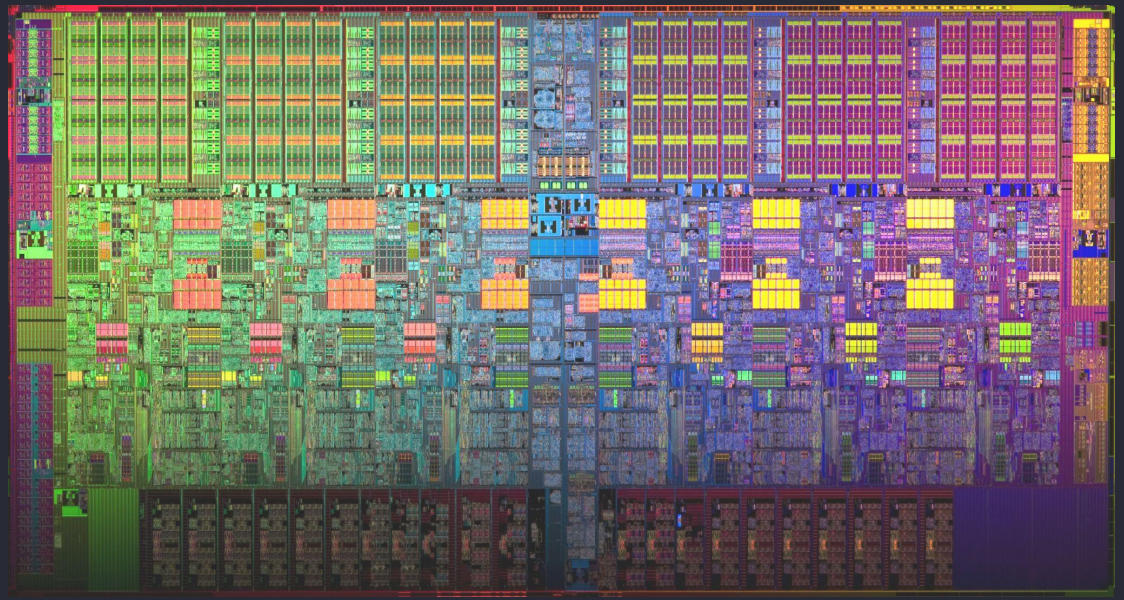
Binary Reversing

+ a little bit of other stuff

Daniel Parks

Review: How CPUs work



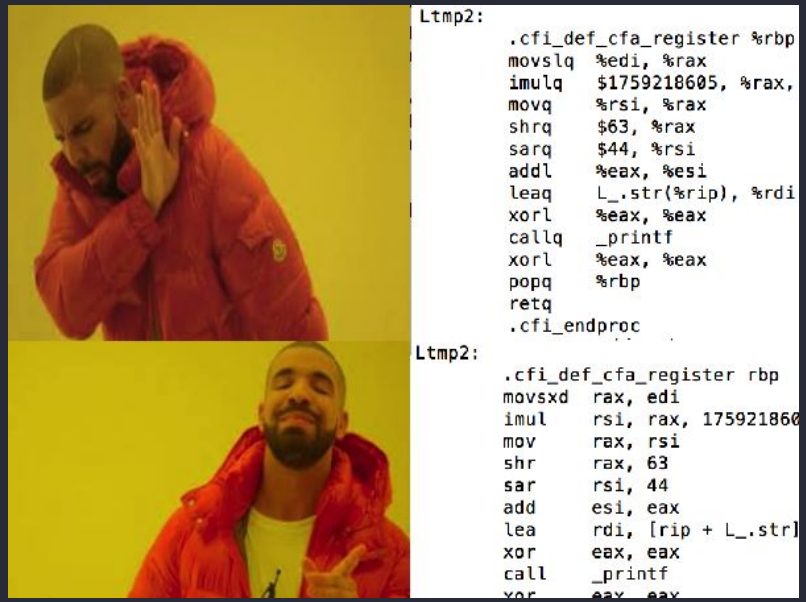
drhart http://drhart.ucoz.com/index/40_years_micro_chip/0-177

CPU basic components

- Registers
- ALU
- Memory (+cache)
- Interrupt handling

- `mov rdi, 10`
- `add rsi, rdi`
- `mov r8, qword ptr [rdi]`
- `syscall / int 0x80`

x86 is annoying



```
Ltmp2:
        .cfi_def_cfa_register %rbp
        movslq  %edi, %rax
        imulq   $1759218605, %rax,
        movq    %rsi, %rax
        shrq    $63, %rax
        sarq    $44, %rsi
        addl    %eax, %esi
        leaq    L_.str(%rip), %rdi
        xorl    %eax, %eax
        callq   _printf
        xorl    %eax, %eax
        popq    %rbp
        retq
        .cfi_endproc
```

```
Ltmp2:
        .cfi_def_cfa_register rbp
        movsxd  rax, edi
        imul    rsi, rax, 175921860
        mov     rax, rsi
        shr     rax, 63
        sar     rsi, 44
        add     esi, eax
        lea     rdi, [rip + L_.str]
        xor     eax, eax
        call    _printf
        xor     eax, eax
```

