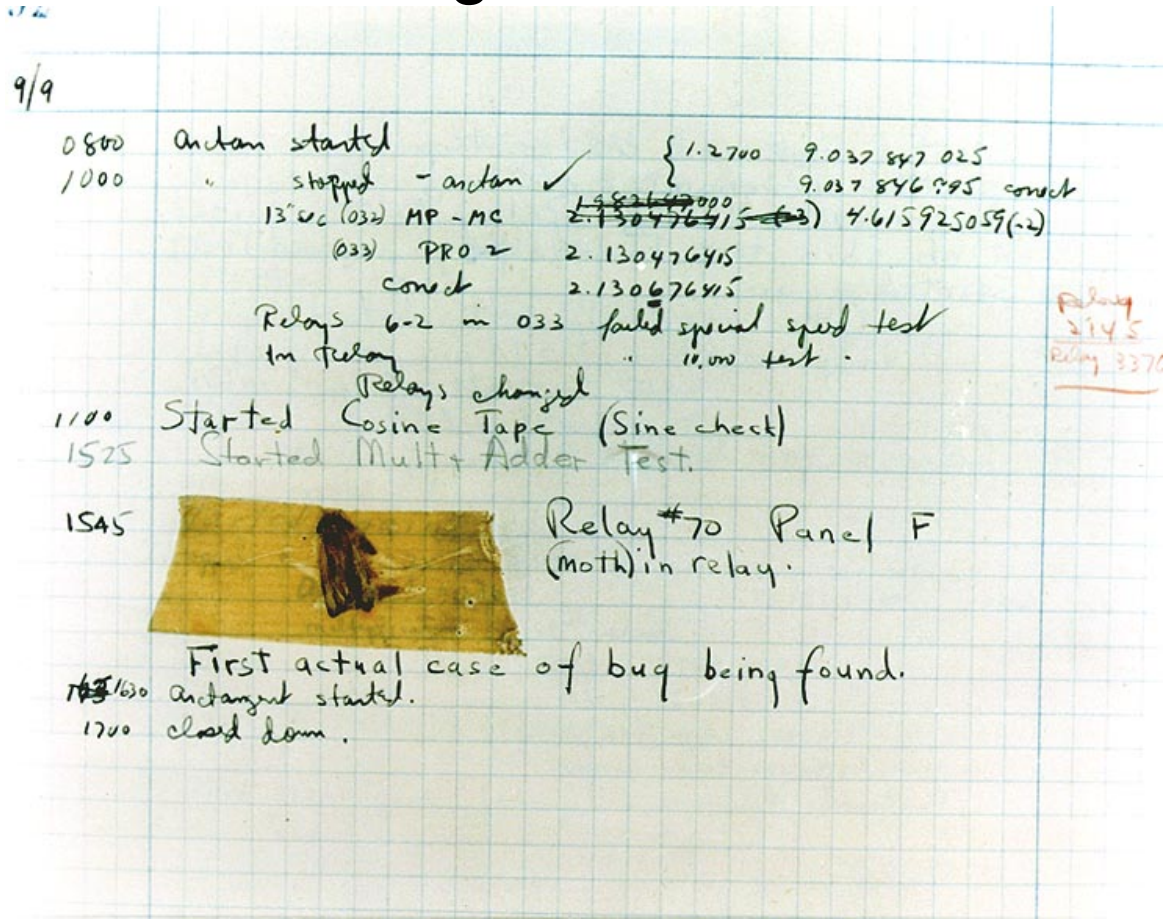
Grundkurs C++

Debugging

Berühmte Software-Bugs



1947: Fehlfunktion des Mark II Relay Calculator aufgrund einer Motte.

Berühmte Software-Fehler

- 1996: Ariane 5 muss 40 Sekunden nach dem Start gesprengt werden.
  - Umwandlung der Geschwindigkeit von double nach signed integer -> Überlauf
  - Software wurde von Ariane 4 übernommen, die diese Geschwindigkeit nicht erreichen konnte
  - Schaden: ca. 370 Millionen USD

Berühmte Software-Fehler

- 1994:
  - Fehler in der Divisionsroutine des Pentium-Prozessors
    - x = 4195835.0
    - y = 3145727.0
    - z = x - (x/y) * y
  - Bugs in CPUs sind aufgrund der Komplexität der Designs auch heute üblich.

