



Übung zu Betriebssystembau

Debuggen mit GDB & GEF

Oktober 2025

Alexander Krause

Arbeitsgruppe Systemsoftware
Technische Universität Dortmund

(Mit Material vom Lehrstuhl 4 der FAU)



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 that 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
 - dir.cc
 - dir.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

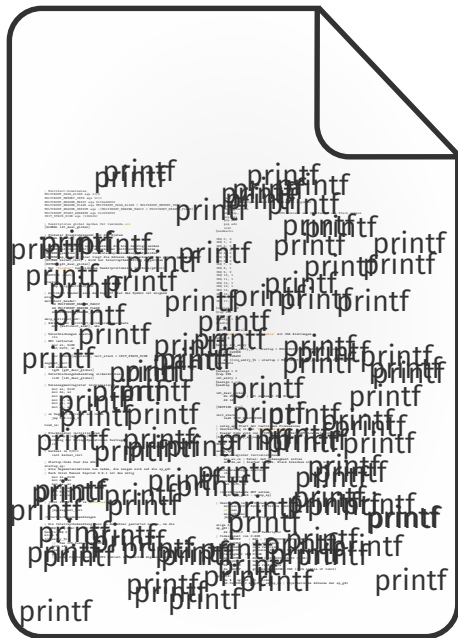
- utils
 - elf.cc
 - elf.h
 - libcc.cc
 - math.h
 - memutil.cc
 - memutil.h
 - sort.h
 - string.cc
 - string.h
 - tar.cc
 - tar.h
- libsys
 - ioStream.h
 - Makefile
 - o_stream.cc
 - o_stream.h
 - o_stream.h
 - strbuf.cc
 - strbuf.h
 - syscall_stubs.asm
 - syscall_stubs.cc
 - types.h
- Makefile
- README.md
- 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

