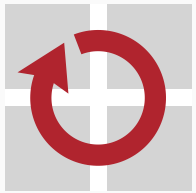
StackViz

April 2020

Lukas Panzer,
Simon Geis

Lehrstuhl für Informatik 4
Friedrich Alexander Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Recall: Anforderungen StackVIZ

- Dynamische Visualisierung des Stacks zur Laufzeit
  - Variable mit Typ, Name und derzeitigem Wert
  - Gesicherte Register (`rbp, rip`)
- Export von Vektorgrafiken
- Auf Wunsch: Visualisierung der Pointer
- Hervorheben von Stackframes

Allgemeines

Anzahl der Codezeilen:

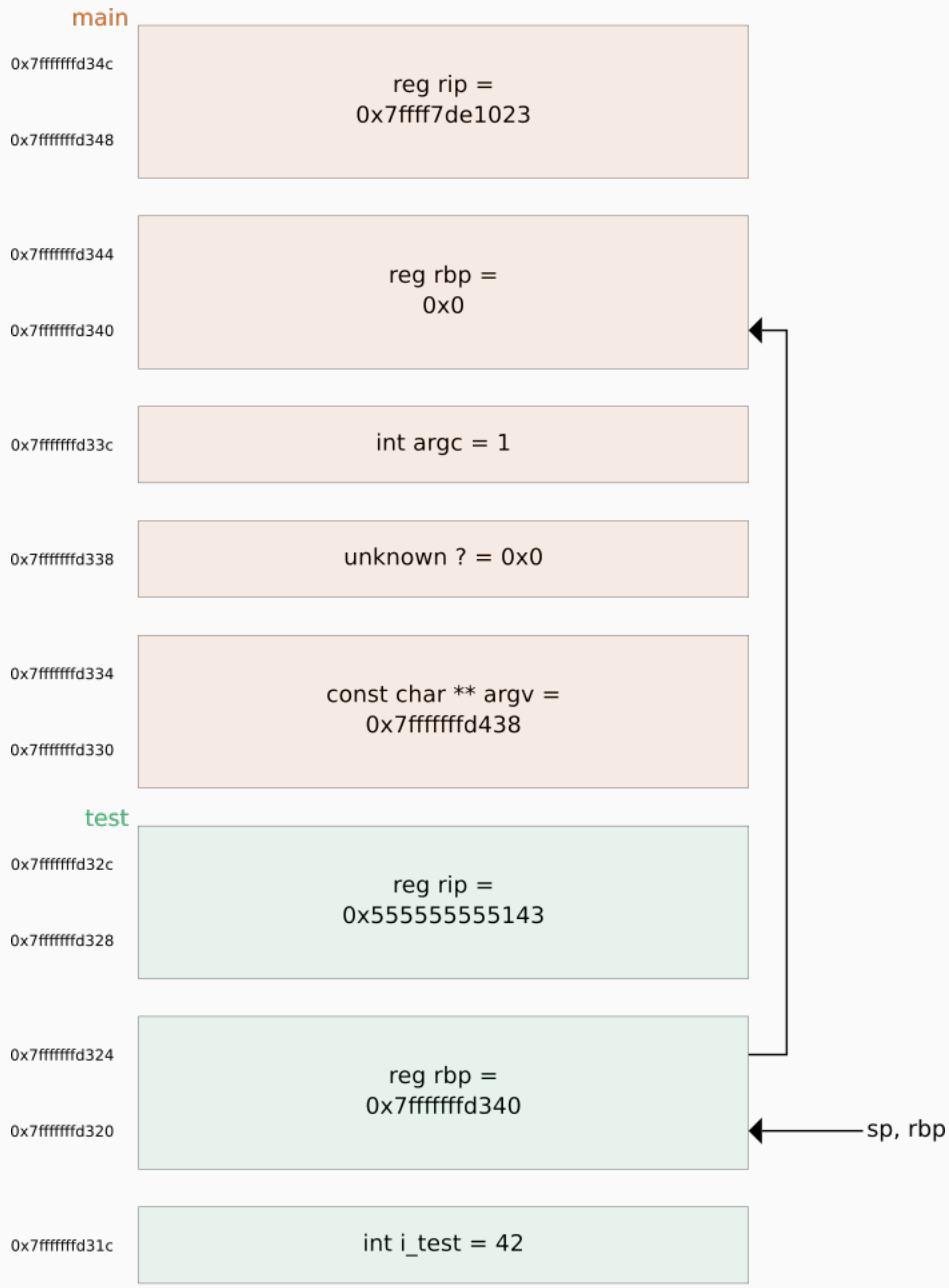
- Python:
  - `gdb.py`: 795 Zeilen
  - `svg.py`: 741 Zeilen
- Javascript: 137 Zeilen
- CSS: 41 Zeilen

