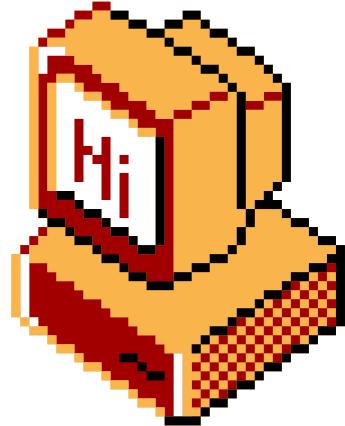
Functions are defined with "<returntype> <name>()" and contents are wrapped in "{}"

- a "main" function is **always** the starting point for a C program
- Reading C becomes intuative



```
● ● ●  
error: expected ';' before 'return'  
5 | puts("Hello, World!")
```

\$./c - Data Types



- Example Data Types

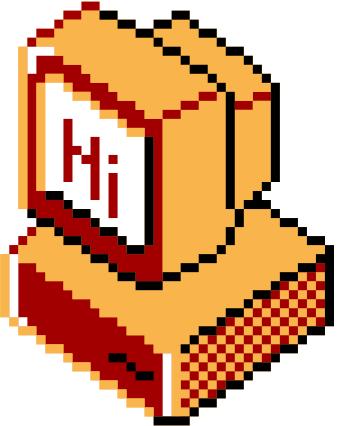
int	2 or 4 bytes	Whole Numbers
float	4 bytes	Numbers with decimals
char	1 byte	Single value (e.g ASCII character)
struct		Collection of elements of different data types

- Specifies the size and type of information to be stored

https://www.w3schools.com/c/c_data_types.php

- Data types can be “casted” for conversion
 - This is done via (<type name>) <expression>

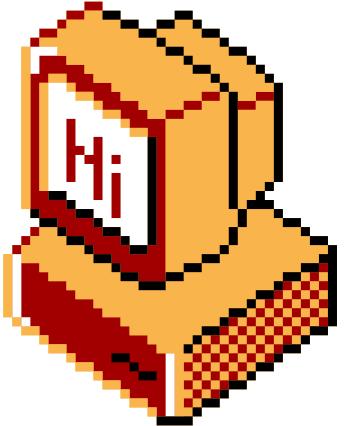
\$./c - Control Flow



- Keywords used to define flows
- Wrapped in {} similar to functions
- if / else
- for loop
- while loop
- break / continue
- switch / case

```
● ● ●  
if (condition){  
    something  
}  
  
while (condition){  
    something  
}  
  
for (int i = 0; i < 1337; i++){  
    something iteration  
}
```

\$./c - Common Pitfalls



Return Values:

- Functions in C expect to be returned to a value
- e.g `int main() {}` is the main function expecting a return value of type `int`

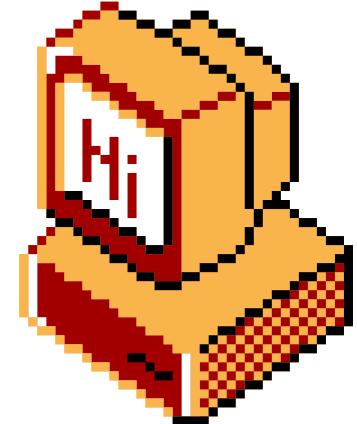
Char Arrays:

- Strings in C are “char arrays”
- This is an Array of characters that make up the string, these arrays end in a “null byte” to terminate the string
- e.g `char string[] = “example”;`

Indexing:

- Indexing arrays and similar are counted from 0

\$./c - Common Pitfalls



Pointers:

- Denoted by “*” character (to create and dereference)
- Pointers are a variable storing the memory address of another variable, denoted by
- Commonly seen as a difficult concept however quite simple
- E.g point to variable “foo” is the value of “foo”s memory address, plenty of googlable resources explain it well