[illegible]

```

// 1 // 2 // 3 // 4 // 5 // 6 // 7 // 8 // 9 // 10 // 11 // 12 // 13 // 14 // 15 // 16 // 17 // 18 // 19 // 20 // 21 // 22 // 23 // 24 // 25 // 26 // 27 // 28 // 29 // 30 // 31 // 32 // 33 // 34 // 35 // 36 // 37 // 38 // 39 // 40 // 41 // 42 // 43 // 44 // 45 // 46 // 47 // 48 // 49 // 50 // 51 // 52 // 53 // 54 // 55 // 56 // 57 // 58 // 59 // 60 // 61 // 62 // 63 // 64 // 65 // 66 // 67 // 68 // 69 // 70 // 71 // 72 // 73 // 74 // 75 // 76 // 77 // 78 // 79 // 80 // 81 // 82 // 83 // 84 // 85 // 86 // 87 // 88 // 89 // 90 // 91 // 92 // 93 // 94 // 95 // 96 // 97 // 98 // 99 // 100 // 101 // 102 // 103 // 104 // 105 // 106 // 107 // 108 // 109 // 110 // 111 // 112 // 113 // 114 // 115 // 116 // 117 // 118 // 119 // 120 // 121 // 122 // 123 // 124 // 125 // 126 // 127 // 128 // 129 // 130 // 131 // 132 // 133 // 134 // 135 // 136 // 137 // 138 // 139 // 140 // 141 // 142 // 143 // 144 // 145 // 146 // 147 // 148 // 149 // 150 // 151 // 152 // 153 // 154 // 155 // 156 // 157 // 158 // 159 // 160 // 161 // 162 // 163 // 164 // 165 // 166 // 167 // 168 // 169 // 170 // 171 // 172 // 173 // 174 // 175 // 176 // 177 // 178 // 179 // 180 // 181 // 182 // 183 // 184 // 185 // 186 // 187 // 188 // 189 // 190 // 191 // 192 // 193 // 194 // 195 // 196 // 197 // 198 // 199 // 200 // 201 // 202 // 203 // 204 // 205 // 206 // 207 // 208 // 209 // 210 // 211 // 212 // 213 // 214 // 215 // 216 // 217 // 218 // 219 // 220 // 221 // 222 // 223 // 224 // 225 // 226 // 227 // 228 // 229 // 230 // 231 // 232 // 233 // 234 // 235 // 236 // 237 // 238 // 239 // 240 // 241 // 242 // 243 // 244 // 245 // 246 // 247 // 248 // 249 // 250 // 251 // 252 // 253 // 254 // 255 // 256 // 257 // 258 // 259 // 260 // 261 // 262 // 263 // 264 // 265 // 266 // 267 // 268 // 269 // 270 // 271 // 272 // 273 // 274 // 275 // 276 // 277 // 278 // 279 // 280 // 281 // 282 // 283 // 284 // 285 // 286 // 287 // 288 // 289 // 290 // 291 // 292 // 293 // 294 // 295 // 296 // 297 // 298 // 299 // 300 // 301 // 302 // 303 // 304 // 305 // 306 // 307 // 308 // 309 // 310 // 311 // 312 // 313 // 314 // 315 // 316 // 317 // 318 // 319 // 320 // 321 // 322 // 323 // 324 // 325 // 326 // 327 // 328 // 329 // 330 // 331 // 332 // 333 // 334 // 335 // 336 // 337 // 338 // 339 // 340 // 341 // 342 // 343 // 344 // 345 // 346 // 347 // 348 // 349 // 350 // 351 // 352 // 353 // 354 // 355 // 356 // 357 // 358 // 359 // 360 // 361 // 362 // 363 // 364 // 365 // 366 // 367 // 368 // 369 // 370 // 371 // 372 // 373 // 374 // 375 // 376 // 377 // 378 // 379 // 380 // 381 // 382 // 383 // 384 // 385 // 386 // 387 // 388 // 389 // 390 // 391 // 392 // 393 // 394 // 395 // 396 // 397 // 398 // 399 // 400 // 401 // 402 // 403 // 404 // 405 // 406 // 407 // 408 // 409 // 410 // 411 // 412 // 413 // 414 // 415 // 416 // 417 // 418 // 419 // 420 //
```



GNU Debugger (GDB)

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

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
- + Bietet verschiedene Kommandos an – siehe Dokumentation von Gef

[Legend: **Modified register** | Code | Heap | Stack | String]

Registerinhalt

```
*eax : 0x0101b520 -> 0x00000000 - 0x00000000
*ebx : 0x01012ba8 -> 0x00000000 - 0x00000000
*ecx : 0x01010870 -> 0x85353557 - 0x85353557
*edx : 0x01ffb900 -> 0x00113000 - 0x00000000 - 0x00000000
*esp : 0x0101af1c -> 0x0100038b -> 0x5008c483 -> 0x5008c483
*ebp : 0x0101af88 -> 0x01011b7c -> 0x01001330 -> 0x0191c8b8 -> 0x00000000 - 0x00000000
*esi : 0x01012ba8 -> 0x00000000 - 0x00000000
*edi : 0x02011200 -> 0x02011200
*ebp : 0x01002d40 -> 0x85353557 - 0x85353557
$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
```

```
0x0101af1c: 0x0000: 0x0100038b -> 0x5008c483 -> 0x5008c483 -> $esp
0x0101af20: 0x0004: 0x01000321 -> 0x00000000
0x0101af24: 0x0008: 0x0101af28 -> 0x00000000 - 0x00000000
0x0101af28: 0x000c: 0x0101af2c -> 0x00000000 - 0x00000000
0x0101af2c: 0x0010: 0x0101af30 -> 0x85353557 - 0x85353557
0x0101af30: 0x0014: 0x0101af34 -> 0x00000000 - 0x00000000
0x0101af34: 0x0018: 0x0101af38 -> 0x00000000 - 0x00000000
0x0101af38: 0x001c: 0x00000000 - 0x00000000
```

```
0x1002130: xchg ax, ax
0x1002150: xchg ax, ax
0x100216f: nop
-> 0x1002d40 <guardian0> push edi
0x1002d41 <guardian1> push esi
0x1002d42 <guardian2> push ebx
0x1002d43 <guardian3> call 0x1010f20 <__x86_get_pc_thunk,bx>
0x1002d48 <guardian8> add ebx, 0xfe60
0x1002d4e <guardian14> sub esp, 0x8
```

```
14
15 #include "guard/guard.h"
16 extern Guard guardian;
17
18 extern "C" void guardianInit(S21 vector, irq_context *context)
19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugbox_report(vector);
23
24     bool wantsEpilogue = gate->prologue();
```

```
[#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
```

```
[#0] 0x1002d40 -> guardian(vector=0x21, context=0x101af28)
[#1] 0x1002b8 -> irq_entry_1()
[#2] 0x1ffb900 -> irq_ctx_irqs[ecx].el
[#3] 0x1005aa8 -> kernel_init()
[#4] 0x101b5e0 -> irq_ctx_irqs[ecx].el
```

