Über Bugs und Crashes

Adrian Schneider

Einleitung

Software-Bugs sind unumgänglich. In diesem Handout sind kurz die wichtigsten Punkte zur Bug-Klassifizierung, Bug-Erkennung, Code- / Laufzeitanalyse und Bug-Tracking-Systemen zusammengefasst.

Bug-Klassifizierung

Bugs sind sehr unterschiedlich. Generell kann man diese nach Schweregrad und Auftreten klassifizieren.

Schweregrad:

- Schweregrad 1: Software-Absturz oder Software-Freeze.
- Schweregrad 2: Hauptfunktionalität der Software ist beschädigt.
- Schweregrad 3+: Zunehmend vernachlässigbar.

Auftreten:

- Immer: Einfach zu reproduzieren.
- System abhängig: Schwierig - Gründe sind meistens äussere Einflüsse.
- Manchmal: Schwierig - Hängt oft mit schlechter Multi-Thread-Programmierung zusammen.
- Der Heisenbug ☺ - Sehr schwierig.

Bug-Erkennung

Bugs sollten noch während dem Entwicklungsprozess entdeckt werden und den Anwender möglichst selten erreichen. Es gibt bestimmte Software-Entwicklungstechniken und Software-Analyse-Tools, welche die Anzahl Bugs signifikant reduzieren können.

- Test-Driven-Development (TDD): Jedes neue Feature beginnt mit der Implementierung von eigenen Testfällen. Oft gebrauchte Testbibliotheken sind *CppUnit*, *QTest* und *jUnit*.
- Build-Systeme: Ein Build-System kompiliert regelmässig den neusten Code und führt die oben erwähnten Testfälle aus. Falls einer der Tests fehlschlägt, werden die Entwickler automatisch informiert und der Bug kann sofort beseitigt werden.

- Source-Code-Analyse: Der Compiler selbst ist eigentlich das beste Werkzeug, um den Code auf Schwachstellen zu prüfen. Beim *gcc* müssen jedoch zuerst die entsprechenden Compiler-Flags gesetzt werden, damit die Warnungen ausgegeben werden (-Wall: aktiviert häufigste Warnungen, -Werror: Warnungen werden als Compile Fehler behandelt).
- Laufzeitanalyse: Ein Debugger, wie zum Beispiel der *gdb*, ist das am besten geeignete Tool um Software-Bugs zu untersuchen. Spezifische Werkzeuge wie *Valgrind* können benutzt werden, um beispielsweise Memory-Leaks zu finden. Falls der Code unbekannt ist und man nur das Binary hat, kann man versuchen, den Bug mit *strace* zu untersuchen.

Bug-Tracking-Systeme

Die Hauptaufgaben eines Bug-Tracking-System (oder Issue-Tracking-System) sind

- Bugs zu erfassen und diese zu verifizieren.
- Bugs unter den Entwicklern zu verteilen.
- formale interne Abläufe zu befolgen.

Wenn man einen Software-Bug gefunden hat, klärt man zuerst ab, ob dieser auch in der neusten Version vorhanden ist. Falls ja, sollte man schauen, ob der Bug bereits bekannt ist. Wenn nicht, so verfasst man dazu einen *Bug-Report*, welcher dann in das Bug-Tracking-System einfliesst. Ein guter Bug-Report sollte

- eine sehr kurze und klare Zusammenfassung des Bugs vorweisen.
- **Detaillierte Schritte** zur Reproduzierbarkeit des Bugs beinhalten.
- das vom User beobachtete und erwartete Software-Verhalten beschreiben.

Wenn man sich als Entwickler an das Flicken eines Bugs macht, resultiert daraus ein sogenannter *Patch*. Wichtig dabei ist, dass der Patch nur minimale Code-Änderungen für diesen spezifischen Bug beinhalten. Was danach mit dem Patch geschieht, variiert stark von Projekt zu Projekt. Generell kann man folgende vier Schritte festhalten:

- Patch schreiben (Bug Fixing)
- Patch einchecken (Source Repository)
- Patch-Code wird kontrolliert (Akzeptiert oder Abgelehnt?)
- Patch wird getestet (Bug gelöst?)