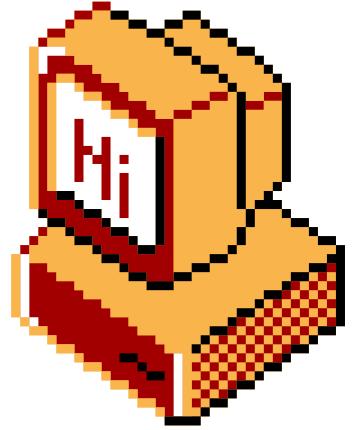
```
● ● ●

int number = 1337;      // Variable
int* ptr = &number;    // Pointer

printf("%p\n", ptr); // (0x7ffe5367e044)

printf("%d\n", *ptr); // 1337
```

\$./ "Hello World" - C <-> ASM



"Hello World" in C vs. Assembly

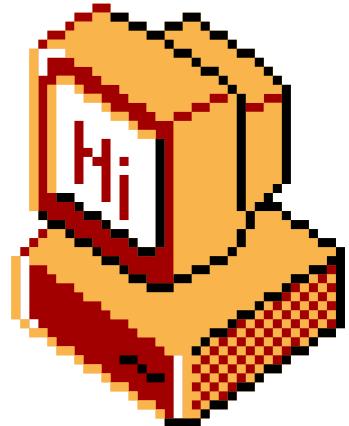
```
#include <stdio.h>

int main()
{
    puts("Hello, World!")
    return 0;
}
```

```
// hello.s - gcc hello.c -O0 -masm=intel -S

.file  "hello.c"
.intel_syntax noprefix
.text
.section      .rodata
.LC0:
.string "Hello, World!"
.text
.globl main
.type   main, @function
main:
.LFB0:
.cfi_startproc
push   rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
mov    rbp, rsp
.cfi_def_cfa_register 6
lea    rax, .LC0[rip]
mov    rdi, rax
call   puts@PLT
mov    eax, 0
pop    rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size  main, .-main
.ident "GCC: (Debian 13.2.0-2) 13.2.0"
.section      .note.GNU-stack,"",@progbits
```

\$ \|: Static Analysis



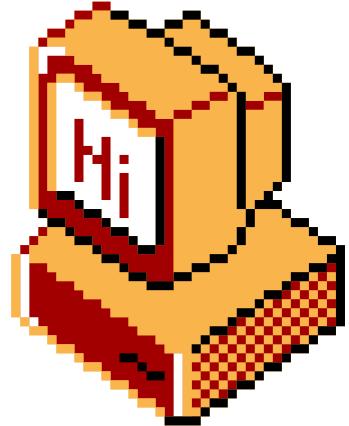
Reverse-engineering through methods of examining available code or binaries *without* executing it.

- Methods: Disassembly & Decompilation
- Techniques: Annotation of Types and Functions

The screenshot shows a static analysis tool interface with several windows:

- Program Trees**: Shows a tree view of symbols including `MAIN_uninitialized1`, `MAIN_uninitialized2`, `StaticR_text0`, `StaticR_data0`, `StaticR_data1`, `StaticR_data2`, `StaticR_data3`, and `StaticR_uninitialized0`.
- Listing: StaticR.rel**: Displays assembly code for functions like `LAB_80006324` and `LAB_8000632c`. The assembly includes instructions like `lwz r3=>DAT_802a4080`, `beq r4,0x0 LAB_80006324`, and `addi r29,r29,0x8 LAB_80006308`.
- Decompile: FUN_800062a0 - (StaticR.r...**: Shows the decompiled C-like code corresponding to the assembly, including loops and memory operations.
- Symbol Tree**: Shows imports, exports, and function symbols for `FUN_800...` and `FUN_805...`.
- Data Type Manager**: Shows data types like `BuiltinTypes`, `StaticR.rel`, and `generic_clib`.
- Console - Scripting**: A command-line interface for scripting.