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

```
19 {
gef> |
```

[Legend: **Modified register** | Code | Heap | Stack | String]

Registerinhalt

```
*eax: 0x0101b520 -> 0x00000000 - 0x00000000
*ebx: 0x01012ba8 -> 0x00000000 - 0x00000000
*ecx: 0x01010870 -> 0x85336557 - 0x85336557
*edx: 0x01ffb900 -> 0x00113000 - 0x00000000 - 0x00000000
*esp: 0x0101af1c -> 0x0100038b -> 0x5808c483 -> 0x5808c483
*ebp: 0x0101af88 -> 0x01011b7c -> 0x01001330 -> 0x0191c8b8 -> 0x00000000 - 0x00000000
*esi: 0x01012ba8 -> 0x00000000 - 0x00000000
*edi: 0x02011200 -> 0x02011200
*ebp: 0x01002d40 -> 0x85336557 - 0x85336557
*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
```

Stackinhalt

```
0x0101af1c +0x0000: 0x0100038b -> 0x5808c483 + 0x5808c483 - *esp
0x0101af20 +0x0004: 0x00000021 -> 0x00000021
0x0101af24 +0x0008: 0x0101af28 -> 0x0101b529 -> 0x00000000 - 0x00000000
0x0101af28 +0x000c: 0x0101b520 -> 0x00000000 - 0x00000000
0x0101af2c +0x0010: 0x01010870 -> 0x85336557 - 0x85336557
0x0101af30 +0x0014: 0x01ffb900 -> 0x00113000 -> 0x00000000 - 0x00000000
0x0101af34 +0x0018: 0x01011b7c -> 0x00000000 - 0x00000000
0x0101af38 +0x001c: 0x00000008 -> 0x00000008
```

```
0x1000130: xchg ax, ax
0x1000131: xchg ax, ax
0x1000132: mov
- 0x1002d40 <guardian0>: push edi
0x1002d41 <guardian1>: push esi
0x1002d42 <guardian2>: push ebx
0x1002d43 <guardian3>: call 0x1010f20 <__x86.get_pc_thunk, bx>
0x1002d48 <guardian8>: add ebx, 0xfe60
0x1002d4e <guardian14>: sub esp, 0x8
```

```
14
15 #include "guard/guard.h"
16 extern Guard guardian;
17
18 extern "C" void guardianInit(S21 vector, irq_context *context)
19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = plugbox_report(vector);
23
24     bool wantsEpilogue = gate->prologue();
```

```
[#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
```

```
[#0] 0x1002d40 -> guardian(vector=0x21, context=0x0101af28)
[#1] 0x100038b -> irq_entry_1()
[#2] 0x01ffb900 -> irq_ctx_tk_irq=0x011b7c
[#3] 0x1005aa8 -> irq_ctx_tk_irq=0x011b7c
[#4] 0x0101b520 -> irq_ctx_tk_irq=0x011b7c
```