Fehler zur Compile-Zeit

- Syntaxfehler (Vergessener Strichpunkt...)
- Inkompatible Datentypen
- Fehlende Dereferenzierung (Objekt statt Zeiger auf Objekt benutzt...)
- usw.
- Aber: Compiler können nicht alle Fehler erkennen.

Fehler zur Laufzeit

- Zugriff außerhalb der Grenzen eines Arrays.
- Dynamisch allozierten Speicher nicht wieder freigegeben (Memory-Leak).
- Dynamisch allozierten Speicher mehrfach freigegeben.
- Dereferenzierung nicht-initialisierter Zeiger.
- Signed Integer Overflow
- usw.

gcc Debug-Optionen

- Parameter -g
  - Erzeugt Debugging-Symbole in der ausführbaren Datei.
- Parameter -Wall
  - Aktiviert einen Satz von Warnungen, die i.d.R. leicht zu beheben sind.
- Parameter -W
  - Aktiviert weitere Warnungen, die nicht von -Wall eingeschlossen werden.
- Parameter -Werror
  - Warnungen werden wie Fehler behandelt und können so nicht mehr übersehen werden.
- Parameter -O2
  - Aktiviert Optimierungen (wichtig, da sonst manche Fehler nicht entdeckt werden).

gcc Debug-Symbole

```
GNU gdb (GDB) 7.0-ubuntu
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show
copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/thabigt/Desktop/valgrind/test...done.
(gdb) run
Starting program: /home/thabigt/Desktop/valgrind/test

Program received signal SIGBUS, Bus error.
0x000000000040061a in main ()
```

gcc Debug-Symbole

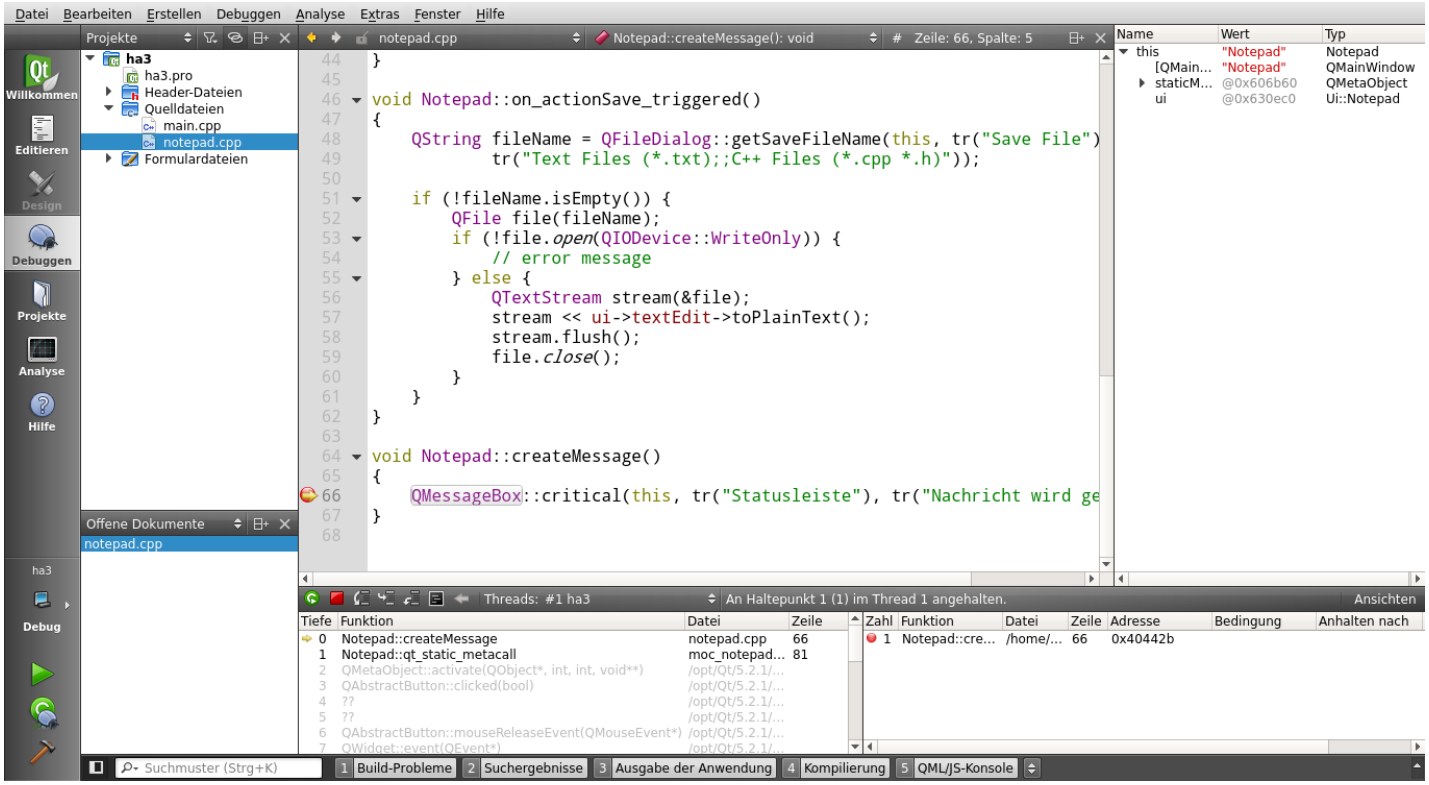
```
GNU gdb (GDB) 7.0-ubuntu
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show
copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/thabigt/Desktop/valgrind/test...done.
(gdb) run
Starting program: /home/thabigt/Desktop/valgrind/test

Program received signal SIGBUS, Bus error.
main () at test.c:23
23      *(&x + 100000) = 1;
```