\$ n/: Disassemblers



- Reading programs as a human is tedious
 - Manually decode the instructions from the binary
 - Keep track of which pointers target where
 - Identifying and documenting where structs are used
 - Naming and documenting functions and different blocks of code
- Use compilers for forward-engineering; disassemblers for reverse-engineering
 - What about **decompilers**?

\$ n/: Disassemblers



IDA Pro
by Hex Rays

US\$365 USD for base version



Binary Ninja
by Vector35

Free for Cloud
US\$300 for Full Ver.



Ghidra
by the National Security Agency

Free (and Open Source!)

\$ N/: Disassemblers



IDA Pro
by Hex Rays

US\$365 USD for base version
Upwards of \$10000 for Pro
version with all features



Binary Ninja
by Vector35

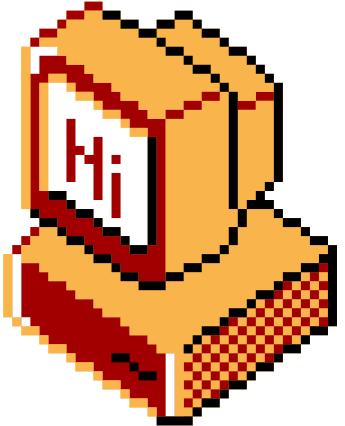
Free for Cloud
US\$300 for Full Ver.



Ghidra
by the National Security Agency

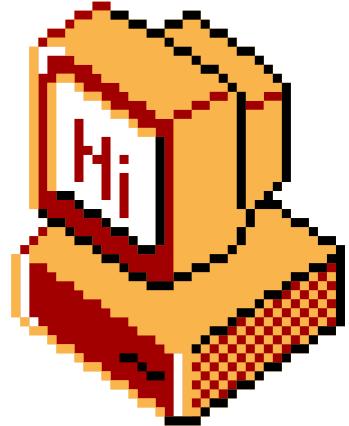
Free (and Open Source!)

\$ \/: Decompilation



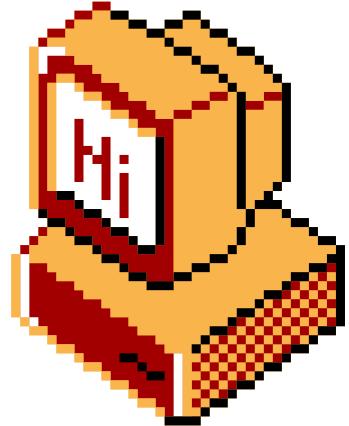
- Compilers have to use an assembler internally
- Decompilers need a disassembly, from a disassembler
- IDA Pro has a *really* good decompiler
 - (+) Fast! Reliable! Concise!
 - (-) Expensive!
- Both Binary Ninja and Ghidra have (okay) decompilers as well
- Ghidra and Binary Ninja Demo version is free...

\$ \/: Decomp Disclaimers



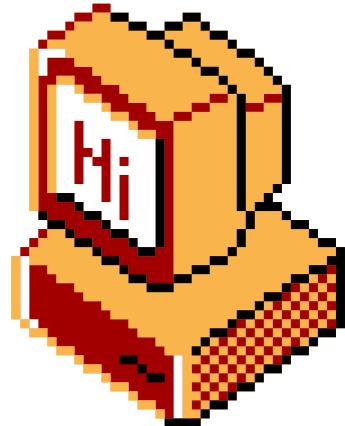
- Decompilation is simply not accurate
 - You still need to annotate
 - Do not entirely rely on decompilation for all of your reverse-engineering
 - Information is not recovered perfectly and can have some parts missing
 - Entire lines of code can be lost to optimisations done in compiling the original code
 - There can be cases where a decompiler will crash when tried on a file, either done deliberately or not
- While a lot more to read and can appear more confusing, every disassembly is more accurate than its decompilation as its a “direct translation” rather than a “best guess”.

