running example

```
$ file mystery
mystery: ELF 64-bit LSB pie executable, x86-64,
version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=9819a3cfb39d01ad2a376c54318f104139422a8f,
for GNU/Linux 3.2.0, stripped
```

LSB = little endian

pie = position-independent executable

interpreter = program that loads this

aside: file(1)

```
$ man file
FILE(1)                      General Commands Manual

NAME
       file – determine file type

....
```

---

looks for “magic numbers” near beginning of file data

hand-managed database of common patterns

from file's source

```
0   name        elf-le
>16 leshort     0       no file type,
!:mime  application/octet-stream
>16 leshort     1       relocatable,
!:mime  application/x-object
>16 leshort     2       executable,
!:mime  application/x-executable
>16 leshort     3       ${x?pie executable:shared object},

...
0   string      \177ELF     ELF
!:strength *2
>4  byte        0       invalid class
>4  byte        1       32-bit
>4  byte        2       64-bit
>5  byte        0       invalid byte order
>5  byte        1       LSB
>>0 use     elf-le
>5  byte        2       MSB
>>0 use     \^elf-le
>7  byte        0       (SYSV)
```

finding strings

```
$ hexdump -c mystery
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  03 00 3e 00 01 00 00 00  c0 60 00 00 00 00 00 00  |..>......`......|
00000020  40 00 00 00 00 00 00 00  08 5e 03 00 00 00 00 00  |@........^......|
00000030  00 00 00 00 40 00 38 00  0d 00 40 00 1e 00 1d 00  |....@.8...@.....|
[... many more lines ...]
00000e60  00 5f 49 54 4d 5f 64 65  72 65 67 69 73 74 65 72  |._ITM_deregister|
00000e70  54 4d 43 6c 6f 6e 65 54  61 62 6c 65 00 5f 5f 67  |TMCloneTable.__g|
00000e80  6d 6f 6e 5f 73 74 61 72  74 5f 5f 00 5f 49 54 4d  |mon_start__._ITM|
00000e90  5f 72 65 67 69 73 74 65  72 54 4d 43 6c 6f 6e 65  |_registerTMClone|
00000ea0  54 61 62 6c 65 00 77 61  64 64 63 68 00 63 6c 65  |Table.waddch.cle|
00000eb0  61 72 6f 6b 00 6e 6f 65  63 68 6f 00 6d 76 70 72  |arok.noecho.mvpr|
[... many more lines ...]
```

exercise: heuristic?

could scan through pages of hexdump for something interesting...

good heuristic for automating this process?

strings utility (1)

```
$ strings mystery
/lib64/ld-linux-x86-64.so.2
*7lT1
A9B*
m8m7
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
waddch
clearok
…
        prints help
        identify object
        left
        down
        right
…
        save game
        quit
```

strings utility (2)

```
$ strings --bytes=40 mystery
character you want help for (* for all):
you feel a wrenching sensation in your gut
your armor appears to be weaker now. Oh my!
you feel a sting in your arm and now feel weaker
Level: %d  Gold: %-5d  Hp: %*d(%*d)  Str: %2d(%d)  Ac: %-2d  Exp: %d/%ld  %s
Ok, if you want to exit that badly, I'll have to allow it
Hello %s, just a moment while I dig the dungeon...
orry, but your terminal window has too few columns.
Sorry, but your terminal window has too few lines.
please specifiy a letter between 'A' and 'Z'
```

...

dedicated reverse engineering tools

specialized toolkits for specifically reverse engineering

more complex analyses than objdump/strings

primary example I'll look at: Ghidra

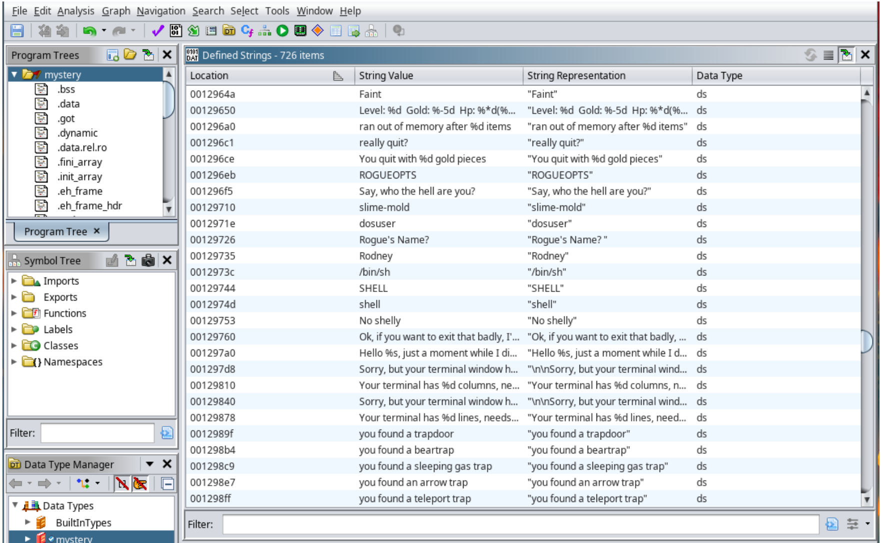
- open source, developed by National Security Agency

has some commercial competitors

- Binary Ninja (`https://binary.ninja`), IDA Pro (`https://hex-rays.com/ida-pro`)
- sometimes free/cheap for educational use
- very expensive for full/commercial licenses

in Ghidra

(after making new project, loading mystery file, Window > Defined Strings)

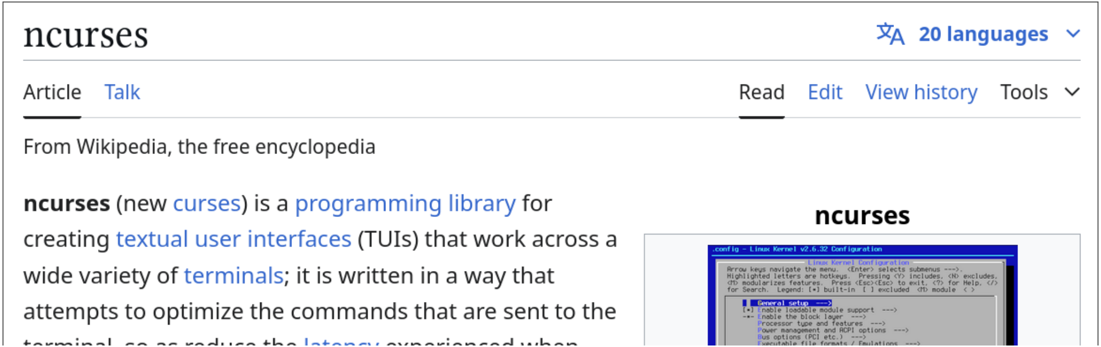


Location	String Value	String Representation	Data Type
0012964a	Faint	"Faint"	ds
00129650	Level: %d Gold: %-5d Hp: %*d(%...	"Level: %d Gold: %-5d Hp: %*d(%...	ds
001296a0	ran out of memory after %d items	"ran out of memory after %d items"	ds
001296c1	really quit?	"really quit?"	ds
001296ce	You quit with %d gold pieces	"You quit with %d gold pieces"	ds
001296eb	ROGUEOPTS	"ROGUEOPTS"	ds
001296f5	Say, who the hell are you?	"Say, who the hell are you?"	ds
00129710	slime-mold	"slime-mold"	ds
0012971e	dosuser	"dosuser"	ds
00129726	Rogue's Name?	"Rogue's Name? "	ds
00129735	Rodney	"Rodney"	ds
0012973c	/bin/sh	"/bin/sh"	ds
00129744	SHELL	"SHELL"	ds
0012974d	shell	"shell"	ds
00129753	No shelly	"No shelly"	ds
00129760	Ok, if you want to exit that badly, I'...	"Ok, if you want to exit that badly, ...	ds
001297a0	Hello %s, just a moment while I di...	"Hello %s, just a moment while I d...	ds
001297d8	Sorry, but your terminal window h...	"\n\nSorry, but your terminal wind...	ds
00129810	Your terminal has %d columns, ne...	"Your terminal has %d columns, n...	ds
00129840	Sorry, but your terminal window h...	"\n\nSorry, but your terminal wind...	ds
00129878	Your terminal has %d lines, needs...	"Your terminal has %d lines, need...	ds
0012989f	you found a trapdoor	"you found a trapdoor"	ds
001298b4	you found a beartrap	"you found a beartrap"	ds
001298c9	you found a sleeping gas trap	"you found a sleeping gas trap"	ds
001298e7	you found an arrow trap	"you found an arrow trap"	ds
001298ff	you found a teleport trap	"you found a teleport trap"	ds

libraries

```
$ objdump --all-headers mystery

…

Dynamic Section:
  NEEDED               libncurses.so.6
  NEEDED               libtinfo.so.6
  NEEDED               libc.so.6

…
```

ncurses?



ncurses

文A 20 languages

Article Talk

Read Edit View history Tools

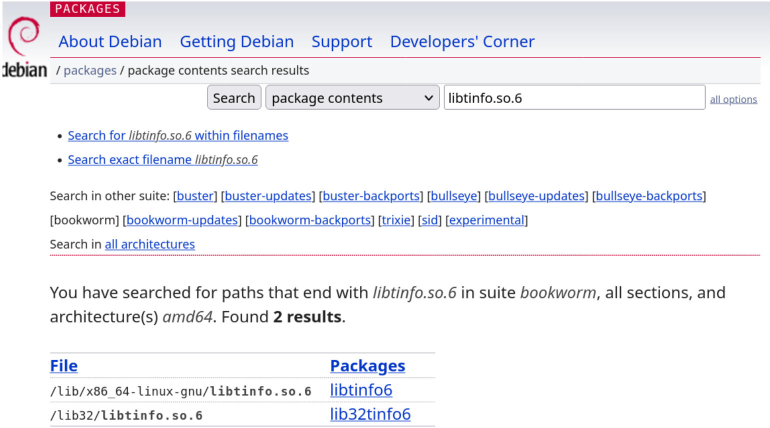
From Wikipedia, the free encyclopedia

ncurses (new curses) is a programming library for creating textual user interfaces (TUIs) that work across a wide variety of terminals; it is written in a way that attempts to optimize the commands that are sent to the terminal, so as reduce the latency experienced when

ncurses

```
.config - Linux Kernel v2.6.32 Configuration
                    Linux Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus --->.
 Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
 <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
 for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

         General setup  --->
     [*] Enable loadable module support  --->
     -*- Enable the block layer  --->
         Processor type and features  --->
         Power management and ACPI options  --->
         Bus options (PCI etc.)  --->
         Executable file formats / Emulations  --->
```

tinfo? (1)



/ packages / package contents search results

- Search for *libtinfo.so.6* within filenames
- Search exact filename *libtinfo.so.6*

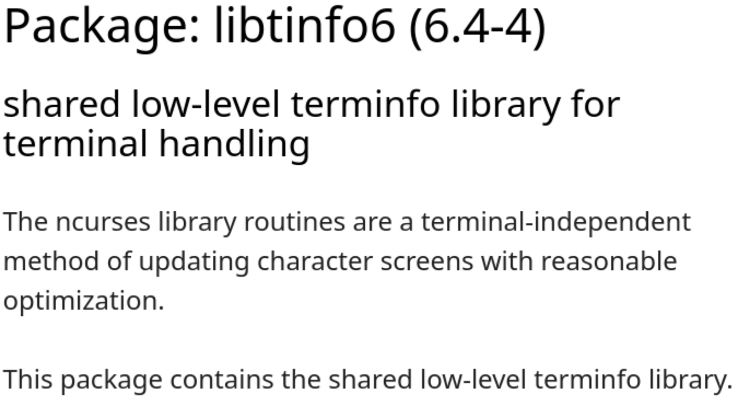
Search in other suite: [buster] [buster-updates] [buster-backports] [bullseye] [bullseye-updates] [bullseye-backports] [bookworm] [bookworm-updates] [bookworm-backports] [trixie] [sid] [experimental]

Search in all architectures

You have searched for paths that end with *libtinfo.so.6* in suite *bookworm*, all sections, and architecture(s) *amd64*. Found **2 results**.

