

Übung zu Betriebssysteme

Entkäfern mit GDB & GEF

07. November 2024

Dustin Nguyen, Christian Eichler

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



Friedrich-Alexander-Universität
Technische Fakultät



Your PC ran into a problem that it couldn't handle, and now it needs to restart.

You can search for the error online: `HAL_INITIALIZATION_FAILED`



Your PC ran into a problem that it couldn't handle, and now it needs to restart.

You can search for the error online: `HAL_INITIALIZATION_FAILED`





Your PC ran into a problem it couldn't handle, and now it needs to restart.

You can search for the error online: `HAL_INITIALIZATION_FAILED`



- common.mk
- kernel
 - boot
 - sections.ld
 - startup.asm
 - startup.cc
 - compiler.h
 - config.h
 - debug
 - assert.cc
 - assert.h
 - gdb
 - handler.asm
 - handler.cc
 - init.cc
 - protocol.cc
 - stub.h
 - kernelpanic.h
 - null_stream.cc
 - null_stream.h
 - output.h
- device
 - cgasttr.cc
 - cgasttr.h
 - console.cc
 - console.h
 - keyboard.cc
 - keyboard.h
 - panic.cc
 - panic.h
 - watch.cc
 - watch.h
- guard
 - gate.h
 - guard.cc
 - guard.h
 - guardian.cc
 - guardian.h
 - secure.h
- machine
 - acpi.cc
 - acpi.h
 - apicssystem.cc
 - apicssystem.h
 - cgasr.cc
 - cgasr.h
 - cpu.asm
 - cpu.h
 - gdt.cc
 - gdt.h
 - ioapic.cc
 - ioapic.h
 - ioapic_registers.h
 - io_port.h
 - keyctrl.cc
 - keyctrl.h
 - keydecoder.cc
 - keydecoder.h
 - key.h
 - lapic.cc
 - lapic.h
 - lapic_registers.h
 - mp_registers.h
 - plugbox.cc

- pluginbox.h
- serial.cc
- serial.h
- spinlock.h
- ticketlock.h
- toc.asm
- toc.cc
- toc.h
- toc.inc
- main.cc
- Makefile
- meeting
 - bell.cc
 - bell.h
 - bellringer.cc
 - bellringer.h
 - messageinfo.h
 - semaphore.cc
 - semaphore.h
 - waitingroom.cc
 - waitingroom.h
- memory
 - allocator.cc
 - allocator.h
 - copy.cc
 - copy.h
 - initmem.cc
 - initmem.h
 - kernelpage.h
 - malloc.cc
 - malloc.h
 - multiboot.h
 - pagecont.cc
 - pagecont.h
 - tdir.cc
 - tdir.h



- pagerfault.h
- page.h
- pageref.cc
- pageref.h
- range.h
- user_buffer.h
- object
 - bbuffer.h
 - o_stream.cc
 - o_stream.h
 - queue.h
 - queueLink.h
 - strbuf.cc
 - strbuf.h
- syscall
 - guarded_bell.cc
 - guarded_bell.h
 - guarded_keyboard.cc
 - guarded_keyboard.h
 - guarded_scheduler.h
 - guarded_semaphore.h
 - syscall.cc
 - syscall.h
- thread
 - assassin.cc
 - assassin.h
 - dispatcher.cc
 - dispatcher.h
 - idlethread.cc
 - idlethread.h
 - scheduler.cc
 - scheduler.h
 - thread.cc
 - thread.h
 - wakeup.h
- types.h
- user
 - app1
 - appl.cc
 - appl.h
 - app2
 - kappl.cc
 - kappl.h

- utils
 - elf.cc
 - elf.h
 - libcc.cc
 - math.h
 - memutil.cc
 - memutil.h
 - sort.h
 - string.cc
 - string.h
 - tar.cc
 - tar.h
- libsys
 - ioStream.h
 - Makefile
 - o_stream.cc
 - o_stream.h
 - o_stream.h
 - strbuf.cc
 - strbuf.h
 - syscall_stubs.asm
 - syscall_stubs.cc
 - types.h
- Makefile
- README.md
- test-stream
 - console_out.cc
 - console_out.h
 - file_out.cc
 - file_out.h
 - Makefile
 - test.cc
- user
 - app0
 - app0.cc
 - app0.h
 - app1
 - app1.cc
 - app1.h
 - app2
 - app2.cc
 - app2.h
 - app3
 - app3.cc
 - app3.h
 - app4
 - app4.cc
 - app4.h
 - imgbuilder.cc
 - init.cc
 - Makefile
 - Makefile.app
 - sections.ld

