
Lost and Found
Oh memset, where did you go??

Disclaimer

The opinions and positions expressed herein are mine only and do not represent the views of any current or previous employer, including Intel Corporation or its affiliates.

This presentation has no intention to advertise or devalue any current or future technology.

No database software was harmed in the making of this presentation. This research is not focused on DSE optimization bugs.

Hello, it's me!

Marion Marschalek

Security Researcher with Intel STORM Team

@pinkflawd | marion@0x1338.at

Why are we here?

Builtins and intrinsics are terribly fascinating and frequently misunderstood

Builtins as a vehicle of attack

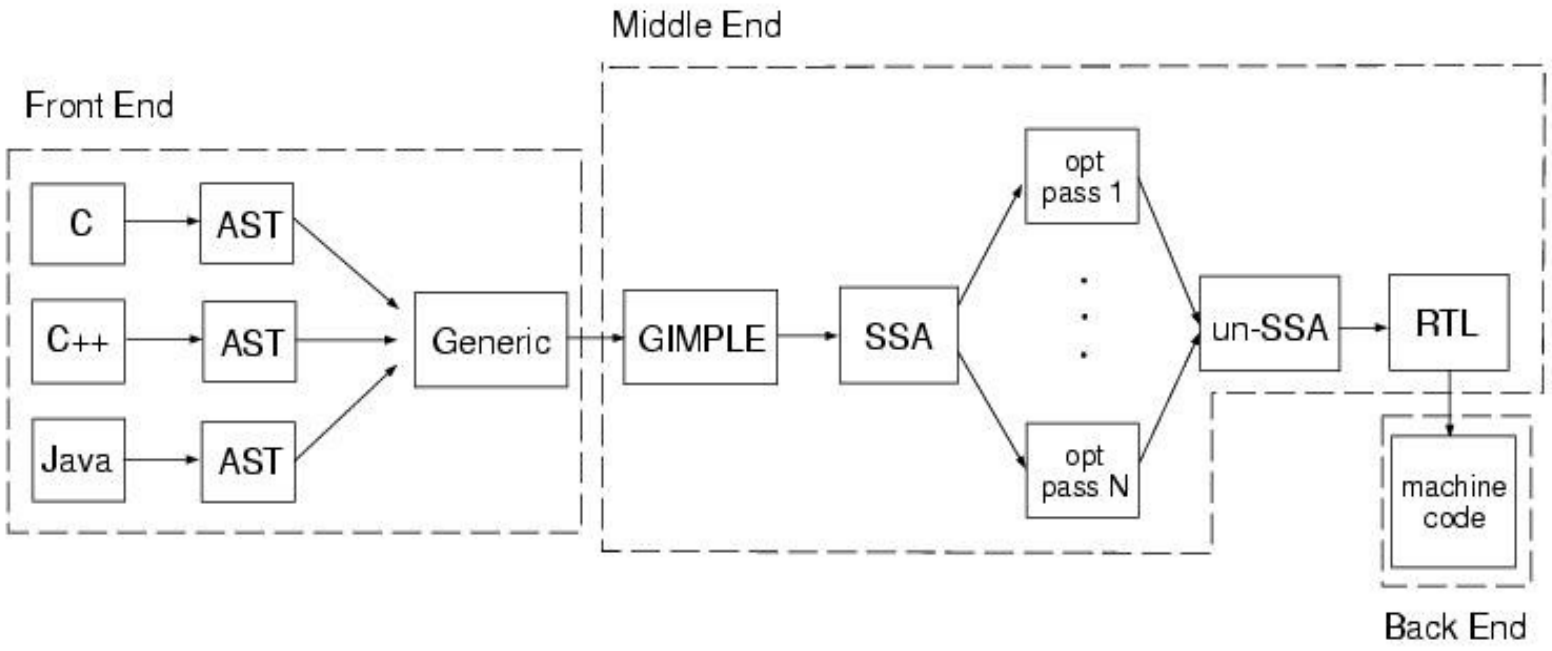
Builtins as a starting point of defense

Builtins & Intrinsics

From the docs:

- GCC provides a large number of **built-in** functions, for internal use, and for **optimization purposes of standard C library** functions
  - __builtin_puts, __builtin_alloca, __builtin_memcpy, etc. etc. etc.
- GCC **intrinsics** are built-in functions that help the developer use domain specific operations, and help the compiler **leverage machine specific functionality**
  - Vector operations, signal processing, interrupt handling, etc. etc. etc.
- Compiler can replace builtins with **custom implementation** if provided

GCC and GCC Builtins



Compiler provides its own implementation of a library function

Tailored to use case and target platform

Mostly used throughout compilation process, not by developers

Optimizing on a case by case basis, eg.:

- printf replacement
- memcpy inlining
- strlen removal
- sizeof removal

How does that look like inside?

builtins.def

builtins.h

builtins.c

xxx-builtin.def

xxxintrin.h

.. and many many more..

```
static rtx
expand_builtin_mempcpy (tree exp, rtx target)
{
  if (!validate_arglist (exp,
                         POINTER_TYPE, POINTER_TYPE, INTEGER_TYPE, VOID_TYPE))
    return NULL_RTX;

  tree dest = CALL_EXPR_ARG (exp, 0);
  tree src = CALL_EXPR_ARG (exp, 1);
  tree len = CALL_EXPR_ARG (exp, 2);

  /* Avoid expanding mempcpy into memcpy when the call is determined
     to overflow the buffer.  This also prevents the same overflow
     from being diagnosed again when expanding memcpy.  */
  if (!check_memop_sizes (exp, dest, src, len))
    return NULL_RTX;

  return expand_builtin_mempcpy_args (dest, src, len,
                                      target, exp, /*endp=*/ 1);
}
```

```
692  DEF_C99_COMPL_BUILTIN        (BUILT_IN_CTANHF, "ctanhf", BT_FN_COMPLEX_FLOAT_COMPLEX_FLOAT, ATTR_MATHFN_FPROUNDING)
693  DEF_C99_COMPL_BUILTIN        (BUILT_IN_CTANHL, "ctanhl", BT_FN_COMPLEX_LONGDOUBLE_COMPLEX_LONGDOUBLE, ATTR_MATHFN_FPROUNDING)
694  DEF_C99_COMPL_BUILTIN        (BUILT_IN_CTANL, "ctanl", BT_FN_COMPLEX_LONGDOUBLE_COMPLEX_LONGDOUBLE, ATTR_MATHFN_FPROUNDING)
695
696  /* Category: string/memory builtins.  */
697  DEF_EXT_LIB_BUILTIN    (BUILT_IN_BCMP, "bcmp", BT_FN_INT_CONST_PTR_CONST_PTR_SIZE, ATTR_PURE_NOTHROW_NONNULL_LEAF)
698  DEF_EXT_LIB_BUILTIN    (BUILT_IN_BCOPY, "bcopy", BT_FN_VOID_CONST_PTR_PTR_SIZE, ATTR_NOTHROW_NONNULL_LEAF)
699  DEF_EXT_LIB_BUILTIN    (BUILT_IN_BZERO, "bzero", BT_FN_VOID_PTR_SIZE, ATTR_NOTHROW_NONNULL_LEAF)
700  DEF_EXT_LIB_BUILTIN    (BUILT_IN_INDEX, "index", BT_FN_STRING_CONST_STRING_INT, ATTR_PURE_NOTHROW_NONNULL_LEAF)
701  DEF_LIB_BUILTIN        (BUILT_IN_MEMCHR, "memchr", BT_FN_PTR_CONST_PTR_INT_SIZE, ATTR_PURE_NOTHROW_NONNULL_LEAF)
702  DEF_LIB_BUILTIN        (BUILT_IN_MEMCMP, "memcmp", BT_FN_INT_CONST_PTR_CONST_PTR_SIZE, ATTR_PURE_NOTHROW_NONNULL_LEAF)
703  DEF_LIB_BUILTIN        (BUILT_IN_MEMCPY, "memcpy", BT_FN_PTR_PTR_CONST_PTR_SIZE, ATTR_RET1_NOTHROW_NONNULL_LEAF)
704  DEF_LIB_BUILTIN        (BUILT_IN_MEMMOVE, "memmove", BT_FN_PTR_PTR_CONST_PTR_SIZE, ATTR_RET1_NOTHROW_NONNULL_LEAF)
705  DEF_EXT_LIB_BUILTIN    (BUILT_IN_MEMPCPY, "mempcpy", BT_FN_PTR_PTR_CONST_PTR_SIZE, ATTR_RETNONNULL_NOTHROW_LEAF)
706  DEF_LIB_BUILTIN        (BUILT_IN_MEMSET, "memset", BT_FN_PTR_PTR_INT_SIZE, ATTR_RET1_NOTHROW_NONNULL_LEAF)
707  DEF_EXT_LIB_BUILTIN    (BUILT_IN_RINDEX, "rindex", BT_FN_STRING_CONST_STRING_INT, ATTR_PURE_NOTHROW_NONNULL_LEAF)
708  DEF_EXT_LIB_BUILTIN    (BUILT_IN_STPCPY, "stpcpy", BT_FN_STRING_STRING_CONST_STRING, ATTR_RETNONNULL_NOTHROW_LEAF)
709  DEF_EXT_LIB_BUILTIN    (BUILT_IN_STPNCPY, "stpncpy", BT_FN_STRING_STRING_CONST_STRING_SIZE, ATTR_RETNONNULL_NOTHROW_LEAF)
710  DEF_EXT_LIB_BUILTIN    (BUILT_IN_STRCASECMP, "strcasecmp", BT_FN_INT_CONST_STRING_CONST_STRING, ATTR_PURE_NOTHROW_NONNULL_LEAF)
711  DEF_LIB_BUILTIN        (BUILT_IN_STRCAT, "strcat", BT_FN_STRING_STRING_CONST_STRING, ATTR_RET1_NOTHROW_NONNULL_LEAF)
712  DEF_LIB_BUILTIN        (BUILT_IN_STRCHR, "strchr", BT_FN_STRING_CONST_STRING_INT, ATTR_PURE_NOTHROW_NONNULL_LEAF)
713  DEF_LIB_BUILTIN        (BUILT_IN_STRCMP, "strcmp", BT_FN_INT_CONST_STRING_CONST_STRING, ATTR_PURE_NOTHROW_NONNULL_LEAF)
```

A vast choice of functions the compiler "understands"

Builtin-spotting Opportunities

- PaX' STACKLEAK (https://pax.grsecurity.net/docs/PaXTeam-H2HC13-PaX-gcc-plugins.pdf, https://code.woboq.org/linux/linux/scripts/gcc-plugins/stackleak_plugin.c.html, https://a13xp0p0v.github.io/2018/11/04/stackleak.html)
- Kostya Serebryany's AddressSanitizer (https://github.com/google/sanitizers/wiki/AddressSanitizer, https://code.woboq.org/gcc/gcc/asan.c.html)

What can an attacker do alongside the compiler's own mechanisms?