Obligatory meme credit: https://www.reddit.com/r/ProgrammerHumor/comments/56fjm5/att_vs_intel_syntax/

Sidenote: CISC vs RISC

- `add r8d, dword ptr [rdi]`
- `mov dword ptr [rsi], r8d`

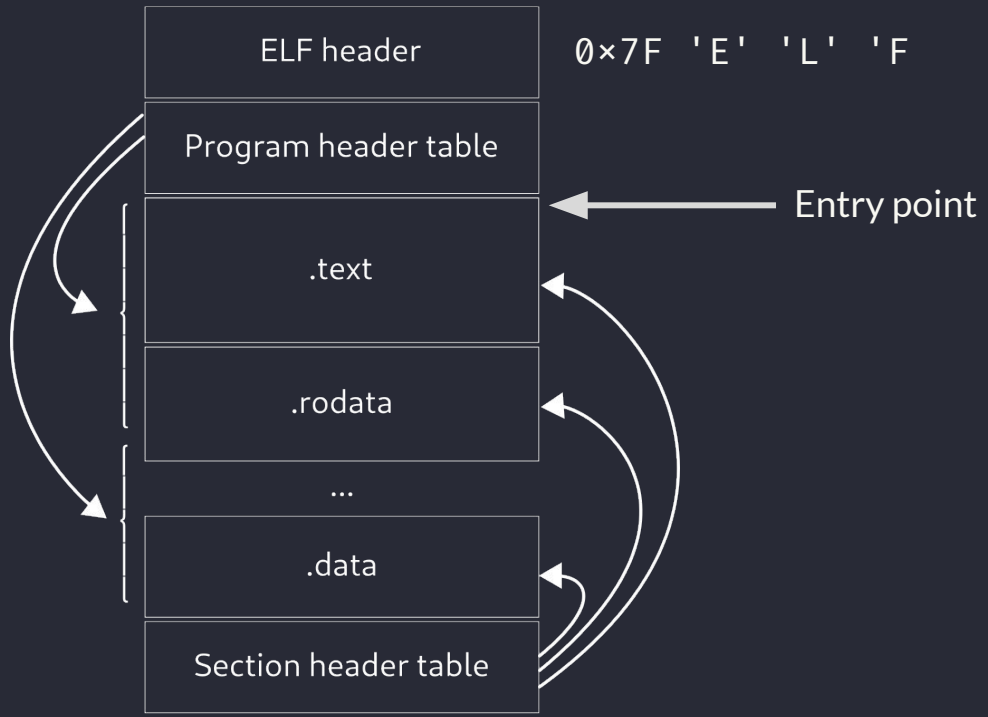
- `lw t1, (a0)`
- `add t0, t0, t1`
- `sw t0, (a1)`

Assembly is not *exactly* human-readable

```
xor    ebp,0×10bd6634
mov    eax,esi
xor    edx,edx
div    ebp
mov    esi,edx
call   <exit@plt+0×b630>
imul   edi,DWORD PTR [eax*4+0×1db80],0×891dabe9
call   <exit@plt+0×b650>
imul   ebp,DWORD PTR [eax*4+0×1db8c],0×9c903052
add    ebp,edi
call   <exit@plt+0×b670>
add    ebp,DWORD PTR [eax*4+0×1db98]
cmp    esi,ebp
sete   bh
call   <exit@plt+0×b690>
xor    bh,BYTE PTR [eax+0×1dba4]
and    bh,bl
call   <exit@plt+0×b6b0>
cmp    bh,BYTE PTR [eax+0×1dba7]
je     <exit@plt+0×2139>
call   <exit@plt+0×2065>
mov    eax,DWORD PTR [eax]
mov    DWORD PTR [esp+0×54],eax
mov    edi,DWORD PTR ds:0×2f104
mov    esi,DWORD PTR ds:0×2f208
call   <exit@plt+0×b6d0>
imul   ebx,DWORD PTR [eax*4+0×1dbac],0×c229bd50
call   <exit@plt+0×b6f0>
mov    eax,DWORD PTR [eax*4+0×1dbb8]
add    eax,ebx
add    eax,0×593a9c34
cmp    esi,eax
setl   BYTE PTR [esp+0×20]
call   <exit@plt+0×b710>
mov    eax,DWORD PTR [eax*4+0×1dbc4]
neg    eax
```

What is a binary?