gcc Debug-Symbole

- Debug-Symbole in externen Libraries:
  - Installieren der -dbg Pakete
  - Kompilieren der Libraries mit -g Option

gdb

- Benutzung von gdb am einfachsten durch Integration in Qt Creator.



gdb

- Wichtige Befehle zur Ausführung in der Kommandozeile:
  - gdb ./executable
    - **run** [Übergabeparameter]
    - **bt** ← zeigt Stack an
    - **print** Variablenname ← gibt Inhalt der Variable aus
    - **quit**

Valgrind

- **Memcheck**
  - Entdeckt unerlaubte Speicherzugriffe, Memory-Leaks, doppelte Speicherfreigaben...
- Cachegrind
  - Cache-Profiler
- **Callgrind**
  - Cachegrind mit zusätzlichen Aufrufstatistiken
- Massif
  - Heap-Profiler
- Helgrind
  - Thread-Debugger

Valgrind Memcheck

```
void f(void)
{
    int* x = new int[10];
    x[10] = 0;
}

int main(void)
{
    f();
    return 0;
}
```

valgrind --leak-check=yes ./main

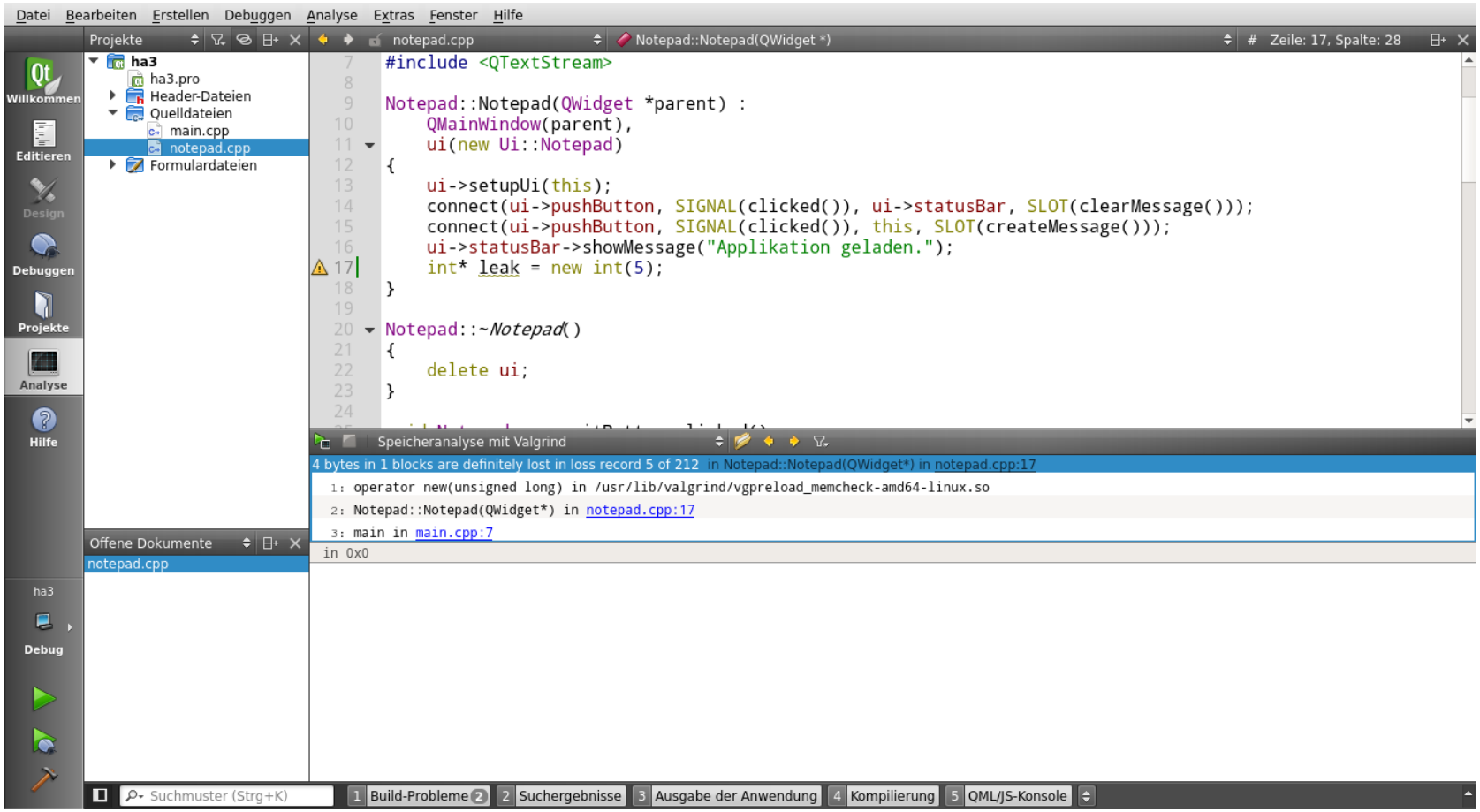
Valgrind Memcheck

```
==13211== Memcheck, a memory error detector
==13211== Copyright (C) 2002-2009, and GNU GPL'd, by Julian
Seward et al.
==13211== Using Valgrind-3.6.0.SVN-Debian and LibVEX; rerun with
-h for copyright info
==13211== Command: ./exec
==13211==