Git-Repro:

- Lizenz: GNU General Public License
- https://gitlab.cs.fau.de/stackviz/stackviz

Features

- Stackvisualisierung



Features

- Stackvisualisierung
- Unterstützung von C, C++, 32 und 64 Bit

Features

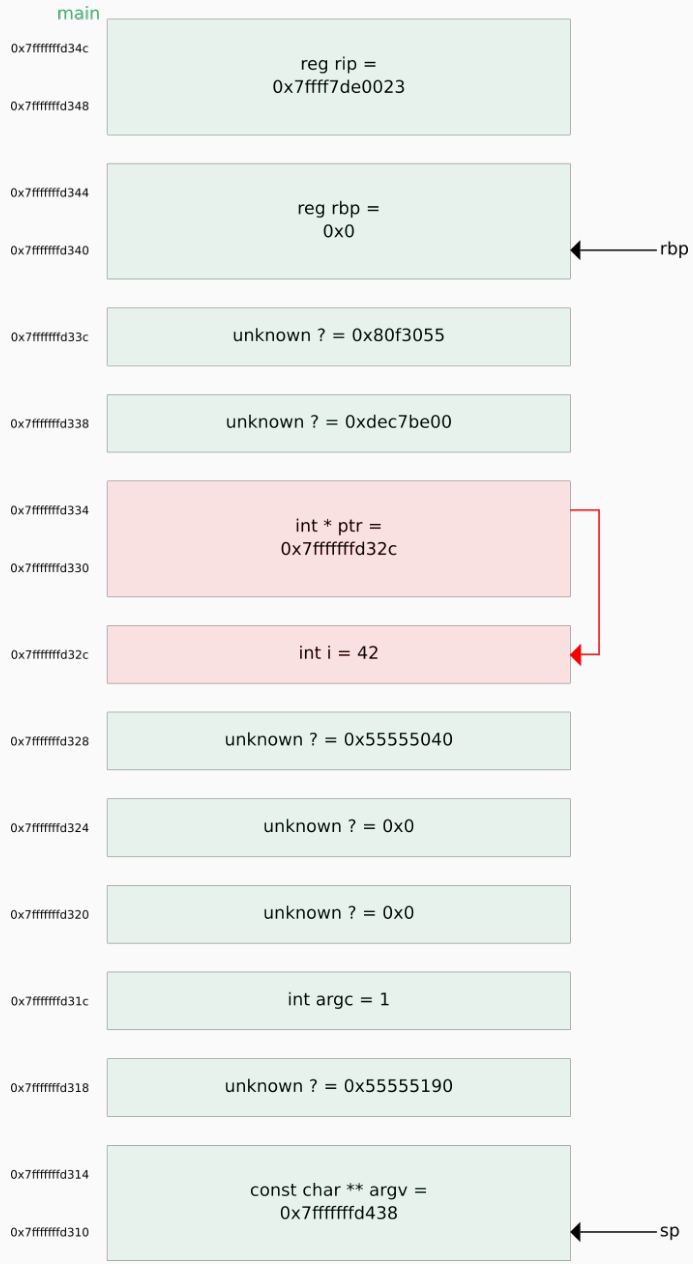
- Stackvisualisierung
- Unterstützung von C, C++, 32 und 64 Bit
- Anzeigen der Environment (auf Wunsch)

Features

- Stackvisualisierung
- Unterstützung von C, C++, 32 und 64 Bit
- Anzeigen der Environment (auf Wunsch)
- Hervorheben des Elements auf das der Pointer zeigt

