
CONSOLE INTERFACE IMPROVEMENTS

DEEPAK CHETHAN (DODOCOCO)



ABOUT ME

- Name: Deepak Chethan (dodococo on telegram)
- Final year computer science undergrad from India
- Have been an Radare2 Contributor since Dec-2018
- Got selected to work on "Console Interface Improvements"

MAIN GOALS

- Generalize the code of popup widget
- Improving autocompletion and dietline modes (VI/Emacs-like hotkeys)
- Support colour scheme in radiff2 graphs
- Further improvements for UTF-8, RTL and Bi-Directional text
- Various bugfixes and improvements for visual, visual panels and graph modes
- Implement table commands and API

TASK I: GENERALIZE THE CODE OF POPUP WIDGET

- Popup widget was already implemented in Visual Offset mode
- Refactored the code and integrated it into RCons dietline
- Can be enabled by setting the config variable `scr.prompt.popup=true`

HOW DOES IT LOOK?

```
[0x1000011e8]> e scr.prompt.popup = true
[0x1000011e8]> eco_
    default
    xvilka
    lima
    bright
    onedark
    consonance
    rasta
    basic
    solarized
    ogray
    tango
    focus
    sepia
    pink
    darkda
```

TASK 2: AUTOCOMPLETION IMPROVEMENTS

- Mount shell (ms) was using its own version of autocomplete which is not complete
- Integrated the RCore autocomplete into mount shell
- Also added autocomplete for mount based commands like (mg, mc, mp, mo etc.,)

```
~ > r2 /bin/ls
-- Don't look at the code. Don't look.
[0x1000011e8]> m posix /
Mounted posix on / at 0x0
[0x1000011e8]> ms
[/]>
ls      cd      cat      get      mount      ?      !      help      q      exit
[/>]> cat
./          ../
/usr/        ./Spotlight-V100/
/.PKInstallSandboxManager-SystemSoftware/ /SAPDevelop/
/Network/    /sbin/
/var/        /Library/
/private/    /.DocumentRevisions-V100/
/Applications/ /opt/
/data/       /tmp/
[/>]> cat |
```

TASK 2: AUTOCOMPLETION IMPROVEMENTS

- Added SDB autocomplete (command k)

```
[0x1000011e8]> k
k      k?      ko      kd      ks      kj
[0x1000011e8]> k anal/
anal/fcns/    anal/meta/    anal/hints/    anal/types/    anal/spec/    anal/cc/    anal/zigns/    anal/classes/
[0x1000011e8]> k anal/spec/
anal/spec/gcc    anal/spec/spec.gcc.c    anal/spec/spec.gcc.d    anal/spec/spec.gcc.f    anal/spec/spec.gcc.g    anal/spec/spec.gcc.ld
anal/spec/spec.gcc.li    anal/spec/spec.gcc.lld    anal/spec/spec.gcc.lli    anal/spec/spec.gcc.llu    anal/spec/spec.gcc.lu    anal/spec/spec.gcc.p
anal/spec/spec.gcc.u
[0x1000011e8]> k anal/spec/█
```

LIVE DEMO TIME

```
[0x1000011e8]> e scr.prompt.popup = true
[0x1000011e8]> eco default
[0x1000011e8]> ms
[/]> q
[0x1000011e8]> m posix /
Mounted posix on / at 0x0
[0x1000011e8]> ms /
[/]> cat ./_
./
/...
/.Spotlight-V100/
./DS_Store
./PKInstallSandboxManager-Syst
./file
./fsevents.d/
./DocumentRevisions-V100/
./vol/
```

```
[0x1000011e8]> k anal/types/
anal/types/*aligned_alloc
anal/types/_Exit
anal/types/_assert_fail
anal/types/_assert_rtn
anal/types/_bzero
anal/types/_error
anal/types/_libc_init
anal/types/_libc_init_array
anal/types/_libc_start_main
anal/types/_maskrune
anal/types/_stack_chk_fail
anal/types/_tolower
anal/types/_toupper
anal/types/_uClibc_main
anal/types/_exit
```