Why are we curious about tracking builtins?

All the Passes

```
*warn_unused_result
*diagnose_omp_blocks
*diagnose_tm_blocks
tree-omplower
tree-lower
tree-tmlower
tree-ehopt
tree-eh
tree-cfg
*warn_function_return
tree-ompexp
tree-printf-return-value1
tree-wallocal
*build_cgraph_edges
*free_lang_data
ipa-visibility
ipa-chkp_versioning
ipa-chkp_ecleanup
ipa-build_ssa_passes
   tree-fixup_cfg1
   tree-ssa
   *nonnullcmp
   tree-ubsan
   *early_warn_uninitialized
   tree-nothrow
   *rebuild_cgraph_edges
ipa-chkp_passes
   tree-fixup_cfg2
   tree-chkp
   *rebuild_cgraph_edges
ipa-opt_local_passes
   tree-fixup_cfg3
   *rebuild_cgraph_edges
   tree-local-fnsummary1
   tree-einline
   tree-early_optimizations
      *remove_cgraph_callee_edges
      tree-objsz1
      tree-ccp1
      tree-forwprop1
      tree-ethread
      tree-esra
      tree-ealias
      tree-fre1
      tree-evrp
      tree-mergephi1
      tree-dse1
      tree-cddce1
      tree-eipa_sra
      tree-tailr1
      tree-switchconv
      tree-ehcleanup1
      tree-profile_estimate
      tree-local-pure-const1
      tree-fnsplit
      *strip_predict_hints
   tree-release_ssa
   *rebuild_cgraph_edges
   tree-local-fnsummary2
ipa-ipa_oacc
   ipa-pta1
   ipa-ipa_oacc_kernels
      tree-oacc_kernels
         tree-ch1
         tree-fre2
         tree-lim1
         tree-dom1
         tree-dce1
         tree-parloops1
         tree-ompexpssa1
         *rebuild_cgraph_edges
ipa-targetclone
ipa-chkp_cleanup
ipa-afdo
ipa-profile
   tree-feedback_fnsplit
ipa-free-fnsummary1
ipa-increase_alignment
ipa-tmipa
ipa-emutls
ipa-whole-program
ipa-profile_estimate
ipa-icf
ipa-devirt
ipa-cp
ipa-cdtor
ipa-hsa
ipa-fnsummary
ipa-inline
ipa-pure-const
ipa-free-fnsummary2
ipa-static-var
ipa-single-use
ipa-comdats
ipa-materialize-all-clones
ipa-pta2
ipa-simdclone
tree-fixup_cfg4
tree-ehdisp
tree-oaccdevlow
tree-ompdevlow
tree-omptargetlink
*all_optimizations
   *remove_cgraph_callee_edges
   *strip_predict_hints
   tree-ccp2
   tree-post_ipa_warn1
   tree-cunrolli
   tree-backprop
   tree-phiprop
   tree-forwprop2
   tree-objsz2
   tree-alias
   tree-retslot
   tree-fre3
   tree-mergephi2
   tree-thread1
   tree-vrp1
   tree-chkpopt
   tree-dce2
   tree-stdarg
   tree-cdce
   tree-cselim
   tree-copyprop1
   tree-ifcombine
   tree-mergephi3
   tree-phiopt1
   tree-tailr2
   tree-ch2
   tree-cplxlower1
   tree-sra
   tree-thread2
   tree-dom2
   tree-isolate-paths
   tree-phicprop1
   tree-dse2
tree-reassoc1
tree-dce3
tree-forwprop3
tree-phiopt2
tree-ccp3
tree-sincos
tree-bswap
tree-laddress
tree-lim2
tree-walloca2
tree-pre
tree-sink
tree-sancov1
tree-asan1
tree-tsan1
tree-dce4
tree-fix_loops
tree-loop
   tree-loopinit
   tree-unswitch
   tree-sccp
   tree-lsplit
   tree-unrolljam
   tree-cddce2
   tree-ivcanon
   tree-ldist
   tree-linterchange
   tree-copyprop2
   tree-graphite0
      tree-graphite
      tree-lim3
      tree-copyprop3
      tree-dce5
   tree-parloops2
   tree-ompexpssa2
   tree-ch_vect
   tree-ifcvt
   tree-vect
      tree-dce6
   tree-pcom
   tree-cunroll
   tree-slp1
   tree-aprefetch
   tree-ivopts
   tree-lim4
   tree-loopdone
tree-no_loop
   tree-slp2
tree-simduid1
tree-veclower21
tree-recip
tree-printf-return-value2
tree-reassoc2
tree-slsr
tree-split-paths
tree-tracer
tree-thread3
tree-dom3
tree-strlen
tree-thread4
tree-vrp2
tree-wrestrict
tree-phicprop2
tree-dse3
tree-cddce3
tree-forwprop4
tree-phiopt3
tree-fab1
   tree-widening_mul
   tree-store-merging
   tree-tailc
   tree-dce7
   tree-crited1
   tree-uninit1
   tree-uncprop1
   tree-local-pure-const2
*all_optimizations_g
   *remove_cgraph_callee_edges
   *strip_predict_hints
   tree-cplxlower2
   tree-veclower22
   tree-ccp4
   tree-post_ipa_warn2
   tree-objsz3
   tree-fab2
   tree-printf-return-value3
   tree-copyprop4
   tree-dce8
   tree-sancov2
   tree-asan2
   tree-tsan2
   tree-crited2
   tree-uninit2
   tree-uncprop2
   tree-local-pure-const3
*tminit
   tree-tmmark
   tree-tmmemopt
   tree-tmedge
tree-simduid2
tree-vtable-verify
tree-lower_vaarg
tree-veclower
tree-cplxlower0
tree-sancov_O0
tree-switchlower
tree-asan0
tree-tsan0
tree-sanopt
tree-ehcleanup2
tree-resx
tree-nrv
tree-optimized
*warn_function_noreturn
tree-hsagen
rtl-expand
*rest_of_compilation
   rtl-vregs
   rtl-into_cfglayout
   rtl-jump
   rtl-subreg1
   rtl-dfinit
   rtl-cse1
   rtl-fwprop1
   rtl-cprop1
   rtl-rtl_pre
   rtl-hoist
   rtl-cprop2
   rtl-store_motion
   rtl-cse_local
   rtl-ce1
   rtl-reginfo
   rtl-loop2
      rtl-loop2_init
      rtl-loop2_invariant
      rtl-loop2_unroll
      rtl-loop2_doloop
      rtl-loop2_done
   rtl-web
   rtl-cprop3
   rtl-stv1
   rtl-cse2
   rtl-dse1
   rtl-fwprop2
   rtl-auto_inc_dec
   rtl-init-regs
   rtl-ud_dce
   rtl-combine
   rtl-stv2
   rtl-ce2
   rtl-bbpart
   rtl-outof_cfglayout
   rtl-split1
   rtl-subreg2
   rtl-no-opt dfinit
   *stack_ptr_mod
   rtl-mode_sw
   rtl-asmcons
   rtl-sms
   rtl-lr_shrinkage
   rtl-sched1
   rtl-early_remat
   rtl-ira
   rtl-reload
   rtl-vzeroupper
   *all-postreload
      rtl-postreload
      rtl-gcse2
      rtl-split2
      rtl-ree
      rtl-cmpelim
      rtl-bt11
      rtl-pro_and_epilogue
      rtl-dse2
      rtl-csa
      rtl-jump2
      rtl-compgotos
      rtl-sched_fusion
      rtl-peephole2
      rtl-ce3
      rtl-rnreg
      rtl-cprop_hardreg
      rtl-rtl_dce
      rtl-bbro
      rtl-bt12
      *leaf_regs
      rtl-split4
      rtl-sched2
      *stack_regs
         rtl-split3
         rtl-stack
   *all-late_compilation
      rtl-alignments
      rtl-vartrack
      *free_cfg
      rtl-mach
      rtl-barriers
      rtl-dbr
      rtl-split5
      rtl-eh_ranges
      rtl-cet
      rtl-shorten
      rtl-nothrow
      rtl-dwarf2
      rtl-final
   rtl-dfinish
*clean_state
```

```
marion@gabrielskx:~/gcc/conference/sqlitedumps/sqlite3_O3$ ls
sqlite3.c.000i.cgraph
sqlite3.c.000i.ipa-clones
sqlite3.c.000i.type-inheritance
sqlite3.c.003t.original
sqlite3.c.004t.gimple
sqlite3.c.006t.omplower
sqlite3.c.007t.lower
sqlite3.c.010t.eh
sqlite3.c.011t.cfg
sqlite3.c.012t.ompexp
sqlite3.c.015i.visibility
sqlite3.c.018i.build_ssa_passes
sqlite3.c.019t.fixup_cfg1
sqlite3.c.020t.ssa
sqlite3.c.022t.nothrow
sqlite3.c.026i.opt_local_passes
sqlite3.c.027t.fixup_cfg3
sqlite3.c.028t.local-fnsummary1
sqlite3.c.029t.einline
sqlite3.c.030t.early_optimizations
sqlite3.c.031t.objsz1
sqlite3.c.032t.ccp1
sqlite3.c.033t.forwprop1
sqlite3.c.034t.ethread
sqlite3.c.035t.esra
sqlite3.c.036t.ealias
sqlite3.c.037t.fre1
sqlite3.c.038t.evrp
sqlite3.c.039t.mergephi1
sqlite3.c.040t.dse1
sqlite3.c.041t.cddce1
sqlite3.c.042t.eipa_sra
sqlite3.c.043t.tailr1
sqlite3.c.044t.switchconv
sqlite3.c.046t.profile_estimate
sqlite3.c.047t.local-pure-const1
sqlite3.c.048t.fnsplit
sqlite3.c.049t.release_ssa
sqlite3.c.050t.local-fnsummary2
sqlite3.c.062i.targetclone
sqlite3.c.067i.free-fnsummary1
sqlite3.c.071i.whole-program
sqlite3.c.072i.profile_estimate
sqlite3.c.073i.icf
sqlite3.c.074i.devirt
sqlite3.c.075i.cp
sqlite3.c.078i.fnsummary
sqlite3.c.079i.inline
sqlite3.c.080i.pure-const
sqlite3.c.081i.free-fnsummary2
sqlite3.c.082i.static-var
sqlite3.c.083i.single-use
sqlite3.c.084i.comdats
sqlite3.c.085i.materialize-all-clones
sqlite3.c.087i.simdclone
sqlite3.c.088t.fixup_cfg4
sqlite3.c.091t.ompdevlow
sqlite3.c.093t.ccp2
sqlite3.c.095t.cunrolli
sqlite3.c.096t.backprop
sqlite3.c.097t.phiprop
sqlite3.c.098t.forwprop2
sqlite3.c.099t.objsz2
sqlite3.c.100t.alias
sqlite3.c.101t.retslot
sqlite3.c.102t.fre3
sqlite3.c.103t.mergephi2
sqlite3.c.104t.thread1
sqlite3.c.105t.vrp1
sqlite3.c.107t.dce2
sqlite3.c.108t.stdarg
sqlite3.c.109t.cdce
sqlite3.c.110t.cselim
sqlite3.c.111t.copyprop1
sqlite3.c.112t.ifcombine
sqlite3.c.113t.mergephi3
sqlite3.c.114t.phiopt1
sqlite3.c.115t.tailr2
sqlite3.c.116t.ch2
sqlite3.c.117t.cplxlower1
sqlite3.c.118t.sra
sqlite3.c.119t.thread2
sqlite3.c.120t.dom2
sqlite3.c.121t.isolate-paths
sqlite3.c.122t.phicprop1
sqlite3.c.123t.dse2
sqlite3.c.124t.reassoc1
sqlite3.c.125t.dce3
sqlite3.c.126t.forwprop3
sqlite3.c.127t.phiopt2
sqlite3.c.128t.ccp3
sqlite3.c.129t.sincos
sqlite3.c.130t.bswap
sqlite3.c.131t.laddress
sqlite3.c.132t.lim2
sqlite3.c.134t.pre
sqlite3.c.135t.sink
sqlite3.c.139t.dce4
sqlite3.c.140t.fix_loops
sqlite3.c.141t.loop
sqlite3.c.142t.loopinit
sqlite3.c.143t.unswitch
sqlite3.c.144t.sccp
sqlite3.c.145t.lsplit
sqlite3.c.146t.unrolljam
sqlite3.c.147t.cddce2
sqlite3.c.148t.ivcanon
sqlite3.c.149t.ldist
sqlite3.c.150t.linterchange
sqlite3.c.151t.copyprop2
sqlite3.c.159t.ch_vect
sqlite3.c.160t.ifcvt
sqlite3.c.161t.vect
sqlite3.c.162t.dce6
sqlite3.c.163t.pcom
sqlite3.c.164t.cunroll
sqlite3.c.165t.slp1
sqlite3.c.167t.ivopts
sqlite3.c.168t.lim4
sqlite3.c.169t.loopdone
sqlite3.c.170t.no_loop
sqlite3.c.171t.slp2
sqlite3.c.173t.veclower21
sqlite3.c.175t.printf-return-value2
sqlite3.c.176t.reassoc2
sqlite3.c.177t.slsr
sqlite3.c.181t.dom3
sqlite3.c.182t.strlen
sqlite3.c.183t.thread4
sqlite3.c.184t.vrp2
sqlite3.c.186t.phicprop2
sqlite3.c.187t.dse3
sqlite3.c.188t.cddce3
sqlite3.c.189t.forwprop4
sqlite3.c.190t.phiopt3
sqlite3.c.191t.fab1
sqlite3.c.192t.widening_mul
sqlite3.c.193t.store-merging
sqlite3.c.194t.tailc
sqlite3.c.195t.dce7
sqlite3.c.196t.crited1
sqlite3.c.198t.uncprop1
sqlite3.c.199t.local-pure-const2
sqlite3.c.225t.switchlower
sqlite3.c.231t.nrv
sqlite3.c.232t.optimized
sqlite3.c.234r.expand
sqlite3.c.235r.vregs
sqlite3.c.236r.into_cfglayout
sqlite3.c.237r.jump
sqlite3.c.238r.subreg1
sqlite3.c.239r.dfinit
sqlite3.c.240r.cse1
sqlite3.c.241r.fwprop1
sqlite3.c.242r.cprop1
sqlite3.c.243r.pre
sqlite3.c.245r.cprop2
sqlite3.c.247r.cse_local
sqlite3.c.248r.ce1
sqlite3.c.249r.reginfo
sqlite3.c.250r.loop2
sqlite3.c.251r.loop2_init
sqlite3.c.252r.loop2_invariant
sqlite3.c.255r.loop2_done
sqlite3.c.257r.cprop3
sqlite3.c.258r.stv1
sqlite3.c.259r.dse2
sqlite3.c.263r.init-regs
sqlite3.c.264r.ud_dce
sqlite3.c.265r.combine
sqlite3.c.267r.ce2
sqlite3.c.268r.bbpart
sqlite3.c.269r.outof_cfglayout
sqlite3.c.270r.split1
sqlite3.c.271r.subreg2
sqlite3.c.273r.mode_sw
sqlite3.c.274r.asmcons
sqlite3.c.279r.ira
sqlite3.c.280r.reload
sqlite3.c.282r.postreload
sqlite3.c.283r.gcse2
sqlite3.c.284r.split2
sqlite3.c.285r.ree
sqlite3.c.286r.cmpelim
sqlite3.c.288r.pro_and_epilogue
sqlite3.c.289r.dse2
sqlite3.c.290r.csa
sqlite3.c.291r.jump2
sqlite3.c.292r.compgotos
sqlite3.c.294r.peephole2
sqlite3.c.295r.ce3
sqlite3.c.297r.cprop_hardreg
sqlite3.c.298r.rtl_dce
sqlite3.c.299r.bbro
sqlite3.c.301r.split4
sqlite3.c.302r.sched2
sqlite3.c.304r.stack
sqlite3.c.305r.alignments
sqlite3.c.306r.vartrack
sqlite3.c.307r.mach
sqlite3.c.308r.barriers
sqlite3.c.313r.shorten
sqlite3.c.314r.nothrow
sqlite3.c.315r.dwarf2
sqlite3.c.316r.final
sqlite3.c.317r.dfinish
sqlite3.c.318t.statistics
```