24 directories, 172 files

```

240  a = 1;
241  b = 2;
242  c = 3;
243  d = 4;
244  e = 5;
245  f = 6;
246  g = 7;
247  h = 8;
248  i = 9;
249  j = 10;
250  k = 11;
251  l = 12;
252  m = 13;
253  n = 14;
254  o = 15;
255  p = 16;
256  q = 17;
257  r = 18;
258  s = 19;
259  t = 20;
260  u = 21;
261  v = 22;
262  w = 23;
263  x = 24;
264  y = 25;
265  z = 26;
266  AA = 27;
267  BB = 28;
268  CC = 29;
269  DD = 30;
270  EE = 31;
271  FF = 32;
272  GG = 33;
273  HH = 34;
274  II = 35;
275  JJ = 36;
276  KK = 37;
277  LL = 38;
278  MM = 39;
279  NN = 40;
280  OO = 41;
281  PP = 42;
282  QQ = 43;
283  RR = 44;
284  SS = 45;
285  TT = 46;
286  UU = 47;
287  VV = 48;
288  WW = 49;
289  XX = 50;
290  YY = 51;
291  ZZ = 52;
292  AA = 53;
293  BB = 54;
294  CC = 55;
295  DD = 56;
296  EE = 57;
297  FF = 58;
298  GG = 59;
299  HH = 60;
300  II = 61;
301  JJ = 62;
302  KK = 63;
303  LL = 64;
304  MM = 65;
305  NN = 66;
306  OO = 67;
307  PP = 68;
308  QQ = 69;
309  RR = 70;
310  SS = 71;
311  TT = 72;
312  UU = 73;
313  VV = 74;
314  WW = 75;
315  XX = 76;
316  YY = 77;
317  ZZ = 78;
318  AA = 79;
319  BB = 80;
320  CC = 81;
321  DD = 82;
322  EE = 83;
323  FF = 84;
324  GG = 85;
325  HH = 86;
326  II = 87;
327  JJ = 88;
328  KK = 89;
329  LL = 90;
330  MM = 91;
331  NN = 92;
332  OO = 93;
333  PP = 94;
334  QQ = 95;
335  RR = 96;
336  SS = 97;
337  TT = 98;
338  UU = 99;
339  VV = 100;
340  WW = 101;
341  XX = 102;
342  YY = 103;
343  ZZ = 104;
344  AA = 105;
345  BB = 106;
346  CC = 107;
347  DD = 108;
348  EE = 109;
349  FF = 110;
350  GG = 111;
351  HH = 112;
352  II = 113;
353  JJ = 114;
354  KK = 115;
355  LL = 116;
356  MM = 117;
357  NN = 118;
358  OO = 119;
359  PP = 120;
360  QQ = 121;
361  RR = 122;
362  SS = 123;
363  TT = 124;
364  UU = 125;
365  VV = 126;
366  WW = 127;
367  XX = 128;
368  YY = 129;
369  ZZ = 130;
370  AA = 131;
371  BB = 132;
372  CC = 133;
373  DD = 134;
374  EE = 135;
375  FF = 136;
376  GG = 137;
377  HH = 138;
378  II = 139;
379  JJ = 140;
380  KK = 141;
381  LL = 142;
382  MM = 143;
383  NN = 144;
384  OO = 145;
385  PP = 146;
386  QQ = 147;
387  RR = 148;
388  SS = 149;
389  TT = 150;
390  UU = 151;
391  VV = 152;
392  WW = 153;
393  XX = 154;
394  YY = 155;
395  ZZ = 156;
396  AA = 157;
397  BB = 158;
398  CC = 159;
399  DD = 160;
400  EE = 161;
401  FF = 162;
402  GG = 163;
403  HH = 164;
404  II = 165;
405  JJ = 166;
406  KK = 167;
407  LL = 168;
408  MM = 169;
409  NN = 170;
410  OO = 171;
411  PP = 172;
412  QQ = 173;
413  RR = 174;
414  SS = 175;
415  TT = 176;
416  UU = 177;
417  VV = 178;
418  WW = 179;
419  XX = 180;
420  YY = 181;
421  ZZ = 182;
422  AA = 183;
423  BB = 184;
424  CC = 185;
425  DD = 186;
426  EE = 187;
427  FF = 188;
428  GG = 189;
429  HH = 190;
430  II = 191;
431  JJ = 192;
432  KK = 193;
433  LL = 194;
434  MM = 195;
435  NN = 196;
436  OO = 197;
437  PP = 198;
438  QQ = 199;
439  RR = 200;
440  SS = 201;
441  TT = 202;
442  UU = 203;
443  VV = 204;
444  WW = 205;
445  XX = 206;
446  YY = 207;
447  ZZ = 208;
448  AA = 209;
449  BB = 210;
450  CC = 211;
451  DD = 212;
452  EE = 213;
453  FF = 214;
454  GG = 215;
455  HH = 216;
456  II = 217;
457  JJ = 218;
458  KK = 219;
459  LL = 220;
460  MM = 221;
461  NN = 222;
462  OO = 223;
463  PP = 224;
464  QQ = 225;
465  RR = 226;
466  SS = 227;
467  TT = 228;
468  UU = 229;
469  VV = 230;
470  WW = 231;
471  XX = 232;
472  YY = 233;
473  ZZ = 234;
474  AA = 235;
475  BB = 236;
476  CC = 237;
477  DD = 238;
478  EE = 239;
479  FF = 240;
480  GG = 241;
481  HH = 242;
482  II = 243;
483  JJ = 244;
484  KK = 245;
485  LL = 246;
486  MM = 247;
487  NN = 248;
488  OO = 249;
489  PP = 250;
490  QQ = 251;
491  RR = 252;
492  SS = 253;
493  TT = 254;
494  UU = 255;
495  VV = 256;
496  WW = 257;
497  XX = 258;
498  YY = 259;
499  ZZ = 260;
500  AA = 261;
501  BB = 262;
502  CC = 263;
503  DD = 264;
504  EE = 265;
505  FF = 266;
506  GG = 267;
507  HH = 268;
508  II = 269;
509  JJ = 270;
510  KK = 271;
511  LL = 272;
512  MM = 273;
513  NN = 274;
514  OO = 275;
515  PP = 276;
516  QQ = 277;
517  RR = 278;
518  SS = 279;
519  TT = 280;
520  UU = 281;
521  VV = 282;
522  WW = 283;
523  XX = 284;
524  YY = 285;
525  ZZ = 286;
526  AA = 287;
527  BB = 288;
528  CC = 289;
529  DD = 290;
530  EE = 291;
531  FF = 292;
532  GG = 293;
533  HH = 294;
534  II = 295;
535  JJ = 296;
536  KK = 297;
537  LL = 298;
538  MM = 299;
539  NN = 300;
540  OO = 301;
541  PP = 302;
542  QQ = 303;
543  RR = 304;
544  SS = 305;
545  TT = 306;
546  UU = 307;
547  VV = 308;
548  WW = 309;
549  XX = 310;
550  YY = 311;
551  ZZ = 312;
552  AA = 313;
553  BB = 314;
554  CC = 315;
555  DD = 316;
556  EE = 317;
557  FF = 318;
558  GG = 319;
559  HH = 320;
560  II = 321;
561  JJ = 322;
562  KK = 323;
563  LL = 324;
564  MM = 325;
565  NN = 326;
566  OO = 327;
567  PP = 328;
568  QQ = 329;
569  RR = 330;
570  SS = 331;
571  TT = 332;
572  UU = 333;
573  VV = 334;
574  WW = 335;
575  XX = 336;
576  YY = 337;
577  ZZ = 338;
578  AA = 339;
579  BB = 340;
580  CC = 341;
581  DD = 342;
582  EE = 343;
583  FF = 344;
584  GG = 345;
585  HH = 346;
586  II = 347;
587  JJ = 348;
588  KK = 349;
5
```