TASK 3: IMPROVING DIETLINE MODES

- Added support for unix commands like:
 - Uniq
 - Sort
 - Join
 - Head
 - tail

TASK 3: IMPROVING DIETLINE MODES

- Radare2 now support vi key bindings !!
- Can be enabled by setting `scr.vi = true`
- The current implementation supports:
Insertion: `i, a, I, A, r and p`
Movement: `h, l, 0, ^, $, w, b, e, W, B, E`
Deletion: `d, dw, db, dB, dW, dh, dl, d$, d^, d0, D`
`c, cw, cb, cB, cW, ch, cl, c$, c^, c0 and x`
`ciw/diw, cib/dib`

TASK 3: IMPROVING DIETLINE MODES

- Integrated dietline into the HUD mode
- Now Hud supports all emacs/vi key bindings!
- Bonus: Added mouse support in HUD as well

TASK 4: RADIFF2 GRAPH IMPROVEMENTS

- Diff graphs in radare2 were super limited
- The only output it supported was xdot

TASK 4: RADIFF2 GRAPH IMPROVEMENTS

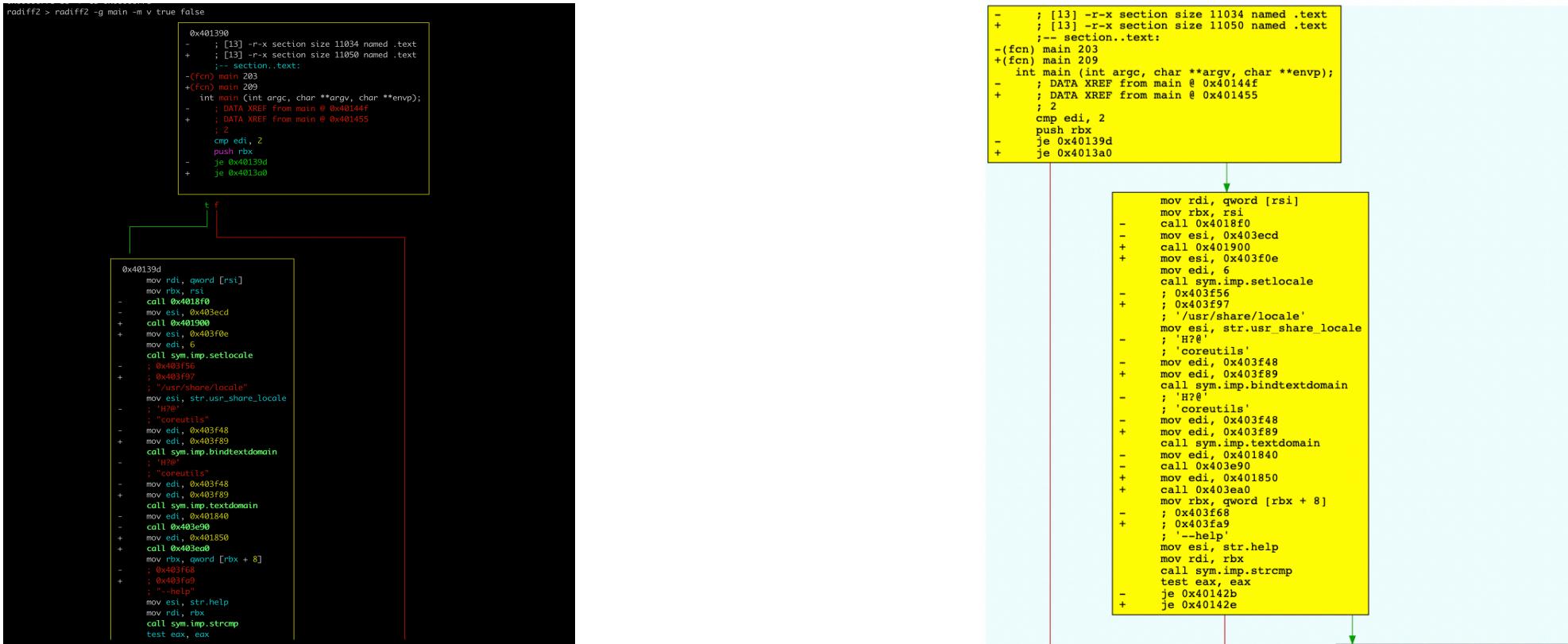
- So implemented more *agd* commands to construct diff graphs:
- Now it supports the output in:
 - r2 commands (*agd**)
 - Graphviz dot (*agdd*)
 - Graph Modelling Language (*agdg*)
 - Json (*agdj*)
 - SDB key-value (*agdk*)
 - Tiny ascii art (*agdt*)
 - Interactive ascii art (*agdv*)