File	**Packages**
`/lib/x86_64-linux-gnu/libtinfo.so.6`	libtinfo6
`/lib32/libtinfo.so.6`	lib32tinfo6

tinfo? (2)



Package: libtinfo6 (6.4-4)

shared low-level terminfo library for terminal handling

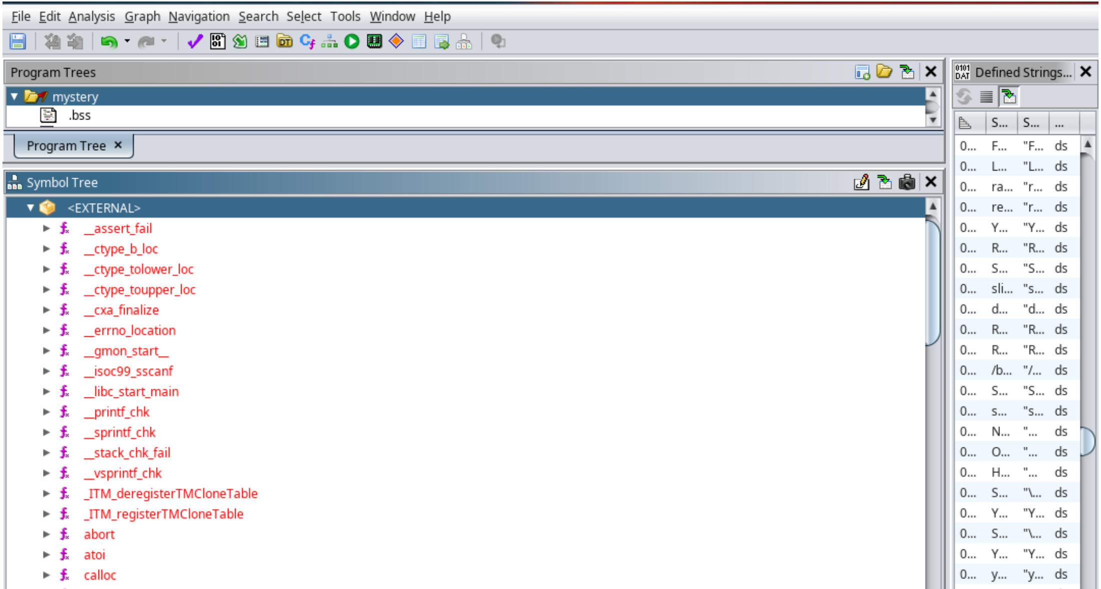
The ncurses library routines are a terminal-independent method of updating character screens with reasonable optimization.

This package contains the shared low-level terminfo library.

library calls

```
$ objdump --dynamic-syms mystery

mystery:     file format elf64-x86-64

DYNAMIC SYMBOL TABLE:
0000000000000000      DF *UND*  0000000000000000 (GLIBC_2.3)  __ctype_toupper_loc
0000000000000000      DF *UND*  0000000000000000 (GLIBC_2.2.5) getenv
0000000000000000      DF *UND*  0000000000000000 (NCURSES6_5.0.19991023) wattrset
0000000000000000      DF *UND*  0000000000000000 (GLIBC_2.2.5) free
0000000000000000      DF *UND*  0000000000000000 (NCURSES6_TINFO_5.0.19991023) flushinp
0000000000000000      DF *UND*  0000000000000000 (GLIBC_2.2.5) localtime
0000000000000000      DF *UND*  0000000000000000 (GLIBC_2.34) __libc_start_main

...
0000000000000000      DF *UND*  0000000000000000 (GLIBC_2.2.5) setuid

...
```

library calls (Ghidra)


File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
mystery
.bss
Program Tree ×
Symbol Tree
Defined Strings...

<EXTERNAL>

- __assert_fail
- __ctype_b_loc
- __ctype_tolower_loc
- __ctype_toupper_loc
- __cxa_finalize
- __errno_location
- __gmon_start__
- __isoc99_sscanf
- __libc_start_main
- __printf_chk
- __sprintf_chk
- __stack_chk_fail
- __vsprintf_chk
- _ITM_deregisterTMCloneTable
- _ITM_registerTMCloneTable
- abort
- atoi
- calloc

	S...	S...	...
0...	F...	"F...	ds
0...	L...	"L...	ds
0...	ra...	"r...	ds
0...	re...	"r...	ds
0...	Y...	"Y...	ds
0...	R...	"R...	ds
0...	S...	"S...	ds
0...	sli...	"s...	ds
0...	d...	"d...	ds
0...	R...	"R...	ds
0...	R...	"R...	ds
0...	/b...	"/...	ds
0...	S...	"S...	ds
0...	s...	"s...	ds
0...	N...	"...	ds
0...	O...	"...	ds
0...	H...	"...	ds
0...	S...	"\...	ds
0...	Y...	"Y...	ds
0...	S...	"\...	ds
0...	Y...	"Y...	ds
0...	y...	"y...	ds

finding library call uses

objdump --disassemble --dyanmic-reloc:

```
0000000000005b00 <setuid@plt>:
    5b00:►      f3 0f 1e fa          ►  endbr64⊠
    5b04:►      f2 ff 25 fd d3 02 00 ►  bnd jmp *0x2d3fd(%rip)
                        # 32f08 <setuid@GLIBC_2.2.5>
    5b0b:►      0f 1f 44 00 00       ►  nopl   0x0(%rax,%rax,1)

...
   2764f:►      e8 ec e3 fd ff       ►  call   5a40 <open@plt>
   27654:►      89 05 fe 48 01 00    ►  mov    %eax,0x148fe(%rip)        # 3bf58 <
   2765a:►      31 c0                ►  xor    %eax,%eax
   2765c:►      e8 2f e1 fd ff       ►  call   5790 <getuid@plt>
   27661:►      89 c7                ►  mov    %eax,%edi
   27663:►      31 c0                ►  xor    %eax,%eax
   27665:►      e8 96 e4 fd ff       ►  call   5b00 <setuid@plt>
   2766a:►      31 c0                ►  xor    %eax,%eax
   2766c:►      e8 cf e2 fd ff       ►  call   5940 <getgid@plt>
   27671:►      48 83 c4 08          ►  add    $0x8,%rsp
```

disassembly issues (1)

```
.global main
main:
    call print_hello
    xorl %eax, %eax
    ret
.Lstr:
    .asciz "Hello!"
print_hello:
    leaq .Lstr(%rip), %rdi  // RDI <- .Lstr address
    jmp puts
```

```
0000000000001139 <main>:
    1139:        e8 0a 00 00 00               call   1148 <print_hello>
    113e:        31 c0                        xor    %eax,%eax
    1140:        c3                           ret
    1141:        48                           rex.W
    1142:        65 6c                        gs insb (%dx),%es:(%rdi)
    1144:        6c                           insb   (%dx),%es:(%rdi)
    1145:        6f                           outsl  %ds:(%rsi),(%dx)
    1146:        2e                           cs
        ...
0000000000001148 <print_hello>:
    1148:        48 8d 3d f2 ff ff ff         lea    -0xe(%rip),%rdi         # 1141 <main+0x8
    114f:        e9 dc fe ff ff               jmp    1030 <puts@plt>
```

disassembly issues

```
0000000000001139 <main>:
    1139:        e8 0a 00 00 00               call   1148 <print_hello>
    113e:        31 c0                        xor    %eax,%eax
    1140:        c3                           ret
    1141:        48                           rex.W
    1142:        65 6c                        gs insb (%dx),%es:(%rdi)
    1144:        6c                           insb   (%dx),%es:(%rdi)
    1145:        6f                           outsl  %ds:(%rsi),(%dx)
    1146:        2e                           cs
            ...
0000000000001148 <print_hello>:
    1148:        48 8d 3d f2 ff ff ff         lea    -0xe(%rip),%rdi        # 1141 <main+0x8
    114f:        e9 dc fe ff ff               jmp    1030 <puts@plt>
```

```
    1139:        e8 0a 00 00 00               call   1148 <__cxa_finalize@plt+0x108>
    113e:        31 c0                        xor    %eax,%eax
    1140:        c3                           ret
    1141:        48                           rex.W
    1142:        65 6c                        gs insb (%dx),%es:(%rdi)
    1144:        6c                           insb   (%dx),%es:(%rdi)
    1145:        6f                           outsl  %ds:(%rsi),(%dx)
    1146:        2e 00 48 8d                  cs add %cl,-0x73(%rax)
    114a:        3d f2 ff ff ff               cmp    $0xfffffff2,%eax
    114f:        e9 dc fe ff ff               jmp    1030 <puts@plt>
```

finding assembly heuristics

objdump strategy, apparently:

- disassemble instructions starting at each symbol
- skip over strings of zero-bytes just before symbol

problem: can misidentify jumped to instructions

- especially if symbols stripped to save space/hinder reverse engineering

exercise: algorithm to fix?

