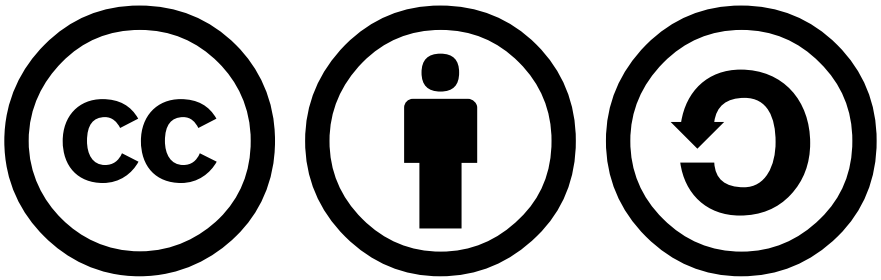
"C - Grundlagen und Konzepte"

Debugging

Marcel Hellwig
1hellwig@informatik.uni-hamburg.de

Universität Hamburg

6. Juni 2013
SOSE 13



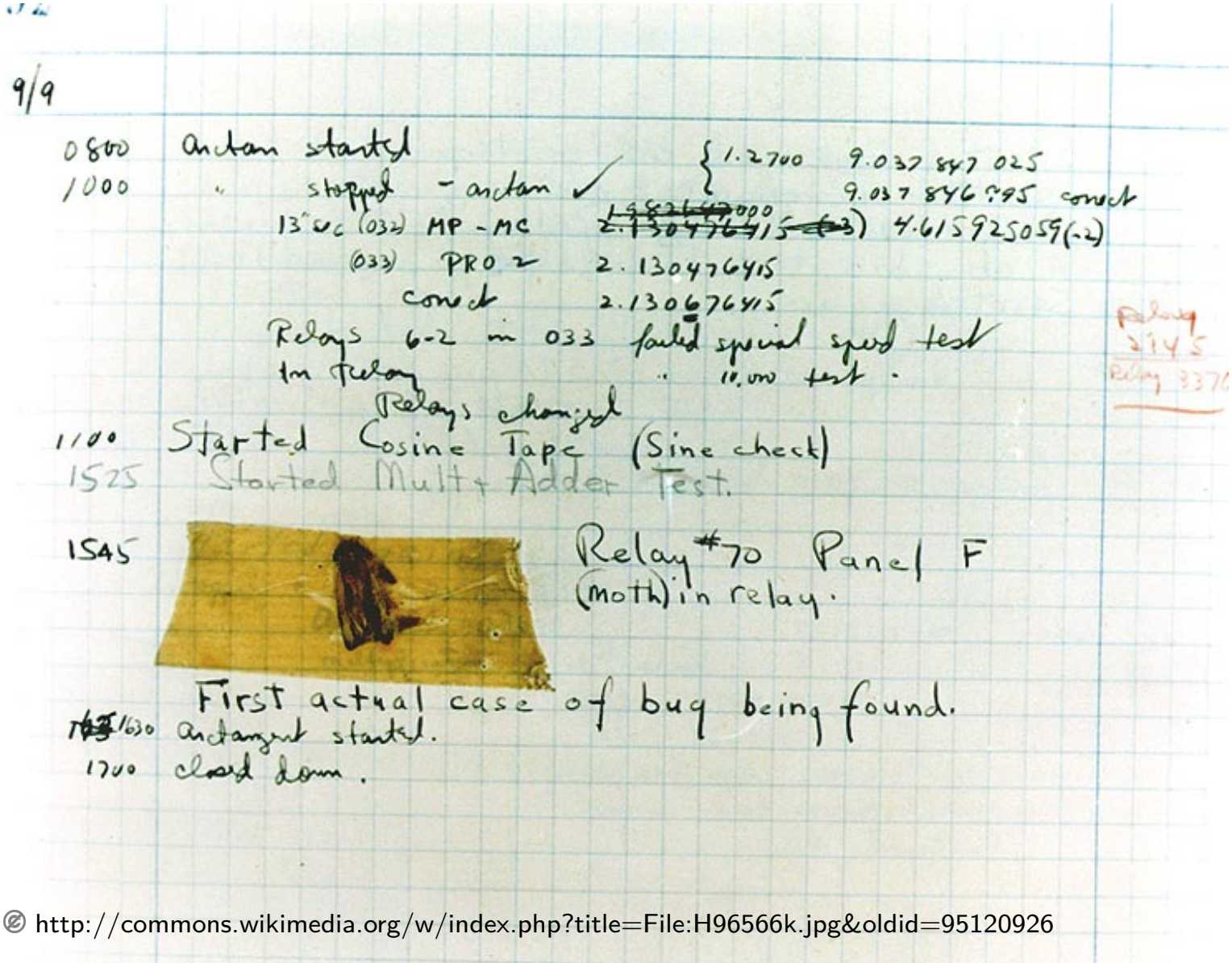
Inhaltsverzeichnis

Was ist Debugging?

Was ist ein Bug?

- Begriff geprägt von Grace Hopper
- Bezeichnet einen Fehler im Computersystemen
- Durch eine Motte in einem Relais entstand der berühmte Satz...

Was ist Debugging?



http://commons.wikimedia.org/w/index.php?title=File:H96566k.jpg&oldid=95120926

Was ist Debugging?

Was ist ein Debugging?

- Aufspüren von Fehlern
- Beseitigen von Fehlern

Zweck von Debugging

Wozu Debugging?

- Auffinden von Programmfehler
- Verfolgung des Programmablaufes
- Nicht erklärbares Verhalten analysieren
- „Interaktion“ mit dem Programm

Definition

Arten:

- manuelles Debugging
- logisches Debugging
- Programmgestütztes Debugging

Begriffe

- Debugger - Tool, das hilft, ein anderes Programm zu analysieren
- Breakpoint - Haltepunkt im Programm, an dem der Debugger anhält
- Next - Führt den Code bis zur nächsten Zeile aus
- Step - Führt den nächsten Befehl aus
- Instruction - Ein Assembler Befehl

Arten von Bugs

Hardware Bugs:

- USB 3.0 Bug bei Intel 4000-Chipsätzen
- TLB Bug bei Amd Phenom Prozessoren
- Sata Bug bei Intel Sandy Bridge
- Der „Ring of Death “bei der XBOX 360
- FPU Bug bei Intel P5 Chips

Arten von Bugs

Software Bugs:

Arten von Bugs

Software Bugs:

- Falscher Makro Wert von „TWO “in der GLIBC

Glibc

```
--- a/sysdeps/ieee754/dbl-64/mpa2.h
+++ b/sysdeps/ieee754/dbl-64/mpa2.h
@@ -36,7 +36,7 @@
 #define  ZERO       0.0             /* 0      */
 #define  ONE        1.0             /* 1      */
 #define  MONE       -1.0            /* -1     */
-#define  TWO         -2.0            /* -2     */
+#define  TWO         2.0            /*  2     */
 #define  TWO5       0x1.0p5         /* 2\^5   */
 #define  TWO10      0x1.0p10        /* 2\^10  */
 #define  TWO18      0x1.0p18        /* 2\^18  */
```

Arten von Bugs

Software Bugs:

- Falscher Makro Wert von „TWO “in der GLIBC
- Apple Programme versagen bei File:///

Apple

Bug im „Data-Detector “führt zu Fehler bei Eingabe von:
`File:///`
Grund:

- Am Anfang wird auf file://(+/) geprüft (case-insensitiv)
- Später wird vorausgesetzt, dass das f klein geschrieben wurde
- Exception wird geworfen
- $\Rightarrow$ Abbruch des Programms

Arten von Bugs

Software Bugs:

- Falscher Makro Wert von „TWO “in der GLIBC
- Apple Programme versagen bei File:///
- CSRF bei D-Link Router

D-Link

```
curl --data "cmd=cat /var/passwd"
    http://<Target IP>/command.php

=> "admin" "THESECRETPASS" "0"
```

Mehrere Fehler kommen zusammen;

- Passwort nicht verschlüsselt
- Keine Authentifizierung

Durch den Request (s.o.) können beliebige Befehle abgesetzt werden, wie:

- ausschalten des WLAN
- sniffen des Traffics
- starten eines telnetd mit root Rechten (!)