Executable and Linkable Format



Symbols and Dynamic Linking

```
daniel@archparks2 ~/Downloads
$ nm AAAAAAAAAAAAAAAA
0000000000404038 B __bss_start
0000000000404038 b completed.8060
0000000000404028 D __data_start
0000000000404028 W data_start
00000000004010b0 t deregister_tm_clones
00000000004010a0 T _dl_relocate_static_pie
0000000000401120 t __do_global_dtors_aux
0000000000403e18 d __do_global_dtors_aux_fini_array_entry
0000000000404030 D __dso_handle
0000000000403e20 d _DYNAMIC
0000000000404038 D _edata
0000000000404040 B _end
                 U execve@@GLIBC_2.2.5
0000000000401248 T _fini
0000000000401150 t frame_dummy
0000000000403e10 d __frame_dummy_init_array_entry
0000000000402174 r __FRAME_END__
000000000040118e T get_flag
                 U gets@@GLIBC_2.2.5
0000000000404000 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
000000000040200c r __GNU_EH_FRAME_HDR
0000000000401000 T _init
```

Image credit: me lol

Elf files are architecture-specific

Exec format error

Mon Apr 24, 2017 6:21 am

Hi, I am using raspberry pi 2 board for my project development. Currently i am doing projects on Linux raspberrypi 4.4.50-v7+ armv7l GNU/Linux.

I installed ARToolkit5 as per steps given by artoolkit.org. I am try to run sample programs given SDK but i am getting following error.

bash: /simpleLite: cannot execute binary file: exec format error

Can u help me

Regards,
Mitul

mitulshinde

Posts: 1
Joined: Mon Apr 24, 2017 5:52 am

Re: Exec format error

Mon Apr 24, 2017 9:43 am

This error is almost always caused by trying to run x86 binaries on the Pi.

It boils down to: You can't install arbitrary Linux binaries (which will almost certainly have been compiled for x86) on the Pi. You either have to install it from one of the known Pi-compatible binary repositories (what you have by default in your sources.list file(s)) -- or to compile the program from source.

If this post appears in the wrong forums category, my apologies.

Martin Frezman

Posts: 1009
Joined: Mon Oct 31, 2016 10:05 am

So, what do you do with a binary?

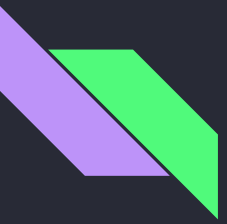
Tools: strings

```
Usage: strings [option(s)] [file(s)]

 Display printable strings in [file(s)] (stdin by default)
```

(also: grep --text)

Tools: file

```
$ file AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA: ELF 64-bit LSB executable, x86-64, version 1
(SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, not stripped

$ file /usr/bin/qemu-aarch64-static
/usr/bin/qemu-aarch64-static: ELF 64-bit LSB executable, x86-64,
version 1 (GNU/Linux), statically linked, for GNU/Linux 3.2.0,
stripped

$ file ~/code/netsec/rlbox/app
/home/daniel/code/netsec/rlbox/app: ELF 64-bit LSB executable, UCB
RISC-V, RVC, double-float ABI, version 1 (SYSV), dynamically
linked, interpreter /libexec/ld-elf.so.1, for FreeBSD 14.0
(1400051), FreeBSD-style, with debug_info, not stripped
```