TASK 4: RADIFF2 GRAPH IMPROVEMENTS

- Even radiff2 graphs supported only xdot output
- Now it even has interactive graphs (mini visual visual mode)
- Overall it supports all these commands

```
Graph Output formats: (-m [mode])
<blank/a>  Ascii art
s            r2 commands
d            Graphviz dot
g            Graph Modelling Language (gml)
j            json
J            json with disarm
k            SDB key-value
t            Tiny ascii art
i            Interactive ascii art
```

NEW ASCII ART GRAPHS VS OLD XDOT OUTPUT



TASK 5: UTF-8 BASED IMPROVEMENTS:

- All the reflines in disarm were of same color :|
- Enhanced it to have different colors for up and down arrows

THE NEW DISARM

```
[0x100001276 [xAdvc]0 0% 316 /bin/ls]> pd $r @ main+142 # 0x100001276
0x100001276          0fb745ca    movzx eax, word [rbp - 0x36]
0x10000127d          85c0        test eax, eax
0x10000127e          7406        je 0x100001284
0x100001284          89054c420000  mov dword [section.10._DATA._data], eax ; [0x1000054d0:4]-80 ; U"P."
0x10000128a          c705a6430000  mov dword [0x100005634], 1 ; [0x100005634:4]=0
0x10000128e          eb26        jmp 0x1000012b6
0x100001290          c60569420000  mov byte [section.11._DATA._bss], 1 ; [0x100005500:1]=-0
0x100001297          488d3d4db3800  lea rdi, str.COLUMNNS ; 0x100004ae9 ; "COLUMNNS"
0x10000129e          e83d320000  call sym.imp.getenv ; [1]
0x1000012a3          4885c0        test rax, rax
0x1000012a6          740e        je 0x1000012b6
0x1000012a8          4889c7        mov rdi, rax
0x1000012ab          e8d0310000  call sym.imp.oatoi ; [2]
0x1000012b0          89051a420000  mov dword [section.10._DATA._data], eax ; [0x1000054d0:4]-80 ; U"P."
0x1000012b6          e85f320000  call sym.imp.getuid ; [3]
0x1000012b7          89051a420000  mov dword [section.10._DATA._data], eax ; [0x1000054d0:4]-80 ; U"P."
0x1000012b9          7507        test al, al
0x1000012b9          89051a420000  mov byte [0x100005504], 1 ; [0x100005504:1]=-0
0x1000012cc          4c8d2d1e3800  lea r13, str.1.ABCFGHLOPRSTUAbcdefghijklmnopqrstuvwxyz ; 0x100004ef1 ; "1@ABCFGHLOPRSTUAbcdefghijklmnopqrstuvwxyz"
0x1000012da          488d1d320700  lea rbx, [0x100001a0c]
0x1000012d0          eb0b        jmp 0x1000012e7
0x1000012d0          e811310000  call sym.func.1000043f2 ; [4]
0x1000012d1          89051a420000  mov dword [0x100005630], 0
0x1000012d3          4889c01      mov edi, r14d
0x1000012d7          4889f7        mov rsi, r15
0x1000012e2          4c89fe        mov rdx, r13
0x1000012ed          4c89ea        mov rdx, r13
0x1000012f0          e8f1310000  call sym.imp getopt ; [5]
0x1000012f5          8d48cf        lea ecx, [rax - 0x31]
0x1000012f8          83f947        cmp ecx, 0x47 ; 71
0x1000012f9          0f8781030000  ja 0x100001682
0x100001301          48653048b     msxd rax, dword [rbx + rcx*4]
0x100001305          48653048b     add rax, rbx
0x100001308          Fc00        shr rax
0x10000130a          c605ef410000  mov byte [section.11._DATA._bss], 1 ; [0x100005500:1]=-0
0x100001311          31c0        xor eax, eax
0x100001313          890517430000  mov dword [0x100005630], eax ; [0x100005630:4]=0
0x100001319          890541430000  mov dword [0x100005660], eax ; [0x100005660:4]=0
0x10000131f          eb05        jmp 0x1000012e7
0x100001321          c6050410000  mov byte [0x100005518], 1 ; [0x100005518:1]=-0
0x100001323          89051a420000  mov dword [0x100005630], eax ; [0x100005630:4]=0
0x100001325          c70524430000  mov dword [0x100005658], 1 ; [0x100005658:4]=0
0x100001334          c705f2420000  mov dword [0x100005630], 0 ; [0x100005630:4]=0
0x10000133e          c605b410000  mov byte [section.11._DATA._bss], 0 ; [0x100005500:1]=-0
0x100001345          eba0        jmp 0x1000012e7
0x100001347          c705e3420000  mov dword [0x100005634], 0 ; [0x100005634:4]=0
0x100001351          eb94        jmp 0x1000012e7
0x100001353          c705ff420000  mov dword [0x10000565c], 1 ; [0x10000565c:4]=-0
0x10000135d          31c0        xor eax, eax
0x10000135f          8905b3420000  mov dword [0x100005618], eax ; [0x100005618:4]=-0
0x100001365          8905d5420000  mov dword [0x100005610], eax ; [0x100005610:4]=0
0x10000136b          e97fffff      jmp 0x1000012e7
0x100001370          c60599410000  mov byte [0x100005510], 1 ; [0x100005510:1]=-0
0x100001377          488d3d9b3700  lea rdi, str.bin_ls ; 0x100004b19 ; "bin_ls"
0x10000137e          488d359b3700  lea rsi, str.Unix2003 ; 0x100004b20 ; "Unix2003"
0x100001385          e802310000  call sym.imp.compat_mode ; [6]
0x10000138a          84c0        test al, al
0x10000138c          0f1845fffffff  je 0x1000012e7
0x100001392          4188cc20        or r12d, 0x20 ; 32
```

TASK 5: UTF-8 BASED IMPROVEMENTS:

- Implemented commands to show ascii-art colour bars with the section ranges
 - *ob=* (for open files)
 - *dt=* (Debug traces)
 - *afl=* (function lists)
 - *afb=* (basic blocks)