Arten von Bugs

Software Bugs:

- Falscher Makro Wert von „TWO “in der GLIBC
- Apple Programme versagen bei File:///
- CSRF bei D-Link Router
- Fremdzugriff auf Vaillant Heizung per Internet

Sicherheitslücken durch versch. Angriffsvektoren:

- Java-Applet
- Abgelaufenes Zertifikat
- Auslesen von Passwörtern möglich (Benutzer, Techniker, Entwickler)
- Alle Heizungen an eigenes DynDNS angeschlossen und erratbar

Vaillant hat Besserung versprochen.

Arten von Bugs

Software Bugs:

- Falscher Makro Wert von „TWO “in der GLIBC
- Apple Programme versagen bei File:///
- CSRF bei D-Link Router
- Fremdzugriff auf Vaillant Heizung per Internet
- Zufallsgenerator bei Debian

Debian OpenSSH

- Fehler in OpenSSH (Debian-)Implementation führte zu einem stark eingeschränktem Lösungsraum von 2^{16} SSH Keys.
- Key in unter 2 Stunden berechnet
- Fehler blieb 2 (!) Jahre unentdeckt (2006-2008)
- Besonders kritisch für SSH-Keys und SSL-Verschlüsselungen

Betroffen waren u.a.

- whitehouse.gov
- ATI-Treiber Download
- Akamai
- Elster

Was heißt das nun konkret?
Ein Angriffsszenario:

- Besorgen des X.509 Zertifikat (public)
- Berechnen des Privaten Keys (max. 2 Stunden)
- Einklinken in den TCP Traffic und erfolgreich als akamai (et al) ausgeben
- Sicheren Traffic mitschneiden, Steuergeheimnisse erfahren und mehr

Debian OpenSSH

- SSL-Zertifkate besitzen Verfallsdatum von einem Jahr
- Allerdings nicht darauf beschränkt
- Es gibt eine Revoking-Idee (wie bei PGP)
- Funktioniert nicht
- $\Rightarrow$ Akamei ein $^3/_4$ Jahr „nicht sicher“
- Auch das Zertifikat einer deutschen Bank war betroffen

Debian OpenSSH

- SSL-Zertifkate besitzen Verfallsdatum von einem Jahr
- Allerdings nicht darauf beschränkt
- Es gibt eine Revoking-Idee (wie bei PGP)
- Funktioniert nicht
- $\Rightarrow$ Akamei ein $^3/_4$ Jahr „nicht sicher“
- Auch das Zertifikat einer deutschen Bank war betroffen
- Es war noch drei Jahre gültig ☹

Manuelles Debugging

- Einsetzen von Makros/o.ä. um sich Zwischenergebnisse anzeigen zu lassen
- Nachverfolgen von Verzweigungen im Programm
- Verstehen warum das Programm sich so verhält

Manuelles Debugging

```
bool sample(int eval, int evalCompare,
            int id, int idCompare)
{
  bool pred = true;
  pred = pred && (eval < 0 && eval >= evalCompare);
  pred = pred && (id   < 0 && id   <= idCompare);
  /* hier ist zuerst gefragt ob eval gültig ist,
   * d.h. größer null und ob eval größer
   * gleich evalCompare ist. Wenn eval nicht
   * gültig ist, soll der Wert ignoriert
   * werden, abert nicht false. Bei Id identisch bis
   * auf den Vergleich mit dem Compare Wert.
   */
  return pred;
}
```

Manuelles Debugging

```
bool sample(int eval, int evalCompare,
            int id, int idCompare)
            //(-1, 5, 2, 4);
{
  bool pred = true;
  VLOG(d, pred);  //pred = true
  pred = pred && (eval < 0 && eval >= evalCompare);
  VLOG(d, pred); //pred = false
  pred = pred && (id   < 0 && id   <= idCompare);
  VLOG(d, pred); //pred = false
  return pred;
}
```