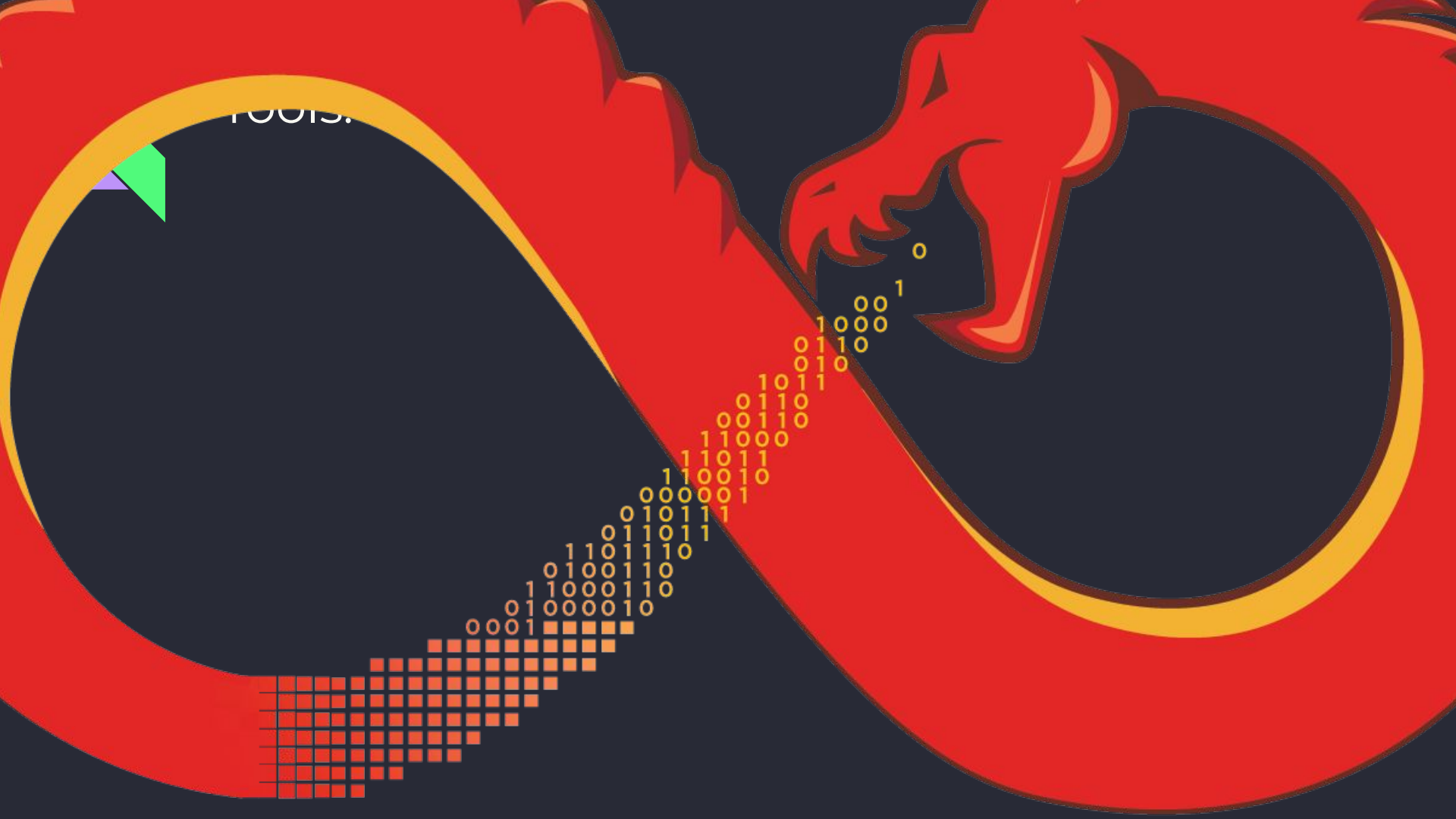


Tools: nm

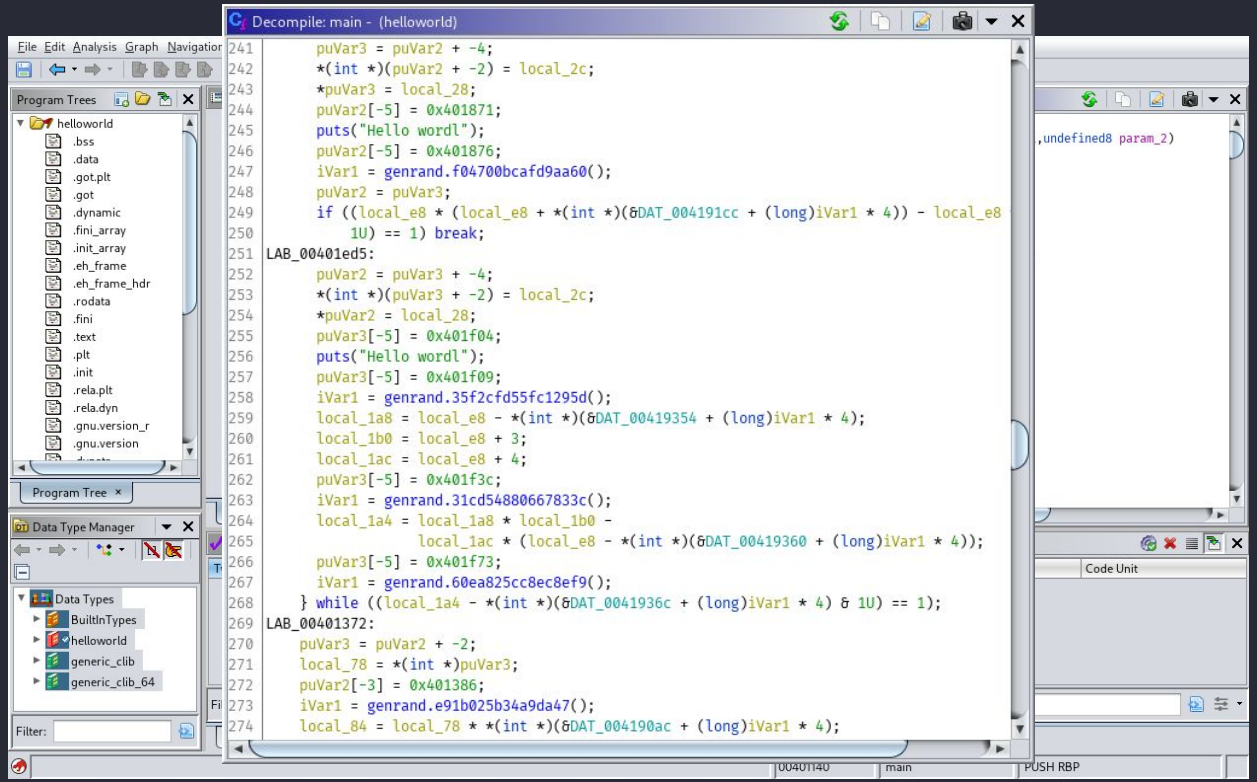
```
Usage: nm [option(s)] [file(s)]

 List symbols in file(s).
```