\$ \|/: Tooling



- All mentioned tools have different features and quirks
- **There is no best tool for this**
 - Different programs, architectures, compilers
- Down to personal preference
- IDA, Binary Ninja and Ghidra all have scripting capabilities
 - Develop your own scripts!

\$ \/: Getting started



Binary Ninja Demo

BINARYNINJA

Welcome to the Binary Ninja demo.

This demo version supports disassembly of x86, x64 and ARMv7 binaries for a variety of platforms. Additional architectures are available in the full release. See the [list of features](#) for more information.

Purchase Binary Ninja to unlock all features. Product comparisons are available on the purchase page.

Visit [Binary Ninja](#) and [Vector 35](#) on the web.

Start

- Start & Main Window

Binary Ninja Demo 3.4.4271 demo

File Edit View Analysis Debugger Plugins Window Help

New Tab

BINARYNINJA

Thank you for trying Binary Ninja.

This demo version supports disassembly of x86, x64 and ARMv7 binaries for a variety of platforms. Additional architectures are available in the full release. See the [list of features](#) for more information.

Questions about Binary Ninja? First check the [frequently asked questions](#) page. You can also join our [Slack](#) to interact with us and our community. See the [user documentation](#) to learn more about how to use Binary Ninja.

Purchase Binary Ninja to unlock all features. Product comparisons are available on the purchase page.

Recent Files

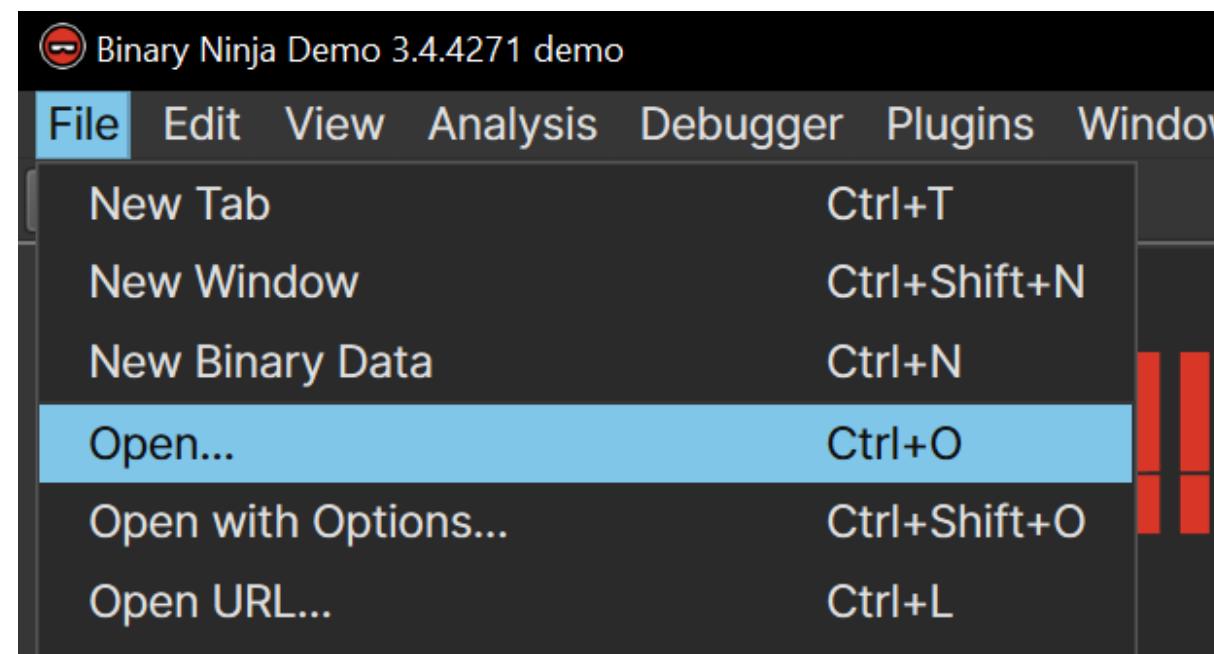
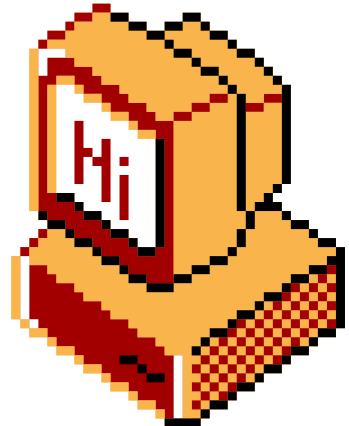
1: C:\Users\torry\Downloads\simple_encrypt.exe