- (Ghidra does this)

some tricky cases (1)

```
_start:
    ...
    movq $main, %rdi
    ...
    call __libc_start_main
    ...
```

```
struct DeviceTypeFuncs {
    void (*Send)(struct DeviceInfo*, char *);
    void (*Recv)(struct DeviceInfo, char *, size_t);
};
void SendToDevice(struct DeviceInfo* info, char *data) {
    (info->funcs->Send)(data);
}
```

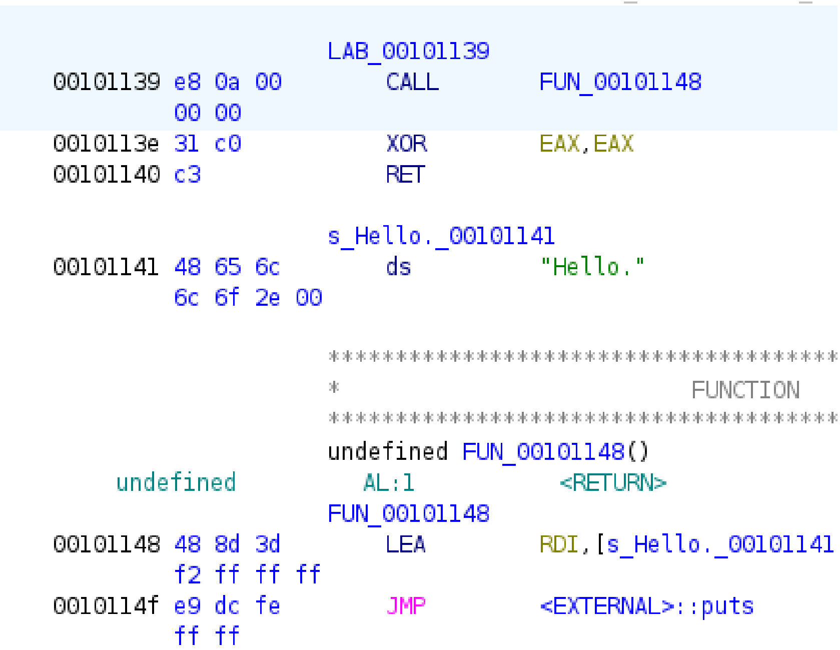
some tricky cases (2)

```
table:
    .int case1 - table
    .int case2 - table
...

    lea table(%rip), %rax
    addq (%rax, %rdi, 4), %rax
    jmp *%rax
```

```
    movq $function + 0x12340, %rax
    movq $0x1234, %r9
    sll $4, %r9
    addq %r9, %rax
    call *%rax
```

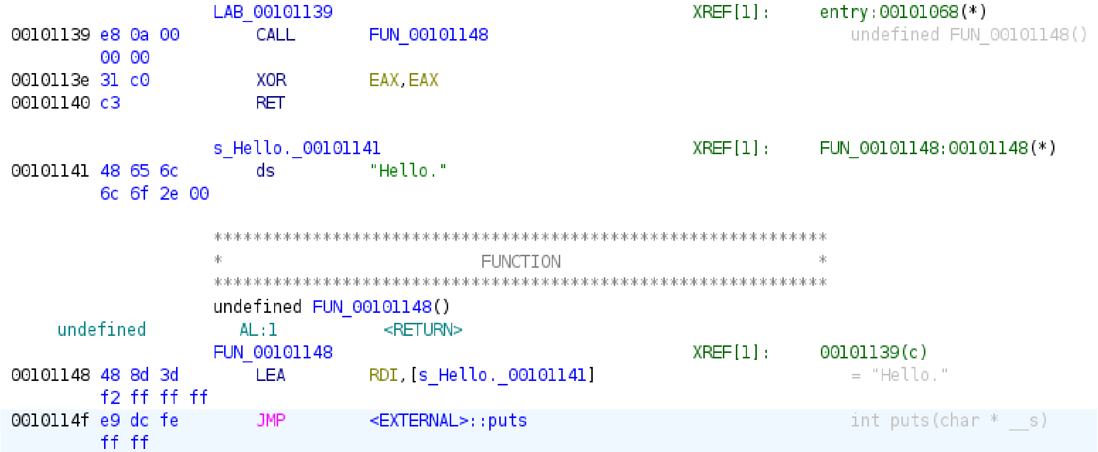
some tricky cases (3)

```
    call complex_func_returning_three
    lea next2-3(%rax), %rax
    jmp *%rax
    .byte 0x39, 0x59, 0x60, 0x89, 0xFF
next2:
    addq ...
```



```
                               LAB_00101139
        00101139 e8 0a 00         CALL       FUN_00101148
                 00 00
        0010113e 31 c0            XOR        EAX,EAX
        00101140 c3               RET

                               s_Hello._00101141
        00101141 48 65 6c         ds         "Hello."
                 6c 6f 2e 00

                               **************************************
                               *                            FUNCTION
                               **************************************
                               undefined FUN_00101148()
             undefined            AL:1            <RETURN>
                               FUN_00101148
        00101148 48 8d 3d         LEA        RDI,[s_Hello._00101141
                 f2 ff ff ff
        0010114f e9 dc fe         JMP        <EXTERNAL>::puts
                 ff ff
```

cross-references (1)



```
                     LAB_00101139                                   XREF[1]:     entry:00101068(*)
00101139 e8 0a 00        CALL       FUN_00101148                                    undefined FUN_00101148()
         00 00
0010113e 31 c0           XOR        EAX,EAX
00101140 c3              RET

                     s_Hello._00101141                              XREF[1]:     FUN_00101148:00101148(*)
00101141 48 65 6c        ds         "Hello."
         6c 6f 2e 00

                     **************************************************************
                     *                          FUNCTION                          *
                     **************************************************************
                     undefined FUN_00101148()
     undefined         AL:1           <RETURN>
                     FUN_00101148                                   XREF[1]:     00101139(c)
00101148 48 8d 3d        LEA        RDI,[s_Hello._00101141]                         = "Hello."
         f2 ff ff ff
0010114f e9 dc fe        JMP        <EXTERNAL>::puts                                int puts(char * __s)
         ff ff
```

cross-references idea

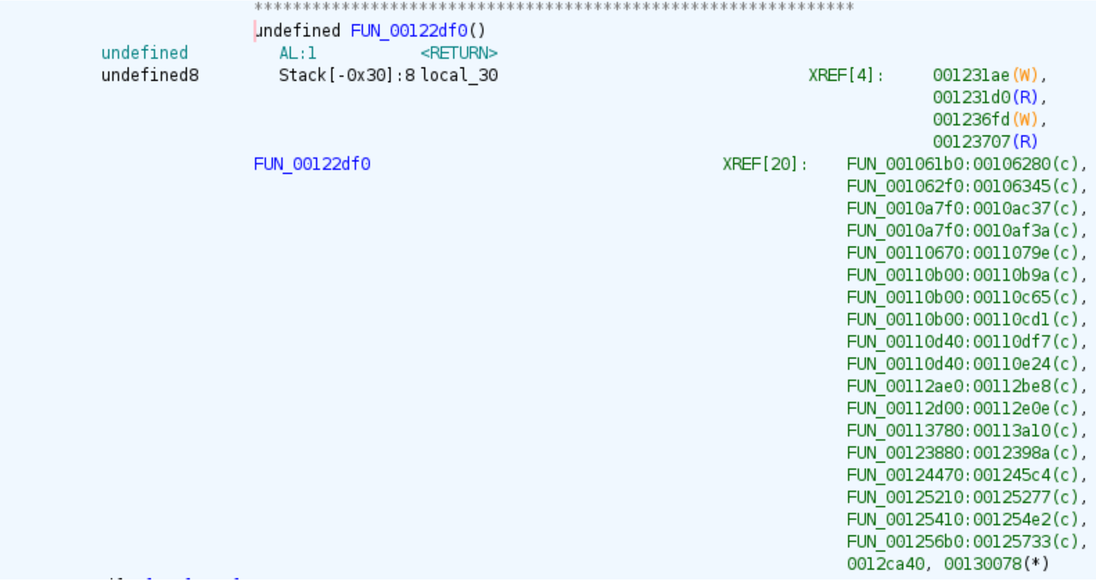
cross-reference idea:

really useful to know where something is used

do-able 'by hand' with objdump and friends, but...

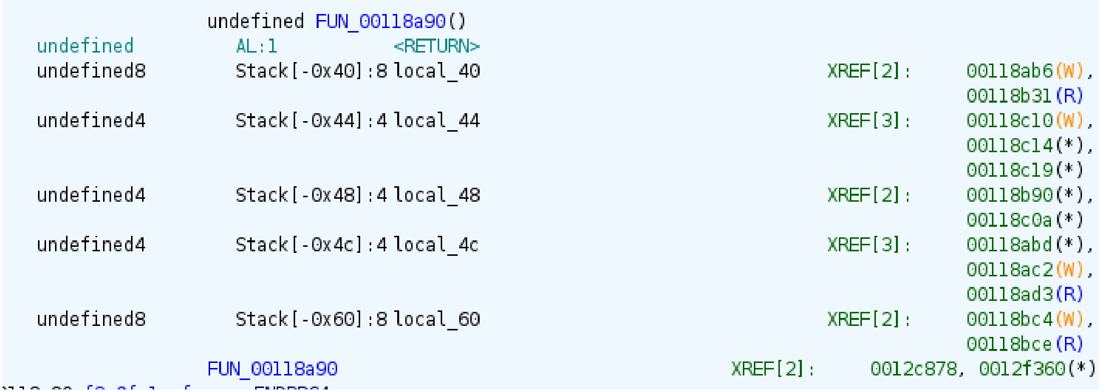
lots of bookkeeping, searching in text files, etc.

more cross-references



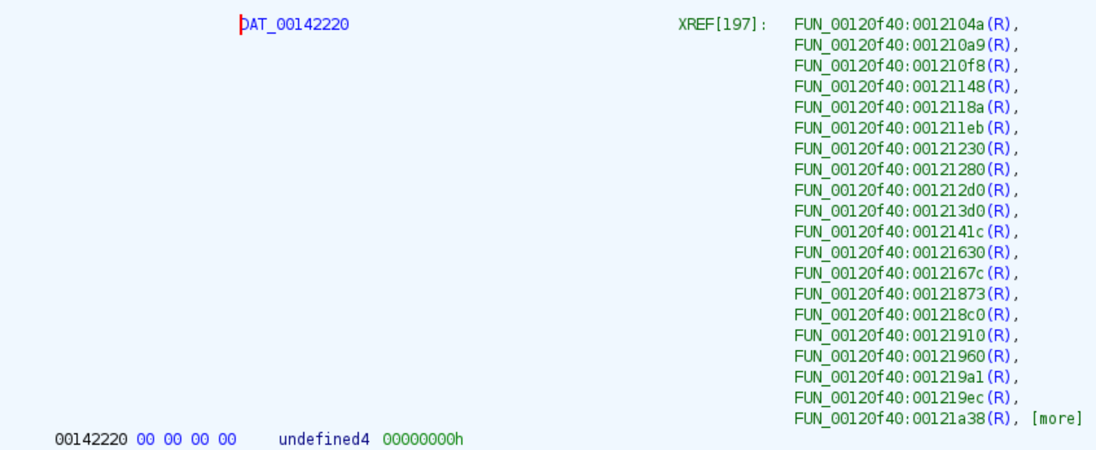
```
                           **************************************************************
                           undefined FUN_00122df0()
          undefined          AL:1             <RETURN>
          undefined8         Stack[-0x30]:8 local_30                          XREF[4]:     001231ae(W),
                                                                                           001231d0(R),
                                                                                           001236fd(W),
                                                                                           00123707(R)
                           FUN_00122df0                                   XREF[20]:    FUN_001061b0:00106280(c),
                                                                                       FUN_001062f0:00106345(c),
                                                                                       FUN_0010a7f0:0010ac37(c),
                                                                                       FUN_0010a7f0:0010af3a(c),
                                                                                       FUN_00110670:0011079e(c),
                                                                                       FUN_00110b00:00110b9a(c),
                                                                                       FUN_00110b00:00110c65(c),
                                                                                       FUN_00110b00:00110cd1(c),
                                                                                       FUN_00110d40:00110df7(c),
                                                                                       FUN_00110d40:00110e24(c),
                                                                                       FUN_00112ae0:00112be8(c),
                                                                                       FUN_00112d00:00112e0e(c),
                                                                                       FUN_00113780:00113a10(c),
                                                                                       FUN_00123880:0012398a(c),
                                                                                       FUN_00124470:001245c4(c),
                                                                                       FUN_00125210:00125277(c),
                                                                                       FUN_00125410:001254e2(c),
                                                                                       FUN_001256b0:00125733(c),
                                                                                       0012ca40, 00130078(*)
```

more cross-references (stack)