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

```
19 {
gef> █
```

[Legend: **Modified register** | Code | Heap | Stack | String]

Registerinhalt

```
*eax : 0x0101b520 -> 0x00000000 - 0x00000000
*ebx : 0x01012ba8 -> 0x00000000 - 0x00000000
*ecx : 0x01010870 -> 0x8b535657 - 0x8b535657
*edx : 0x01ffb900 -> 0x00113000 - 0x00000000 - 0x00000000
*esp : 0x0101af1c -> 0x0100038b -> 0x5808c483 -> 0x5808c483
*ebp : 0x0101af88 -> 0x01011b7c -> 0x01001330 -> 0x0191c8b8 -> 0x00000000 - 0x00000000
*esi : 0x01012ba8 -> 0x00000000 - 0x00000000
*edi : 0x02011200 -> 0x02011200
*ebp : 0x01002d40 -> 0x8b535657 - 0x8b535657
$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
```

Stackinhalt

```
0x0101af1c +0x0000: 0x0100038b -> 0x5808c483 -> 0x5808c483 -> *esp
0x0101af20 +0x0004: 0x00000021 -> 0x00000021
0x0101af24 +0x0008: 0x0101af28 -> 0x0101b529 -> 0x00000000 - 0x00000000
0x0101af28 +0x000c: 0x0101b520 -> 0x00000000 - 0x00000000
0x0101af2c +0x0010: 0x01010870 -> 0x8b535657 -> 0x8b535657
0x0101af30 +0x0014: 0x01ffb900 -> 0x00113000 -> 0x00000000 - 0x00000000
0x0101af34 +0x0018: 0x01011b7c -> 0x0b06fffa -> 0x0b06fffa
0x0101af38 +0x001c: 0x00000008 -> 0x00000008
```

Stelle im Assembly

```
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
```

source: ./guard/guardian.cc:19

```
14
15 #include "guard/guard.h"
16 extern Guard guardian;
17
18 extern "C" void guardianInit(S21 vector, irq_context *context)
19 {
20     (void) vector;
21     (void) context;
22     Gate* gate = pluginbox_report(vector);
23
24     bool wantsEpilogue = gate->prologue();
```

breakpoints

```
[#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
```

trace

```
[#0] 0x1002d40 -> guardian(vector=0x21, context=0x0101af28)
[#1] 0x100038b -> __x86.get_pc_thunk.bx()
[#2] 0x01ffb900 -> __x86.get_pc_thunk.bx()
[#3] 0x1005aa8 -> __x86.get_pc_thunk.bx()
[#4] 0x0101b520 -> __x86.get_pc_thunk.bx()
```

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

```
19 {
gef> |
```

[Legend: **Modified register** | Code | Heap | Stack | String]

```
*eax : 0x0101b520 -> 0x00000000 - 0x00000000
*ebx : 0x01012ba8 -> 0x00000000 - 0x00000000
*ecx : 0x01010870 -> 0x8b539657 - 0x8b539657
*edx : 0x01ffb900 -> 0x00119000 - 0x00000000 - 0x00000000
*esp : 0x0101af1c -> 0x0100038b -> 0x5808c483 -> 0x5808c483
*ebp : 0x0101af88 -> 0x01011b7c -> 0x01001330 -> 0x0191c8b8 -> 0x00000000 - 0x00000000
*esi : 0x01012ba8 -> 0x00000000 - 0x00000000
*edi : 0x02011200 -> 0x02011200
*ebp : 0x01002d40 -> 0x8b539657 - 0x8b539657
$eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
fs: 0x0008 kss: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010 fs: 0x0010
```