Tools: objdump (old-school disassembling)

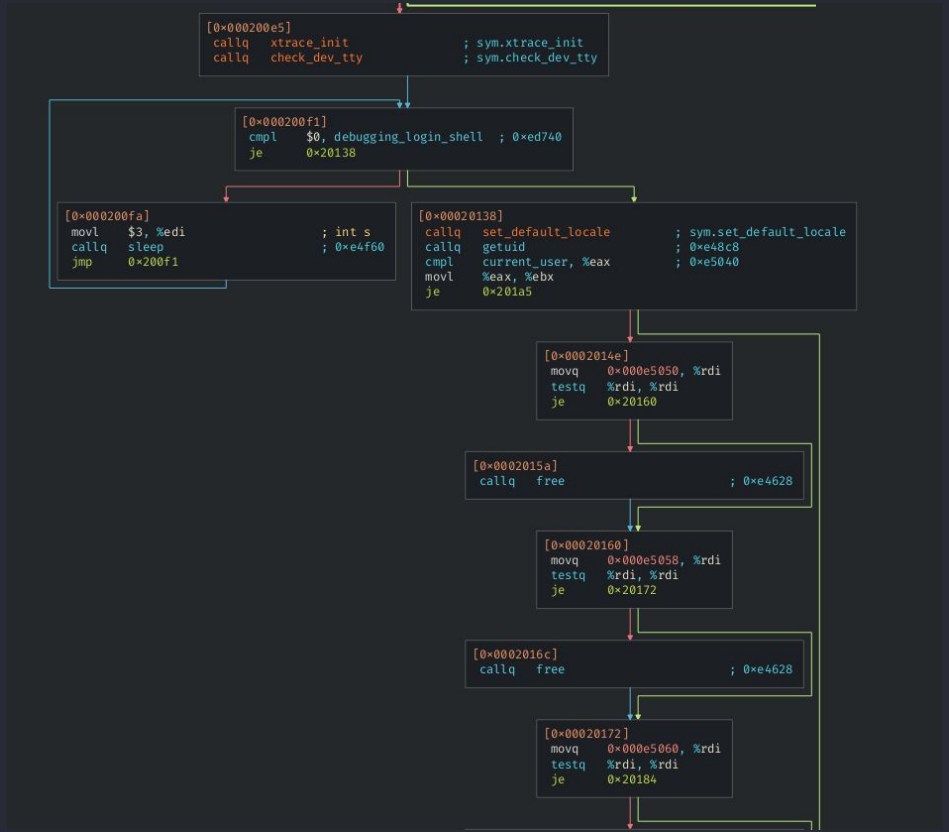
```
daniel@archparks2 ~/Downloads
$ objdump -h /bin/bash

/bin/bash:     file format elf64-x86-64

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .interp       0000001c  0000000000000318  0000000000000318  00000318  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.gnu.property 00000040  0000000000000338  0000000000000338  00000338  2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .note.gnu.build-id 00000024  0000000000000378  0000000000000378  00000378  2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .note.ABI-tag 00000020  000000000000039c  000000000000039c  0000039c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .gnu.hash     000031d8  00000000000003c0  00000000000003c0  000003c0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .dynsym       0000b7d8  0000000000003598  0000000000003598  00003598  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .dynstr       0000742e  000000000000ed70  000000000000ed70  0000ed70  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .gnu.version  00000f52  000000000001619e  000000000001619e  0001619e  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .gnu.version_r 000000d0  00000000000170f0  00000000000170f0  000170f0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  9 .rela.dyn     00008ad8  00000000000171c0  00000000000171c0  000171c0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 10 .init         0000001b  0000000000020000  0000000000020000  00020000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 11 .text         000929e5  0000000000020020  0000000000020020  00020020  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 12 .fini         0000000d  00000000000b2a08  00000000000b2a08  000b2a08  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 13 .rodata       00017f80  00000000000b3000  00000000000b3000  000b3000  2**5
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 14 .eh_frame_hdr 000032dc  00000000000caf80  00000000000caf80  000caf80  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 15 .eh_frame     00013070  00000000000ce260  00000000000ce260  000ce260  2**3
```