Open... Open an existing file.
Options... Open an existing file with custom options.
New Create a new binary file.

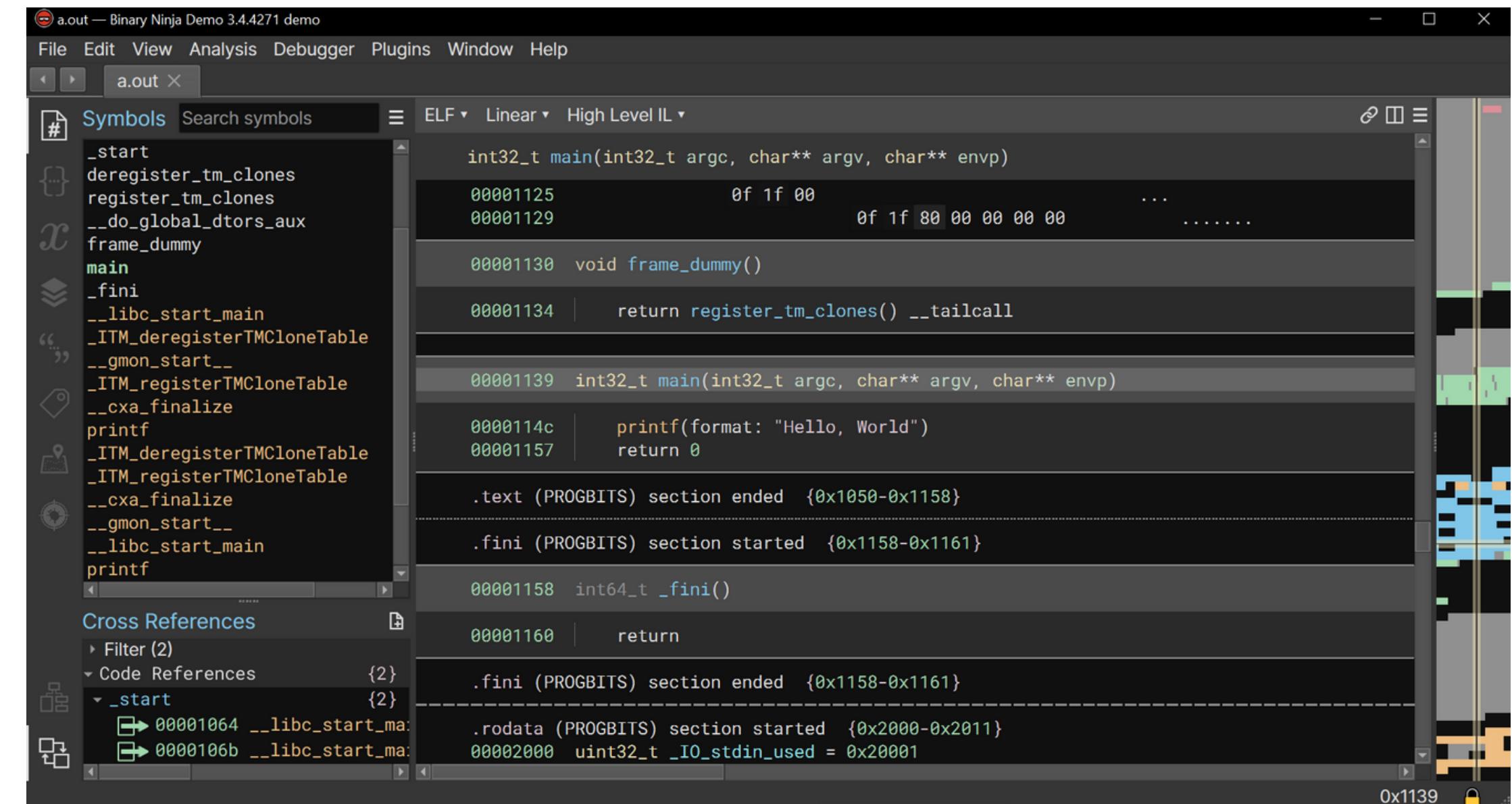
DEMO VERSION Version 3.4.4271 demo, Build ID b7fd028d