Compiler debug files

Relevant Concepts

1000-foot view: GIMPLE, SSA & RTL, optimization levels

10-foot view: Alice's Wonderland

998-foot view on compiler optimization

Level	CFLAG	Description
0	-O0	No optimizations enabled.
1	-O1	Optimizations are enabled that reduce code and execution time, but don't significantly increase compilation time.
2	-O2	All optimizations are enabled that reduce code and execution time, excluding those that involve a tradeoff between code size and speed.
3	-O3	Enables all optimizations at Level 2, plus those that can drastically increase code size and those that may not always improve performance.
s	-Os	Optimizes for size. Similar to Level 2, except without optimizations that could increase code size, plus additional optimizations to reduce code size.
fast	-Ofast	Optimization Level 3, plus additional optimizations that may violate the language standards.
g	-Og	Enables any optimizations that do not interfere with the debugger or significantly increase the compilation time.

```
$ gcc -Q -O2 --help=optimizers
```

shows optimizations enabled for e.g. O2

GCC recognizes O0, O1, O2, O3, Os, Ofast and Og

In detail, the number of possible optimizations is YUGE

Table: https://www.sciencedirect.com/science/article/pii/B9780128007266000124

GIMPLE – SIMPLE for GCC

The three address code representation

Target- and language independent optimization

```
# Calculate one solution to the [[quadratic equation]].
x = (-b + sqrt(b^2 - 4*a*c)) / (2*a)
```

```
t1 := b * b
t2 := 4 * a
t3 := t2 * c
t4 := t1 - t3
t5 := sqrt(t4)
t6 := 0 - b
t7 := t5 + t6
t8 := 2 * a
t9 := t7 / t8
x  := t9
```