Registerinhalt

Stackinhalt

```
0x0101af1c +0x0000: 0x0100038b -> 0x5808c483 -> 0x5808c483 -> *esp
0x0101af20 +0x0004: 0x00000021 -> 0x00000021
0x0101af24 +0x0008: 0x0101af28 -> 0x0101b520 -> 0x00000000 -> 0x00000000
0x0101af28 +0x000c: 0x0101b520 -> 0x00000000 -> 0x00000000
0x0101af2c +0x0010: 0x01010870 -> 0x8b539657 -> 0x8b539657
0x0101af30 +0x0014: 0x01ffb900 -> 0x00119000 -> 0x00000000 -> 0x00000000
0x0101af34 +0x0018: 0x01011b7c -> 0x0b65ff5a -> 0x0b65ff5a
0x0101af38 +0x001c: 0x00000008 -> 0x00000008
```

Stelle im Assembly

```
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
```

Stelle in C/C++

```
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();
```

```
[*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
```

```
[*0] 0x1002d40 -> guardian(vector=0x21, context=0x0101af28)
[*1] 0x100038b -> __x86.get_pc_thunk, bx
[*2] 0x01ffb900 -> __x86.get_pc_thunk, ebx
[*3] 0x005a00 -> __x86.get_pc_thunk, ebx
[*4] 0x0101b520 -> __x86.get_pc_thunk, esi
```

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

```
19 {
gef> █
```

[Legend: **Modified register** | Code | Heap | Stack | String]

Registerinhalt

```
*eax: 0x0101b520 -> 0x00000000 - 0x00000000
*ebx: 0x01012ba8 -> 0x00000000 - 0x00000000
*ecx: 0x01010870 -> 0x85336557 - 0x85336557
*edx: 0x01ffb900 -> 0x00113000 - 0x00000000 - 0x00000000
*esp: 0x0101af1c -> 0x0100038b - 0x5808c483 - 0x5808c483
*ebp: 0x0101af88 -> 0x01011b7c - 0x01001330 - 0x0191c8b8 - 0x00000000 - 0x00000000
*esi: 0x01012ba8 -> 0x00000000 - 0x00000000
*edi: 0x02011200 -> 0x02011200
*ebp: 0x01002d40 -> 0x85336557 - 0x85336557
$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
```

Stackinhalt

```
0x0101af1c +0x0000: 0x0100038b -> 0x5808c483 - 0x5808c483 - *$esp
0x0101af20 +0x0004: 0x00000021 -> 0x00000021
0x0101af24 +0x0008: 0x0101af28 -> 0x0101b529 - 0x00000000 - 0x00000000
0x0101af28 +0x000c: 0x0101b520 -> 0x00000000 - 0x00000000
0x0101af2c +0x0010: 0x01010870 -> 0x85336557 - 0x85336557
0x0101af30 +0x0014: 0x01ffb900 -> 0x00113000 - 0x00000000 - 0x00000000
0x0101af34 +0x0018: 0x01011b7c -> 0x0b65ff5a - 0x0b65ff5a
0x0101af38 +0x001c: 0x00000008 -> 0x00000008
```

Stelle im Assembly

```
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
```

Stelle in C/C++

```
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();
```

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
```

```
[#0] 0x1002d40 -> guardian(vector=0x21, context=0x0101af28)
[#1] 0x100038b -> __x86.get_pc_thunk,bl
[#2] 0x01ffb900 -> __x86.get_pc_thunk,ebx
[#3] 0x1005aa8 -> _start_init()
[#4] 0x0101b520 -> __x86.get_pc_thunk,esi
```

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

```
19 {
gef> |
```

[Legend: **Modified register** | Code | Heap | Stack | String]