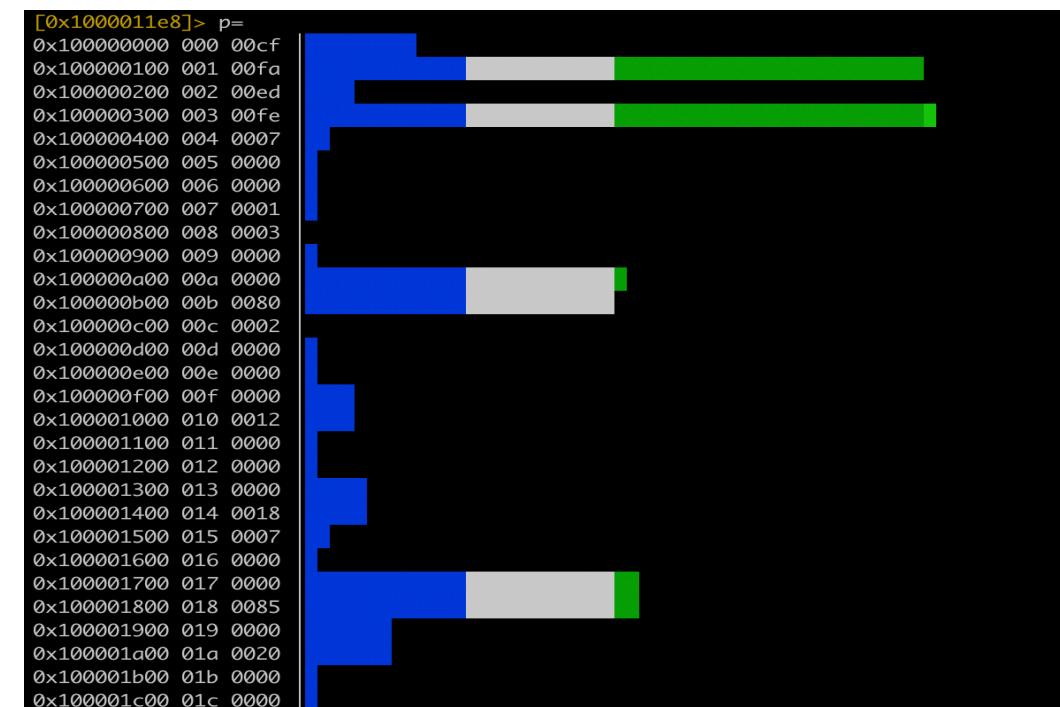
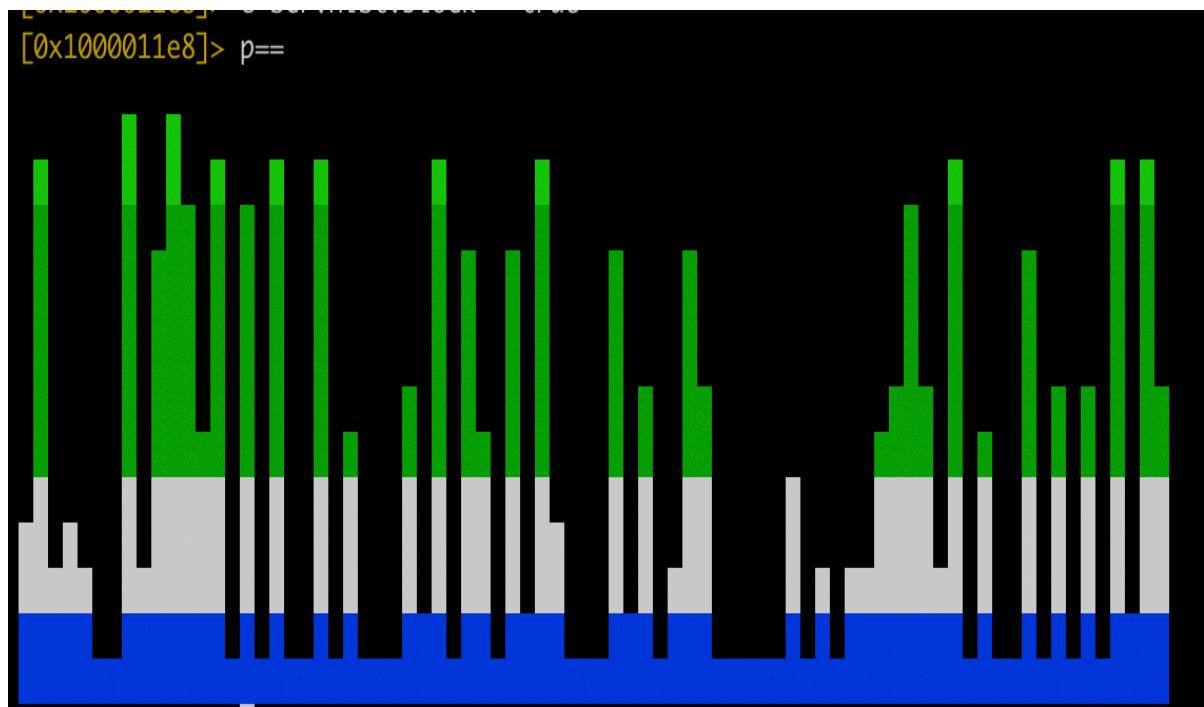
ASCII-ART COLOUR BARS

```
[0x10000011e8]- ob-          3 /bin/ls
***0x1000000000: #####0x1000009730
0x10000011e8 |-----^-----| 0x1000012e8
[0x10000011e8]- ocl-          64 sym._mh_execute_header
000 0x1000000000 #: 64 sym.fcn.10000026f
001 0x100000026f |---# 64 fcn.10000066a
002 0x100000066e |---# 64 sym.fcn.100000680
003 0x1000000108 #: 64 sym.fcn.1000001fd
004 0x10000001fd |---# 64 sym.fcn.1000002f2
005 0x100000012f |---# 64 sym.fcn.100000f74
006 0x100000074 #: 64 sym.fcn.100000fc2
007 0x1000000fc2 #: 64 sym.fcn.1000001907
008 0x1000001007 #: 64 sym.fcn.1000001055
009 0x1000001055 #: 64 sym.fcn.1000001055
010 0x100000109a #: 64 sym.fcn.100000109a
011 0x10000010e8 #: 64 sym.fcn.10000010e8
012 0x100000111a #: 64 sym.fcn.100000111a
013 0x1000001155 #: 64 sym.fcn.1000001155
014 0x100000119a #: 64 sym.fcn.100000119a
015* 0x10000011e8 #####0x10000011e8
016 0x1000001b2c #####0x10000011d5
017 0x1000001df0 |---# 64 sym.fcn.1000001e5f
018 0x1000001e5f #####0x1000002866
019 0x1000002884 |---# 64 sym.fcn.1000002950
020 0x1000002950 |---# 64 sym.fcn.1000002950
021 0x1000002d6d |---# 64 sym.fcn.1000002d6d
022 0x1000003358 |---# 64 sym.fcn.1000003358