printf

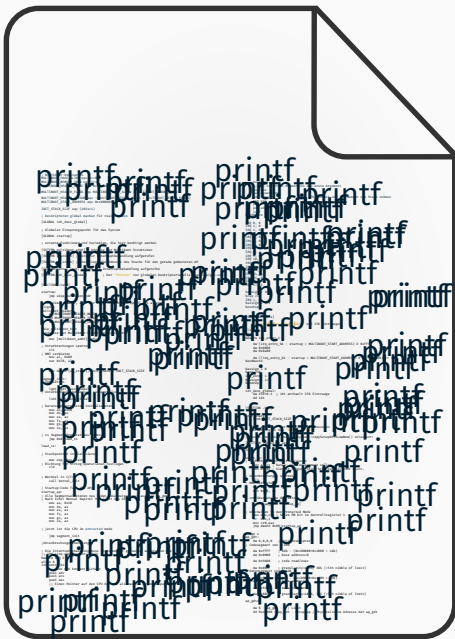
```
printf
```

[illegible]

```
printf
```

```
printf
```

printf



Entkäfern mit GDB & GEF

GNU Debugger (GDB)

- + Inspizieren des Systemzustands während das System läuft
- Nur rudimentäres TUI

```
(gdb) c
Continuing.

Thread 1 hit Breakpoint 1, guardian (vector=33, context=0x101cf58 <cpu_stack+3912>) at guard/guardian.cc:15
15      Gate* gate = Plugbox::report(vector);
(gdb) █
```

Entkäfern mit GDB & GEF

GNU Debugger (GDB)

- + Inspizieren des Systemzustands während das System läuft
- Nur rudimentäres TUI

```
(gdb) c
Continuing.

Thread 1 hit Breakpoint 1, guardian (vector=33, context=0x101cf58 <cpu_stack+3912>) at guard/guardian.cc:15
15      Gate* gate = Plugbox::report(vector);
(gdb) █
```

GDB Enhanced Features (GEF)

- + Erweitert GDB um ein brauchbar(er)es Interface
- z.T. visuell überladen

Entkäfern mit GDB & GEF

GNU Debugger (GDB)

- + Inspizieren des Systemzustands während das System läuft
- Nur rudimentäres TUI

```
(gdb) c
Continuing.

Thread 1 hit Breakpoint 1, guardian (vector=33, context=0x101cf58 <cpu_stack+3912>) at guard/guardian.cc:15
15      Gate* gate = Plugbox::report(vector);
(gdb) █
```

GDB Enhanced Features (GEF)

- + Erweitert GDB um ein brauchbar(er)es Interface
- z.T. visuell überladen

Editor/IDE-Integration

- + bekannte Umgebung
- frickelig

```

eax : 0x0101b520 + 0x00000000 + 0x00000000
ebx : 0x01012ba8 + 0x00000000 + 0x00000000
ecx : 0x01010870 + 0x8b535657 + 0x8b535657
edx : 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
esp : 0x0101af1c + 0x0100038b + 0x5808c483 + 0x5808c483
ebp : 0x0101afa8 + 0x01011b7c + 0x01001930 + 0x0191c8b8 + 0x00000000 + 0x00000000
esi : 0x01012ba8 + 0x00000000 + 0x00000000
edi : 0x02011200 + 0x02011200
ebp : 0x01002d40 + 0xe8535657 + 0xe8535657
eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
$cs: 0x0008 $ss: 0x0010 $ds: 0x0010 $es: 0x0010 $fs: 0x0010 $gs: 0x0010

```

registers

Registerinhalt

```

0x0101af1c +0x0000: 0x0100038b + 0x5808c483 + 0x5808c483 + 0x00000000
0x0101af20 +0x0004: 0x00000021 + 0x00000021 + 0x00000000 + 0x00000000
0x0101af24 +0x0008: 0x0101afa8 + 0x01010520 + 0x00000000 + 0x00000000
0x0101af28 +0x000c: 0x0101b520 + 0x00000000 + 0x00000000
0x0101af2c +0x0010: 0x01010870 + 0x8b535657 + 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
0x0101af34 +0x0018: 0x01011200 + 0xe8535657 + 0xe8535657
0x0101af38 +0x001c: 0x00000000 + 0x00000000

```

stack

```

0x1002d3b xchg ax, ax
0x1002d3d xchg ax, ax
0x1002d3f mov
+ 0x1002d40 <guardian*0> push edi
0x1002d41 <guardian*1> push esi
0x1002d42 <guardian*2> push ebx
0x1002d43 <guardian*3> call 0x1010f20 <__x86_get_pc_thunk, b0>
0x1002d48 <guardian*8> add ebx, 0xfe60
0x1002d4e <guardian*14> sub esp, 0x8

```

code:x86:32

```

14
15 #include "guard/guard.h"
16 extern Guard guardian;
17
18 extern "C" void guardian(uint32_t vector, irq_context *context)
19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugbox_report(vector);
23
24     bool wantsEpilogue = gate->prologue();