Manuelles Debugging

```
bool sample(int eval, int evalCompare,
            int id, int idCompare)
            //(-1, 5, 2, 4);
{
  bool pred = true;
  VLOG(d, pred);                  //pred = true
  pred = pred && (eval < 0 && eval >= evalCompare);
  VLOG(d, eval < 0);              //eval < 0 = true
  VLOG(d, eval >= evalCompare);
                                  // eval >= evalCompare = false
  pred = pred && (id   < 0 && id   <= idCompare);
  VLOG(d, id < 0);                //id < 0 = false
  VLOG(d, id >= idCompare); //id >= idCompare = false
  return pred;
}
```

Manuelles Debugging

```
bool sample(int eval, int evalCompare,
            int id, int idCompare)
{
  bool pred = true;
  pred = pred && (eval < 0 || eval >= evalCompare);
  pred = pred && (id   < 0 || id   <= idCompare);
  return pred;
}
```

Manuelles Debugging

Manuelles Debugging ist sinnvoll, . . .

- wenn kein „echter“Debugger zur Verfügung steht
- man keinen Debugger nutzen möchte
- Debugger nicht nutzbar ist

Logisches Debugging

Logisches Debuggen ...

- Analysieren des Quelltextes
- Logisches durchgehen des Programms
- Fehler finden durch Überlegungen

Logisches Debugging

Logisches Debuggen ...

- Analysieren des Quelltextes
- Logisches durchgehen des Programms
- Fehler finden durch Überlegungen

- Starke Ähnlichkeit mit manuellem Debugging
- Kein kompilieren des Programms notwendig
- Somit auch nicht kompilierbarer Code „Debugbar “

Logisches Debugging

Durch logisches Debugging werden Fehler gefunden wie:

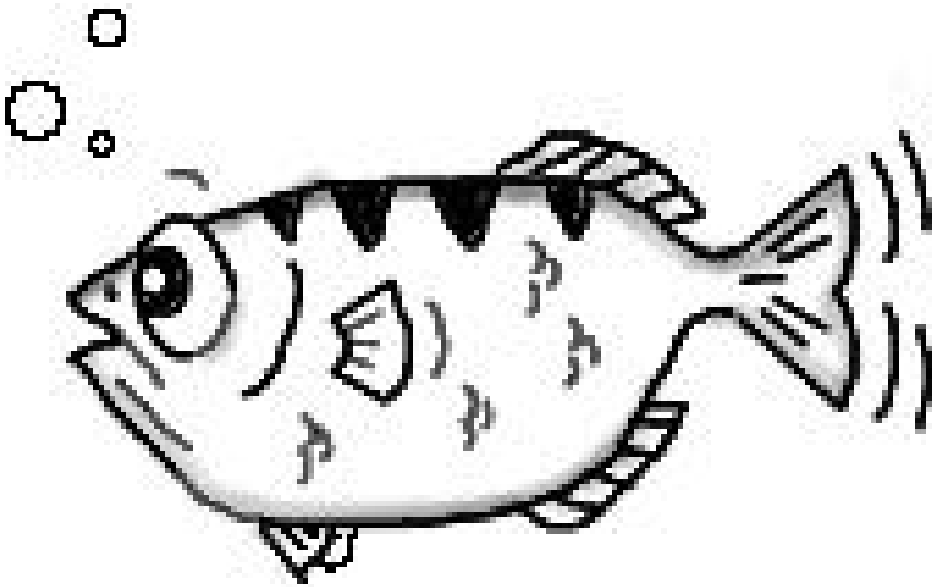
- if(a = 1)
- *p++
- if(i < 0 && i > 1)

Programunterstütztes Debuggen

Debuggen des Programms mit Hilfe eines Programms
Typische Programme sind:

- GDB
- DDD
- debug.exe
- HiTOP
- M$ Visual Studio
- IDA

gdb

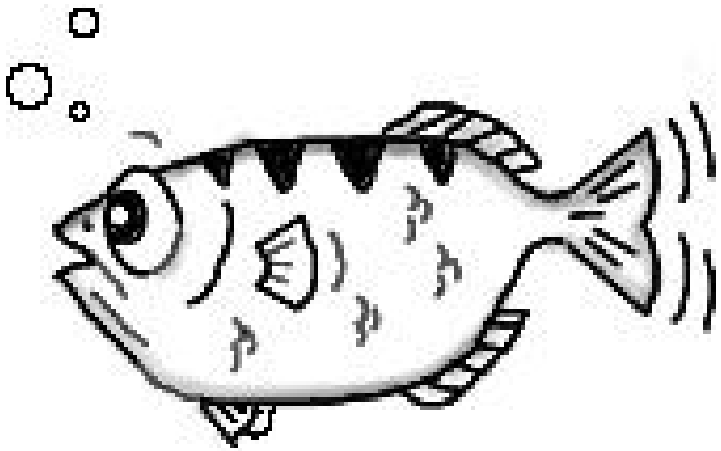


http://www.gnu.org/software/gdb/images/archer.jpg

gdb

?

gdb



„For a fish, the archer fish is known to shoot down bugs from low hanging plants by spitting water at them.“

http://www.gnu.org/software/gdb/mascot/

gdb

GDB - Gnu Debugger
Unterstützt neben C auch: C++, OC, FORTRAN, Java, Pascal, Ada, D, Go
Möglichkeiten:

- Breakpoints setzen
- Quelltext anzeigen (soweit vorhanden/mitkompiliert)
- Stacktrace ansehen
- Variablen anzeigen/verändern
- Statements auswerten

GDB wird oft als Backend verwendet in IDEs.
Valgrind nutzt GDB

gdb

```
GNU gdb (GDB) Fedora (7.6-24.fc19)
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) help
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
```