023 0x1000003498 |---# 64 sym.fcn.1000003498
024 0x100000357c |---# 64 sym.fcn.100000357c
025 0x10000035b2 |---# 64 sym.fcn.10000035b2
026 0x1000003660 |---# 64 sym.fcn.1000003660
027 0x100000373d |---# 64 sym.fcn.100000373d
028 0x100000350 |---# 64 sym.fcn.100000350
029 0x1000003bcd |---# 64 sym.fcn.1000003bcd
030 0x100000337 |---# 64 sym.fcn.100000337
031 0x10000035d |---# 64 sym.fcn.10000035d
032 0x10000036a |---# 64 sym.fcn.10000036a
033 0x1000004026 |---# 64 sym.fcn.1000004026
034 0x100000417d |---# 64 sym.fcn.100000417d
035 0x1000004420 |---# 64 sym.fcn.1000004420
036 0x1000004426 |---# 64 sym.fcn.1000004426
037 0x100000442c |---# 64 sym.fcn.100000442c
038 0x1000004432 |---# 64 sym.fcn.1000004432
039 0x1000004438 |---# 64 sym.fcn.1000004438
040 0x100000443e |---# 64 sym.fcn.100000443e
041 0x1000004444 |---# 64 sym.fcn.1000004444
042 0x1000004448 |---# 64 sym.fcn.1000004448
043 0x1000004450 |---# 64 sym.fcn.1000004450
044 0x1000004456 |---# 64 sym.fcn.1000004456
045 0x100000445c |---# 64 sym.fcn.100000445c
046 0x1000004462 |---# 64 sym.fcn.1000004462
047 0x1000004468 |---# 64 sym.fcn.1000004468
048 0x100000446e |---# 64 sym.fcn.100000446e
049 0x1000004474 |---# 64 sym.fcn.1000004474
050 0x100000447a |---# 64 sym.fcn.100000447a