```

source: ./guard/guardian.cc:19

```

[ #0] Id 1, Name: "", stopped, reason: NPE/NPPOINT
[ #1] Id 2, Name: "", stopped, reason: NPE/NPPOINT
[ #2] Id 3, Name: "", stopped, reason: NPE/NPPOINT
[ #3] Id 4, Name: "", stopped, reason: NPE/NPPOINT

```

threads

```

[ #0] 0x1002d40 - guardian(vector=0x21, context=0x0101af28)
[ #1] 0x100038b - irq_enter_33()
[ #2] 0x01ffb000 - add $01b, 01b [eax=011], 01
[ #3] 0x1005aa8 - kernel_init()
[ #4] 0x01b5e0 - add $01b, 01b [eax], a1

```

trace

Thread 1 hit Breakpoint 2, guardian (vector=0x21, context=0x0101af28) at ./guard/guardian.cc:19

```

19 {
gef+

```

```

eax : 0x0101b520 + 0x00000000 + 0x00000000
ebx : 0x01012ba8 + 0x00000000 + 0x00000000
ecx : 0x01010870 + 0x8b535657 + 0x8b535657
edx : 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
esp : 0x0101af1c + 0x0100038b + 0x5808c483 + 0x5808c483
ebp : 0x0101afa8 + 0x01011b7c + 0x01001930 + 0x0191c8b8 + 0x00000000 + 0x00000000
esi : 0x01012ba8 + 0x00000000 + 0x00000000
edi : 0x02011200 + 0x02011200
ebp : 0x01002940 + 0xe8535657 + 0xe8535657
eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
$cc: 0x0008 $cs: 0x0010 $ds: 0x0010 $es: 0x0010 $fs: 0x0010 $gs: 0x0010

```

registers

Registerinhalt

```

0x0101af1c +0x0000: 0x0100038b + 0x5808c483 + 0x5808c483 - fesp
0x0101af20 +0x0004: 0x00000021 + 0x00000021
0x0101af24 +0x0008: 0x0101afa8 + 0x0101b520 + 0x00000000 + 0x00000000
0x0101af28 +0x000c: 0x0101b520 + 0x00000000 + 0x00000000
0x0101af2c +0x0010: 0x01010870 + 0x8b535657 + 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
0x0101af34 +0x0018: 0x010114dd + 0xe0b6ff5a + 0xe0b6ff5a
0x0101af38 +0x001c: 0x00000008 + 0x00000008

```

stack

Stackinhalt

```

0x1002d3b xchg ax, ax
0x1002d3d xchg ax, ax
0x1002d3f mov rax, rax
- 0x1002d40 <guardian*0> push edi
0x1002d41 <guardian*1> push esi
0x1002d42 <guardian*2> push ebx
0x1002d43 <guardian*3> call 0x1010f20 <__x86.get_pc_thunk.b0>
0x1002d48 <guardian*8> add ebx, 0xfe60
0x1002d4e <guardian*14> sub esp, 0x8

```

code:0032

source: ./guard/guardian.cc:19

```

14
15 #include "guard/guard.h"
16 extern Guard guardian;
17
18 extern "C" void guardian(uint32_t vector, irq_context *context)
19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugin.report(vector);
23
24     bool wantsEpilogue = gate->prologue();

```

threads

```

[#0] Id 1, Name: "", stopped, reason: NPE/NP0INT
[#1] Id 2, Name: "", stopped, reason: NPE/NP0INT
[#2] Id 3, Name: "", stopped, reason: NPE/NP0INT
[#3] Id 4, Name: "", stopped, reason: NPE/NP0INT

```

```

[#0] 0x1002d40 - guardian(vector=0x21, context=0x0101af28)
[#1] 0x100038b - irq_enter_33()
[#2] 0x1ffb000 - add $01b, %r10 (%eax=0x11), 01
[#3] 0x1005aa8 - kernel_init()
[#4] 0x010b5e0 - add $01b, %r10 (%eax), a1

```

trace

Thread 1 hit Breakpoint 2, guardian (vector=0x21, context=0x0101af28) at ./guard/guardian.cc:19

```

19 {
gef+

```

```

%eax : 0x0101b520 + 0x00000000 + 0x00000000
%ebx : 0x01012ba8 + 0x00000000 + 0x00000000
%ecx : 0x01010870 + 0x8b535657 + 0x8b535657
%edx : 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
%esp : 0x0101af1c + 0x0100038b + 0x5808c483 + 0x5808c483
%ebp : 0x0101af28 + 0x0101b7c + 0x01001930 + 0x0191c8b8 + 0x00000000 + 0x00000000
%esi : 0x01012ba8 + 0x00000000 + 0x00000000
%edi : 0x02011200 + 0x02011200
%eip : 0x01002d40 + 0xe8535657 + 0xe8535657
%eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
%cs: 0x0008 %ss: 0x0010 %ds: 0x0010 %es: 0x0010 %fs: 0x0010 %gs: 0x0010

```

registers

Registerinhalt

```

0x0101af1c +0x0000: 0x0100038b + 0x5808c483 + 0x5808c483 - %esp
0x0101af20 +0x0004: 0x00000021 + 0x00000021
0x0101af24 +0x0008: 0x0101af28 + 0x0101b520 + 0x00000000 + 0x00000000
0x0101af28 +0x000c: 0x0101b520 + 0x00000000 + 0x00000000
0x0101af2c +0x0010: 0x01010870 + 0x8b535657 + 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
0x0101af34 +0x0018: 0x010114dd + 0xe0b6ff5a + 0xe0b6ff5a
0x0101af38 +0x001c: 0x00000008 + 0x00000008