Selfmade

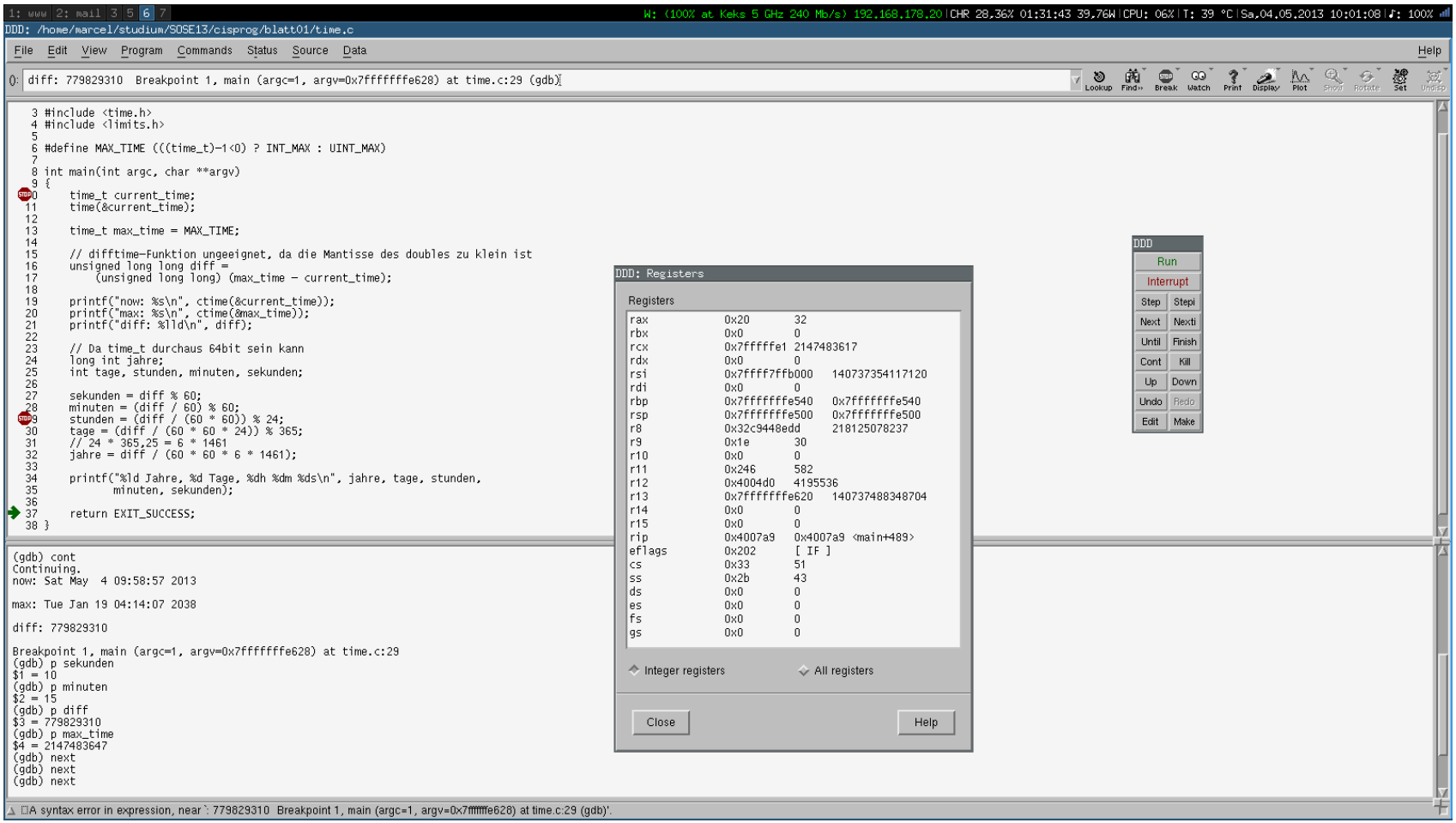
ddd

DDD - Data Display Debugger
Kein Debugger an sich, sondern ein Frontend für

- GDB - GNU Debugger
- DBX - Debugger von 1981 (Oracle)
- XDB - So unbekannt, dass ihn keiner kennt
- Pydb - Python Debugger
- ...

Durch Plugins gut erweiterbar für allerlei Sprachen (kompilierte/Skripte)