==13211== Invalid write of size 4
==13211==    at 0x40088E: f() (main.cpp:4)
==13211==    by 0x4008A8: main (main.cpp:9)
==13211==  Address 0x595e068 is 0 bytes after a block of size 40
alloc'd
==13211==    at 0x4C27939: operator new[](unsigned long)
(vg_replace_malloc.c:305)
==13211==    by 0x40088D: f() (main.cpp:3)
==13211==    by 0x4008A8: main (main.cpp:9)
```

Valgrind Memcheck

```
==13211== HEAP SUMMARY:
==13211==     in use at exit: 40 bytes in 1 blocks
==13211==   total heap usage: 1 allocs, 0 frees, 40 bytes
allocated
==13211==
==13211== 40 bytes in 1 blocks are definitely lost in loss record
1 of 1
==13211==    at 0x4C27939: operator new[](unsigned long)
(vg_replace_malloc.c:305)
==13211==    by 0x40088D: f() (main.cpp:3)
==13211==    by 0x4008A8: main (main.cpp:9)
==13211==
==13211== LEAK SUMMARY:
==13211==    definitely lost: 40 bytes in 1 blocks
==13211==    indirectly lost: 0 bytes in 0 blocks
==13211==      possibly lost: 0 bytes in 0 blocks
==13211==    still reachable: 0 bytes in 0 blocks
==13211==         suppressed: 0 bytes in 0 blocks
```

Valgrind Integration in Qt Creator



Valgrind und gdb Integration in Qt Creator

- Anleitung unter:

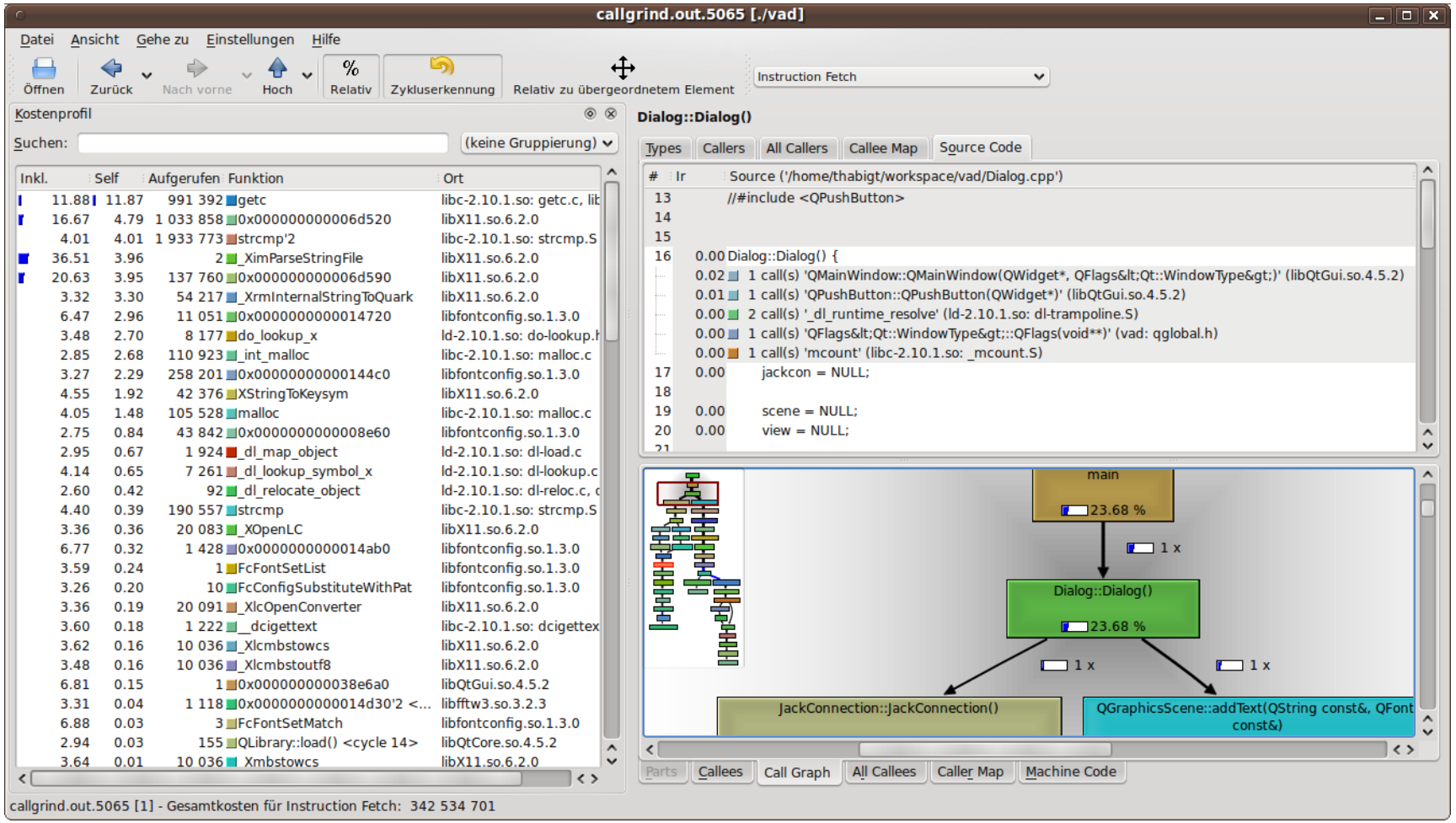
http://doc.qt.io/qtcreator/creator-debugging.html

Valgrind Callgrind

```
valgrind --tool=callgrind ./main
```

- Erzeugt Datei callgrind.out.[Prozess-ID]
- Kann mit kcachegrind visualisiert werden.

Valgrind Callgrind



Valgrind - Probleme

- Programme laufen bis zu 100-mal langsamer ab.
- Das kann besonders bei echtzeitfähiger Software zu neuen Problemen führen.
- Überwältigende Zahl von Ausgaben vor allem durch externe Bibliotheken. Diese können durch sog. Suppression-Dateien unterdrückt werden.