Debugging

Debugging

- Es liegt in der Natur der Software-Entwicklung, dass Programme Fehler enthalten

Debugging

- Es liegt in der Natur der Software-Entwicklung, dass Programme Fehler enthalten
- Es gibt eine Reihe von Möglichkeiten, Fehler zu finden:
  - Programm wird einer Quelltextanalyse unterzogen

Debugging

- Es liegt in der Natur der Software-Entwicklung, dass Programme Fehler enthalten
- Es gibt eine Reihe von Möglichkeiten, Fehler zu finden:
  - Programm wird einer Quelltextanalyse unterzogen
  - Einstreuung von Ausgabeanweisungen (z.B. puts(), ...Achtung bei printf(...., \n);

Debugging

- Es liegt in der Natur der Software-Entwicklung, dass Programme Fehler enthalten
- Es gibt eine Reihe von Möglichkeiten, Fehler zu finden:
  - Programm wird einer Quelltextanalyse unterzogen
  - Einstreuung von Ausgabeanweisungen (z.B. puts(), ...Achtung bei printf(...., \n);
  - Einsetzen von Debuggern

Der gdb

- **Vorbereitung**
  - Übersetzen mit Option -g (`$> gcc -g meinprog.c`)
- **Aufruf**

```
$> gdb <Programmname>
GNU gdb 6.3
....
(gdb)
```

- **Programm Starten**

```
run
```

gdb

- **Quellcodeanzeige**
  - Um den aktuellen Haltepunkt herum: `list`
  - Beliebige Datei und Zeile: `list Datei:Zeile`
  - sonst: `help list`
- **Schrittweises Ausführen**
  - Debugger zeigt nächste auszuführende Zeile an
    `next` führt die ganze Zeile aus
    `step` springt in evtl. aufgerufene Funktion
    `finish` springt aus Funktion wieder heraus

gdb

- **Unterbrechungen / Haltepunkte**
  - Durch Benutzer definierbarer Punkt, an dem die
  - Ausführung angehalten wird
    `break Dateiname:Zeile`
    `break Funktionsname`
  - Programm hält ohne Fehler an, wenn Haltepunkt erreicht
  - Programm kann im Gegensatz zu einem Fehlerfall fortgesetzt werden:
  - Gesetzte Haltepunkte anzeigen lassen:
    `info breakpoints` zeigt alle Haltepunkte an
  - Haltepunkte löschen: `delete <nummer>`
  - Nummern werden werden nur einmal vergeben

gdb

- **Untersuchung der Variablen**
  - Inhalt jeder Variable mit `print <varname>` anzeigbar
  - Zeiger und Fledindizes wie im Quellcode verwendbar
    Beispiel: `print feld[4]`
  - Variablen modifizieren:
    Beispiel: set <varname>=<expression> (`set var i=5`)

gdb

- **Stackoperationen**
  - Stack: Funktions-Aufruf-Pfad bis zur aktuellen Stelle von main() aus
  - Anzeige mit `backtrace` oder `bt`
    aktuelle Funktion steht ganz oben mit höchster Nummer
    `break Funktionsname`
  - Herumwandern im Stack (Wechsel der Funktionsebene)
    `up <anzahl>` zur aufrufenden Funktion wechseln
    `down <anzahl>` zur aufgerufenden Funktion wechseln
  - Wichtig: beim Wechsel der Funktionsebene wird kein Code ausgeführt