https://en.wikipedia.org/wiki/Three-address_code

RTL – Register Transfer Language

RTL passes “implement” the machine definition

Machine definition reflects the processor ABI

Algebraic description of target instructions

Target dependent optimization, register allocation, machine code generation

```
(insn 26 25 27 2 (set (mem/c:TI (reg:DI 97) [0 MEM[(void *)&buf2]+0 S16 A128])
        (const_wide_int 0x206120736920736968742 06f6c6c6548)) "hellocompiler.c":23 -1
(nil))
```

```c
#include<stdio.h>
#include<string.h>

void main(void) {
    char name[30];
    char buf1[300], buf2[300], buf3[2000];
    char* longstring = "Hello this is a longer string to test wl
    char* moarr =   "adgweubsssssskaserjefbgggggggddddddddddddd
            "halweufhadrcfghvbfzvlhubnednfkzjesbf,hbdcv,hawevf,
            "aeknflalssodfingnlsufgnlarbngkjadfnvkjlaenrgilarrf
            "asrlgjnzndlfvnlejrkbglkzjdfbcfvkljabjerklgkbzdfkjv
            "aweednvafjgnagnkaejkjngkjaerngakej;kngak;ngkja;nb;

    int i, superlargesize = 5000;
    char superlarge[superlargesize];
    char otherlarge[superlargesize];

    memset(buf1, 0, 300);
    memset(buf2, 0, 300);

    // memcpy 1 -- 300 empty
    memcpy(buf1, buf2, 300);

    // memcpy 2 -- 13 Hello world
    memset(name, 0, 30);
    memcpy(name, "hello world!", 13);
    printf("%s\n", name);

    // memcpy 3 -- 78 long string
    memcpy(buf2, longstring, strlen(longstring));
    printf("%s\n", buf2);

    // memcpy 4 -- 600 longer string
    memcpy(buf3, moarr, strlen(moarr));
    printf("%s\n", buf3);

    // memcpy 5 -- 5000 super long string, repeating char
    for (i=0; i<superlargesize; i++) {
        superlarge[i]='x';
    }
    printf("%s\n", superlarge);

    memcpy(otherlarge, superlarge, strlen(superlarge));
    printf("%s\n", otherlarge);
}
```

hellocompiler.c

5 memcpy calls

300 byte empty buffer

13 byte hello world!

78 byte random string

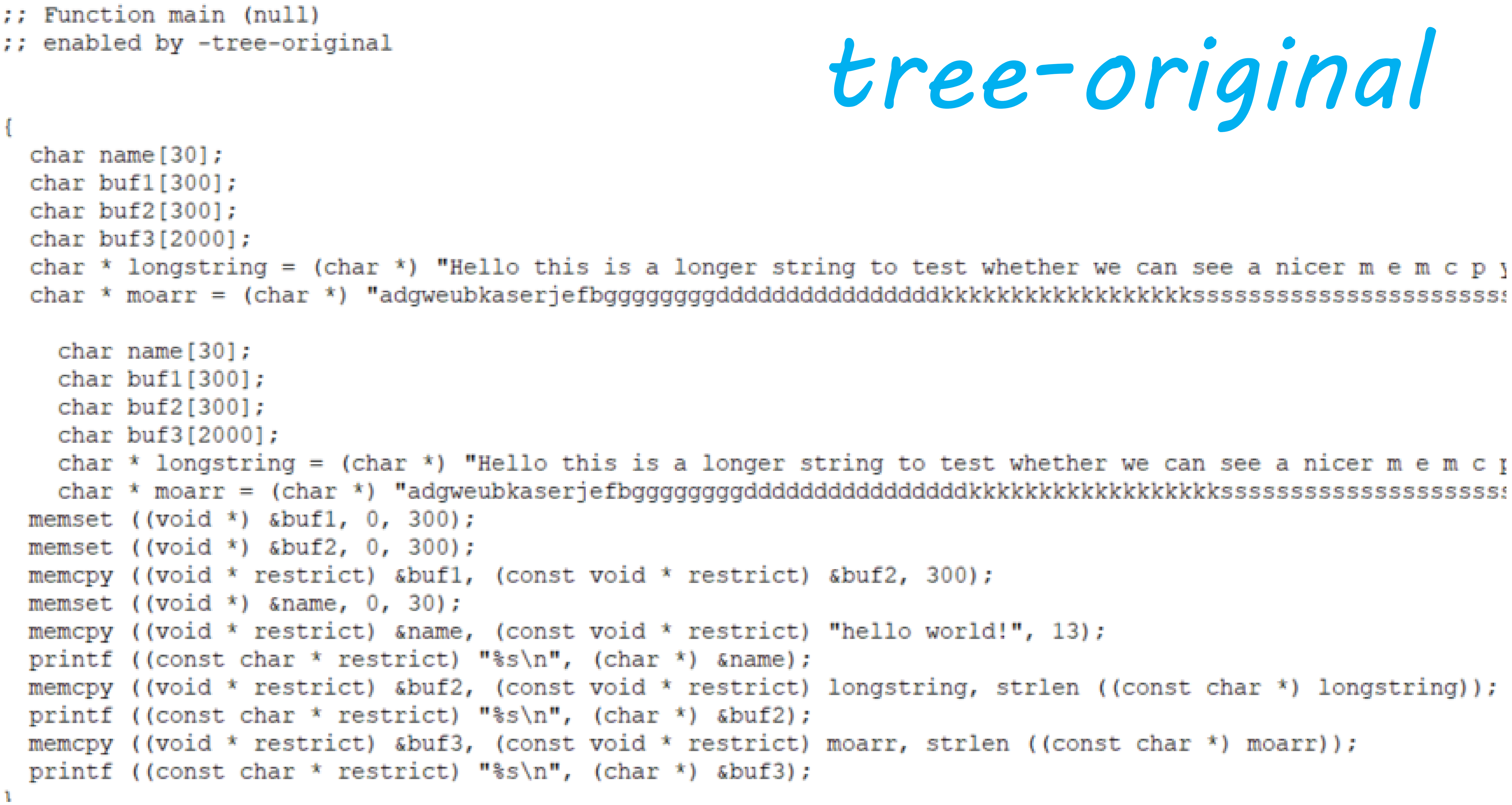
600 byte random string

5000 byte random string

Don’t forget the printfs to create dependencies!

Note: ignoring anything but “memcpy”, “memset”, etc.

tree-original



```
;; Function main (null)
;; enabled by -tree-original


{
  char name[30];
  char buf1[300];
  char buf2[300];
  char buf3[2000];
  char * longstring = (char *) "Hello this is a longer string to test whether we can see a nicer m e m c p
  char * moarr = (char *) "adgweubkaserjefbggggggggdddddddddddddddkkkkkkkkkkkkkkkkksssssssssssssssssssss

    char name[30];
    char buf1[300];
    char buf2[300];
    char buf3[2000];
    char * longstring = (char *) "Hello this is a longer string to test whether we can see a nicer m e m c
    char * moarr = (char *) "adgweubkaserjefbggggggggdddddddddddddddkkkkkkkkkkkkkkkkkssssssssssssssssss
  memset ((void *) &buf1, 0, 300);
  memset ((void *) &buf2, 0, 300);
  memcpy ((void * restrict) &buf1, (const void * restrict) &buf2, 300);
  memset ((void *) &name, 0, 30);
  memcpy ((void * restrict) &name, (const void * restrict) "hello world!", 13);
  printf ((const char * restrict) "%s\n", (char *) &name);
  memcpy ((void * restrict) &buf2, (const void * restrict) longstring, strlen ((const char *) longstring));
  printf ((const char * restrict) "%s\n", (char *) &buf2);
  memcpy ((void * restrict) &buf3, (const void * restrict) moarr, strlen ((const char *) moarr));
  printf ((const char * restrict) "%s\n", (char *) &buf3);
}
```

tree-gimple

```
main ()
{
  char name[30];
  char buf1[300];
  char buf2[300];
  char buf3[2000];
  char * longstring;
  char * moarr;

  try
    {
      longstring = "Hello this is a longer string to test whether we ca
      moarr = "adgweubkaserjefbgggggggdddddddddddddddddkkkkkkkkkkkkkkkk
      memset (&buf1, 0, 300);
      memset (&buf2, 0, 300);
      MEM[(char * {ref-all})&buf1] = MEM[(char * {ref-all})&buf2];
      memset (&name, 0, 30);
      memcpy (&name, "hello world!", 13);
      __builtin_puts (&name);
      _1 = strlen (longstring);
      memcpy (&buf2, longstring, _1);
      __builtin_puts (&buf2);
      _2 = strlen (moarr);
      memcpy (&buf3, moarr, _2);
      __builtin_puts (&buf3);
    }
  finally
    {
      name = {CLOBBER};
      buf1 = {CLOBBER};
      buf2 = {CLOBBER};
      buf3 = {CLOBBER};
    }
}
```