```
*eax: 0x0101b520 -> 0x00000000 - 0x00000000
*ebx: 0x01012ba8 -> 0x00000000 - 0x00000000
*ecx: 0x01010870 -> 0x85336557 - 0x85336557
*edx: 0x01ffb000 -> 0x00113000 - 0x00000000 - 0x00000000
*esp: 0x0101af1c -> 0x0100038b - 0x5808c483 - 0x5808c483
*ebp: 0x0101afa8 -> 0x01011b7c - 0x01001330 - 0x0191c8b8 - 0x00000000 - 0x00000000
*esi: 0x01012ba8 -> 0x00000000 - 0x00000000
*edi: 0x02011200 -> 0x02011200
*ebp: 0x01002d40 -> 0x85336557 - 0x85336557
$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
```

Registerinhalt

Stackinhalt

```
0x0101af1c +0x0000: 0x0100038b -> 0x5808c483 -> 0x5808c483 -> *esp
0x0101af20 +0x0004: 0x00000021 -> 0x00000021
0x0101af24 +0x0008: 0x0101af28 -> 0x0101b529 -> 0x00000000 -> 0x00000000
0x0101af28 +0x000c: 0x0101b530 -> 0x00000000 -> 0x00000000
0x0101af2c +0x0010: 0x01010870 -> 0x85336557 -> 0x85336557
0x0101af30 +0x0014: 0x01ffb000 -> 0x00113000 -> 0x00000000 -> 0x00000000
0x0101af34 +0x0018: 0x010114d4 -> 0xe0b5ff5a -> 0xe0b5ff5a
0x0101af38 +0x001c: 0x00000008 -> 0x00000008
```

```
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
```

```
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();
```

```
[#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
```

```
[#0] 0x1002d40 - guardian(vector=0x21, context=0x0101af28)
[#1] 0x100038b - irq_entry_33()
[#2] 0x1ffb000 - add BYTE PTR [eax+0x11], dl
[#3] 0x1005aa8 - kernel_init()
[#4] 0x01b5e0 - add BYTE PTR [eax], al
```

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

Stelle im Assembly

Stelle in C/C++

Threads

Backtrace

[Legend: Modified register | Code | Heap | Stack | String]

Registerinhalt

```
*eax : 0x0101b520 -> 0x00000000 - 0x00000000
*ebx : 0x01012ba8 -> 0x00000000 - 0x00000000
*ecx : 0x01010870 -> 0x8b535657 - 0x8b535657
*edx : 0x01ffb000 -> 0x00113000 - 0x00000000 - 0x00000000
*esp : 0x0101af1c -> 0x0100038b - 0x5808c483 - 0x5808c483
*ebp : 0x0101af08 -> 0x01011b7c - 0x01001330 - 0x0191c8b8 - 0x00000000 - 0x00000000
*esi : 0x01012ba8 -> 0x00000000 - 0x00000000
*edi : 0x02011200 -> 0x02011200
*ebp : 0x01002d40 -> 0x8b535657 - 0x8b535657
$eflags: [carry parity adjust zero sign trap interrupt direction overflow resume virtualx86 identification]
cs: 0x0008  eip: 0x0010  fs: 0x0010  gs: 0x0010  iopl: 0x0010  iopl: 0x0010  iopl: 0x0010
```

Stackinhalt

```
0x0101af1c +0x0000: 0x0100038b -> 0x5808c483 - 0x5808c483 - *esp
0x0101af20 +0x0004: 0x00000021 -> 0x00000021
0x0101af24 +0x0008: 0x0101af28 -> 0x0101b529 - 0x00000000 - 0x00000000
0x0101af28 +0x000c: 0x00000000 -> 0x00000000 - 0x00000000
0x0101af2c +0x0010: 0x01010870 -> 0x8b535657 - 0x8b535657
0x0101af30 +0x0014: 0x01ffb000 -> 0x00113000 - 0x00000000 - 0x00000000
0x0101af34 +0x0018: 0x01011b7c -> 0x0b56ff5a - 0x0b56ff5a
0x0101af38 +0x001c: 0x00000008 -> 0x00000008
```

Stelle im Assembly