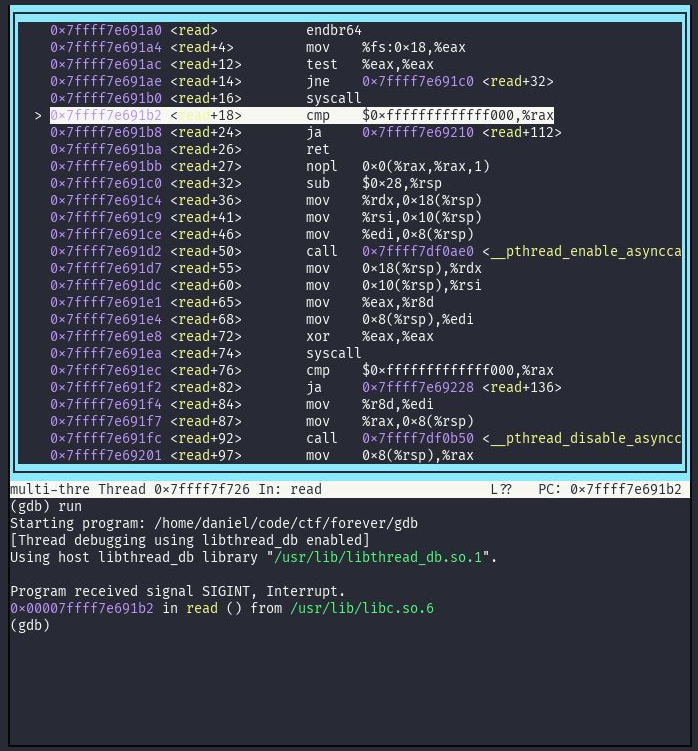
Tools: Ghidra



Tools: cutter (/ iaito / whatever)



Tools: gdb



Basic command overview:

`layout <src|asm>` - enter TUI mode
`C-x o` - switch between windows in TUI mode
`tui disable` - exit TUI mode
`break [address|file.c:#|symbol]` - breakpoint
`commands` - run commands on breakpoint
`run [args...]` - start program [with arguments]
`start` - put a breakpoint on main and run
`next` - go to next line
`step` - go to next line, stepping into functions
`nexti/stepi` - like above but with instructions
`continue` - resume program
`kill` - die, bad program!
`help` - learn more commands

Tools: gdb: Printing & Examining

```
print[/format]
[address|variable|$register]

x[/[count]format]
[address|variable|$register]
```