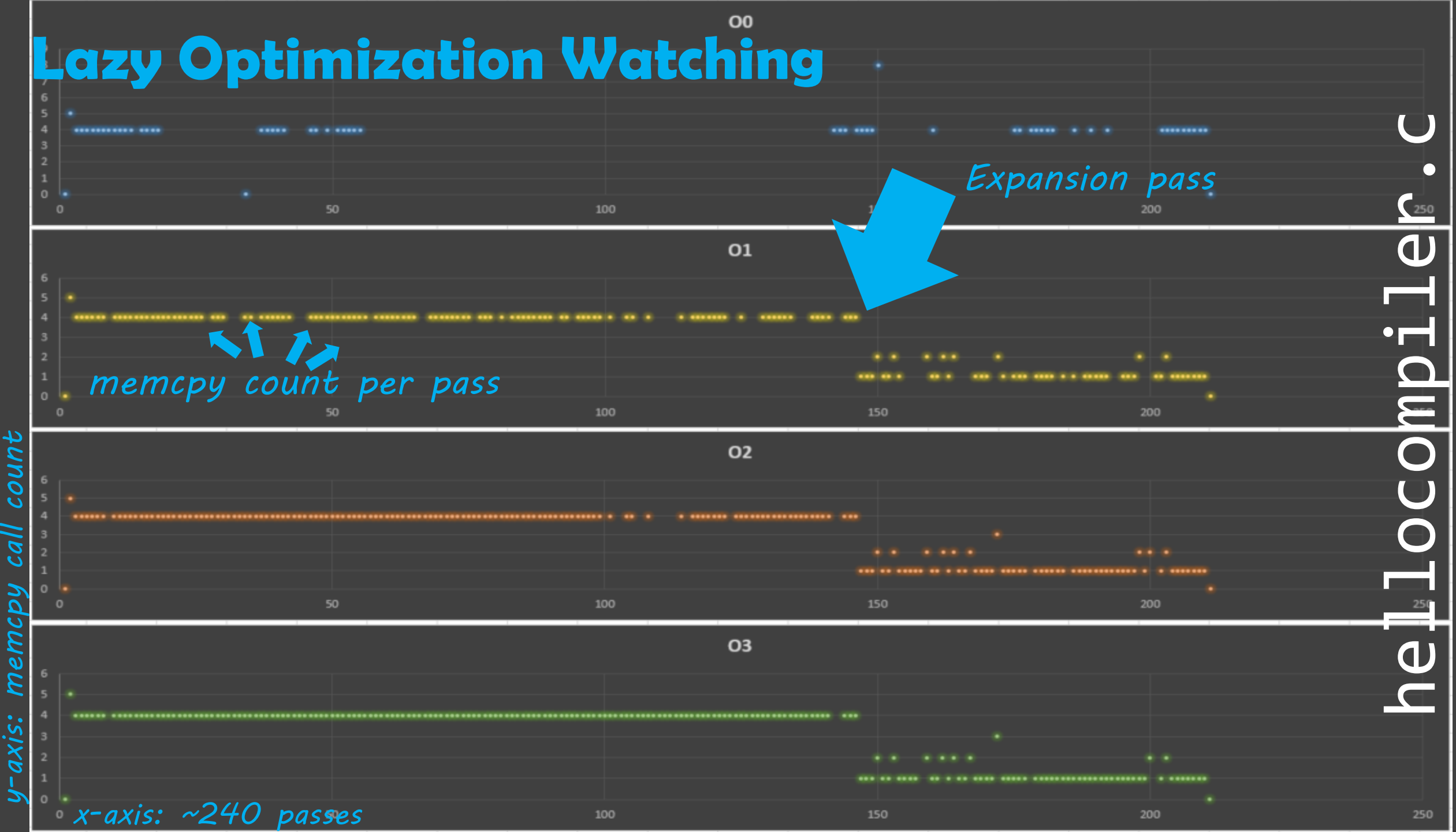
tree-optimized

```
;; Function main (main, funcdef_no=11, decl_uid=2575,
cgraph_uid=11, symbol_order=11) (executed once)

main ()
{
  char buf3[2000];
  char buf2[300];
  char name[30];

  <bb 2> [local count: 1073741825]:
  memset (&buf2, 0, 300);
  memset (&MEM[(void *)&name + 12B], 0, 18);
  memcpy (&name, "hello world!", 13);
  __builtin_puts (&name);
  memcpy (&buf2, "Hello this is a longer string to test whethe
  __builtin_puts (&buf2);
  memcpy (&buf3, "adgweubkaserjefbgggggggdddddddddddddddddkkkk
  __builtin_puts (&buf3);
  name ={v} {CLOBBER};
  buf2 ={v} {CLOBBER};
  buf3 ={v} {CLOBBER};
  return;

}
```

rtl-expand

```
(insn 26 25 27 2 (set (mem/c:TI (reg:DI 97) [0 MEM[(void *)&buf2]+0 S16 A128])
        (const_wide_int 0x206120736920736968742 06f6c6c6548)) "hellocompiler.c":23 -1
     (nil))
(insn 27 26 28 2 (set (mem/c:TI (plus:DI (reg:DI 97)
                (const_int 16 [0x10])) [0 MEM[(void *)&buf2]+16 S16 A128])
        (const_wide_int 0x6f7420676e69727473207265676e6f6c)) "hellocompiler.c":23 -1
     (nil))
(insn 28 27 29 2 (set (mem/c:TI (plus:DI (reg:DI 97)
                (const_int 32 [0x20])) [0 MEM[(void *)&buf2]+32 S16 A128])
        (const_wide_int 0x65772072656874656877207473657420)) "hellocompiler.c":23 -1
     (nil))
(insn 29 28 30 2 (set (mem/c:TI (plus:DI (reg:DI 97)
                (const_int 48 [0x30])) [0 MEM[(void *)&buf2]+48 S16 A128])
        (const_wide_int 0x726563696e206120656573206e616320)) "hellocompiler.c":23 -1
     (nil))
(insn 30 29 31 2 (set (mem/c:TI (plus:DI (reg:DI 97)
                (const_int 64 [0x40])) [0 MEM[(void *)&buf2]+64 S16 A128])
        (const_wide_int 0x697562207920702063206d2065206d20)) "hellocompiler.c":23 -1
     (nil))
(insn 31 30 32 2 (set (mem/c:SI (plus:DI (reg:DI 97)
                (const_int 80 [0x50])) [0 MEM[(void *)&buf2]+80 S4 A128])
        (const_int 1852404844 [0x6e69746c])) "hellocompiler.c":23 -1
     (nil))
(insn 32 31 33 2 (set (mem/c:HI (plus:DI (reg:DI 97)
                (const_int 84 [0x54])) [0 MEM[(void *)&buf2]+84 S2 A32])
        (const_int 8481 [0x2121])) "hellocompiler.c":23 -1
     (nil))
(insn 33 32 34 2 (set (mem/c:QI (plus:DI (reg:DI 97)
                (const_int 86 [0x56])) [0 MEM[(void *)&buf2]+86 S1 A16])
        (const_int 10 [0xa])) "hellocompiler.c":23 -1
     (nil))
(insn 34 33 35 2 (parallel [
            (set (reg:DI 98)
                (plus:DI (reg/f:DI 82 virtual-stack-vars)
                    (const_int -2304 [0xfffffffffffff700])))
            (clobber (reg:CC 17 flags))
        ]) "hellocompiler.c":24 -1
     (nil))
(insn 35 34 36 2 (set (reg:DI 5 di)
        (reg:DI 98)) "hellocompiler.c":24 -1
     (nil))
(call_insn 36 35 37 2 (set (reg:SI 0 ax)
        (call (mem:QI (symbol_ref:DI ("puts") [flags 0x41] <function_decl 0x7fb1fd6cdf00 __builtin_puts>) [0 __builtin_puts S1 A8])
            (const_int 0 [0]))) "hellocompiler.c":24 -1
     (expr_list:REG_CALL_DECL (symbol_ref:DI ("puts") [flags 0x41] <function_decl 0x7fb1fd6cdf00 __builtin_puts>)
        (nil))
    (expr_list:DI (use (reg:DI 5 di))
        (nil)))
```