051 0x1000004480 |---# 64 sym.fcn.1000004480
052 0x1000004486 |---# 64 sym.fcn.1000004486
053 0x100000448c |---# 64 sym.fcn.100000448c
054 0x1000004492 |---# 64 sym.fcn.1000004492
055 0x1000004498 |---# 64 sym.fcn.1000004498
056 0x100000449e |---# 64 sym.fcn.100000449e
[0x10000011e8]- imp-          64 sym.imp._assert_rtn
057 0x10000044a0 |---# 64 sym.imp._bzero
058 0x10000044a2 |---# 64 sym.imp._error
059 0x10000044a3 |---# 64 sym.imp._maskrune
060 0x10000044a4 |---# 64 sym.imp._snprintf_chk
061 0x10000044a5 |---# 64 sym.imp._stack_chk_fail
062 0x10000044a6 |---# 64 sym.imp._tolower
063 0x10000044a7 |---# 64 sym.imp._free
064 0x10000044a8 |---# 64 sym.imp._get_entry
065 0x10000044a9 |---# 64 sym.imp._get_flag_np
066 0x10000044a9 |---# 64 sym.imp._get_flagset_np
067 0x10000044a9 |---# 64 sym.imp._get_link_np
068 0x10000044a9 |---# 64 sym.imp._get_perm_np
069 0x10000044a9 |---# 64 sym.imp._get_permset
070 0x10000044a9 |---# 64 sym.imp._get_qualifier
071 0x10000044a9 |---# 64 sym.imp._get_tag_type
072 0x10000044a9 |---# 64 sym.imp._atoi
073 0x10000044a9 |---# 64 sym.imp._calloc
074 0x10000044a9 |---# 64 sym.imp._compat_mode
075 0x10000044a9 |---# 64 sym.imp._err
076 0x10000044a9 |---# 64 sym.imp._exit
```