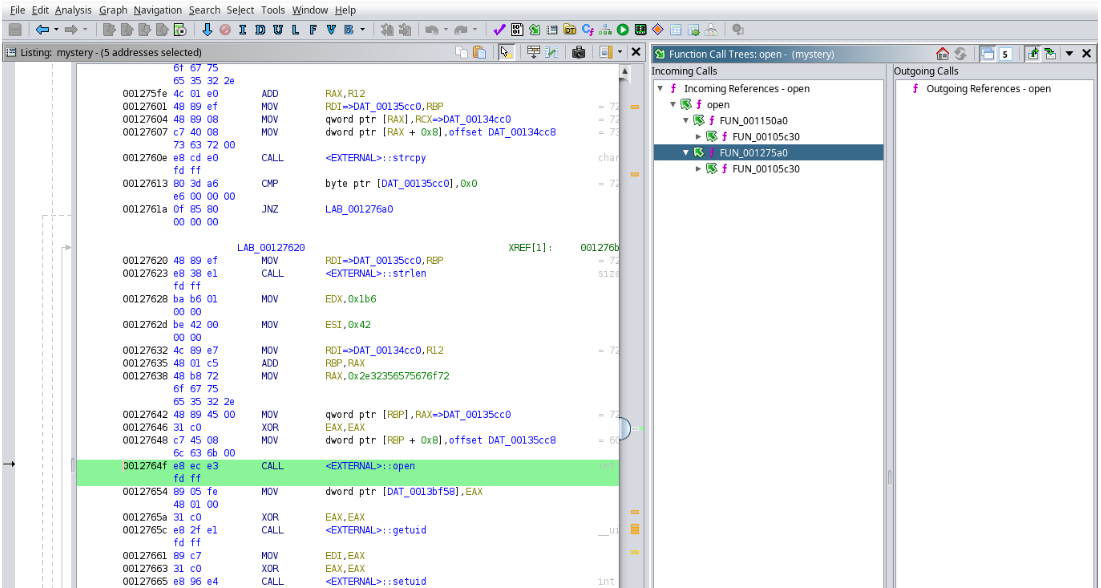
```
                 undefined FUN_00118a90()
   undefined        AL:1         <RETURN>
   undefined8       Stack[-0x40]:8 local_40                              XREF[2]:     00118ab6(W),
                                                                                      00118b31(R)
   undefined4       Stack[-0x44]:4 local_44                              XREF[3]:     00118c10(W),
                                                                                      00118c14(*),
                                                                                      00118c19(*)
   undefined4       Stack[-0x48]:4 local_48                              XREF[2]:     00118b90(*),
                                                                                      00118c0a(*)
   undefined4       Stack[-0x4c]:4 local_4c                              XREF[3]:     00118abd(*),
                                                                                      00118ac2(W),
                                                                                      00118ad3(R)
   undefined8       Stack[-0x60]:8 local_60                              XREF[2]:     00118bc4(W),
                                                                                      00118bce(R)
                 FUN_00118a90                                    XREF[2]:     0012c878, 0012f360(*)
```

more cross-references (global)



```
                    DAT_00142220                                  XREF[197]:   FUN_00120f40:0012104a(R),
                                                                               FUN_00120f40:001210a9(R),
                                                                               FUN_00120f40:001210f8(R),
                                                                               FUN_00120f40:00121148(R),
                                                                               FUN_00120f40:0012118a(R),
                                                                               FUN_00120f40:001211eb(R),
                                                                               FUN_00120f40:00121230(R),
                                                                               FUN_00120f40:00121280(R),
                                                                               FUN_00120f40:001212d0(R),
                                                                               FUN_00120f40:001213d0(R),
                                                                               FUN_00120f40:0012141c(R),
                                                                               FUN_00120f40:00121630(R),
                                                                               FUN_00120f40:0012167c(R),
                                                                               FUN_00120f40:00121873(R),
                                                                               FUN_00120f40:001218c0(R),
                                                                               FUN_00120f40:00121910(R),
                                                                               FUN_00120f40:00121960(R),
                                                                               FUN_00120f40:001219a1(R),
                                                                               FUN_00120f40:001219ec(R),
                                                                               FUN_00120f40:00121a38(R), [more]
    00142220 00 00 00 00     undefined4 00000000h
```

function callers?


File Edit Analysis Graph Navigation Search Select Tools Window Help
Listing: mystery - (5 addresses selected)
Function Call Trees: open - (mystery)
5

```
                         6f 67 75
                         65 35 32 2e
        001275fe 4c 01 e0        ADD        RAX,R12
        00127601 48 89 ef        MOV        RDI=>DAT_00135cc0,RBP                       = 72
        00127604 48 89 08        MOV        qword ptr [RAX],RCX=>DAT_00134cc0           = 72
        00127607 c7 40 08        MOV        dword ptr [RAX + 0x8],offset DAT_00134cc8   = 73
                 73 63 72 00
        0012760e e8 cd e0        CALL       <EXTERNAL>::strcpy                          char
                 fd ff
        00127613 80 3d a6        CMP        byte ptr [DAT_00135cc0],0x0                 = 72
                 e6 00 00 00
        0012761a 0f 85 80        JNZ        LAB_001276a0
                 00 00 00

                             LAB_00127620                              XREF[1]:     001276b
        00127620 48 89 ef        MOV        RDI=>DAT_00135cc0,RBP                       = 72
        00127623 e8 38 e1        CALL       <EXTERNAL>::strlen                          size
                 fd ff
        00127628 ba b6 01        MOV        EDX,0x1b6
                 00 00
        0012762d be 42 00        MOV        ESI,0x42
                 00 00
        00127632 4c 89 e7        MOV        RDI=>DAT_00134cc0,R12                       = 72
        00127635 48 01 c5        ADD        RBP,RAX
        00127638 48 b8 72        MOV        RAX,0x2e32356575676f72
                 6f 67 75
                 65 35 32 2e
        00127642 48 89 45 00     MOV        qword ptr [RBP],RAX=>DAT_00135cc0           = 72
        00127646 31 c0           XOR        EAX,EAX
        00127648 c7 45 08        MOV        dword ptr [RBP + 0x8],offset DAT_00135cc8   = 60
                 6c 63 6b 00
        0012764f e8 ec e3        CALL       <EXTERNAL>::open                            int
                 fd ff
        00127654 89 05 fe        MOV        dword ptr [DAT_0013bf58],EAX
                 48 01 00
        0012765a 31 c0           XOR        EAX,EAX
        0012765c e8 2f e1        CALL       <EXTERNAL>::getuid                          __ui
                 fd ff
        00127661 89 c7           MOV        EDI,EAX
        00127663 31 c0           XOR        EAX,EAX
        00127665 e8 96 e4        CALL       <EXTERNAL>::setuid                          int
```

Incoming Calls

```
▼ ƒ Incoming References - open
   ▼ ƒ open
      ▼ ƒ FUN_001150a0
         ▶ ƒ FUN_00105c30
      ▼ ƒ FUN_001275a0
         ▶ ƒ FUN_00105c30
```

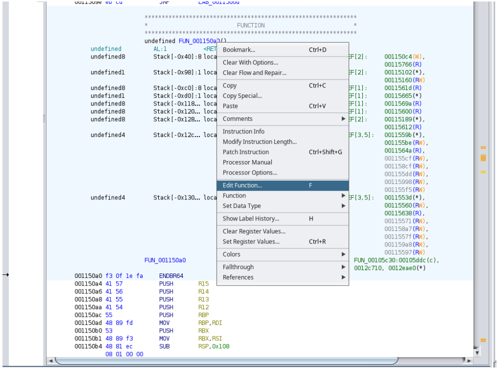
Outgoing Calls

```
ƒ Outgoing References - open
```

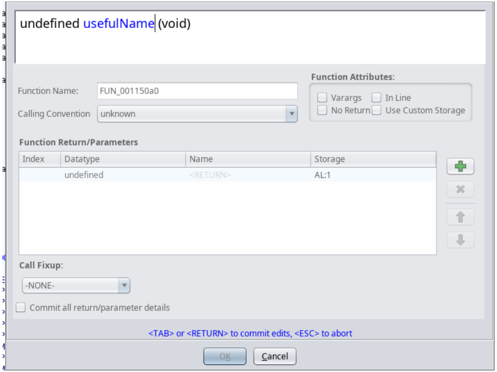
FUN_12345678

Ghidra names functions without symbols based on address

we can adjust that...



```
                          ****************************************************************
                          *                          FUNCTION                            *
                          ****************************************************************
                          undefined FUN_001150a0()
        undefined            AL:1           <RET
        undefined8           Stack[-0x40]:8 loca                EF[2]:     001150c4(W),
                                                                           00115766(R)
        undefined1           Stack[-0x98]:1 loca                EF[2]:     00115102(*),
                                                                           00115160(RW)
        undefined8           Stack[-0xc0]:8 loca                EF[1]:     0011561d(R)
        undefined1           Stack[-0xd0]:1 loca                EF[1]:     00115665(*)
        undefined8           Stack[-0x118... loca               EF[1]:     0011569a(R)
        undefined8           Stack[-0x120... loca               EF[1]:     00115600(R)
        undefined8           Stack[-0x128... loca               EF[2]:     00115189(*),
                                                                           00115612(R)
        undefined4           Stack[-0x12c... loca               EF[3,5]:   0011559b(*),
                                                                           001155be(RW),
                                                                           0011564a(R),
                                                                           001155cf(RW),
                                                                           001158cf(RW),
                                                                           001155dd(RW),
                                                                           00115998(RW),
                                                                           001155f5(RW)
        undefined4           Stack[-0x130... loca               EF[3,5]:   0011553d(*),
                                                                           00115560(RW),
                                                                           00115638(R),
                                                                           00115571(RW),
                                                                           001158a7(RW),
                                                                           0011557f(RW),
                                                                           001159a8(RW),
                                                                           00115597(RW)
                      FUN_001150a0                              FUN_00105c30:00105ddc(c),
                                                                0012c710, 0012eae0(*)
    001150a0 f3 0f 1e fa      ENDBR64
    001150a4 41 57            PUSH       R15
    001150a6 41 56            PUSH       R14
    001150a8 41 55            PUSH       R13
    001150aa 41 54            PUSH       R12
    001150ac 55               PUSH       RBP
    001150ad 48 89 fd         MOV        RBP,RDI
    001150b0 53               PUSH       RBX
    001150b1 48 89 f3         MOV        RBX,RSI
    001150b4 48 81 ec         SUB        RSP,0x108
             08 01 00 00
```



undefined usefulName (void)

Function Name: FUN_001150a0

Calling Convention: unknown

Function Attributes:

□ Varargs □ In Line

□ No Return □ Use Custom Storage

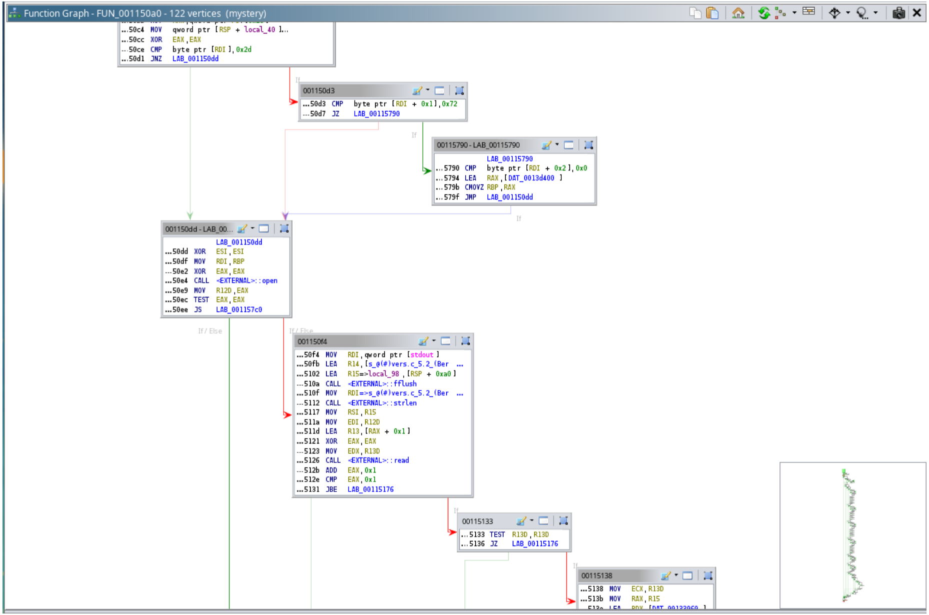
Function Return/Parameters

Index	Datatype	Name	Storage
	undefined	<RETURN>	AL:1

Call Fixup: -NONE-

□ Commit all return/parameter details

<TAB> or <RETURN> to commit edits, <ESC> to abort



Function Graph - FUN_001150a0 - 122 vertices (mystery)

```
...50c4 MOV   qword ptr [RSP + local_40 ]...
...50cc XOR   EAX,EAX
...50ce CMP   byte ptr [RDI ],0x2d
...50d1 JNZ   LAB_001150dd
```

001150d3

```
...50d3 CMP   byte ptr [RDI + 0x1],0x72
...50d7 JZ    LAB_00115790
```

00115790 - LAB_00115790

```
              LAB_00115790
...5790 CMP   byte ptr [RDI + 0x2],0x0
...5794 LEA   RAX,[DAT_0013d400 ]
...579b CMOVZ RBP,RAX
...579f JMP   LAB_001150dd
```

001150dd - LAB_00...

```
              LAB_001150dd
...50dd XOR   ESI,ESI
...50df MOV   RDI,RBP
...50e2 XOR   EAX,EAX
...50e4 CALL  <EXTERNAL>::open
...50e9 MOV   R12D,EAX
...50ec TEST  EAX,EAX
...50ee JS    LAB_001157c0
```

001150f4

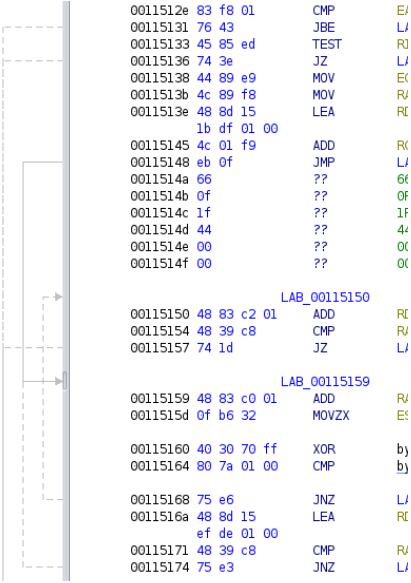
```
...50f4 MOV   RDI,qword ptr [stdout ]
...50fb LEA   R14,[s_@(#)vers.c_5.2_(Ber  ...
...5102 LEA   R15=>local_98 ,[RSP + 0xa0 ]
...510a CALL  <EXTERNAL>::fflush
...510f MOV   RDI=>s_@(#)vers.c_5.2_(Ber  ...
...5112 CALL  <EXTERNAL>::strlen
...5117 MOV   RSI,R15
...511a MOV   EDI,R12D
...511d LEA   R13,[RAX + 0x1]
...5121 XOR   EAX,EAX
...5123 MOV   EDX,R13D
...5126 CALL  <EXTERNAL>::read
...512b ADD   EAX,0x1
...512e CMP   EAX,0x1
...5131 JBE   LAB_00115176
```

00115133

```
...5133 TEST  R13D,R13D
...5136 JZ    LAB_00115176
```

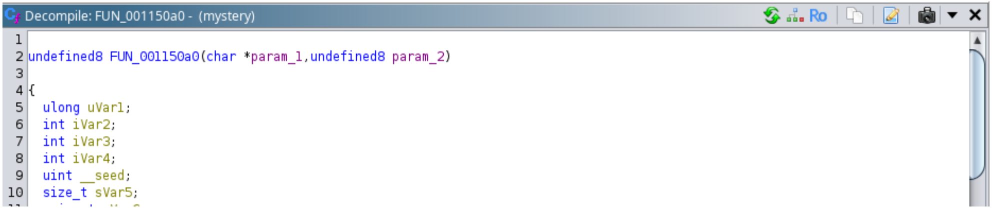
00115138

```
...5138 MOV   ECX,R13D
...513b MOV   RAX,R15
...513e LEA   RDX,[DAT_00133060 ]
```



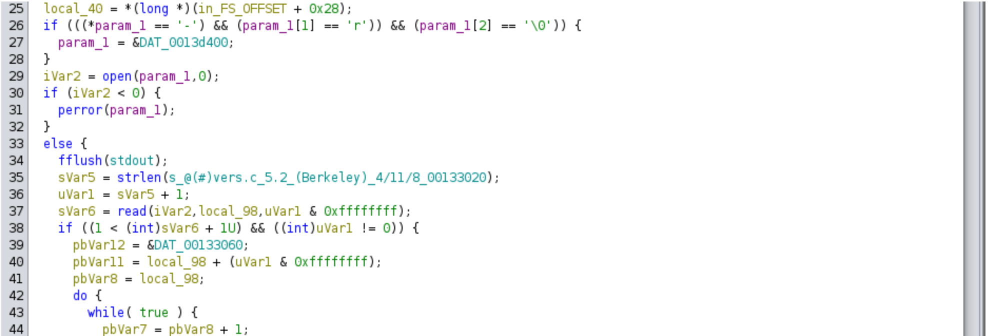
```
0011512e 83 f8 01       CMP        EA
00115131 76 43          JBE        LA
00115133 45 85 ed       TEST       RI
00115136 74 3e          JZ         LA
00115138 44 89 e9       MOV        EC
0011513b 4c 89 f8       MOV        RA
0011513e 48 8d 15       LEA        RD
         1b df 01 00
00115145 4c 01 f9       ADD        RC
00115148 eb 0f          JMP        LA
0011514a 66             ??         66
0011514b 0f             ??         0F
0011514c 1f             ??         1F
0011514d 44             ??         44
0011514e 00             ??         00
0011514f 00             ??         00

                     LAB_00115150
00115150 48 83 c2 01    ADD        RD
00115154 48 39 c8       CMP        RA
00115157 74 1d          JZ         LA

                     LAB_00115159
00115159 48 83 c0 01    ADD        RA
0011515d 0f b6 32       MOVZX      ES

00115160 40 30 70 ff    XOR        by
00115164 80 7a 01 00    CMP        by

00115168 75 e6          JNZ        LA
0011516a 48 8d 15       LEA        RD
         ef de 01 00
00115171 48 39 c8       CMP        RA
00115174 75 e3          JNZ        LA
```

decompiler



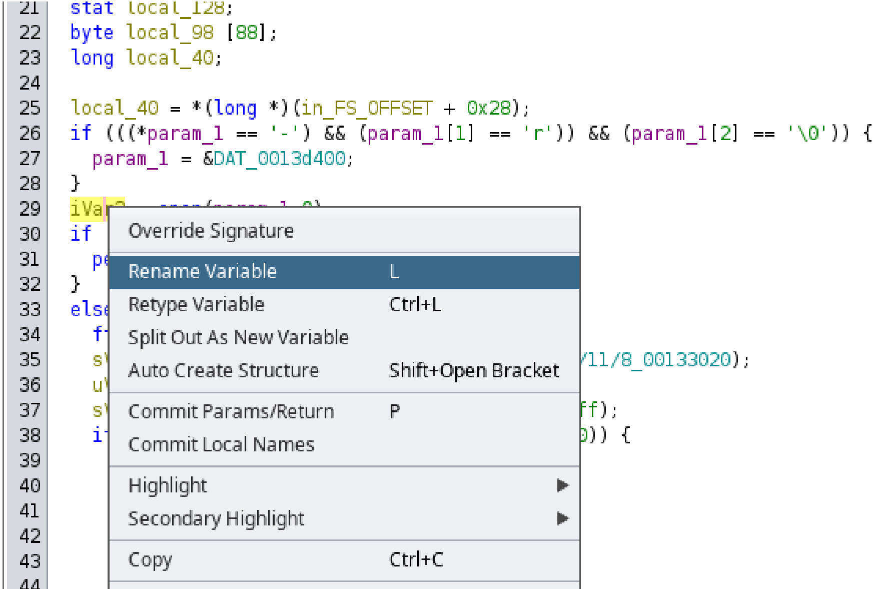
```

undefined8 FUN_001150a0(char *param_1,undefined8 param_2)

{
  ulong uVar1;
  int iVar2;
  int iVar3;
  int iVar4;
  uint __seed;
  size_t sVar5;
```

...



```
  local_40 = *(long *)(in_FS_OFFSET + 0x28);
  if (((*param_1 == '-') && (param_1[1] == 'r')) && (param_1[2] == '\0')) {
    param_1 = &DAT_0013d400;
  }
  iVar2 = open(param_1,0);
  if (iVar2 < 0) {
    perror(param_1);
  }
  else {
    fflush(stdout);
    sVar5 = strlen(s_@(#)vers.c_5.2_(Berkeley)_4/11/8_00133020);
    uVar1 = sVar5 + 1;
    sVar6 = read(iVar2,local_98,uVar1 & 0xffffffff);
    if ((1 < (int)sVar6 + 1U) && ((int)uVar1 != 0)) {
      pbVar12 = &DAT_00133060;
      pbVar11 = local_98 + (uVar1 & 0xffffffff);
      pbVar8 = local_98;
      do {
        while( true ) {
          pbVar7 = pbVar8 + 1;
```

refining decompile (1)



```
stat local_128;
byte local_98 [88];
long local_40;

local_40 = *(long *)(in_FS_OFFSET + 0x28);
if (((*param_1 == '-') && (param_1[1] == 'r')) && (param_1[2] == '\0')) {
  param_1 = &DAT_0013d400;
}
iVar2 = open(param_1,0);
if
  pe
}
else
  f
  s                                           /11/8_00133020);
  u
  s                                           ff);
  i                                           0)) {
```

refining decompile (2)