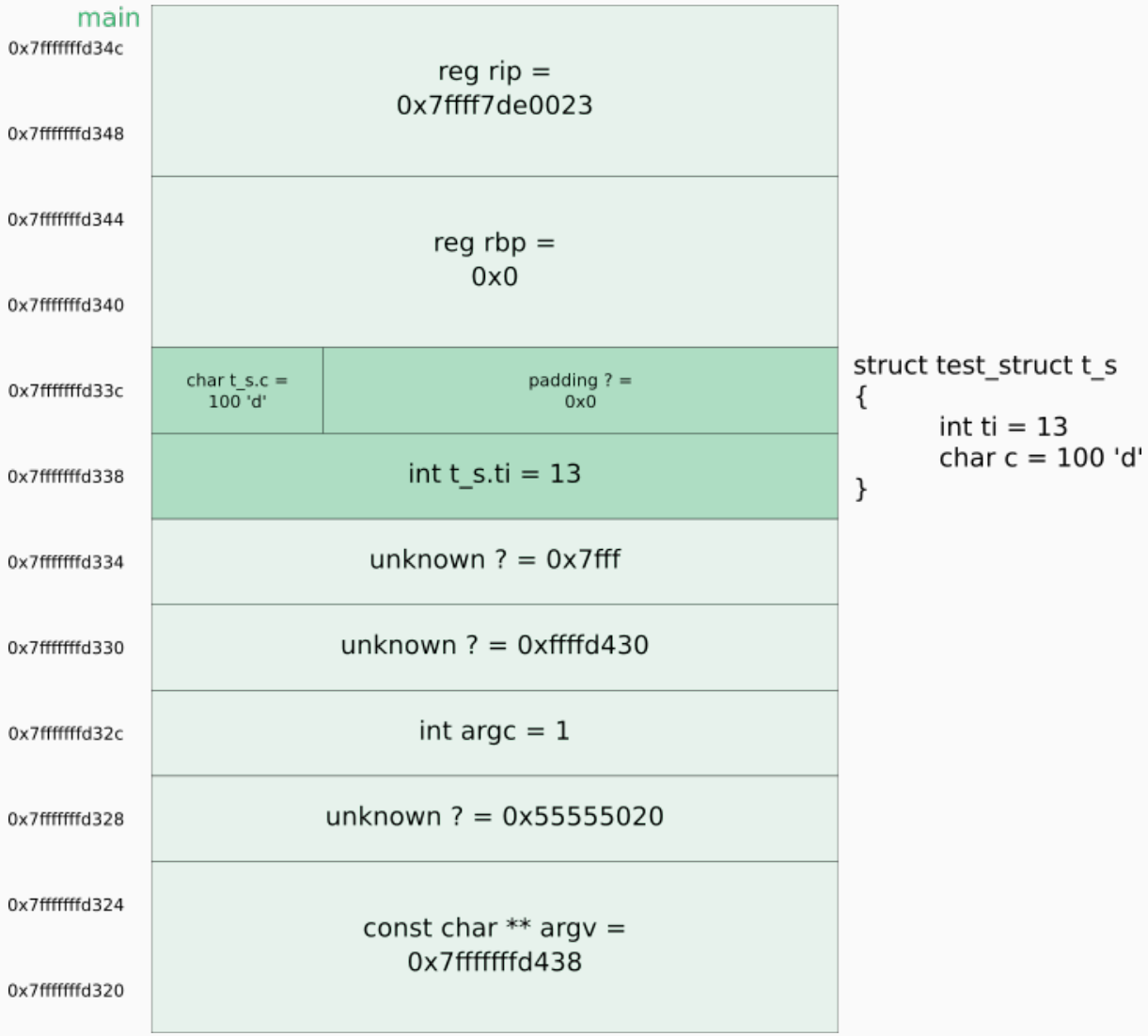
Features

- Stackvisualisierung
- Unterstützung von C, C++, 32 und 64 Bit
- Anzeigen der Environment (auf Wunsch)
- Hervorheben des Elements auf das der Pointer zeigt
- Hervorheben zusammengehöriger Werte (Elemente eines Arrays, struct Elemente)



Features

- Anzeigen von struct-Informationen



Bedienung

```
cd </path/to/StackViz>
gdb your-binary
# Breakpoint an der gewuenschten Stelle setzen
(gdb) b main
(gdb) r
# falls .gdbinit nicht aktiviert ist, Skripte manuell laden
# source gdb.py
# source svg.py
(gdb) createSVG
Display links? ([y], n): y
Display environment variables? (y, [n]): n
```

Weitere Informationen im README

Kommunikation mit GDB: GDB python API

Bsp: Informationen der Variablen des aktuellen Stackframes

```
import gdb

frame = gdb.selected_frame()
block = frame.block()

while(block):
    for symbol in block:
        s_name = symbol.name
        s_value = symbol.value(frame)
        s_type = symbol.type
        s_address = frame.read_var(s_name, block).address

    block = block.superblock
```

Erzeugung der Vektorgrafiken: svgwrite

Beispiel: Rechteck mit Text in der Mitte innerhalb einer Gruppe

```
import svgwrite

dwg = svgwrite.drawing('Stack.svg')

group = dwg.g(class_="classname", id="id-1")
rect = dwg.rect((0, 0), (100, 20),
                stroke='black', stroke_width=0.2,
                fill='orange', fill_opacity='0.1')

text = dwg.text('test', insert=(50, 10), fill='red',
                text_anchor='middle')
group.add(rect)
group.add(text)
dwg.add(group)
dwg.save()
```