```

stack

Stackinhalt

```

0x0002f8b: xchg ax, ax
0x0002f8d: xchg ax, ax
0x0002f8f: nop
- 0x0002d40 <guardian+0>: push edi
0x0002d41 <guardian+1>: push esi
0x0002d42 <guardian+2>: push ebx
0x0002d43 <guardian+3>: call 0x01010f20 <__x86.get_pc_thunk.bx>
0x0002d48 <guardian+8>: add ebx, 0xfe60
0x0002d4e <guardian+14>: sub esp, 0x8

```

code:x86:32

Stelle im Assembly

source: ./guard/guardian.cc:19

```

14
15 #include "guard/guard.h"
16 extern Guard guardian;
17
18 extern "C" void guardian(uint32_t vector, irq_context *context)
19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugin.report(vector);
23
24     bool wantsEpilogue = gate->prologue();

```

threads

```

[#0] Id 1, Name: "", stopped, reason: NO_MORE_EVENT
[#1] Id 2, Name: "", stopped, reason: NO_MORE_EVENT
[#2] Id 3, Name: "", stopped, reason: NO_MORE_EVENT
[#3] Id 4, Name: "", stopped, reason: NO_MORE_EVENT

```

```

[#0] 0x0002d40 - guardian(vector=0x21, context=0x0101af28)
[#1] 0x000038b - irq_enter_33()
[#2] 0x01ffb000 - add $01b, 01b(%eax), 01
[#3] 0x0005aa8 - kernel_init()
[#4] 0x010b5e0 - add $01b, 01b(%eax), 01

```

trace

Thread 1 hit Breakpoint 2, guardian (vector=0x21, context=0x0101af28) at ./guard/guardian.cc:19

```

19 {
gef+

```

```

eax : 0x0101b520 + 0x00000000 + 0x00000000
ebx : 0x01012ba8 + 0x00000000 + 0x00000000
ecx : 0x01010870 + 0x8b535657 + 0x8b535657
edx : 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
esp : 0x0101af1c + 0x0100038b + 0x5808c483 + 0x5808c483
ebp : 0x0101af28 + 0x0101b7c + 0x01001930 + 0x0191c8b8 + 0x00000000 + 0x00000000
esi : 0x01012ba8 + 0x00000000 + 0x00000000
edi : 0x02011200 + 0x02011200
ebp : 0x01002940 + 0xe8535657 + 0xe8535657
eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
$cc: 0x0008 $ss: 0x0010 $ds: 0x0010 $es: 0x0010 $fs: 0x0010 $gs: 0x0010

```

registers

Registerinhalt

```

0x0101af1c +0x0000: 0x0100038b + 0x5808c483 + 0x5808c483 - fesp
0x0101af20 +0x0004: 0x00000021 + 0x00000021
0x0101af24 +0x0008: 0x0101af28 + 0x0101b520 + 0x00000000 + 0x00000000
0x0101af28 +0x000c: 0x0101b520 + 0x00000000 + 0x00000000
0x0101af2c +0x0010: 0x01010870 + 0x8b535657 + 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
0x0101af34 +0x0018: 0x010114dd + 0xe0b6ff5a + 0xe0b6ff5a
0x0101af38 +0x001c: 0x00000008 + 0x00000008

```

stack

Stackinhalt

```

0x1002d3b: xchg ax, ax
0x1002d3d: xchg ax, ax
0x1002d3f: nop
- 0x1002d40 <guardian+0>: push edi
0x1002d41 <guardian+1>: push esi
0x1002d42 <guardian+2>: push ebx
0x1002d43 <guardian+3>: call 0x1010f20 <__x86.get_pc_thunk.bx>
0x1002d48 <guardian+8>: add ebx, 0xfe60
0x1002d4e <guardian+14>: sub esp, 0x8

```

code:x86:32

Stelle im Assembly

source:./guard/guardian.cc:19

```

14
15 #include "guard/guard.h"
16 extern Guard guard;
17
18 extern "C" void guardian(uint32_t vector, irq_context *context)
- 19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugbox.report(vector);
23
24     bool wantsEpilogue = gate->prologue();

```

Stelle in C/C++

```

[*0] Id 1, Name: "", stopped, reason: BREAKPOINT
[*1] Id 2, Name: "", stopped, reason: BREAKPOINT
[*2] Id 3, Name: "", stopped, reason: BREAKPOINT
[*3] Id 4, Name: "", stopped, reason: BREAKPOINT

```

threads

```

[*0] 0x1002d40 - guardian(vector=0x21, context=0x0101af28)
[*1] 0x100038b - irq_enter_33()
[*2] 0x1ffb000 - add $R15, $R15 ($w=0x11), 0
[*3] 0x1005aa8 - kernel_init()
[*4] 0x01b5e0 - add $R15, $R15 ($w=), 0

```

trace

Thread 1 hit Breakpoint 2, guardian (vector=0x21, context=0x0101af28) at ./guard/guardian.cc:19

```

19 {
gef+

```

```

eax : 0x0101b520 + 0x00000000 + 0x00000000
ebx : 0x01012ba8 + 0x00000000 + 0x00000000
ecx : 0x01010870 + 0x8b535657 + 0x8b535657
edx : 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
esp : 0x0101af1c + 0x0100038b + 0x5808c483 + 0x5808c483
ebp : 0x0101af28 + 0x0101b7c + 0x01001930 + 0x0191c8b8 + 0x00000000 + 0x00000000
esi : 0x01012ba8 + 0x00000000 + 0x00000000
edi : 0x02011200 + 0x02011200
ebp : 0x01002940 + 0xe8535657 + 0xe8535657
eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
fs: 0x0008 fs: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010

```

registers

Registerinhalt

```

0x0101af1c +0x0000: 0x0100038b + 0x5808c483 + 0x5808c483 - fesp
0x0101af20 +0x0004: 0x00000021 + 0x00000021
0x0101af24 +0x0008: 0x0101af28 + 0x0101b520 + 0x00000000 + 0x00000000
0x0101af28 +0x000c: 0x0101b520 + 0x00000000 + 0x00000000
0x0101af2c +0x0010: 0x01010870 + 0x8b535657 + 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
0x0101af34 +0x0018: 0x010114dd + 0xe0b6ff5a + 0xe0b6ff5a
0x0101af38 +0x001c: 0x00000008 + 0x00000008