can setup names, types for functions

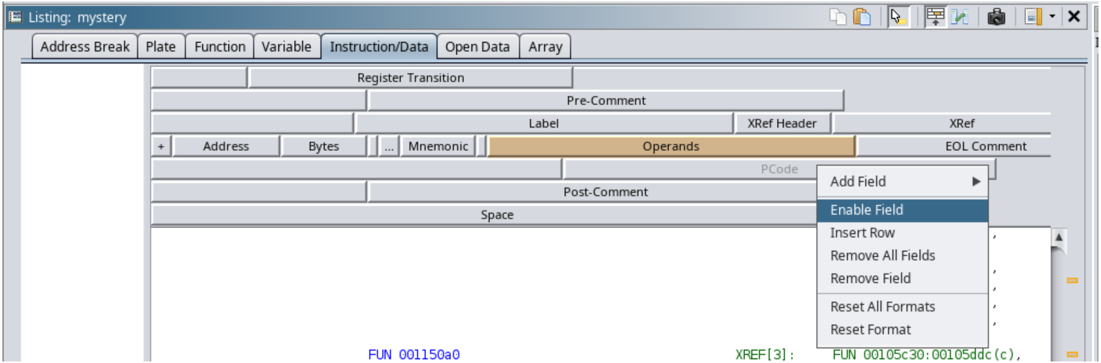
types can include marking array

- Ghidra doesn't seem great at inferring this all the time

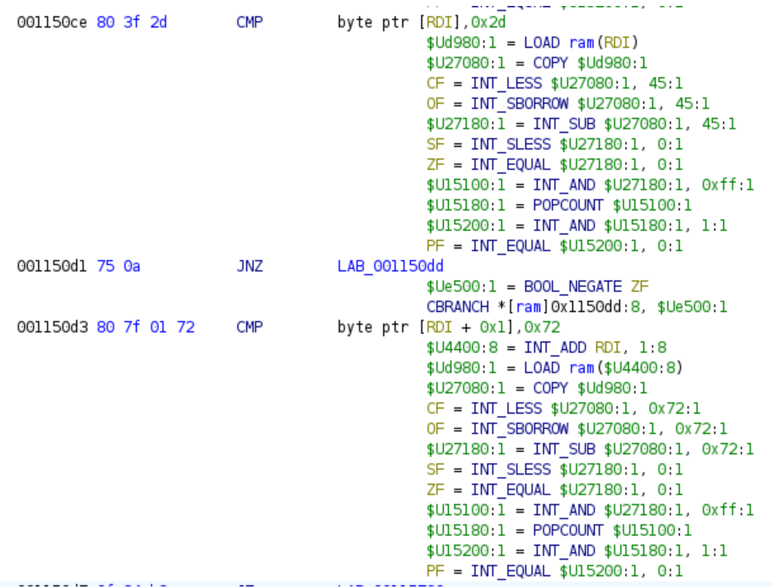
also for local/global variables

- for globals, can right-click in listing view too

interlude: editing disassembly format



PCode



```
001150ce 80 3f 2d        CMP          byte ptr [RDI],0x2d
                                             $Ud980:1 = LOAD ram(RDI)
                                             $U27080:1 = COPY $Ud980:1
                                             CF = INT_LESS $U27080:1, 45:1
                                             OF = INT_SBORROW $U27080:1, 45:1
                                             $U27180:1 = INT_SUB $U27080:1, 45:1
                                             SF = INT_SLESS $U27180:1, 0:1
                                             ZF = INT_EQUAL $U27180:1, 0:1
                                             $U15100:1 = INT_AND $U27180:1, 0xff:1
                                             $U15180:1 = POPCOUNT $U15100:1
                                             $U15200:1 = INT_AND $U15180:1, 1:1
                                             PF = INT_EQUAL $U15200:1, 0:1
001150d1 75 0a           JNZ          LAB_001150dd
                                             $Ue500:1 = BOOL_NEGATE ZF
                                             CBRANCH *[ram]0x1150dd:8, $Ue500:1
001150d3 80 7f 01 72     CMP          byte ptr [RDI + 0x1],0x72
                                             $U4400:8 = INT_ADD RDI, 1:8
                                             $Ud980:1 = LOAD ram($U4400:8)
                                             $U27080:1 = COPY $Ud980:1
                                             CF = INT_LESS $U27080:1, 0x72:1
                                             OF = INT_SBORROW $U27080:1, 0x72:1
                                             $U27180:1 = INT_SUB $U27080:1, 0x72:1
                                             SF = INT_SLESS $U27180:1, 0:1
                                             ZF = INT_EQUAL $U27180:1, 0:1
                                             $U15100:1 = INT_AND $U27180:1, 0xff:1
                                             $U15180:1 = POPCOUNT $U15100:1
                                             $U15200:1 = INT_AND $U15180:1, 1:1
                                             PF = INT_EQUAL $U15200:1, 0:1
```

Intermediate Representations

Ghidra converts instructions to this PCode language

- describes effects of each instruction for other parts of Ghidra
- allows ‘easy’ support for ARM, MIPS, …

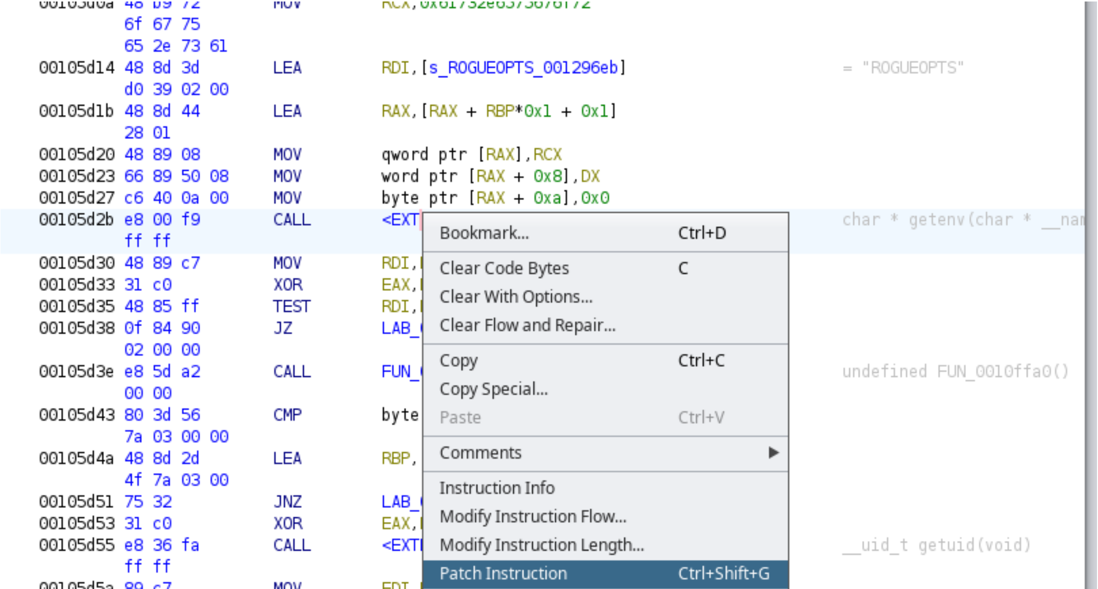
function graph we saw using PCode information, probably

decompiler is basically a PCode to C compiler

- does the same kind of optimizations/etc. normal compiler does
- different output language

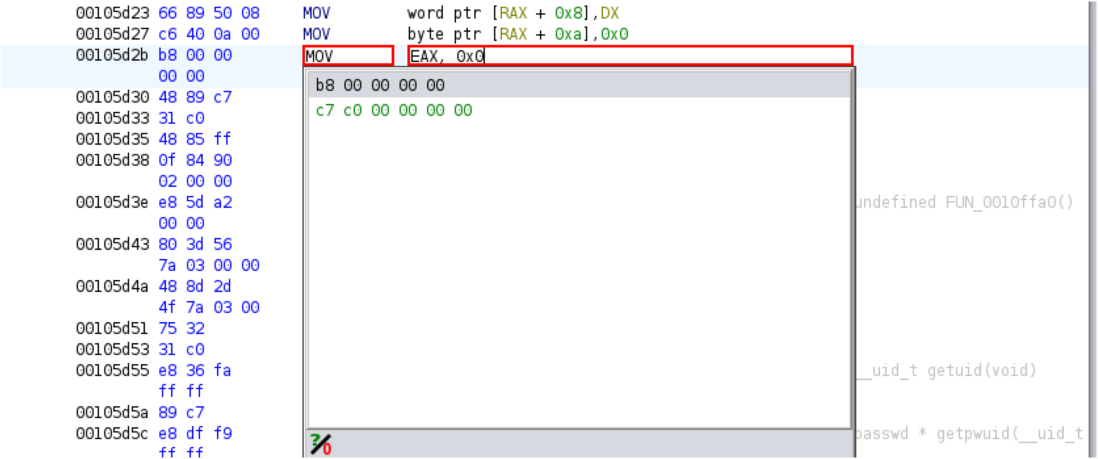
Ghidra has ‘find similar functions’ tool that probably uses this

patch instruction?



```
00105d0a 48 b9 72       MOV     RCX,0x61732e6575676f72
         6f 67 75
         65 2e 73 61
00105d14 48 8d 3d       LEA     RDI,[s_ROGUEOPTS_001296eb]                        = "ROGUEOPTS"
         d0 39 02 00
00105d1b 48 8d 44       LEA     RAX,[RAX + RBP*0x1 + 0x1]
         28 01
00105d20 48 89 08       MOV     qword ptr [RAX],RCX
00105d23 66 89 50 08    MOV     word ptr [RAX + 0x8],DX
00105d27 c6 40 0a 00    MOV     byte ptr [RAX + 0xa],0x0
00105d2b e8 00 f9       CALL    <EXT                                              char * getenv(char * __nam
         ff ff
00105d30 48 89 c7       MOV     RDI,
00105d33 31 c0          XOR     EAX,
00105d35 48 85 ff       TEST    RDI,
00105d38 0f 84 90       JZ      LAB_
         02 00 00
00105d3e e8 5d a2       CALL    FUN_                                              undefined FUN_0010ffa0()
         00 00
00105d43 80 3d 56       CMP     byte
         7a 03 00 00
00105d4a 48 8d 2d       LEA     RBP,
         4f 7a 03 00
00105d51 75 32          JNZ     LAB_
00105d53 31 c0          XOR     EAX,
00105d55 e8 36 fa       CALL    <EXT                                              __uid_t getuid(void)
         ff ff
00105d5a 89 c7          MOV     EDI,
```

patch instruction?



```
00105d23 66 89 50 08    MOV     word ptr [RAX + 0x8],DX
00105d27 c6 40 0a 00    MOV     byte ptr [RAX + 0xa],0x0
00105d2b b8 00 00       MOV     EAX, 0x0
         00 00                  b8 00 00 00 00
00105d30 48 89 c7               c7 c0 00 00 00 00
00105d33 31 c0
00105d35 48 85 ff
00105d38 0f 84 90
         02 00 00
00105d3e e8 5d a2                                                      undefined FUN_0010ffa0()
         00 00
00105d43 80 3d 56
         7a 03 00 00
00105d4a 48 8d 2d
         4f 7a 03 00
00105d51 75 32
00105d53 31 c0
00105d55 e8 36 fa                                                      __uid_t getuid(void)
         ff ff
00105d5a 89 c7
00105d5c e8 df f9                                                      passwd * getpwuid(__uid_t
         ff ff
```

why is this useful?

can export modified version of binary to test

ghidra has support for debugging or emulating running program

- emulation is another application of PCode representation
- debugging requires some work to configure

debuggers / emulators

major way to analyzing software — run it!

possibly using debugger to analyze memory/registers/etc.

possibly in restricted environment

- either limit access to system calls, *or*
- run on virtual (okay-to-lose) hardware

selected debugger features (1)

watchpoint (GDB/LLDB `watch`)

