Programmierung 2 (SS 2019)
Universität des Saarlandes
Fakultät MI

Zusatztutorium C
gdb

Prof. Dr. Jörg Hoffmann
M. Sc. Rebecca Eifler
M. Sc. Julia Wichlacz

Dieses Blatt bietet ihnen eine kurze Übersicht über die wichtigsten Befehle des GNU-Debuggers gdb. gdb kann Ihnen dabei helfen kann, Fehler in ihren C–Projekten zu finden oder Errors zu lokalisieren.

1 gdb ausführen

Um ein Programm mit `gdb` auszuführen, muss zunächst das Programm kompiliert werden. Mit Hilfe der so erhalteten ausführbaren Datei, der `executable`, kann man dann gdb wie folgt aufrufen:

```
gdb <executable>
```

Erhält das Programm zusätzlich Programmparameter, muss folgender Befehl verwendet werden:

```
gdb --args <executable> <arg1> <arg2> ...
```

wobei `arg1`, `arg2` usw. durch die entsprechenden Parameter zu ersetzen sind.

2 Die wichtigsten Befehle in gdb

Im Folgenden erhalten Sie eine kurze Übersicht über die meist verwendetsten Befehle in `gdb`. Mit Hilfe von `help <command>` können Sie genauere Informationen über den Befehl `<command>` erhalten. Um alle Befehle aufzulisten kann der Befehl `help` verwendet werden.

Allgemeine Befehle

- `quit`: Beendet gdb
- `run`: Führt das Programm bis zu einem Error oder einem Breakpoint aus
- `kill`: Beendet jetzige Programmausführung
- `continue`: Führt das Programm bis zum nächsten Breakpoint (oder Error) aus
- `finish`: Führt die Funktion aus, in der man sich momentan befindet
- `step <x>`: Führt die nächsten `<x>` Zeilen aus (nur 1, falls `<x>` nicht gegeben); springt dabei auch in Unterfunktionen
- `next <x>`: Führt die nächsten `<x>` Zeilen aus (nur 1, falls `<x>` nicht gegeben); ignoriert dabei Unterfunktionsaufrufe

Breakpoints und Watchpoints

- `break -`: Setzt einen breakpoint in der jetzigen Zeile
- `break <x>`: Setzt einen Breakpoint in Zeile `<x>`
- `break <function>`: Setzt einen breakpoint an den Anfang der Funktion `<function>`
- `delete <x>`: Löscht den breakpoint mit Nummer `<x>`
- `watch <condition>`: Führt das Programm aus, bis die Bedingung `condition` nicht mehr erfüllt ist
- `info break`: Listet alle breakpoints auf (durchnummeriert)
- `info watch`: Listet alle watchpoints auf (durchnummeriert)

Ausgaben

- `print <var>`: Printet den momentanen Inhalt der Variable `<var>` einmalig
- `watch <var>`: Printet den momentanen Inhalt der Variable `<var>` nach jedem Schritt, in dem sie sich ändert
- `disable <var>`: Macht `watch <var>` rückgängig
- `list`: Printet die nächsten 10 Codezeilen
- `list <function>`: Printet den Code der Funktion `function`

3 Hinweise

Mit Hilfe der Tastenkombination `Strg + X + A` erhält man eine schönere Ansicht des Debuggers, in der man den Code einfacher nachvollziehen kann.
In diesem Modus kann es passieren, dass sich die Anzeige des Codes verschiebt oder sich der Code überlappt. Dies kann durch `Strg + L` gefixt werden.
Befinden sich in dem Code prints, die eigentlich nicht benötigt werden, da der Inhalt jeder Variable auch direkt in `gdb` überprüft werden können (siehe `print`), ist es sinnvoll, den Befehl

```
tty /dev/null
```

direkt nach dem Starten von `gdb` zu verwenden, der die Print-Ausgabe in das Terminal verhindert.