Debugging und Speicherfehler

Seminar Effiziente Programmierung

Kadir Duman

01.12.2016

Inhalt

Debugging
- Allgemein
- Was ist ein Bug?
- Was ist Debugging?

Speicherfehler
- Allgemein
- Unterschiedliche Typen von Speicherfehlern:
  - Gültigkeitsbereich, Null-Pointer, Speicherlecks etc.
- Speicherfehler finden (AddressSan, LeakSan, MemorySan)

Effizientes Fehlerfinden
- Suchraum einschränken
- Delta Debugging
- Continuous Testing

GDB/Valgrind

Debugging – Allgemein

- Kaum fehlerfreie Programme
  - Methoden notwendig um effizient Fehler zu erkennen

- Fehler können entstehen durch:
  - Hinzufügen von Funktionen
  - Ersetzen des Quellcodes, durch ein kompakteren
  - Veränderungen in einer benutzten Library

Debugging – Was ist ein Bug?

- Ein Programmfehler zur Laufzeit (Laufzeitfehler)
- Fehlerhaftes Verhalten/ Ausführung wird angehalten
- Ursache oft nicht offensichtlich

Debugging – Was ist Debugging?

- Fehlerlokalisierung
  - Zeitaufwendiger Teil
- Bugfixing
  - Komplizierter Teil
- Früh genug debuggen
  - Programm in den späteren Phasen sehr komplex

Speicherfehler – Allgemein

- Keine automatische Speicherverwaltung in C
  - Häufige Fehler in der Speicherverwaltung

Welche Arten von Speicherfehler gibt es?

Speicherfehler – nicht initialisierte Variablen

- Variablen müssen nicht initialisiert werden
  - Lokale Variablen bekommen einen beliebigen Wert
- Verhindern durch Warnungen des Compilers

Speicherfehler – Gültigkeitsbereich

- Verschatten
  - Verfügbarkeit von Variablen kann eingeschränkt werden
- Unabsichtliches Verschatten
  - Falsche Variablen werden verändert
- Verhindern in dem man verständliche Bezeichner wählt
- Compilerwarnungen beachten (GCC)

Speicherfehler – Null-Pointer

- Pointer haben den Wert des Null-Pointers
- Zugriff auf Null-Pointer
  - Ausführung des Programms wird abgebrochen (Seitenfehler)

Speicherfehler – Speicherlecks

- Nach Allocation und Benutzung, Speicher wieder freigeben
  - Bleibt sonst reserviert
- Je öfter dieser Fehler passiert, desto weniger Speicher zur Verfügung
- Vorsicht bei lang laufendem Programmen!

Speicherfehler – Zugriff nach Freigabe

- Gefahr: Zugriff nach Freigabe
  - Denn: Speicher könnte schon für etwas anderes in Verwendung sein
  - Muss nicht unbedingt abstürzen
    - » Speicher kann noch dem Programm „gehören“

Speicherfehler – Doppelte Freigabe

- Einen schon freigegebenen Speicher, erneut freigeben
  - Resultat nicht definiert

    →unerwünschtes Verhalten

Speicherfehler finden – AddressSanitizer

- Compiler kann Speicherfehler finden
- Sanitizer suchen nach Speicherfehler
  - » Statisch oder zur Laufzeit

- AddressSanitizer findet:
  - „Use-after-free“
  - „Double-free“
  - „Memory-leaks“ [2]

Speicherfehler finden – Leak- und MemorySanitizer

- LeakSanitizer:
  - Kann zur Laufzeit Speicherlecks finden [1]

- MemorySanitizer:
  - Kann Fehler wie z.b. „nicht-initialisierte-Variablen“ finden [3]

Effizientes Fehlerfinden – Suchraum einschränken

- Suchraum einschränken und Zeit sparen
- Zwei Versionen mit funktionalen Tests
  - Ältere Version: Tests erfolgreich
  - Neue Version: Tests nicht (alle) erfolgreich
- Abweichungen(„Delta“) kann bestimmt werden

Effizientes Fehlerfinden – Delta Debugging

- Automatisches Verfahren zum Auffinden der Fehler verursachenden Änderungen
- Regionen ausschließen, die nicht für den Fehler verantwortlich sind

Effizientes Fehlerfinden – Definition

Interferenz:

Fehler werden nur in Kombination von Änderungen verursacht

Effizientes Fehlerfinden – Delta Debugging

Schritt	C	Konfiguration	Test
1	C1	1 2 3 4	+
2	C2	. . . . 5 6 7 8	+

Effizientes Fehlerfinden – Delta Debugging

Schritt	C	Konfiguration▫	Test
1	C1	1 2 3 4	+
2	C2	. . . . 5 6 7 8	+
3	C3	1 2 . . 5 6 7 8	+
4	C4	. . 3 4 5 6 7 8	-

Effizientes Fehlerfinden – Delta Debugging

Schritt	C	Konfiguration	Test
1	C1	1 2 3 4	+
2	C2	. . . . 5 6 7 8	+
3	C3	1 2 . . 5 6 7 8	+
4	C4	. . 3 4 5 6 7 8	-
5	C5	. . . 4 5 6 7 8	+
6	C6	. . 3 . 5 6 7 8	-

Effizientes Fehlerfinden – Delta Debugging

Schritt	C	Konfiguration▫	Test
1	C1	1 2 3 4	+
2	C2	. . . . 5 6 7 8	+
3	C3	1 2 . . 5 6 7 8	+
4	C4	. . 3 4 5 6 7 8	-
5	C5	. . . 4 5 6 7 8	+
6	C6	. . 3 . 5 6 7 8	-
7	C7	1 2 3 4 . . 7 8	+
8	C8	1 2 3 4 5 6 . .	-

Effizientes Fehlerfinden – Delta Debugging

Schritt	C	Konfiguration	Test
1	C1	1 2 3 4	+
2	C2	. . . . 5 6 7 8	+
3	C3	1 2 . . 5 6 7 8	+
4	C4	. . 3 4 5 6 7 8	-
5	C5	. . . 4 5 6 7 8	+
6	C6	. . 3 . 5 6 7 8	-
7	C7	1 2 3 4 . . 7 8	+
8	C8	1 2 3 4 5 6 . .	-
9	C9	1 2 3 4 5 . . .	+
10	C10	1 2 3 4 . 6 . .	-

Effizientes Fehlerfinden – Delta Debugging

Schritt	C	Konfiguration	Test
1	C1	1 2 3 4	+
2	C2	. . . . 5 6 7 8	+
3	C3	1 2 . . 5 6 7 8	+
4	C4	. . 3 4 5 6 7 8	-
5	C5	. . . 4 5 6 7 8	+
6	C6	. . 3 . 5 6 7 8	-
7	C7	1 2 3 4 . . 7 8	+
8	C8	1 2 3 4 5 6 . .	-
9	C9	1 2 3 4 5 . . .	+
10	C10	1 2 3 4 . 6 . .	-
Ergebnis		. . 3 . . 6 . .	

Zeitnahes Fehlerfinden – Continuous Testing

- Automatische Kompilierung im Hintergrund (z.b. Eclipse)
  - Vom Compiler erkennbare Fehler, sofort sichtbar
- Tests können automatisch priorisiert werden
  - Most-Recent-Failure-First-Ansatz
- Stetige Durchführung der Tests

GNU Debugger

- Das innere des Programms zur Laufzeit
  - Sich ändernde Werte
  - Momentan ausgeführte Anweisung
- Bietet Breakpoints zur Ablaufverfolgung
- Ermöglicht ein Eingreifen in die Ausführung

Valgrind

- Eine Werkzeugsammlung
  - Besteht aus verschiedenen Tools
- Schwer zu lokalisierende Fehler können gefunden werden

Literatur

[1] LeakSanitizer, http://clang.llvm.org/docs/LeakSanitizer.html, 24-11-2016

[2] AddressSanitizer, http://clang.llvm.org/docs/AddressSanitizer.html, 24-11-2016

[3] MemorySanitizer, http://clang.llvm.org/docs/MemorySanitizer.html, 24-11-2016

[4] Speicherfehler, http://llvm.org/devmtg/2011-11/Serebryany_FindingRacesMemoryErrors.pdf, 24-11-2016

[5] Debugging, www2.in.tum.de/hp/file?fid=1239, 24-11-2016

[6] Fehlerlokalisierung, https://www.fernuni-hagen.de/ps/arbeiten/master_bertschler.shtml, 24-11-2016

[7] Bugs und Fehlersuche, https://codingtutor.de/debugging/, 24-11-2016

[8] Debugging with GDB, https://www.eecs.umich.edu/courses/eecs373/readings/Debugger.pdf, 24-11-2016

[9] Valgrind, http://valgrind.org/docs/memcheck2005.pdf, 24-11-2016

[10] Speicherverwaltung, https://de.wikibooks.org/wiki/C-Programmierung:_Speicherverwaltung, 24-11-2016