Copyright © 2015-2023 Vector 35 Inc

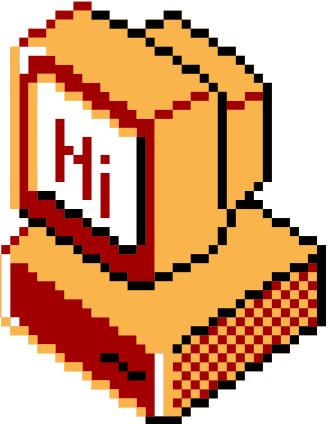
\$ n/: Getting started



- File -> Open (Ctr+O)
 - LoadBinary & View



\$ \/: Getting started



The screenshot shows the Binary Ninja interface with the following numbered callouts:

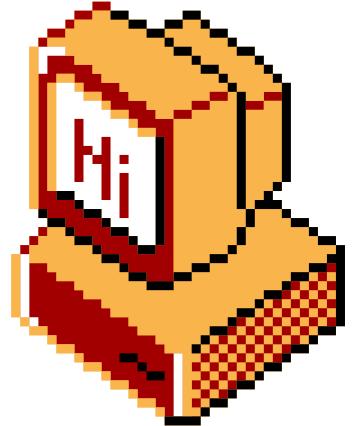
- 1 Points to the **Symbols** panel on the left.
- 2 Points to the **Registers** panel on the left.
- 3 Points to the **Cross References** panel on the left.
- 4 Points to the **High Level IL** view on the right, which displays assembly-like code for the `main` function.
- 5 Points to the **File** menu at the top.

The IL code listing in the main window includes:

```
int32_t main(int32_t argc, char** argv, char** envp)
00001125          0f 1f 00
00001129          0f 1f 80 00 00 00 00 ...
00001130  void frame_dummy()
00001134    | return register_tm_clones() __tailcall
00001139  int32_t main(int32_t argc, char** argv, char** envp)
0000114c    | printf(format: "Hello, World")
00001157    | return 0
.text (PROGBITS) section ended {0x1050-0x1158}
.fin (PROGBITS) section started {0x1158-0x1161}
00001158  int64_t _fini()
00001160    | return
.fin (PROGBITS) section ended {0x1158-0x1161}
.rod (PROGBITS) section started {0x2000-0x2011}
00002000  uint32_t _IO_stdin_used = 0x20001
```

1. Context Menus
2. Menu
3. Cross References
4. Window
5. Selection

`^/: pause`



Workshop/Networking will now commence!