```

stack

Stackinhalt

```

0xd002d3b: xchg ax, ax
0xd002d3d: xchg ax, ax
0xd002d3f: nop
- 0xd002d40 <guardian+0>: push edi
0xd002d41 <guardian+1>: push esi
0xd002d42 <guardian+2>: push ebx
0xd002d43 <guardian+3>: call 0xd010f20 <__x86.get_pc_thunk.bx>
0xd002d48 <guardian+8>: add ebx, 0xfe60
0xd002d4e <guardian+14>: sub esp, 0x8

```

code:x86:32

Stelle im Assembly

source:./guard/guardian.cc:19

```

14
15 #include "guard/guard.h"
16 extern Guard guard;
17
18 extern "C" void guardian(uint32_t vector, irq_context *context)
- 19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugbox.report(vector);
23
24     bool wantsEpilogue = gate->prologue();

```

Stelle in C/C++

threads

```

[*0] Id 1, Name: "", stopped, reason: BREAKPOINT
[*1] Id 2, Name: "", stopped, reason: BREAKPOINT
[*2] Id 3, Name: "", stopped, reason: BREAKPOINT
[*3] Id 4, Name: "", stopped, reason: BREAKPOINT

```

Threads

```

[*0] 0xd002d40 - guardian(vector=0x21, context=0xd01af28)
[*1] 0xd00038b - irq_context_33()
[*2] 0xd01ffb00 - add $R15, $R11, 0
[*3] 0xd005aa8 - kernel_init()
[*4] 0xd01b5e0 - add $R15, $R11, 0

```

Thread 1 hit Breakpoint 2, guardian (vector=0x21, context=0xd01af28) at ./guard/guardian.cc:19

```

19 {
gef+

```

```

eax: 0x0101b520 + 0x00000000 + 0x00000000
ebx: 0x01012ba8 + 0x00000000 + 0x00000000
ecx: 0x01010870 + 0x8b535657 + 0x8b535657
edx: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
esp: 0x0101af1c + 0x0100038b + 0x5808c483 + 0x5808c483
ebp: 0x0101af28 + 0x0101b7c + 0x01001930 + 0x0191c8b8 + 0x00000000 + 0x00000000
esi: 0x01012ba8 + 0x00000000 + 0x00000000
edi: 0x02011200 + 0x02011200
ebp: 0x01002d40 + 0xe8535657 + 0xe8535657
eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
fs: 0x0008 gs: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010

```

registers

Registerinhalt

```

0x0101af1c +0x0000: 0x0100038b + 0x5808c483 + 0x5808c483 - fesp
0x0101af20 +0x0004: 0x00000021 + 0x00000021
0x0101af24 +0x0008: 0x0101af28 + 0x0101b520 + 0x00000000 + 0x00000000
0x0101af28 +0x000c: 0x0101b520 + 0x00000000 + 0x00000000
0x0101af2c +0x0010: 0x01010870 + 0x8b535657 + 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
0x0101af34 +0x0018: 0x010114dd + 0xe0b6ff5a + 0xe0b6ff5a
0x0101af38 +0x001c: 0x00000008 + 0x00000008

```

stack

Stackinhalt

```

0x1002d3b: xchg ax, ax
0x1002d3d: xchg ax, ax
0x1002d3f: nop
- 0x1002d40 <guardian+0>: push edi
0x1002d41 <guardian+1>: push esi
0x1002d42 <guardian+2>: push ebx
0x1002d43 <guardian+3>: call 0x1010f20 <__x86.get_pc_thunk.bx>
0x1002d48 <guardian+8>: add ebx, 0xfe60
0x1002d4e <guardian+14>: sub esp, 0x8

```

code:x86:32

Stelle im Assembly

source:./guard/guardian.cc:19

```

14
15 #include "guard/guard.h"
16 extern Guard guard;
17
18 extern "C" void guardian(uint32_t vector, irq_context *context)
- 19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugbox.report(vector);
23
24     bool wantsEpilogue = gate->prologue();

```

Stelle in C/C++

threads

```

[*0] Id 1, Name: "", stopped, reason: BREAKPOINT
[*1] Id 2, Name: "", stopped, reason: BREAKPOINT
[*2] Id 3, Name: "", stopped, reason: BREAKPOINT
[*3] Id 4, Name: "", stopped, reason: BREAKPOINT

```

Threads

trace

```

[*0] 0x1002d40 - guardian(vector=0x21, context=0x101af28)
[*1] 0x100038b - irq_entry_33()
[*2] 0x1ffb000 - add BYTE PTR [eax+0x11], di
[*3] 0x1005aa8 - kernel_init()
[*4] 0x101b5e0 - add BYTE PTR [eax], al

```

Backtrace

Thread 1 hit Breakpoint 2, guardian (vector=0x21, context=0x101af28) at ./guard/guardian.cc:19

```

19 {
getf

```

```

eax : 0x0101b520 + 0x00000000 + 0x00000000
ebx : 0x01012ba8 + 0x00000000 + 0x00000000
ecx : 0x01010870 + 0x8b535657 + 0x8b535657
edx : 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
esp : 0x0101af1c + 0x0100038b + 0x5808c483 + 0x5808c483
ebp : 0x0101af28 + 0x0101b7c + 0x01001930 + 0x0191c8b8 + 0x00000000 + 0x00000000
esi : 0x01012ba8 + 0x00000000 + 0x00000000
edi : 0x02011200 + 0x02011200
ebp : 0x01002d40 + 0xe8535657 + 0xe8535657
eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
fs: 0x0008 fs: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010