- breakpoint triggered by variable/expression changing

breakpoints on system calls (GDB `catch syscall` …)

searching memory for strings (GDB `find`, LLDB `memory find`)

selected debugger fetures (2)

saving 'core' files (GDB `generate-core-file NAME`)

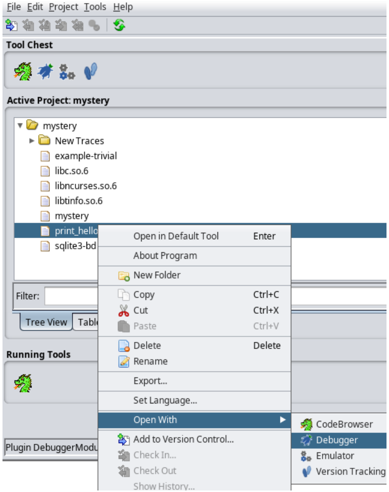
- full copy of program's memory, can reload in debugger later

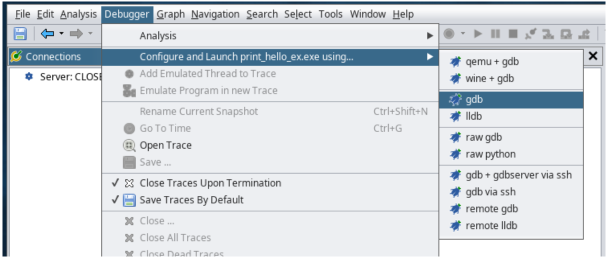
copying memory to/from file (GDB `dump`/`append`/`restore`; LLDB `memory read`/`write`)

attaching to programs / remote debugging

forcing jump to address/return from function (GDB `jump`/`return`)

Ghidra debugger integration



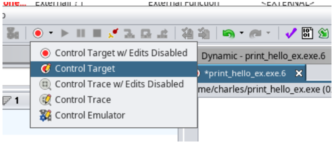

File Edit Analysis Debugger Graph Navigation Search Select Tools Window Help
Connections
Server: CLOSE
Analysis
Configure and Launch print_hello_ex.exe using...
Add Emulated Thread to Trace
Emulate Program in new Trace
Rename Current Snapshot Ctrl+Shift+N
Go To Time Ctrl+G
Open Trace
Save ...
✓ Close Traces Upon Termination
✓ Save Traces By Default
Close ...
Close All Traces
Close Dead Traces
qemu + gdb
wine + gdb
gdb
lldb
raw gdb
raw python
gdb + gdbserver via ssh
gdb via ssh
remote gdb
remote lldb

aside: Ghidra debugger installation

relies on GDB python support + some python packages installed

see installation docs

Ghidra traces


Control Target w/ Edits Disabled
Control Target
Control Trace w/ Edits Disabled
Control Trace
Control Emulator
Dynamic - print_hello_ex.exe.6
*print_hello_ex.exe.6
me/charles/print_hello_ex.exe (0:

Ghidra — debugging session creates a ‘trace’

can be saved to look at later

creates a list of ‘snapshots’ for every time debugger stopped

snapshots are *incomplete*

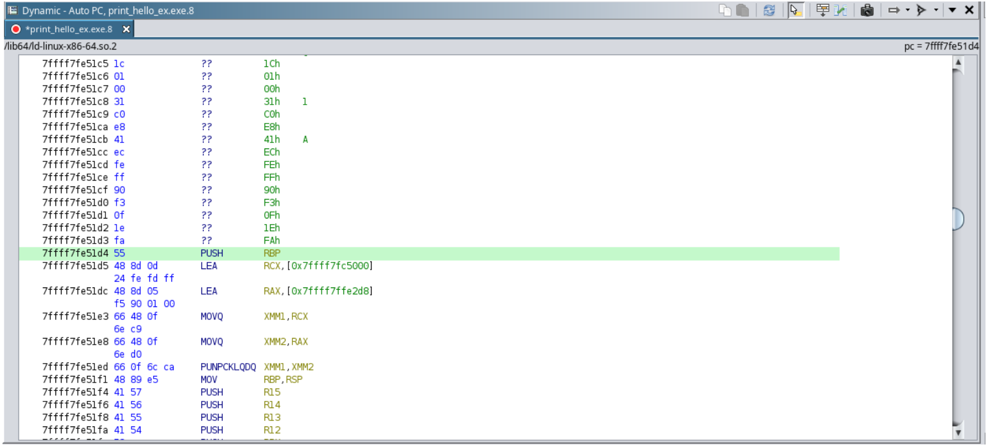
need to force read of memory/etc. to have info included in snapshot

can switch between ‘Control Target’ and ‘Control Trace’ modes

Control Trace — go back to old snapshots, examine state

Control Target — control live program in debugger

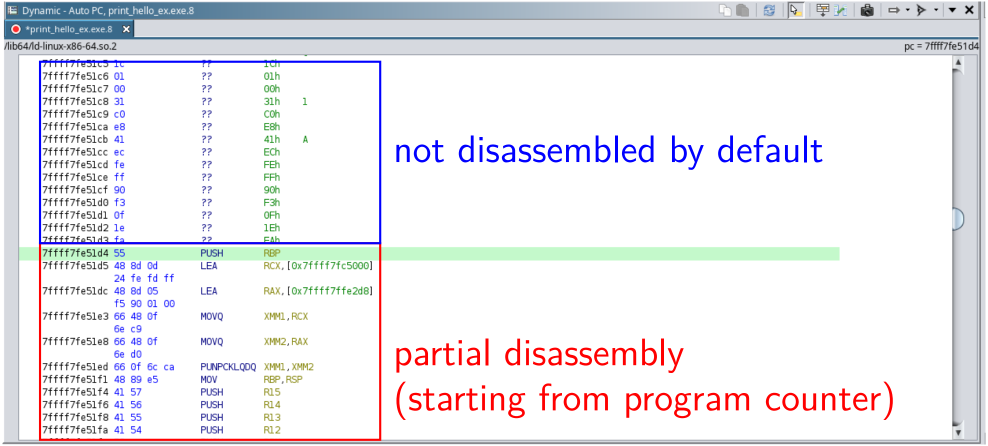
Ghidra dynamic view



Dynamic - Auto PC, print_hello_ex.exe.8
*print_hello_ex.exe.8
/lib64/ld-linux-x86-64.so.2
pc = 7ffff7fe51d4

```
7ffff7fe51c5 1c            ??         1Ch
7ffff7fe51c6 01            ??         01h
7ffff7fe51c7 00            ??         00h
7ffff7fe51c8 31            ??         31h    1
7ffff7fe51c9 c0            ??         C0h
7ffff7fe51ca e8            ??         E8h
7ffff7fe51cb 41            ??         41h    A
7ffff7fe51cc ec            ??         ECh
7ffff7fe51cd fe            ??         FEh
7ffff7fe51ce ff            ??         FFh
7ffff7fe51cf 90            ??         90h
7ffff7fe51d0 f3            ??         F3h
7ffff7fe51d1 0f            ??         0Fh
7ffff7fe51d2 1e            ??         1Eh
7ffff7fe51d3 fa            ??         FAh
7ffff7fe51d4 55            PUSH       RBP
7ffff7fe51d5 48 8d 0d      LEA        RCX,[0x7ffff7fc5000]
             24 fe fd ff
7ffff7fe51dc 48 8d 05      LEA        RAX,[0x7ffff7ffe2d8]
             f5 90 01 00
7ffff7fe51e3 66 48 0f      MOVQ       XMM1,RCX
             6e c9
7ffff7fe51e8 66 48 0f      MOVQ       XMM2,RAX
             6e d0
7ffff7fe51ed 66 0f 6c ca   PUNPCKLQDQ XMM1,XMM2
7ffff7fe51f1 48 89 e5      MOV        RBP,RSP
7ffff7fe51f4 41 57         PUSH       R15
7ffff7fe51f6 41 56         PUSH       R14
7ffff7fe51f8 41 55         PUSH       R13
7ffff7fe51fa 41 54         PUSH       R12
```

Ghidra dynamic view



Dynamic - Auto PC, print_hello_ex.exe.8
*print_hello_ex.exe.8
/lib64/ld-linux-x86-64.so.2
pc = 7ffff7fe51d4

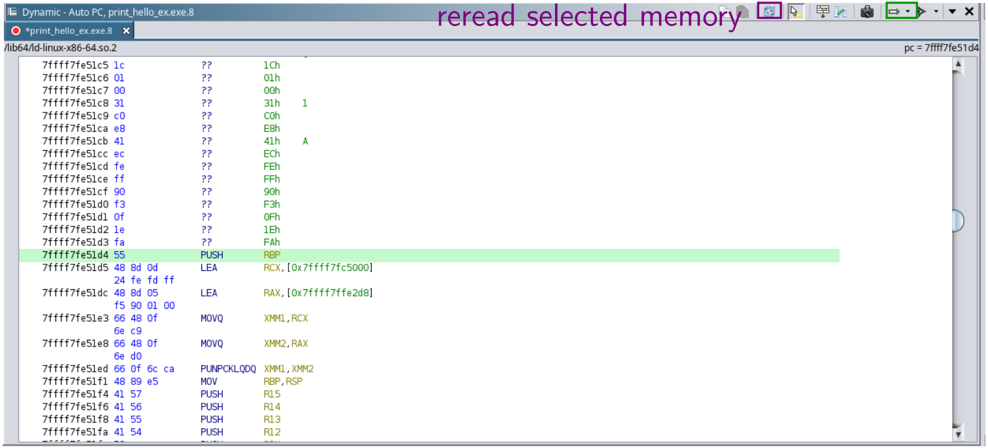
```
7ffff7fe51c5 1c              ??         1Ch
7ffff7fe51c6 01              ??         01h
7ffff7fe51c7 00              ??         00h
7ffff7fe51c8 31              ??         31h    1
7ffff7fe51c9 c0              ??         C0h
7ffff7fe51ca e8              ??         E8h
7ffff7fe51cb 41              ??         41h    A
7ffff7fe51cc ec              ??         ECh
7ffff7fe51cd fe              ??         FEh
7ffff7fe51ce ff              ??         FFh
7ffff7fe51cf 90              ??         90h
7ffff7fe51d0 f3              ??         F3h
7ffff7fe51d1 0f              ??         0Fh
7ffff7fe51d2 1e              ??         1Eh
7ffff7fe51d3 fa              ??         FAh
7ffff7fe51d4 55              PUSH       RBP
7ffff7fe51d5 48 8d 0d        LEA        RCX,[0x7ffff7fc5000]
             24 fe fd ff
7ffff7fe51dc 48 8d 05        LEA        RAX,[0x7ffff7ffe2d8]
             f5 90 01 00
7ffff7fe51e3 66 48 0f        MOVQ       XMM1,RCX
             6e c9
7ffff7fe51e8 66 48 0f        MOVQ       XMM2,RAX
             6e d0
7ffff7fe51ed 66 0f 6c ca     PUNPCKLQDQ XMM1,XMM2
7ffff7fe51f1 48 89 e5        MOV        RBP,RSP
7ffff7fe51f4 41 57           PUSH       R15
7ffff7fe51f6 41 56           PUSH       R14
7ffff7fe51f8 41 55           PUSH       R13
7ffff7fe51fa 41 54           PUSH       R12
```

not disassembled by default

partial disassembly (starting from program counter)