Lazy Optimization Watching



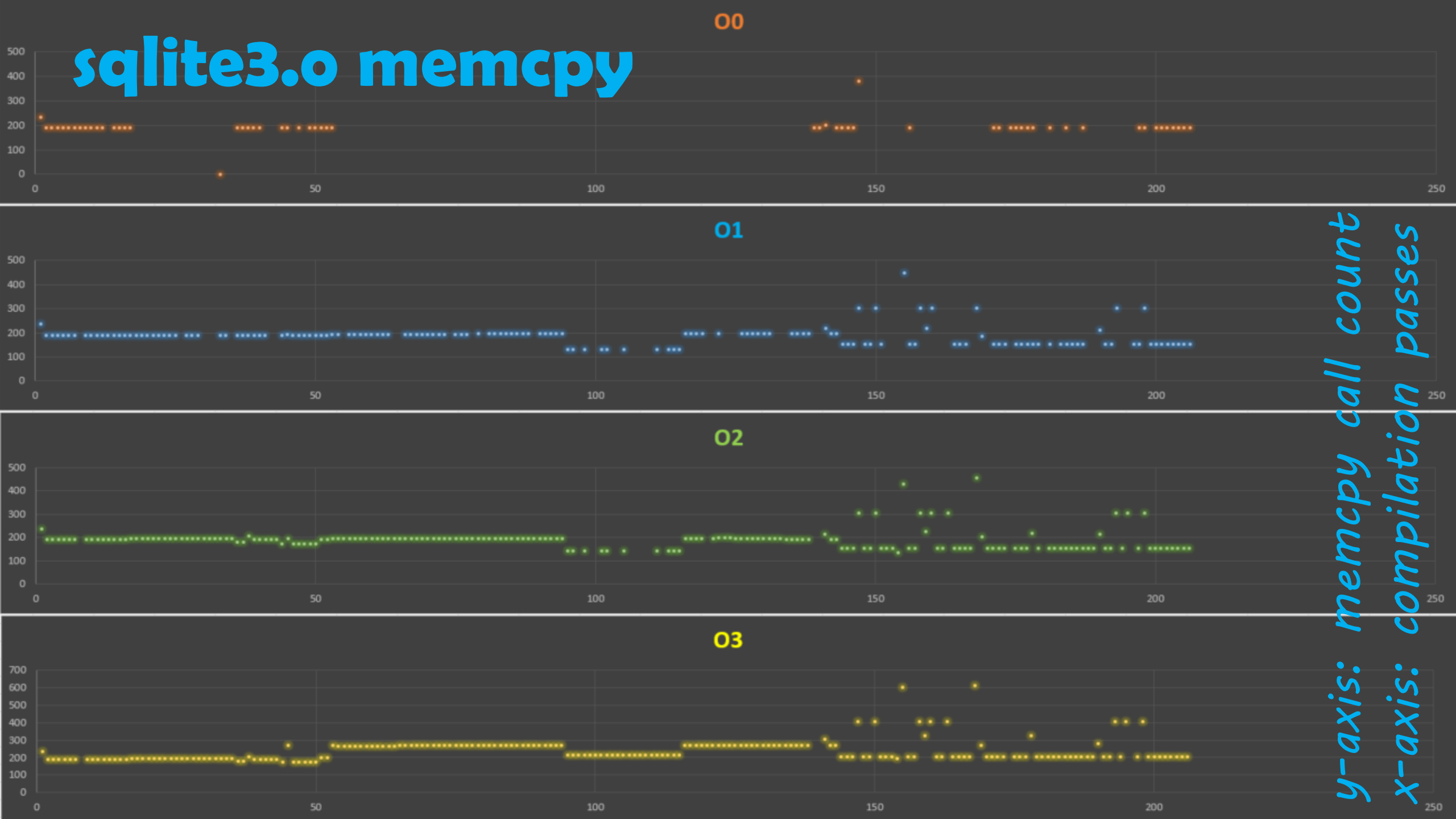
helloocompiler.c

The testbunny: Sqlite3

```
sqlite3.c
shell.c
lemon.c
mkkeywordhash.c
tclsqlite.c
```

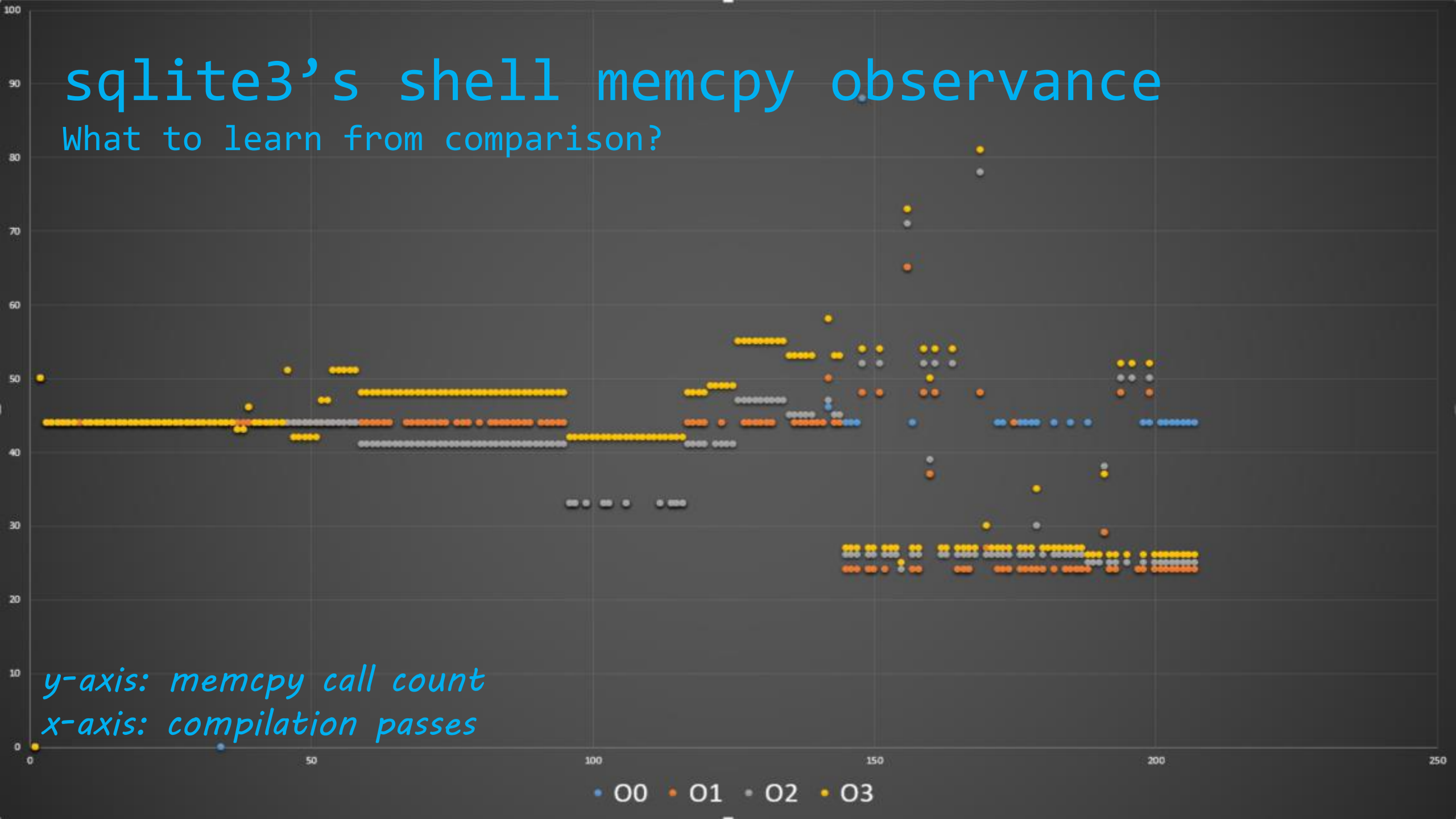
```
sqlite3 / sqlite3.o
sqlite3
lemon
mkkeywordhash
tclsqlite.o
```

sqlite3.o memcpy



sqlite3’s shell memcpy observance

What to learn from comparison?



y-axis: memcpy call count
x-axis: compilation passes

Grep for Compiler Research ^^

Fairly straight forward

Not necessarily the most accurate

Text parsing only ever gets so far:

- Code duplications for optimization purposes
- Different representations in same log file

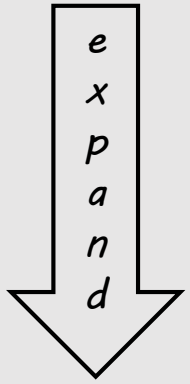
but there IS fun stuff we can do with that data...

Homemade off-by-plenty bugs

Developer controlled

Piggy-backing onto compiler behavior

```
__builtin_fun(x, y, magic_value);
```



magic_value found?
modify output

```
mov ...
mov ...
mov ...
(evil) mov dst, src
```

```c
int AUTH = 0;

int main(void)
{
    char *string1 = aligned_alloc(32, 96);
    char *string2 = "eldit934h2rkfhgppq;a3fjjdklsajieeisbnufblabghbalweglqjwebdfjhlkj"
                    "alekdgowei;anebkbirnwlekgn;qnwol";

    memcpy(string1, string2, 96);

    printf("Result %s\n", string2);

    if (AUTH != 0) {
        execl("/usr/games/xmabacus", "xmabacus", NULL);
    } else {
        printf("Nooope go away\n");
    }

    return 0;
}
```

victim variable

builtin to "fix"

magic number

DEMO TIME

Look ma, I made memcpy faster!

```
lea    rsi,[rip+0xf66]        # 2008 <_IO_stdin_used+0x
lea    rdi,[rip+0xfc7]        # 2070 <_IO_stdin_used+0x
mov    QWORD PTR [rax-0xff4],0x1

movaps XMMWORD PTR [rax],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xff1]        # 20b0 <_IO_

mov    QWORD PTR [rax-0xfec],0x0

movaps XMMWORD PTR [rax+0x10],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xfea]        # 20c0 <_IO_

movaps XMMWORD PTR [rax+0x20],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xfee]        # 20d0 <_IO_

movaps XMMWORD PTR [rax+0x30],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xff2]        # 20e0 <_IO_

movaps XMMWORD PTR [rax+0x40],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xff6]        # 20f0 <_IO_

movaps XMMWORD PTR [rax+0x50],xmm0
xor    eax,eax
call   1040 <printf@plt>
mov    eax,DWORD PTR [rip+0x2f41]        # 404c <AUTH>
test   eax,eax
je     112d <main+0xad>
xor    edx,edx
lea    rsi,[rip+0xf6e]        # 2086 <_IO_stdin_used+0x
lea    rdi,[rip+0xf5c]        # 207b <_IO_stdin_used+0x
xor    eax,eax
call   1060 <execl@plt>
```

Tracking from within

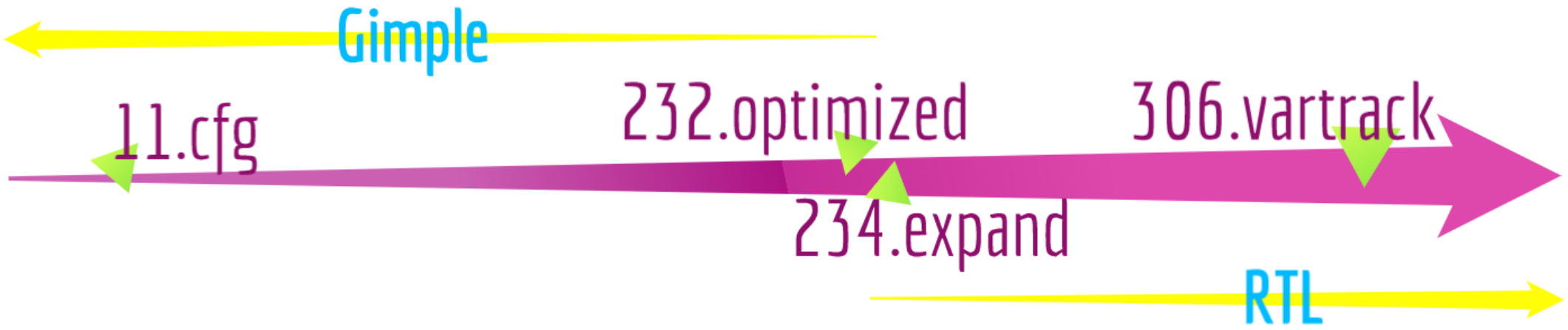
GCC's location_t available throughout compilation

Provides source file, line number, function name, etc.