TASK 5: UTF-8 BASED IMPROVEMENTS:

- Added hist graphs support for p= and p==
- Can be enabled by setting *scr.hist.block = true*

HIST GRAPHS



TASK 5: UTF-8 BASED IMPROVEMENTS:

- Added the UTF-8 support to p= and p==, which are used when hist mode is disabled

```
0x1000011e8 0.11 0000  
[0x1000011e8]> p==  
  
[0x1000011e8]>
```

TASK 6: TABLE API

- Due to time constraints, I was not able to work on this
- But pancake built it and it supports a lot of cool features like:
 - Special querying language to manipulate the table data
 - Exporting the graph data into CSV, SDB and other formats

TABLE API IN ACTION

```
[0x1000012a3]> p-h
```

offset	flags	funcs	cmts	syms	str
0x100000000	2	0	0	1	0
0x100000ee0	17	0	1	13	0
0x100001a08	1	0	0	1	0
0x100001dc0	2	0	0	2	0
0x100002530	1	0	0	1	0
0x1000028e8	2	0	0	2	0
0x100003058	2	0	0	2	0
0x100003410	5	0	0	5	0
0x1000037c8	1	0	0	1	0
0x100003b80	6	0	0	6	0
0x100003f38	2	0	0	2	0
0x1000042f0	79	0	2	77	0
0x1000046a8	6	0	1	0	5
0x100004a60	53	0	1	0	53
0x100004e18	86	0	4	0	24
0x1000051d0	28	0	3	0	0
0x100005588	1	0	1	0	0
0x100005cf8	1	0	0	0	0
	295	0	13	113	82

FUTURE WORK

- Add UTF-8 support for table API
- Integrate table API wherever necessary
- Build a tree API

ANY QUESTIONS?





THANK YOU

