Debugging (Fehlersuche)

Aufspüren und Beseitigen von Programmfehlern (*bugs*, ursprünglich Käfer, Insekt, Wanze)

Der Begriff *debug* soll von Grace Hopper (US Navy Konteradmiralin) kreiert worden sein, die gerne die Geschichte von einem Techniker erzählte, der 1947 eine Störung am Computer behoben haben soll, indem er ein totes Insekt (eine Motte) zwischen den Kontakten eines Relais entfernte.

Debugger (Fehlersuchprogramm)

Programmierwerkzeug, das den Entwickler beim Testen von Programmen unterstützt. Zu den wichtigen Möglichkeiten eines Debuggers gehören:

- Unterbrechung der Programmausführung (an einem sogenannten *breakpoint* (Haltepunkt)).
- Zugriff auf Variablen (Inhalt ansehen/verändern)
- Einzelschrittausführung des Quellcodes

Verwendung des DDD Debuggers unter Linux:

DDD: GUI (*Graphical User Interface*) für **gdb** (zeilenorientierter Debugger von GNU) oder einen anderen kommandozeilenbasierten Debugger (z.B. **dbx**).

Zur graphischen Oberfläche gehören:

- *Source File window*
  - Quelltext
- *Command Tool window*
  - Buttons für die wichtigsten Kommandos
- *Debugger Console window*
  - gdb Kommandos
  - Tastatureingabe (stdin) und Bildschirmausgabe (stdout)

1. Voraussetzung zum Debuggen: Übersetzen mit -g

```
cc -g -o beispiel beispiel.c
```

2. DDD starten

```
ddd beispiel
```

oder

```
ddd
```

und anschließend ausführbares Programm über den Dialog **File→Open Program** laden.

3. Breakpoint setzen

Es gibt verschiedene Möglichkeiten einen Breakpoint zu setzen:

1. Linksklick auf den linken Rand der Zeile, **vor** deren Ausführung das Programm angehalten werden soll. Anschließend auf `Break`-Button (Stop-Zeichen) in der Werkzeugleiste klicken. Es sollte ein rotes Stop-Zeichen vor der ausgewählten Zeile erscheinen. (Der Breakpoint kann mit der gleichen Vorgehensweise wieder gelöscht werden.)
2. Rechte Maustaste auf dem linken Zeilenrand gedrückt halten und den Eintrag `Set breakpoint` in dem erscheinenden Menü auswählen.
3. Name einer Funktion anklicken (oder in das Eingabefeld in der Werkzeugleiste eintippen) und auf `Break`-Button klicken. Es erscheint ein Stop-Zeichen links vor der ersten ausführbaren Anweisung dieser Funktion.

4. Programmausführung starten

Auch hier gibt es verschiedene Möglichkeiten:

1. `Run` Button anklicken
2. über den Dialog **Program→Run**.
   In das Eingabefeld mit dem Text `Run with Arguments` können Komandozeilenargumente (z.B. auch eine Eingabeumlenkung) eingegeben werden.
3. run-Kommando im Debugger-Konsolenfenster eingeben, z.B.: `run < input.txt`

5. Einzelschrittausführung

mit den Kommandos:

1. `Next` : führt die nächste Anweisung aus. Im Gegensatz zum Kommando `Step` wird die Anweisung in jedem Fall vollständig ausgeführt, auch wenn diese Unterprogrammaufrufe enthält.
2. `Step` : führt die nächste Anweisung aus. Enthält diese Anweisung Unterprogrammaufrufe wird die Ausführung jedoch unmittelbar nach dem Verzweigen in ein Unterprogramm unterbrochen.
3. `Finish` : kehrt zur rufenden Funktion zurück, d.h. die Programmausführung wird fortgesetzt bis hinter die Stelle, von der die gerade aktive Funktion aufgerufen wurde.

Mit `Cont` kann die Programmausführung wieder aufgenommen werden.

6. Anzeigen von Variablen

1. Mauszeiger auf Variablennamen schieben. Inhalt der Variable wird eingeblendet.
2. Rechte Maustaste über Variablenname gedrückt halten und in dem erscheinenden Menü `Display` auswählen. Es wird ein zusätzlicher Bereich eingeblendet, in dem die Variable mit ihrem jeweils aktuellen Wert angezeigt wird.

3. Ein Feldelement, z.B. `a[i]` kann angezeigt werden, indem der Text `a[i]` mit der Maus markiert (oder in das Eingabefeld in der Werkzeugleiste eingetippt) und anschließend auf den `Display`-Button geklickt wird.
4. Über **Data→Display Local Variables** können alle lokalen Variablen angezeigt werden.
5. `print`-Kommando im Debugger-Konsolenfenster eingeben, z.B.: `print (char) a[i]`
   oder `(char) a[i]` in das Eingabefeld in der Werkzeugleiste eintippen und `Print`-Button (oder `Display`) anklicken.

7. Wenn Änderungen im Programm notwendig sind

- Programm editieren und neu übersetzen.
- `Run` ⇒ der geänderte Quelltext wird automatisch geladen.
- Soll ein anderes Programm getestet werden, sollte `ddd` beendet und erneut aufgerufen werden.

8. Verlassen des Debuggers

über **File→Exit**.

9. Weitere Informationen zu gdb-Kommandos

`help` im Debugger-Konsolenfenster eintippen.