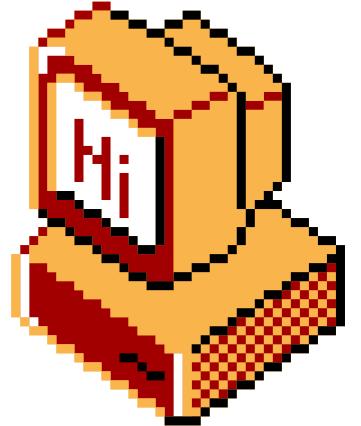
Filedrop! Find the exercise and challenge files here:

- <https://emu.team/filedrop>
- 2 Exercises +crackme challenge ! (solutions soon)

Download “Binary Ninja”: (cross platform)

- <https://binary.ninja/demo/>

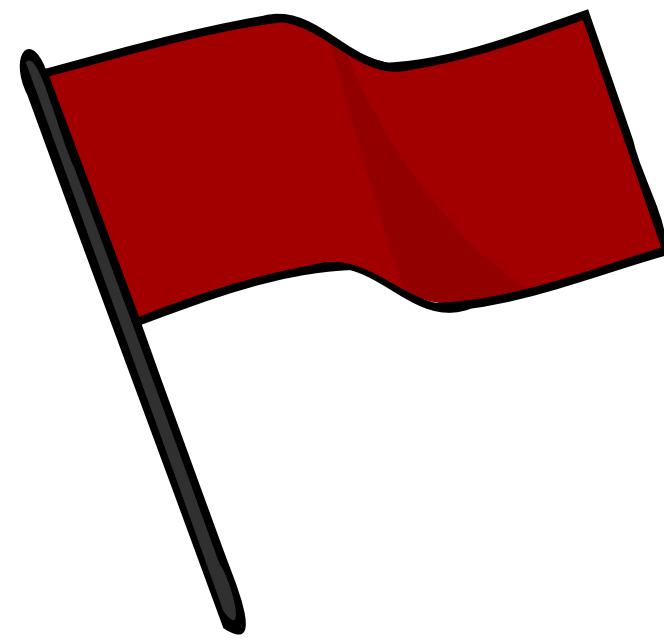
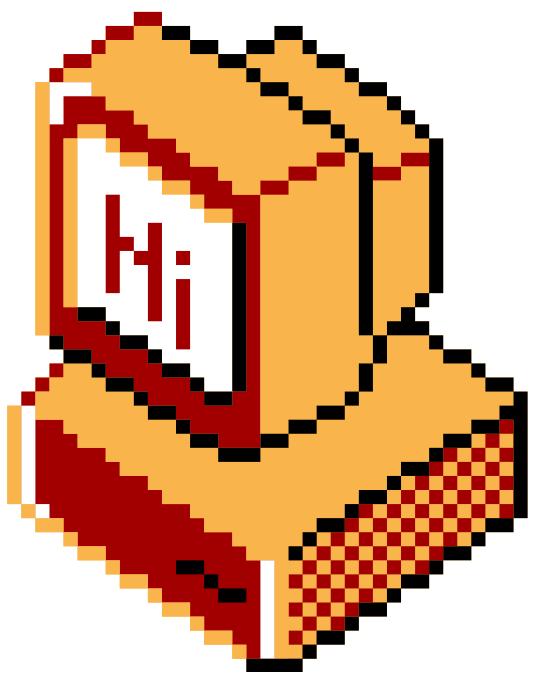
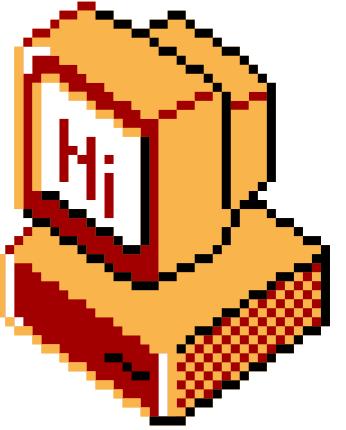
`/*/: Compiling C online`



If you cannot compile C code on your machine, use this website:
<https://cplayground.com/>

Online decompiler (may give better results than Binary Ninja):
<https://dogbolt.org/>

\$ n/: questions



Questions!

`^/: shutdown`

Thank you!