ddd



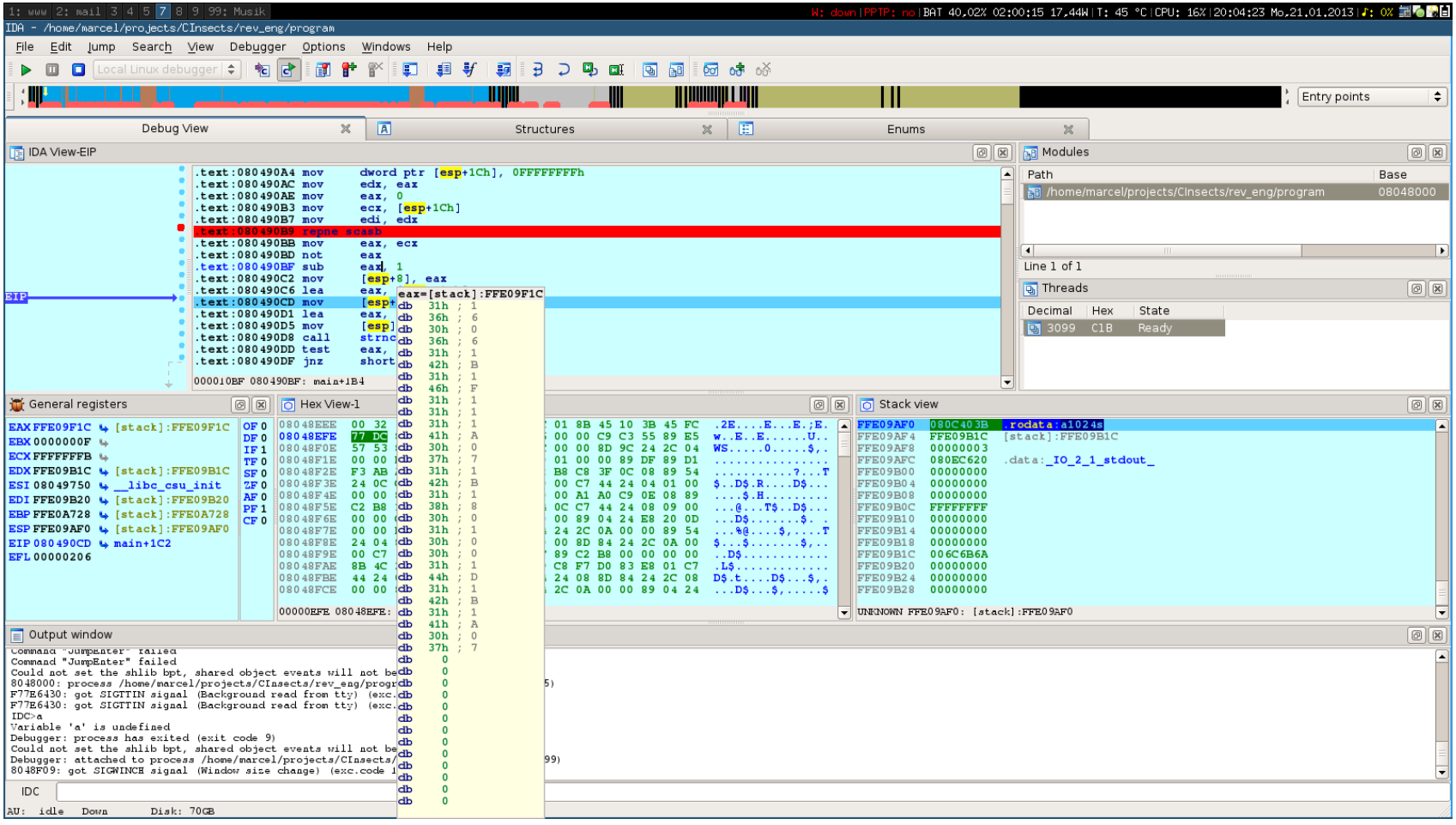
Selfmade

ida

The Interactive Disassembler

- Kommerzielles Produkt
- Disassembler mit (guter) GUI
- Auch als Debugger nutzbar
- erweiterbar durch Python Plugins
- Visualisierung des Programmflusses

ida



Selfmade

Einzelnachweise

5 - http://commons.wikimedia.org/w/index.php?title=File:H96566k.jpg&oldid=95120926
10 - http://www.golem.de/print.php?a=99552
10 - http://www.dailytech.com/Understanding++AMDs+TLB+Processor+Bug/article9915.htm
10 - http://techreport.com/news/20326/intel-finds-flaw-in-sandy-bridge-chipsets-halts-shipments
10 - http://www.heise.de/newsticker/meldung/Microsoft-gesteht-Probleme-mit-Hardwarefehlern-der-Xbox-36html
10 - http://en.wikipedia.org/wiki/Pentium_FDIV_bug

Einzelnachweise

11 - http://www.heise.de/newsticker/meldung/
Schwache-Krypto-Schluessel-unter-Debian-Ubuntu-und-Co-2073
html
11 - http://www.bhkw-infothek.de/nachrichten/18555/
2013-04-15-kritische-sicherheitslucke-ermoglicht-fremdzugr
11 - http://www.s3cur1ty.de/m1adv2013-003
11 - http://openradar.appspot.com/13128709
11 - http:
//sourceware.org/ml/glibc-cvs/2013-q1/msg00115.html
32 - http://www.gnu.org/software/gdb/images/archer.jpg
21 - http://blog.fefe.de/?ts=b6c9ec7e

Quellen

https://en.wikipedia.org/wiki/Debugging
https://en.wikipedia.org/wiki/Grace_Hopper

Any Questions, anyone?

Zusammenfassung

Arten von Bugs

- Hardware
- Software

Arten von Debugging

- Manuell
- Logisch
- Programmgesteuert