Ghidra dynamic view

select where in memory to view



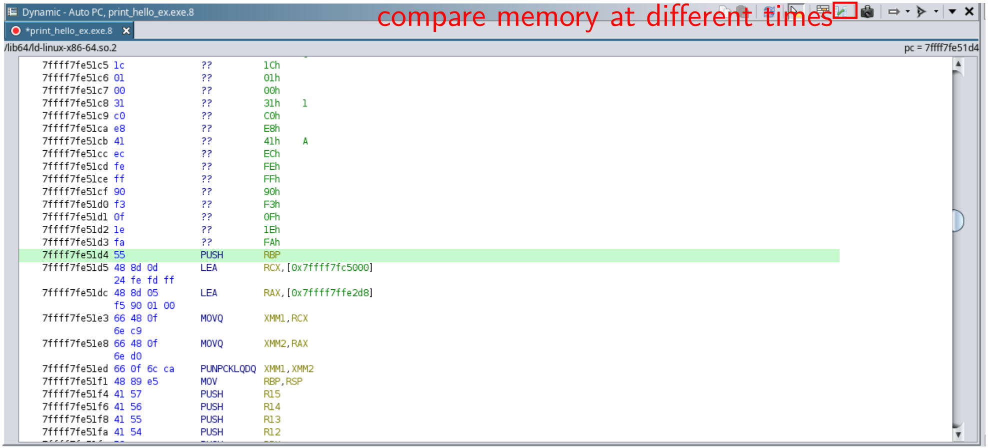
reread selected memory

/lib64/ld-linux-x86-64.so.2

pc = 7ffff7fe51d4

```
7ffff7fe51c5 1c          ??          1Ch
7ffff7fe51c6 01          ??          01h
7ffff7fe51c7 00          ??          00h
7ffff7fe51c8 31          ??          31h    1
7ffff7fe51c9 c0          ??          C0h
7ffff7fe51ca e8          ??          E8h
7ffff7fe51cb 41          ??          41h    A
7ffff7fe51cc ec          ??          ECh
7ffff7fe51cd fe          ??          FEh
7ffff7fe51ce ff          ??          FFh
7ffff7fe51cf 90          ??          90h
7ffff7fe51d0 f3          ??          F3h
7ffff7fe51d1 0f          ??          0Fh
7ffff7fe51d2 1e          ??          1Eh
7ffff7fe51d3 fa          ??          FAh
7ffff7fe51d4 55          PUSH        RBP
7ffff7fe51d5 48 8d 0d    LEA         RCX,[0x7ffff7fc5000]
             24 fe fd ff
7ffff7fe51dc 48 8d 05    LEA         RAX,[0x7ffff7ffe2d8]
             f5 90 01 00
7ffff7fe51e3 66 48 0f    MOVQ        XMM1,RCX
             6e c9
7ffff7fe51e8 66 48 0f    MOVQ        XMM2,RAX
             6e d0
7ffff7fe51ed 66 0f 6c ca PUNPCKLQDQ  XMM1,XMM2
7ffff7fe51f1 48 89 e5    MOV         RBP,RSP
7ffff7fe51f4 41 57       PUSH        R15
7ffff7fe51f6 41 56       PUSH        R14
7ffff7fe51f8 41 55       PUSH        R13
7ffff7fe51fa 41 54       PUSH        R12
```

Ghidra dynamic view

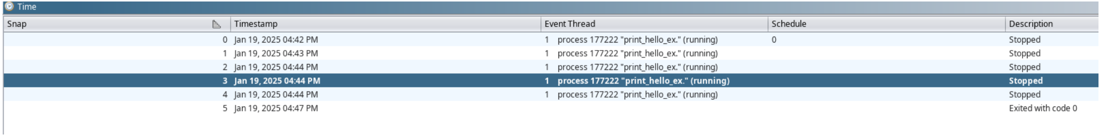


compare memory at different times

Dynamic - Auto PC, print_hello_ex.exe.8
*print_hello_ex.exe.8
/lib64/ld-linux-x86-64.so.2
pc = 7ffff7fe51d4

```
7ffff7fe51c5 1c          ??         1Ch
7ffff7fe51c6 01          ??         01h
7ffff7fe51c7 00          ??         00h
7ffff7fe51c8 31          ??         31h    1
7ffff7fe51c9 c0          ??         C0h
7ffff7fe51ca e8          ??         E8h
7ffff7fe51cb 41          ??         41h    A
7ffff7fe51cc ec          ??         ECh
7ffff7fe51cd fe          ??         FEh
7ffff7fe51ce ff          ??         FFh
7ffff7fe51cf 90          ??         90h
7ffff7fe51d0 f3          ??         F3h
7ffff7fe51d1 0f          ??         0Fh
7ffff7fe51d2 1e          ??         1Eh
7ffff7fe51d3 fa          ??         FAh
7ffff7fe51d4 55          PUSH       RBP
7ffff7fe51d5 48 8d 0d    LEA        RCX,[0x7ffff7fc5000]
             24 fe fd ff
7ffff7fe51dc 48 8d 05    LEA        RAX,[0x7ffff7ffe2d8]
             f5 90 01 00
7ffff7fe51e3 66 48 0f    MOVQ       XMM1,RCX
             6e c9
7ffff7fe51e8 66 48 0f    MOVQ       XMM2,RAX
             6e d0
7ffff7fe51ed 66 0f 6c ca PUNPCKLQDQ XMM1,XMM2
7ffff7fe51f1 48 89 e5    MOV        RBP,RSP
7ffff7fe51f4 41 57       PUSH       R15
7ffff7fe51f6 41 56       PUSH       R14
7ffff7fe51f8 41 55       PUSH       R13
7ffff7fe51fa 41 54       PUSH       R12
```

Ghidra snapshots/saved traces:



Time

Snap	Timestamp	Event Thread	Schedule	Description
0	Jan 19, 2025 04:42 PM	1 process 177222 "print_hello_ex." (running)	0	Stopped
1	Jan 19, 2025 04:43 PM	1 process 177222 "print_hello_ex." (running)		Stopped
2	Jan 19, 2025 04:44 PM	1 process 177222 "print_hello_ex." (running)		Stopped
3	Jan 19, 2025 04:44 PM	1 process 177222 "print_hello_ex." (running)		Stopped
4	Jan 19, 2025 04:44 PM	1 process 177222 "print_hello_ex." (running)		Stopped
5	Jan 19, 2025 04:47 PM			Exited with code 0

automatic partial snapshots whenever pausing debugger

can force read of range of memory to make snapshot contain memory image

reverse debugging?

old idea: ‘reverse debugging’

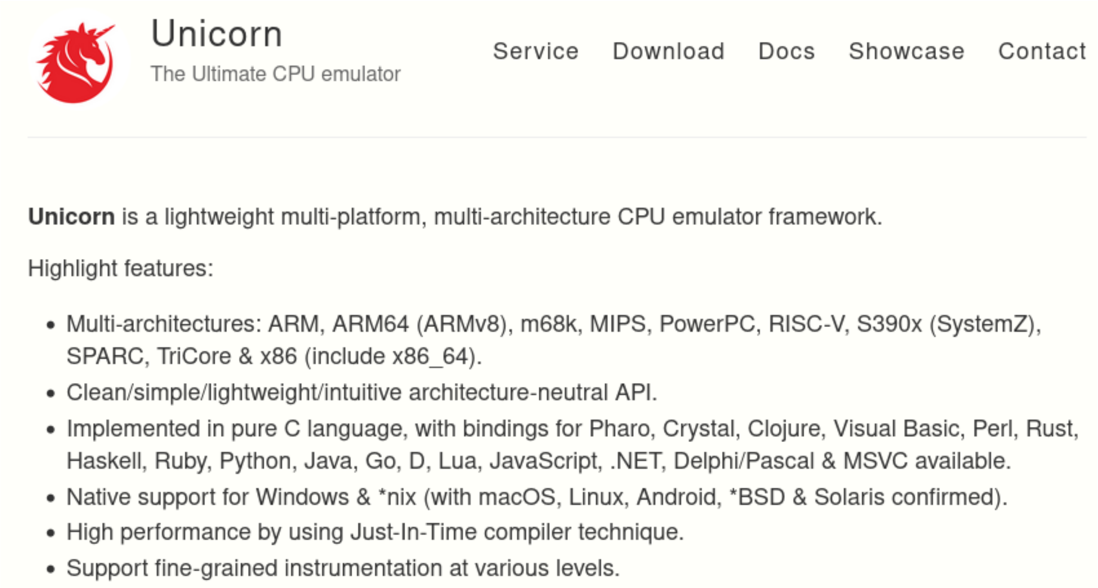
in addition to `step`/`continue`,
debugger could have `reverse-step`/`reverse-continue`

typically implemented by recording ‘trace’ of execution

some implementations (with varyingly middling performance

- `https://rr-project.org` for x86-64 Linux (needs sysadmin to set some things)
- QEMU for full virtual machines (not just one program)
- built-in to GDB, but not maintained/possibly broken with modern systems

unicorn as tool



Unicorn

The Ultimate CPU emulator

Unicorn is a lightweight multi-platform, multi-architecture CPU emulator framework.

Highlight features:

- Multi-architectures: ARM, ARM64 (ARMv8), m68k, MIPS, PowerPC, RISC-V, S390x (SystemZ), SPARC, TriCore & x86 (include x86_64).
- Clean/simple/lightweight/intuitive architecture-neutral API.
- Implemented in pure C language, with bindings for Pharo, Crystal, Clojure, Visual Basic, Perl, Rust, Haskell, Ruby, Python, Java, Go, D, Lua, JavaScript, .NET, Delphi/Pascal & MSVC available.
- Native support for Windows & *nix (with macOS, Linux, Android, *BSD & Solaris confirmed).
- High performance by using Just-In-Time compiler technique.
- Support fine-grained instrumentation at various levels.

unicorn example (1)

```
$ cat test.s
    mov $10000, %edi
    imul $2, %rdi, %rdi
$ gcc -c test.s; objcopy -j .text test.o -O binary test.bin
```

```
code = Path('test.bin').read_bytes()
uc = Uc(UC_ARCH_X86, UC_MODE_64)
uc.mem_map(0x10000, 1024 * 1024)
uc.mem_write(0x10000, code)
uc.emu_start(0x10000, 0x10000 + len(code))
print("RDI",uc.reg_read(UC_X86_REG_RDI))
```

```
RDI 20000
```

unicorn example (2)

```
...
uc.hook_add(UC_HOOK_CODE, hook_code_func)
def hook_code_func(uc, addr, size, user_data):
    print(f"{addr:x} ({size} byte instruction): "
          f"{codecs.encode(
                uc.mem_read(addr, size), 'hex'
             ).decode()}")
uc.emu_start(0x10000, 0x10000 + len(code))
```

```
10000 (5 byte instruction): bf10270000
10005 (4 byte instruction): 486bff02
```

example tool: qiling

https://qiling.io

uses Unicorn emulator but adds…

emulation for a lot of system calls

- including (hopefully) limiting file accesses to specific “virtual root” directory

loaders for common executable/bootloader formats

idea: get log of malware activity / add custom behaviors

PANDA.re

fork of emulator QEMU

supports whole-system record+replay

idea: run virtual machine with malware

replay run with analyses that can look at all instructions run

examples:

identify where dat from a specific file was used

search memory for string throughout execution

function call history

backup slides