Animationen

`javascript, css`

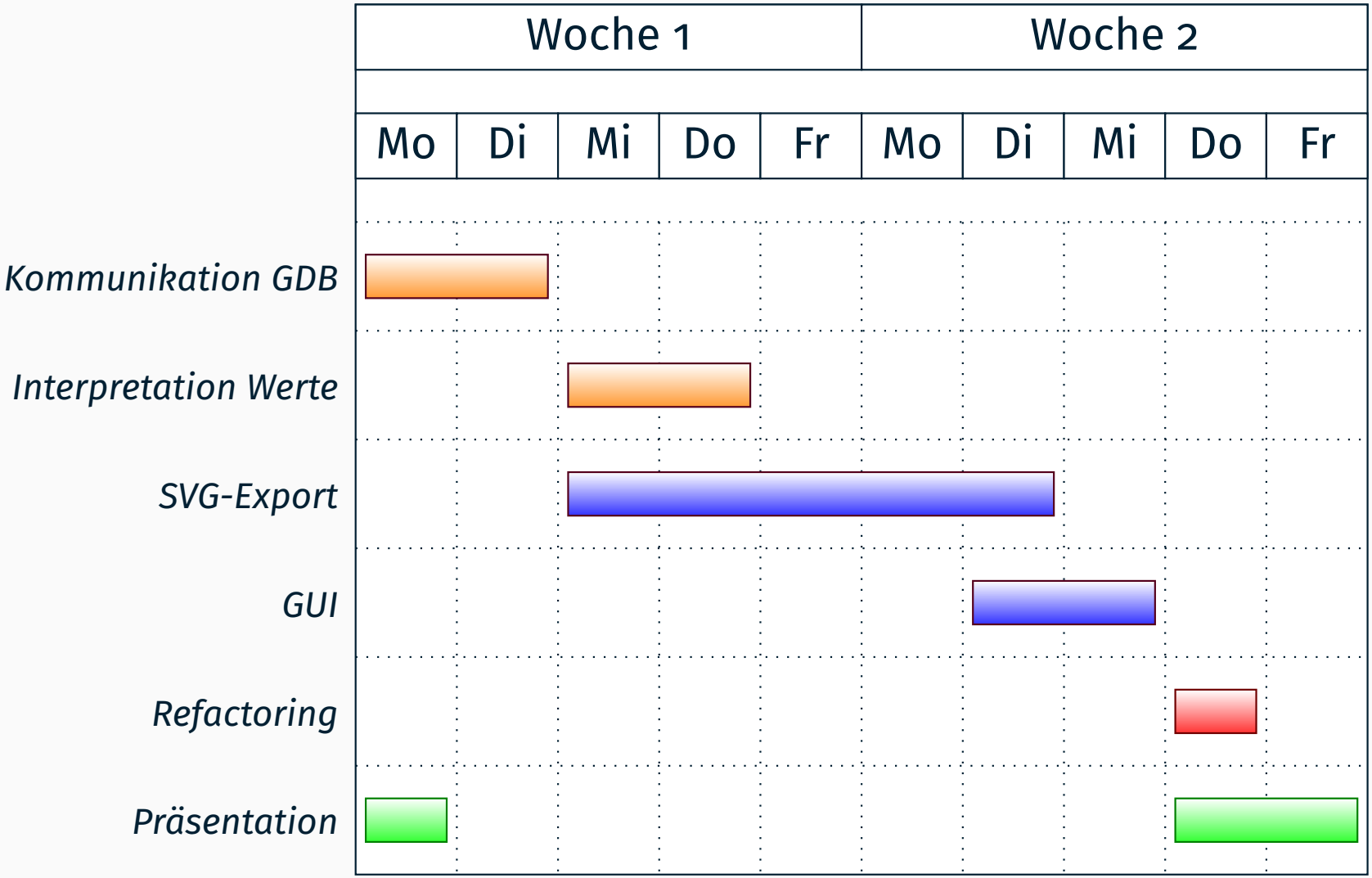
- Zusammengehörige Werte (array, struct) innerhalb einer Gruppe (’type_group’) speichern; Werte in dieser Gruppe verändern

```
.type_group:hover rect {
    fill-opacity:0.4;
}
```

- on-hover: Javascript färbt Verweis und Zielelement rot
- Javascript verbirgt Links beim Anzeigen von struct-Informationen

