Debugging Praktikum "C-Programmierung"

Eugen Betke, Nathanael Hübbe, Michael Kuhn, Jakob Lüttgau, Jannek Squar

> Wissenschaftliches Rechnen Fachbereich Informatik Universität Hamburg

> > 2018-11-05



- 1 Motivation
- 2 Debugging
 - Beispiel
 - Verwendung von GDB
- 3 Queller

Das große Krabbeln

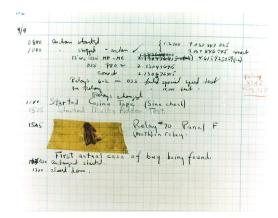


Abbildung: Dokumentation eines "echten" Computer-Bugs [Com47]

Bugs

- Debugging im HPC-Umfeld potentiell schwierig:
 - Komplexes Zusammenspiel: Rechenknoten, Netzwerk, Speicher, Software-Stack, etc.
 - Nichtdeterminismus
- Bug-Sorten:
 - Syntax-Error
 - Runtime-Error
- Problematisch, wenn Programm nicht abstürzt
- Bugfreiheit ist unentscheidbar

Was gibt es für Bugs?

Syntax-Error

- Fehler während Compile-Zeit
- Fehlerhafte Syntax
- Compiler kann Grammatik nicht ableiten
- Relativ leicht behebbar
- ⇒ Binary kann nicht gebaut werden!

Runtime-Error

- Fehler während Laufzeit
- Vielfältige Ursachen:
 - Semantik-/Spezifikation-/Logik-Fehler (Deadlocks, Race Conditions, ...)
 - Speicherzugriffsfehler
 - Hardware-Defekt
 - **.**.
- \blacksquare Symptom \neq Ursache
- ⇒ Binary kann gebaut werden!

- 1 Motivation
- 2 Debugging
 - Beispiel
 - Verwendung von GDB
- 3 Queller

Beispiel

recursion.c

```
int testFunc(int i)
       if(i > 0)
            return testFunc(i-1)+i;
6
       else if(i == 0)
8
            return 0;
10
11
```

Jannek Squar Debugging 7 / 15

Erste Maßnahmen

- Mit -wall kompilieren
- printf an strategischen Punkten einfügen
- Unterstützung durch Tools:
 - Deterministischer, kleinschrittiger Code-Durchlauf
 - Valgrind
 - GDB (GNU Debugger)
 - Open Source
 - Command line Debugger
 - Backend für andere Debugger

Erste Maßnahmen - Beobachtungen

Programm stürzt nicht ab

- Ausgabe verändert sich mit unterschiedlichen Optimierungsleveln
- printf beeinträchtigt Übersicht
- -wall gibt wichtigen Hinweis

Erste Schritte mit GDB

- Kompilieren mit -og -g oder -og -ggdb¹
- Interessante GDB-Flags:
 - Übergabe von Parametern: --args ./app arg1 ... argN
 - Ordner mit Sourcen: --directory=DIR
 - Weitere Literatur: gdb --help und man gdb
- Neukompilierung während Debugging möglich

¹Alternativ -00 -g verwenden

How to GDB (I)

Ausführung

- **r**un
- **c**ontinue
- next
- **s**tep
- finish
- **q**uit/kill

Ausgabe

- **p**rint [/f] [var]
- display [/f] var
- undisplay n
- enable/disable/info display

How to GDB (II)

Break-/Watch-Points

- break [file:]line | [file:]func [if condition]
- **tb**reak ...
- watch var
- info b
- disable/enable/delete n
- clear [[file:]line|[file:]func]
- delete [n]

Programm-Stack

- **b**ack**t**race [n]
- frame [n]
- up/down [n]

Weiteres

- An laufendes Programm anhängen:
 - gdb ./runningProcess PID
 - attach PID

```
#include <stdio.h>
    #include <unistd.h>
    int main()
6
         for(int i = 0; i < 10; i++)
8
             printf("Aktueller Wert für i: %d\n". i):
             sleep(1):
10
             if(i == 9)
11
12
                  i = 0:
13
14
15
         return 0:
16
```

- set var=value
- print *pointer
- print *dynArray@length
- info threads
- thread n
- GDB-Frontends [Kal18]
- Online-Variante [ss18]
- GDB Quick Reference [Pes00]
- Blog: How Does a C Debugger Work? [Pou14]

- 1 Motivation
- 2 Debugging
 - Beispiel
 - Verwendung von GDB
- 3 Quellen



Quellen I

- [Com47] Wikimedia Commons. The First Computer Bug, 1947.
 https://commons.wikimedia.org/wiki/File:H96566k.jpg.
- [Kal18] Marina Kalashina. Gdb front ends and other tools, 2018. https://sourceware.org/gdb/wiki/GDB%20Front%20Ends.
- [Pes00] Roland H. Pesch. GDB QUICK REFERENCE, 2000. http://users.ece.utexas.edu/~adnan/gdb-refcard.pdf.
- [Pou14] Kevin Pouget. How does a c debugger work?, 2014. https://blog.0x972.info/?d=2014/11/13/10/40/50-how-does-a-debugger-work.
- [ss18] Mritunjay singh sengar. Onlinegdb online compiler and debugger for c/c++, 2018. https://www.onlinegdb.com.