```

registers

Registerinhalt

```

0x0101af1c +0x0000: 0x0100038b + 0x5808c483 + 0x5808c483 - fesp
0x0101af20 +0x0004: 0x00000021 + 0x00000021
0x0101af24 +0x0008: 0x0101af28 + 0x0101b520 + 0x00000000 + 0x00000000
0x0101af28 +0x000c: 0x0101b520 + 0x00000000 + 0x00000000
0x0101af2c +0x0010: 0x01010870 + 0x8b535657 + 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 + 0x00119000 + 0x00000000 + 0x00000000
0x0101af34 +0x0018: 0x010114dd + 0xe0b6ff5a + 0xe0b6ff5a
0x0101af38 +0x001c: 0x00000008 + 0x00000008

```

stack

Stackinhalt

```

0x1002d3b: xchg ax, ax
0x1002d3d: xchg ax, ax
0x1002d3f: nop
- 0x1002d40 <guardian+0>: push edi
0x1002d41 <guardian+1>: push esi
0x1002d42 <guardian+2>: push ebx
0x1002d43 <guardian+3>: call 0x1010f20 <__x86.get_pc_thunk.bx>
0x1002d48 <guardian+8>: add ebx, 0xfe60
0x1002d4e <guardian+14>: sub esp, 0x8

```

code:x86:32

Stelle im Assembly

source:./guard/guardian.cc:19

```

14
15 #include "guard/guard.h"
16 extern Guard guardian;
17
18 extern "C" void guardian(uint32_t vector, irq_context *context)
- 19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugbox.report(vector);
23
24     bool wantsEpilogue = gate->prologue();

```

Stelle in C/C++

threads

```

[#0] Id 1, Name: "", stopped, reason: BREAKPOINT
[#1] Id 2, Name: "", stopped, reason: BREAKPOINT
[#2] Id 3, Name: "", stopped, reason: BREAKPOINT
[#3] Id 4, Name: "", stopped, reason: BREAKPOINT

```

Threads

trace

```

[#0] 0x1002d40 - guardian(vector=0x21, context=0x101af28)
[#1] 0x100038b - irq_entry_33()
[#2] 0x1ffb000 - add BYTE PTR [eax+0x11], di
[#3] 0x1005aaa8 - kernel_init()
[#4] 0x101b5e0 - add BYTE PTR [eax], al

```

Backtrace

Thread 1 hit Breakpoint 2, guardian (vector=0x21, context=0x101af28) at ./guard/guardian.cc:19

```

19 {
gef>

```

Eingabezeile

Breakpoints: (gdb) break <location>

Breakpoints

Unterbrechen der Ausführung, sobald eine bestimmte **Codestelle** erreicht wird.

- Funktionsname
 - absolute/relative Codezeile
 - *Adresse
- } optionaler Prefix: Quelldatei

Unterbricht vor dem Ausführen von...

```
(gdb) b main
```

Funktion main

```
(gdb) b main.cc:main
```

... aus main.cc

```
(gdb) b 63
```

Zeile 63 in aktueller Datei

```
(gdb) b main.cc:63
```

Zeile 63 in main.cc

```
(gdb) b +3
```

In 3 Zeilen

```
(gdb) b *0x100a9ca
```

An Adresse 0x100a9ca

Temporäre & Bedingte Breakpoints

Temporäre Breakpoints: `(gdb) tbreak <location>`

Werden nach dem 1. Auslösen entfernt, sonst wie „normale“ Breakpoints.

Temporäre & Bedingte Breakpoints

Temporäre Breakpoints: `(gdb) tbreak <location>`

Werden nach dem 1. Auslösen entfernt, sonst wie „normale“ Breakpoints.

Bedingte Breakpoints: `(gdb) break <location> if <cond>`

Unterbrechung nur falls Bedingung erfüllt ist, z.B:

```
(gdb) break interrupt_handler if vector == 33
```

Unterbricht nur, falls die Funktion `interrupt_handler` aufgrund von Tastatureingabe (Vektor 33) betreten wurde.

Temporäre & Bedingte Breakpoints

Temporäre Breakpoints: `(gdb) tbreak <location>`

Werden nach dem 1. Auslösen entfernt, sonst wie „normale“ Breakpoints.

Bedingte Breakpoints: `(gdb) break <location> if <cond>`

Unterbrechung nur falls Bedingung erfüllt ist, z.B:

```
(gdb) break interrupt_handler if vector == 33
```

Unterbricht nur, falls die Funktion `interrupt_handler` aufgrund von Tastatureingabe (Vektor 33) betreten wurde.



Achtung: Nichttriviale Break- oder Watchpoints werden ohne Hardwareunterstützung umgesetzt → Langsam!

Watchpoints (Data Breakpoints)

Unterbricht wenn Speicherbereich geschrieben (oder gelesen) wird:

watch <location> Schreibzugriff

rwatch <location> Lesezugriff

awatch <location> Schreib- oder Lesezugriff

```
(gdb) watch guard
```

```
(gdb) watch guard if guard.locked == 1
```

Verwalten von Break-/Watchpoints

ignore	<id> <N>	Breakpoint N mal ignorieren
enable	<id> <id> ..	Breakpoints aktivieren
disable	<id> <id> ..	Breakpoints deaktivieren
delete	<id> <id> ..	Breakpoints löschen



Schrittweise Ausführung

step *[count]* – Nächste Zeile

stepi *[count]* – Nächste Instruktion

next *[count]* – Nächste Zeile (ohne Funktionen zu betreten)

nexti *[count]* – Nächste Instruktion (ohne Funktionen zu betreten)

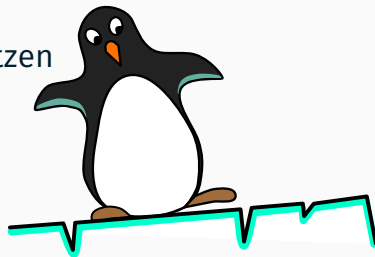
until *[count]* – Wiederhole **next** bis zur **textuell** nächsten Zeile

