Typische Speicherfehler

Proseminar „C-Programmierung – Grundlagen und Konzepte“

Michael Kuhn
michael.kuhn@informatik.uni-hamburg.de

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

2011-05-20

1 Einführung
- Überblick
- Fehlerklassen

2 Compiler

3 Valgrind

- Die manuelle Speicherverwaltung von C erlaubt größtmögliche Kontrolle
  - . . . aber auch sich selbst in den Fuß zu schießen
- Programmierer muss sich selbst um Speicherzuteilung und -freigabe kümmern
- Es findet keine Referenzzählung oder automatische Speicherbereinigung statt
  - Speicher kann freigegeben werden, auch wenn er noch erreichbar ist
  - Andererseits kann auch nicht mehr erreichbarer Speicher belegt bleiben

- Nicht initialisierte Variablen
- Benutzung nach Freigabe
- Grenzüberschreitungen
- Speicherlecks
- Doppelte Freigabe
- Benutzung außerhalb des Gültigkeitbereichs
- NULL-Zeiger

1 Einführung

2 Compiler
- Einführung
- Benutzung

3 Valgrind

- Standardmäßig werden nur wenige Warnungen ausgegeben
- Es existieren sehr viele optionale Warnungen
  - In `man gcc` werden ca. 150 aufgelistet
  - Die meisten davon muss man nicht kennen ☺
- Viele davon können schon frühzeitig auf Probleme hinweisen
- Einige gebräuchliche sollten **immer** aktiviert werden

GCC-Aufruf

```
gcc -std=c99 -pedantic -Wall -Wextra -Wshadow
    -o my_app my_app.c
```

- `-std=c99`
  - Veranlasst den Compiler den C99-Standard zu verwenden
- `-pedantic`
  - Gibt alle Warnungen aus, die vom ISO-C-Standard vorgeschrieben werden
- `-Wall`
  - Gibt Warnungen aus, die relativ einfach zu verhindern sind
- `-Wextra`
  - Gibt zusätzliche Warnungen aus
- `-Wshadow`
  - Gibt eine Warnung aus, wenn eine lokale Variable eine andere überschattet

1 Einführung

2 Compiler

3 Valgrind
- Vorstellung
- Benutzung

- “Valgrind is a flexible program for debugging and profiling Linux executables. It consists of a core, which provides a synthetic CPU in software, and a series of debugging and profiling tools. The architecture is modular, so that new tools can be created easily and without disturbing the existing structure.”[1]
- Wir werden uns auf das `memcheck`-Werkzeug zur Erkennung von Speicherlecks und -fehlern beschränken
- Wichtig: Valgrind verlangsamt den Programmablauf teilweise enorm

[1] Quelle: `man valgrind`

- Valgrind bietet eine Vielzahl von Optionen
- Die wichtigsten werden hier kurz vorgestellt

Valgrind-Aufruf

```
valgrind
    --tool=memcheck
    --leak-check=full
    --show-reachable=yes
    --track-origins=yes
    --trace-children=yes
    --num-callers=20
    --suppressions=my_app.sup
    --gen-suppressions=yes
    ./my_app
```

- `--tool=memcheck`
  - Wählt das `memcheck`-Werkzeug aus
- `--leak-check=full`
  - Gibt zusätzliche Informationen zu Speicherlecks aus
- `--show-reachable=yes`
  - Zeigt auch noch belegten Speicher an, auf den am Programmende noch verwiesen wird
- `--track-origins=yes`
  - Gibt zusätzlich den Ursprung von nicht initialisiertem Speicher aus

- `--trace-children=yes`
  - Sorgt dafür, dass auch Unterprogramme ausgewertet werden
- `--num-callers=20`
  - Gibt die Tiefe des Aufrufbaums an, der dargestellt wird
- `--suppressions=my_app.sup`
  - Unterdrückt bestimmte Fehler, die in der angegebenen Datei aufgelistet sind
- `--gen-suppressions=yes`
  - Gibt optional Regeln zum Unterdrücken von aufgetretenen Fehlern aus