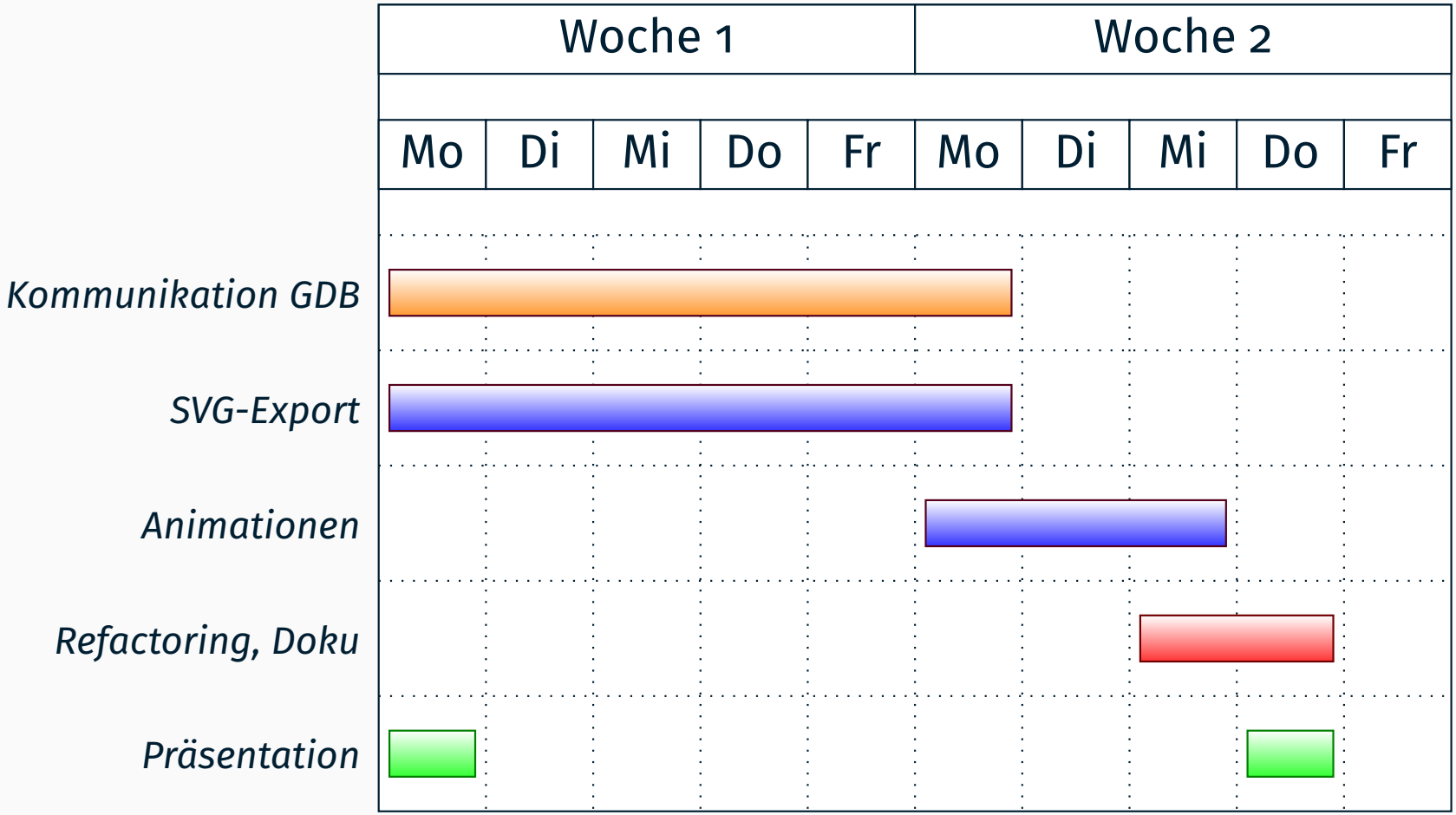
Zeitplan

Ursprünglicher Plan:



Zeitplan

Tatsächlicher Ablauf



Probleme

- Unterschiedliche Darstellung von Adressen in GDB
  $\rightsquigarrow$ Richtige Werte extrahieren

  ```
  0x7fffffffe9c8 "_=/usr/bin/gdb"
  ```

- Werte liegen auf dem Stack, von denen wir keine Informationen haben $\rightsquigarrow$ `'unknown ?'`
- Unterschiedliche Registernamen für andere Architekturen (`ebp` vs. `rbp` ...)
- Kein Traceback bei Laufzeitfehlern innerhalb von GDB
  $\rightsquigarrow$ erschwert Debugging

Mögliche nächste Schritte

- GUI/Webeinbindung mit passender Kommunikation zu GDB
- Überprüfung auf andere Architekturen (ARM, …)
- Weite der Rechtecke den Variablennamen anpassen
- …

Fragen?