↑ _____ Optional: Anzahl Wiederholungen

finish – Bis zum return des aktuellen Stackframes

advance <location> – Bis zu <location>

continue – Ausführung (bis zum Haltepunkt) fortsetzen



Der skip Befehl

```
unsigned int numCPUs = System::getNumberOfCPUs();  
kout << "numCPUs:_" << numCPUs << endl;  
ApplicationProcessor::boot();
```

Der skip Befehl

```
unsigned int numCPUs = System::getNumberOfCPUs();  
kout << "numCPUs:_" << numCPUs << endl;  
ApplicationProcessor::boot();
```

Übergehe aktuelle Funktion:

```
(gdb) skip
```

Übergehe eine einzelne Funktion:

```
(gdb) skip function OutputStream::operator<<
```

Übergehe alle Funktionen aus einer Datei:

```
(gdb) skip file object/outputstream.cc
```

Der info Befehl

info args	Auflistung der Aufrufargumente
info locals	Auflistung aller lokalen Variablen
info registers	Auflistung der Registerwerte
info breakpoints	Auflistung der Breakpoints
info threads	Auflistung der Threads/CPU's
info skip	Auflistung der zu überspringenden Funktionen
...	siehe (gdb) help info

Ergänzend:

backtrace	Auflistung geschachtelter Stackframes
frame <id>	Wechsel zu Stackframe
thread <id>	Wechsel zu Thread/CPU

Ausgabe von Werten in Speicher / Register

```
(gdb) print/<Format> <Ausdruck>
```

```
(gdb) examine/<Anzahl><Format><Einheit> <Ausdruck>
```

Werte für <Format>

x	Ganzzahl (hex)
d	Ganzzahl (mit VZ, dezimal)
u	Ganzzahl (ohne VZ, dezimal)
t	Ganzzahl (binär) (two)
a	Adresse (hex) + Offset zum Startsymbol
f	Float
i	Instruktion

Werte für <Einheit>

b	Byte	8-bit
h	Halfword	16-bit
w	Word	32-bit
g	Giant word	64-bit

Bestimmen des Typs eines Symbols

```
ptype <Symbol>
```

Verändern des Zielsystems: (gdb) set

Verändern eines Registers

```
(gdb) set $esp = 0xdeadbeef
```

Verändern einer Variable / Speicherbereichs

```
(gdb) set numCPUs = 2
```

```
(gdb) set *((int *) 0x1013fdc) = 42
```

GDB vs. Optimierungen

Generell: Optimierungen sind doof fürs Debuggen:

- Inlining von Funktionen
- Elimination von Variablen
- ...

Relevante Compileroptionen

- g Generiere Debuginformationen
- O0 Optimierungen aus
- Og Nur Optimierungen, die das Debuggen nicht stören

GDB vs. Optimierungen

Generell: Optimierungen sind doof fürs Debuggen:

- Inlining von Funktionen
- Elimination von Variablen
- ...

Relevante Compileroptionen

- g Generiere Debuginformationen
- O0 Optimierungen aus
- Og Nur Optimierungen, die das Debuggen nicht stören
- O2 Fast alle Optimierungen

Laden der .gdbinit

Automatische Ausführung von gdb-Befehlen in .gdbinit

- Vordefinierte Breakpoints
- Eigene Skripte

~/.config/gdb/gdbinit oder ~/.gdbinit

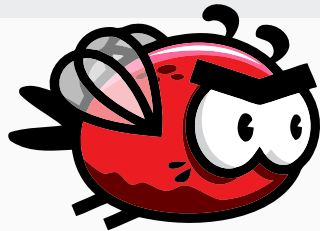
auto-load local-gdbinit	Lade lokale .gdbinit-Datei
add-auto-load-safe-path ~/stubs	Schalte Pfad zum Laden frei
set disassembly-flavor intel	Nutze Intel-, statt AT&T-Syntax
set print asm-demangle on	Löse Name Mangling auf

Beispielhafte projektlokale .gdbinit

```
b assertion_failed  
b Core::die
```

Die bittere Wahrheit...

Ihr werdet entkäfern müssen



Die bittere Wahrheit...

Ihr werdet entkäfern müssen

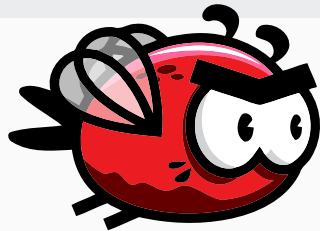
- `make qemu-gdb-noopt`

→ Profit?

- `make qemu-gdb`

- `make VERBOSE= QEMUCPUS=1 all`

- `make qemu-wait + make gdb` oder externer Debugger



Vorgegebenes GEF

```
source tools/gdbinit-gef.py
```

Fragen?



<https://sourceware.org/gdb/current/onlinedocs/gdb.pdf>
<https://www.qemu.org/docs/master/system/monitor.html>

Anhang

Snippet: Ausgabe von Details zur Exception im Interrupt Handler

```
// Optional: Print exceptions on DBG stream to support debugging
if (vector <= Core::Interrupt::SECURITY_EXCEPTION) {
    DBG << "Exception_" << dec << vector;
    switch (vector) {
        case 0:  DBG << "_(Div_by_0)"; break;
        case 6:  DBG << "_(InvalidOpcode)"; break;
        case 10: DBG << "_(InvalidTSS)"; break;
        case 13: DBG << "_(GeneralProtectionFault)"; break;
        case 14: DBG << "_(PageFault)"; break;
        default: break;
    }
    if (context->error_code != 0) {
        DBG << "_" << bin << context->error_code << "_";
    }
    DBG << "_@" << hex << context->ip << flush;
    DBG << endl;
}
```