Plugging into four passes to take measurements

- cfg, optimized, expand, vartrack

Two types of parsers: GIMPLE & RTL



	optsetting	binaryout	pass	token	sourcefile	linenumber
	Filter	Filter	Filter	Filter	Filter	Filter
2072	O3	sqlite3	post_expand	memcpy	shell.c	10903
2073	O3	sqlite3	post_vartrack	memcpy	shell.c	10903
2074	O3	sqlite3	post_vartrack	memcpy	shell.c	10903
2075	O3	sqlite3	post_optimized	memcpy	shell.c	9884
2076	O3	sqlite3	post_optimized	memcpy	shell.c	9885
2077	O3	sqlite3	post_optimized	memcpy	shell.c	18187
2078	O3	sqlite3	post_optimized	memcpy	shell.c	18200
2079	O3	sqlite3	post_optimized	memcpy	shell.c	18205
2080	O3	sqlite3	post_expand	memcpy	shell.c	18200
2081	O3	sqlite3	post_expand	memcpy	shell.c	18205
2082	O3	sqlite3	post_vartrack	memcpy	shell.c	18205
2083	O3	sqlite3	post_vartrack	memcpy	shell.c	18200
2084	O3	sqlite3	post_optimized	memcpy	shell.c	18426
2085	O3	sqlite3	post_optimized	memcpy	shell.c	18427
2086	O3	sqlite3	post_optimized	memcpy	shell.c	18775
2087	O3	sqlite3	post_cfg	memcpy	sqlite3.c	74941
2088	O3	sqlite3	post_cfg	memcpy	sqlite3.c	27854
2089	O3	sqlite3	post_cfg	memcpy	sqlite3.c	28229
2090	O3	sqlite3	post_cfg	memcpy	sqlite3.c	27141
2091	O3	sqlite3	post_cfg	memcpy	sqlite3.c	28283
2092	O3	sqlite3	post_cfg	memcpy	sqlite3.c	28264
2093	O3	sqlite3	post_cfg	memcpy	sqlite3.c	28305

“Big Data”

Flat dataset:

- builtin
- optimization setting
- output binary
- source file : line number

Future reseach .....

Memcpy calls in sqlite3's lemon tool

optimization

source file and line

measured passes

	source	1_post_cfg	2_post_optimized	3_post_expand	4_post_vartrack
O0	lemon.c:2644	1	1	1	1
O0	lemon.c:2659	1	1	1	1
O0	lemon.c:89	1	1	1	1
	source	1_post_cfg	2_post_optimized	3_post_expand	4_post_vartrack
O1	lemon.c:2644	1	1	0	0
O1	lemon.c:2659	1	1	1	1
O1	lemon.c:89	1	1	1	1
	source	1_post_cfg	2_post_optimized	3_post_expand	4_post_vartrack
O2	lemon.c:2644	1	1	0	0
O2	lemon.c:2659	1	1	1	1
O2	lemon.c:89	1	2	2	2
	source	1_post_cfg	2_post_optimized	3_post_expand	4_post_vartrack
O3	lemon.c:2644	1	1	0	0
O3	lemon.c:2659	1	1	1	1
O3	lemon.c:89	1	13	6	6

memcpy calls in that location

WAIT WHAT??

```
;; Function lemon_sprintf (null)
;; enabled by -tree-original


{
  struct  ap[1];
  int rc;

  # DEBUG BEGIN STMT;
    struct  ap[1];
  # DEBUG BEGIN STMT;
    int rc;
  # DEBUG BEGIN STMT;
  __builtin_va_start ((struct  *) &ap, format);
  # DEBUG BEGIN STMT;
  rc = lemon_vsprintf (str, format, (struct  *) &ap);
  # DEBUG BEGIN STMT;
  __builtin_va_end ((struct  *) &ap);
  # DEBUG BEGIN STMT;
  return rc;
}
```

Original code

lemon_sprintf(..)

inlining

lemon_vsprintf(..)

inlining / loop unrolling

lemon_addtext(..)

memcpy(..)

Location dependent actions might shift if the compiler reorders things, in case no data dependency is present

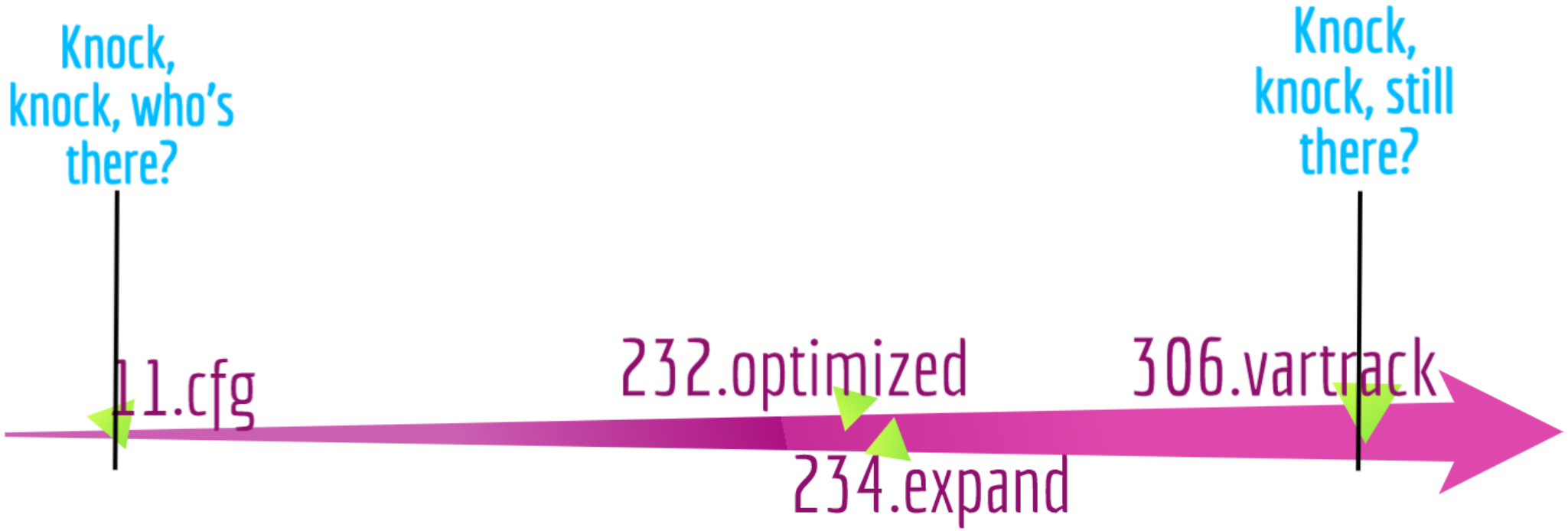
	source	1_post_cfg	2_post_optimized	3_post_expand	4_post_vartrack
O2	lemon.c:2644	1	1	0	0
O2	lemon.c:2659	1	1	1	1
O2	lemon.c:89	1	2	2	2
	source	1_post_cfg	2_post_optimized	3_post_expand	4_post_vartrack
O3	lemon.c:2644	1	1	0	0
O3	lemon.c:2659	1	1	1	1
O3	lemon.c:89	1	13	6	6

inlining

inlining + loop unrolling

But most importantly, what about the zeros?

How to know whether inlined or eliminated?



O2	sqlite3.c:168195	1	1	1	1	remains:3
O2	sqlite3.c:168366	1	1	0	0	remains:7
O2	sqlite3.c:168728	1	0	0	0	eliminated
O2	sqlite3.c:169143	1	1	0	0	remains:1
O2	sqlite3.c:169437	1	1	0	0	remains:2

memset in sqlite3.c line 168728 when compiled with -O2 disappears ...

```
168722 /*
168723 ** Malloc and Free functions
168724 */
168725 static void *fts3HashMalloc(sqlite3_int64 n){
168726   void *p = sqlite3_malloc64(n);
168727   if( p ){
168728     memset(p, 0, n);
168729   }
168730   return p;
168731 }
```

... in fact, the entire function is gone ...

```
167518 static void *fts3MallocZero(sqlite3_int64 nByte){
167519   void *pRet = sqlite3_malloc64(nByte);
167520   if( pRet ) memset(pRet, 0, nByte);
167521   return pRet;
167522 }
```

... since it had been replaced with another one O.o

The Linux Kernel Archives

About Contact us FAQ Releases Signatures Site news

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel:

5.3.6

Experiments on Linux kernel 5.3.6

GCC plugin infrastructure plug n play (thx to PaX!)

Used default opt setting -O2

%%% ?