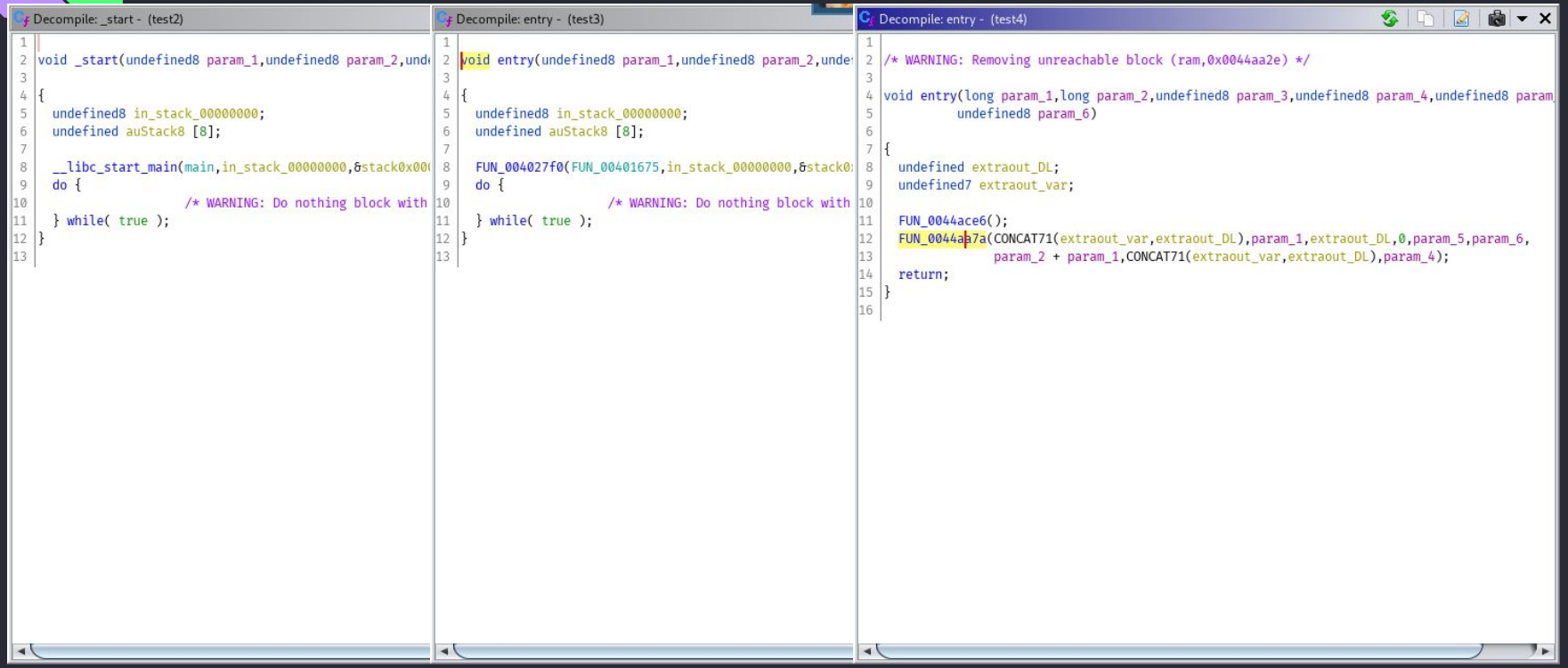
```
x - hexadecimal

d/u - signed/unsigned decimal

t - binary

c - character

f - float
```

There are more of these!
See `help data`

Other tools

- Binwalk - like file but it looks everywhere in the file for file signatures. Useful for inspecting firmware images
- Any hex editor (although some support structs which can be very helpful!)
- readelf - parses and displays misc. elf info
- upx - can create and extract self-extracting binaries
- angr - programmatic reversing on steroids

Reversing Tips and Tricks: `__libc_start_main`

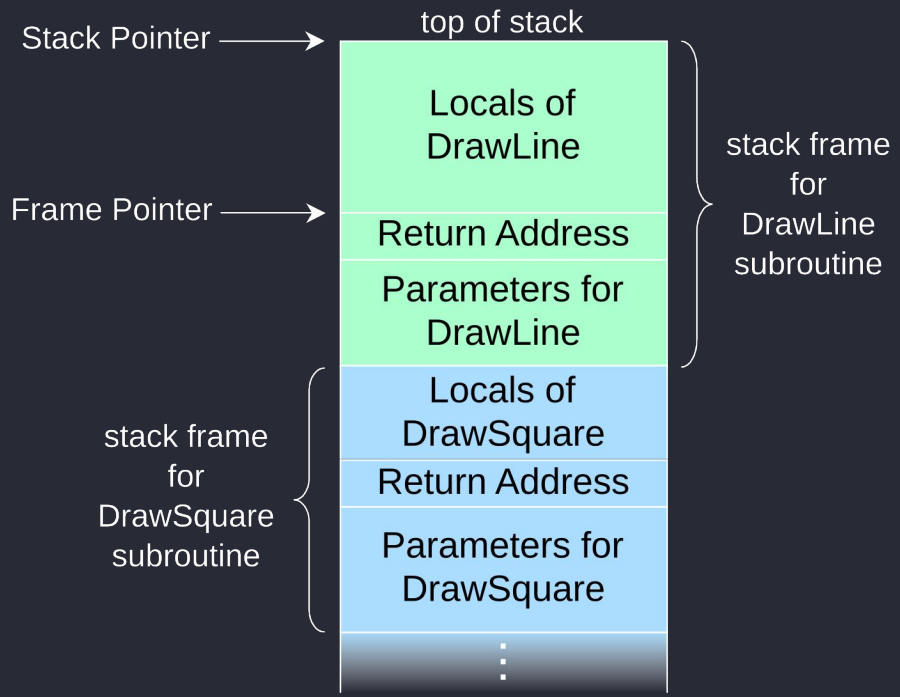


Reversing Tips and Tricks: Stack Protector

Ignore this!

