

Debugging

Jan Hendrik Dithmar

02.02.2006

Testfälle

Ein Test kann nur die Anwesenheit von Fehlern nachweisen, nicht jedoch deren Abwesenheit.

Dijkstra (1972)

Problemverwaltung

Bug List - Mozilla Firefox

mozilla.org

Bug List

Sun Jan 29 2006 06:55:00 PST

11 bugs found.

ID	Sev	Pri	Plt	Assignee	Status	Resolution	Summary
325116	maj	--	PC	msscott@mozilla.org	UNCO		-setDefaultMail option no longer works in Thunderbird 1.5
325113	nor	--	Mac	nobody@mozilla.org	UNCO		Repair themes after v1.6 installation OS X.3.9 when rever...
325115	nor	--	PC	nobody@mozilla.org	UNCO		Cairo regression - probably z-index
325118	maj	--	PC	nobody@mozilla.org	UNCO		Bad Example: have no graphics cannot open any pictures on...
325120	maj	--	PC	nobody@mozilla.org	UNCO		the bookmark option DOESN'T WORK AT ALL after fiddling wi...
325121	nor	--	PC	nobody@mozilla.org	UNCO		XML parser says "No element found" on valid RSS feed
325125	nor	--	PC	nobody@mozilla.org	UNCO		firefox does not remember filled forms
325126	maj	--	PC	nobody@mozilla.org	UNCO		Search engide dosn't open
325117	maj	--	All	benjamin@smedbergs.us	ASSI		libgtkmozembed no longer exists but still referenced in p...
325122	nor	--	PC	nobody@mozilla.org	RESO	WORK	About: Gives XML Parsing Error
325124	nor	--	PC	nobody@mozilla.org	RESO	DUPL	Some menus occasionally lose the ability to definitively ...

11 bugs found.

Long Format [CSV](#) [RSS](#) [iCalendar](#) [Change Columns](#) [Edit Search](#) Remember search as

Actions: [Home](#) | [New](#) | [Search](#) | bug # [Find](#) | [Reports](#) | [Requests](#) | [New Account](#) | [Log In](#)

Fertig bugzilla.mozilla.org Print 1,781s AdBlock

Quelle: <http://bugzilla.mozilla.org>

Debugging – ein Überblick

1. Ein Fehler tritt auf
2. Hypothese aufstellen
3. Eingrenzen der Fehlerursache

Das systematische Eingrenzen von Fehlerursachen nennt man auch die *wissenschaftliche Methode* der Fehlersuche.

Die wissenschaftliche Methode

1. Hypothese aufstellen
2. Hypothese prüfen
 - Der Test schlägt fehl: Hypothese bestätigt
 - Der Test ist erfolgreich: Hypothese widerlegt
 - Inkonsistenz: Test zeigt weder korrektes noch fehlerhaftes Verhalten
3. Fahre mit der neuen Hypothese fort bei Schritt 2, bis der Fehler hinreichend eingegrenzt ist.

Wie prüft man Hypothesen?

- Anreichern des Quelltextes mit Debugging-Anweisungen

```
#ifdef DEBUG
```

```
std::cout << "a=" << a << std::endl;
```

```
std::cout << "b=" << b << std::endl;
```

```
#endif
```

Typische Untersuchungstechniken

- Ausgabe des Programmzustands
z.B. in einem Debugger nach dem Halten eines Programms
- Validierung des Programmzustands
am besten mit Zusicherungen (assertions)
- Änderung des Programmzustands
z.B. Ausblenden von Programmfragmenten

Beispiel

- Programm:
Berechnung der Differenz **a-b**.
- Situation:
In meinem Quelltext ist irgendwo ein Fehler.
- Ziel:
Eingrenzen der Fehlerursache mit Hilfe der *wissenschaftlichen Methode*.

Auf Debugging-Anweisungen wird aus Platzgründen verzichtet.

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    #ifdef DEBUG
        std::cout << "a=" << a << std::endl;
        std::cout << "b=" << b << std::endl;
    #endif
    return (b-a);
}

int main() {
    std::cout << "Differenz(5,3) = " << differenz(5,3) << std::endl;

    return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (b-a);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Hypothese prüfen

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (b-a);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (b-a);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Fehlerhypothese bestätigen!

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (a-b);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (a-b);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (a-b);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Hypothese prüfen

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (a-b);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (a-b);
}

int main() {
    int a = 0;
    int b = 0;

    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Hypothese prüfen

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
    return (a-b);
}

int main() {
    int a = 0;
    int b = 0;

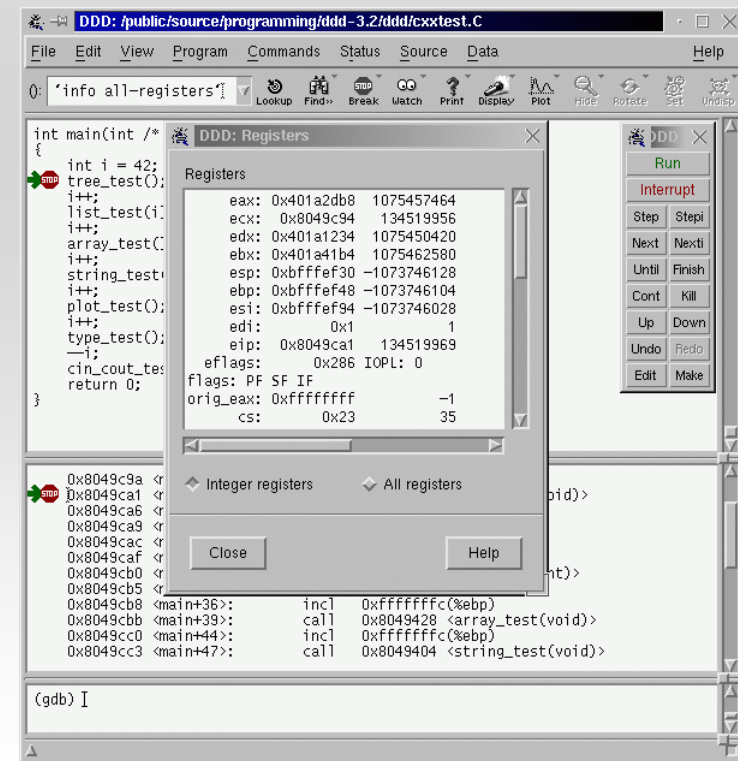
    std::cout << "a: "; std::cin >> a;
    std::cout << "b:"; std::cin >> b;

    std::cout << "a - b = " << (differenz(a,b)) << std::endl;

    return 0;
}
```

Fehler erfolgreich beseitigt!

Quelle: <http://www.gnu.org/software/ddd/>



Arbeiten mit einem Debugger

Ein Debugger ist kein Ersatz für gutes Nachdenken. In einigen Fällen ist aber Nachdenken auch kein Ersatz für einen guten Debugger. Die effektivste Kombination ist gutes Nachdenken und ein guter Debugger.

McConnell (1993)

Debugging-Prozess

Wie kann der Debugging-Prozess effektiver gestaltet werden?

Durch:

- Testfälle und Verwendung von Assertions
- Problemverwaltung
- Debugger
- scharfes Nachdenken