Linux kernel 5.3.6, some numbers

	memset calls via grep	memset calls in cfg	memset calls post optimized	memset calls after expand	memset calls final	Eliminated memset	Inlined memsets
Total	*19732*	12870	12476	1374	1364	434	11072
Loss			*394*	*11102*	*10*		
			% call loss since cfg	% call loss since cfg	% call loss since cfg	% memset eliminated	% memset inlined
% of cfg total			3.0614%	86.2626%	0.0777%	3.3722%	86.0295%

10717	smb2transport.o:smb2transport.c:244	1	1	0	0	2
10718	smb2transport.o:smb2transport.c:443	1	1	0	0	2
10719	smb2transport.o:smb2transport.c:444	1	1	0	0	2
10720	smb2transport.o:smb2transport.c:543	1	1	0	0	2
10721	smb2transport.o:smb2transport.c:587	1	1	0	0	10
10722	smbencrypt.o:smbencrypt.c:149	1	0	0	0	eliminated
10723	smbencrypt.o:smbencrypt.c:150	1	1	0	0	2
10724	smbencrypt.o:smbencrypt.c:151	1	1	0	0	4
10725	smbencrypt.o:smbencrypt.c:198	1	1	0	0	2
10726	smbencrypt.o:smbencrypt.c:199	1	1	0	0	4
10727	smc91c92_cs.o:smc91c92_cs.c:1552	1	1	0	0	1
10728	sme.o:etherdevice.h:240	1	1	0	0	3
10729	sme.o:sme.c:1121	1	1	0	0	4
10730	sme.o:sme.c:260	1	1	0	0	10
10731	sme.o:sme.c:369	1	1	0	0	6
10732	sme.o:sme.c:703	1	1	0	0	2
10733	sme.o:sme.c:710	1	1	0	0	2
10734	sme.o:sme.c:716	1	1	0	0	4
10735	sme.o:sme.c:929	1	1	0	0	2
10736	sme.o:sme.c:936	1	1	0	0	2
10737	sme.o:sme.c:942	1	1	0	0	1
10738	smincio_main.o:smincio_main.c:531	1	1	0	0	4

Lost memset in ./drivers/crypto/ccp/ccp-crypto-aes-cmac.c

```
static int ccp_aes_cmac_export(struct ahash_request *req, void *out)
{
    struct ccp_aes_cmac_req_ctx *rctx = ahash_request_ctx(req);
    struct ccp_aes_cmac_exp_ctx state;

    /* Don't let anything leak to 'out' */
    memset(&state, 0, sizeof(state));

    state.null_msg = rctx->null_msg;
    memcpy(state.iv, rctx->iv, sizeof(state.iv));
    state.buf_count = rctx->buf_count;
    memcpy(state.buf, rctx->buf, sizeof(state.buf));

    /* 'out' may not be aligned so memcpy from local variable */
    memcpy(out, &state, sizeof(state));

    return 0;
}
```

eliminated / implied

inlined

```
var_38          = qword ptr -38h
var_2C          = qword ptr -2Ch
var_24          = dword ptr -24h
var_20          = qword ptr -20h
var_18          = qword ptr -18h
canary          = qword ptr -10h

                call    __fentry__      ; PIC mode
                push    rbp
                mov     rbp, rsp
                sub     rsp, 30h        ; rdi - ahash_request *req
                mov     rdx, [rdi+0B0h] ; reading of request data
                mov     rcx, [rdi+0DCh]
                mov     rax, gs:28h
                mov     [rsp+38h+canary], rax
                xor     eax, eax
                mov     r9d, [rdi+0D8h]
                mov     rax, [rdi+0A8h]
                mov     r8, [rdi+0E4h]
                mov     edi, [rdi+50h]
                mov     [rsp+38h+var_2C], rdx ; space for state struct apparently held on stack
                mov     [rsp+38h+var_38+4], rax ; no sizeof, memcpy or memset calls preserved
                mov     dword ptr [rsp+38h+var_38], edi
                mov     rdx, [rsp+38h+var_38]
                mov     [rsp+38h+var_24], r9d
                mov     [rsi], rdx      ; rsi - void *out
                mov     rdx, [rsp+8]    ; memcpy(out, &state, sizeof.. inlined here
                mov     [rsp+38h+var_20], rcx
                mov     [rsi+8], rdx
                mov     rdx, [rsp+38h+var_2C+4]
                mov     [rsp+38h+var_18], r8
                mov     [rsi+10h], rdx
                mov     [rsi+18h], rcx
                mov     [rsi+20h], r8
                mov     rcx, [rsp+38h+canary]
                xor     rcx, gs:28h
                jnz     short loc_94
```

Loading of req data from first function argument

state struct occupies stack vars var_18 to var_38

inlined memcpy copying state struct data to void *out

Lost memset in smbencrypt.c

The compiler removes memset(p14..), and inlines the other two

We see the memcpy(p14, passwd,..) inlined so as passwd is copied to a stack buffer in chunks

The compiler also inlined the call to E_P16

```
int
SMBencrypt(unsigned char *passwd, const unsigned char *c8, unsigned char *p24)
{
    int rc;
    unsigned char p14[14], p16[16], p21[21];

    memset(p14, '\0', 14);
    memset(p16, '\0', 16);
    memset(p21, '\0', 21);

    memcpy(p14, passwd, 14);
    rc = E_P16(p14, p16);
    if (rc)
        return rc;

    memcpy(p21, p16, 16);
    rc = E_P24(p21, c8, p24);

    return rc;
}
```

eliminated / implied

inlined

inlined

```
static int
E_P16(unsigned char *p14, unsigned char *p16)
{
    int rc;
    unsigned char sp8[8] =
        { 0x4b, 0x47, 0x53, 0x21, 0x40, 0x23, 0x24, 0x25 };

    rc = smbhash(p16, sp8, p14);
    if (rc)
        return rc;
    rc = smbhash(p16 + 8, sp8, p14 + 7);
    return rc;
}
```

SMBEncrypt asm

```
sp8= qword ptr -5Bh
p14_1= qword ptr -53h
p14_2= dword ptr -4Bh
p14_3= word ptr -47h
p16_1= qword ptr -45h
p16_2= qword ptr -3Dh
p21_1= byte ptr -35h
p21_2= qword ptr -2Dh
p21_3= dword ptr -25h
p21_4= byte ptr -21h
cookie= qword ptr -20h

call    __fentry__      ; PIC mode
push    rbp
mov     rbp, rsp
push    r12
mov     r12, rdx
push    rbx
mov     rbx, rsi
sub     rsp, 48h
mov     rax, gs:28h
mov     [rsp+60h+cookie], rax
xor     eax, eax
mov     rax, [rdi]
lea     rdx, [rsp+60h+p14_1]
lea     rsi, [rsp+60h+sp8]
mov     [rsp+60h+p16_1], 0
mov     [rsp+60h+p14_1], rax
mov     eax, [rdi+8]
mov     [rsp+60h+p16_2], 0
mov     [rsp+60h+p14_2], eax
movzx   eax, word ptr [rdi+0Ch]
lea     rdi, [rsp+60h+p16_1]
mov     qword ptr [rsp+60h+p21_1], 0
mov     [rsp+60h+p14_3], ax
mov     rax, 252423402153474Bh
mov     [rsp+60h+p21_2], 0
mov     [rsp+60h+p21_3], 0
mov     [rsp+60h+p21_4], 0
mov     [rsp+60h+sp8], rax
call    smbhash
test    eax, eax
jz      short loc_3AF
```

p14 memset-0 omitted

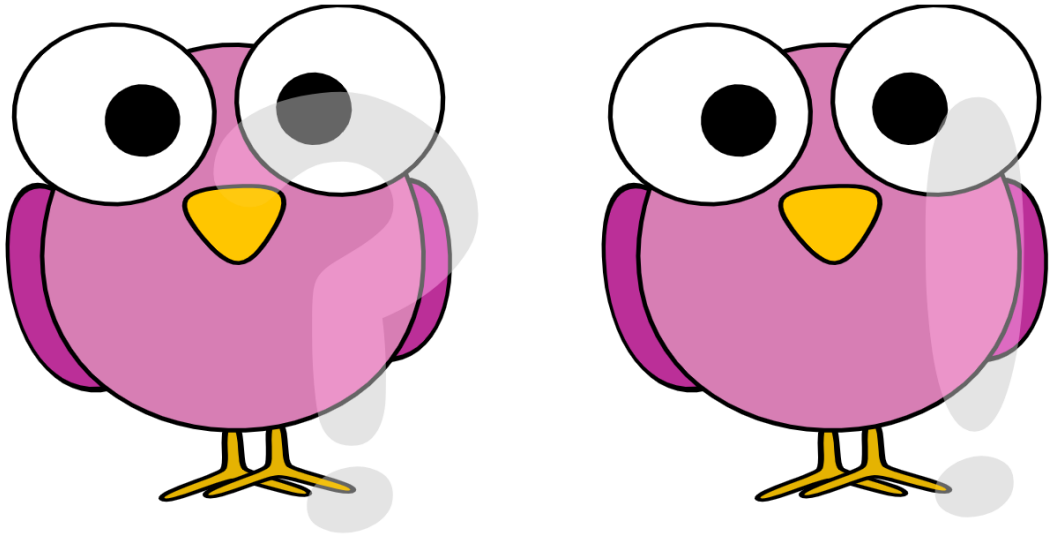
RDI holds first argument, passwd is copied to p14_1, p14_2 and p14_3

```
loc_3AF:
lea     rdx, [rsp+60h+p14_1+7]
lea     rdi, [rsp+60h+p16_2]
lea     rsi, [rsp+60h+sp8]
call    smbhash
test    eax, eax
```

- 14 bytes are copied out of the passwd buffer using mov instructions, and overwrite the given memory
- This, regardless of what the memory contained before

… and 431 other cases, but first:

- Careful tracking: wrapper functions or self-made implementations
- Understanding of application needed in order to track the right information
- Gatherable information in reality very one dimensional, much more analysis of code desireable




Thank you!