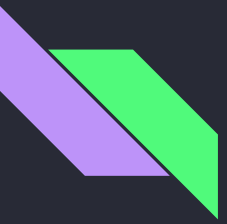
```
if (local_40 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
```

Reversing Tips and Tricks: Frame Pointer vs. No Frame Pointer



Reversing Tips and Tricks: Ghidra has a Debugger

I can’t vouch for it but you should probably try it



Reversing Tips and Tricks: Cutter supports the Ghidra decompiler

I can vouch for it and you should try it

Good if you need to do some low-level analysis and the CFGs would help

Reversing Tips and Tricks: Take Advantage of Ghidra

- Rename variables
- Retype variables
- Create structs
- Reformat literals

What is deobfuscation?

What is *ob*fuscation?

Deobfuscation Tips and Tricks: Googleable patterns

```
z="

";Nz='ORAN';Vz='="$(';ez='CYAN';DCz='\e[9';fBz='F**';NCz='writ';pz=' """;dz='e[35';rz='HITE';XCz='e 10';XBz='
▄";iBz='ch. ';jCz=' Com';Hz='N="$';SBz=' ▙▄▄';rCz='L 10';VBz='▟▌ ▐';YBz='espe';UCz='p 0.';rBz='ous
';kz='K="$';EBz='▌ ▙';hz='E="$';nCz='0\e[';DDz=' :|:';jz='BLAC';hBz=' bit';ZCz='[93m';LBz='▟ ▐';kBz='
mes';SCz=' 100';kCz='mand';tCz='e[93';lCz=' cod';RCz=' -qL';vCz='ast ';Ez='e[31';OBz=' ▐▀▙';xBz='"
';Sz='[33m';wz='  ▗▟';pBz='e an';WBz='▀▀  ';Rz=' \''\e';eCz=' bla';xCz='whil';hCz=' sec';MBz='▚
';CDz=':(){';oz='echo';sBz='" | ';Qz='intf';UBz=' ▙▟';dCz='will';bBz='ven ';PBz='    ▐';MCz='over';QCz='|
pv';fz='6m\'')';nz='r';vz='▓▓ ';Yz='\e[3';gBz=' you';TBz=' ▟▙▄';CCz='91m~';oCz='93m"';ACz='[92m';sz='
';FBz='▌    ';dBz='0 -k';BCz='[\e[';Zz='4m\'')';yz='▀";yCz='e tr';tz='▙▟ ';Pz='$(pr';DBz='▒
▙▄';bz='MAGE';qBz='onym';Fz='m\'')"';OCz='e\e[';QBz=' ▛▒';Jz='ntf ';ICz='omma';RBz=' ▟▟';vBz='5
&';KCz='unct';jBz='Dont';eBz='20 "';IBz='  ▐';JCz='nd f';tBz='pv -';gz='WHIT';Wz='prin';lBz='s
wi';FCz='e[0m';iCz=' !!\';sCz='0';ABz=' "$R';cz='NTA=';VCz='5';uCz='m Bl';HBz='▙▄";Xz='tf \''';nBz='s.
w';Cz='rint';fCz='st i';LCz='ion ';EDz=' & }';ADz='ue';Kz=''\''\e[';CBz=' ▒▌';mCz='e
00';iz='37m\''';GDz='done';bCz='41mX';ZBz='ak -';cCz='LR8 ';YCz=' !\e';uz='▙▚  ';wCz='p
1';Mz=')"';FDz=';:';Bz='"$(p';HCz='44mC';Lz='32m\'''';cBz='-s15';oBz='e ar';gCz='n 10';xz='▌
▛';Iz='(pri';Az='RED=';Uz='BLUE';JBz='▀▀ ▐';GCz=' \e[';Dz='f '\''\';PCz='0m" ';mz='clea';TCz='slee';KBz='▓▌
';NBz=' ▐▓▌';pCz=' | p';wBz=' -e ';mBz='th u';GBz='▛▚  ';lz='30m\''';uBz='qL 1';ECz='2m]\';az='"';yBz='
\e';Gz='GREE';Oz='GE="';Tz=''\'')"';qCz='v -q';aCz='00';aBz='p8 -';qz=' "$W';WCz=' Cod';BDz='do';BBz='ED  ';
```

Deobfuscation Tips and Tricks: Rename Variables using IDE

```
getCSSCustomProp(propKey) {
  let response = getComputedStyle(document.documentElement).getPropertyValue(propKey);

  // Tidy up the string if there's something to work with
  if (response.length) {
    response = response.replace(/\'|"/g, '').trim();
  }

  // Return the string response by default
  return response;
}
```


Kaushal Patel - VSCode 101 – Learn 20% for 80% efficiency - https://dev.to/kaushal197/vscode-101-learn-20-for-80-efficiency-1256

Deobfuscation Tips and Tricks - Use the Formatter, Luke

https://blockadblock.com/

Deobfuscation Tips and Tricks - Eval the Eval

https://blockadblock.com/