```
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
```

Stelle in C/C++

```
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();
```

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
```

Backtrace

```
[*0] 0x1002d40 - guardian(vector=0x21, context=0x0101af28)
[*1] 0x100038b - irq_entry_33()
[*2] 0x1ffb000 - add BYTE PTR [eax+0x11], dl
[*3] 0x1005aa8 - kernel_init()
[*4] 0x0101b5e0 - add BYTE PTR [eax], al
```

Eingabezeile

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

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 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) b 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
```



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



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 (zum nächsten Breakpoint) fortsetzen



Der skip Befehl

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



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

Übergehe eine einzelne Funktion:

```
(gdb) skip Guard::enter
```

Übergehe alle Funktionen aus einer Datei:

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



<code>info locals</code>	Auflistung aller lokalen Variablen
<code>info registers</code>	Auflistung der Registerwerte
<code>info breakpoints</code>	Auflistung der Breakpoints
<code>info threads</code>	Auflistung der Threads
<code>info skip</code>	Auflistung der übersprungenen Funktionen
...	siehe <code>(gdb) info</code>

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

```
(gdb) x/<Anzahl><Format> <Ausdruck>
```

Werte für <Format>

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

Bestimmen des Typs eines Symbols

```
ptype <Symbol>
```

Verändern des Zielsystems: (gdb) set

Verändern eines Registers

```
(gdb) set $esp = oxdeadbeef
```

Verändern einer Variable / Speicherbereichs

```
(gdb) set numCPUs = 2    (gdb) set *((int *) 0x1013fdc) = 42
```



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

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

Die bittere Wahrheit...

Ihr werdet debuggen müssen



Die bittere Wahrheit...

Ihr werdet debuggen müssen

Anlegen einer eigenen `.gdbinit`:

Die bittere Wahrheit...

Ihr werdet debuggen müssen

Anlegen einer eigenen .gdbinit:

- Auf *mars* oder *sylabXX* im Labor:

```
cp /fs/stubs/tools/gdbinit ~/.gdbinit
```



Die bittere Wahrheit...

Ihr werdet debuggen müssen

Anlegen einer eigenen .gdbinit:

- Auf *mars* oder *sylabXX* im Labor:

```
cp /fs/stubs/tools/gdbinit ~/.gdbinit
```

- Lokal:

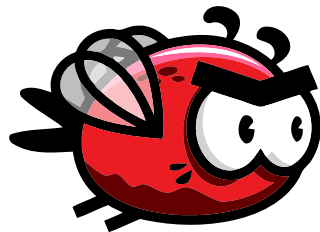
- `scp mars.cs.tu-dortmund.de:/fs/stubs/tools/gdbinit ~/.gdbinit`
- `scp ↵`
`mars.cs.tu-dortmund.de:/fs/stubs/tools/gdbinit-gef.py`
`~/gdbinit-gef.py`
- Pfad in `~/.gdbinit` anpassen



Die bittere Wahrheit...

Ihr werdet debuggen müssen

- `make qemu-gdb-noopt`
- Profit?
- `make qemu-gdb`



Fragen?



Snippet: Ausgabe von Details zur Exception im Interrupt Handler

```
1 // Optional: Print exceptions on DBG stream to support debugging
2 if (vector < Core::Interrupt::EXCEPTIONS) {
3     dout << "Exception " << dec << vector;
4     switch (vector) {
5         case 0:  dout << " (Div by 0)"; break;
6         case 6:  dout << " (Invalid Opcode)"; break;
7         case 10: dout << " (Invalid TSS)"; break;
8         case 13: dout << " (General Protection Fault)"; break;
9         case 14: dout << " (Page Fault)"; break;
10        default: break;
11    }
12    if (context->error_code != 0) {
13        dout << " [" << bin << context->error_code << "];";
14    }
15    dout << " @ " << hex << context->ip << flush;
16    dout << endl;
17    Core::die();
18 }
```

