



Einfuehrung strukturierte
Programmierung

[Tutorium]

vision
vision

& [Gruppe 06, 07, 08, 15]



Fragen

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

Offene Punkte aus
Vorlesung, AG, NG, ...





Linux Pipes

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

- `./program > log.txt`
 - Schreibt den Output des Programms in eine Datei
- `./program 1>log.txt 2>error.log`
 - Schreibt den stdout des Programms nach `log.txt` und den stderr nach `error.log`
- `valgrind --tool=memcheck --leak-check=yes ./troubles 1>log.txt 2>error.log`



Linux Pipes

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

- `./program < input.txt`
 - Uebergibt den Inhalt von `input.txt` an das Programm
 - `./own_solution < sampleTxt > log.txt`
`./ref_solution < sampleTxt > log2.txt`
- Differ
 - Vergleicht 2 Dateien ob sie sich unterscheiden
 - `Diff [datei1] [datei2]`
 - `Diff log.txt log2.txt`



Dynamischer speicher

dynamicMemory.c

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

■ malloc

- Fordert das erste mal Speicher an
- `[Pointer] = malloc([size])`
- `memory = malloc(sizeof(int));`

■ realloc

- Vergroessert den speicher
- `[pointer]=realloc([pointer],[new size])`
- `memory = realloc(memory, sizeof(int) * 2);`

■ free

- Gibt speicher wieder frei
- `Free([pointer]);`



Valgrind

<http://www.cprogramming.com/debugging/valgrind.html>

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

■ Compilerflags fuer gcc

- -Wall alle Fehler
- -o name des binary
- -g debug informationen
- -fstack-protector-all
preuft statische Arraygrenzen

■ Optionen fuer Valgrind

- --tool=memcheck
aktiviert die Speicherueberwachung
- --leak-check=yes
prueft auf speicherloecher



Valgrind

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

```
gcc -Wall -g -o [progrname] [sourcefile].c
valgrind --tool=memcheck --leak-check=yes ./mem
```

```
altihar@pluto:~/esp
[altihar@pluto esp]$ valgrind --tool=memcheck --leak-check=yes ./dynMem
==12006== Memcheck, a memory error detector.
==12006== Copyright (C) 2002-2006, and GNU GPL'd, by Julian Seward et al.
==12006== Using LibVEX rev 1658, a library for dynamic binary translation.
==12006== Copyright (C) 2004-2006, and GNU GPL'd, by OpenWorks LLP.
==12006== Using valgrind-3.2.1, a dynamic binary instrumentation framework.
==12006== Copyright (C) 2000-2006, and GNU GPL'd, by Julian Seward et al.
==12006== For more details, rerun with: -v
==12006==
1
2
==12006==
==12006== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 12 from 1)
==12006== malloc/free: in use at exit: 0 bytes in 0 blocks
==12006== malloc/free: 2 allocs, 2 frees, 12 bytes allocated.
==12006== For counts of detected errors, rerun with: -v
==12006== All heap blocks were freed -- no leaks are possible.
[altihar@pluto esp]$
```



Valgrind

memoryErrors.c, memoryErrorsII.c, memoryTricks.c, memoryTroubles.c

```
drawSimple:>eod
```

```
==7766== Invalid write of size 1
```

```
==7766==    at 0x8048696: getInputCommand (drawingprogramsimple.c:121)
==7766==    by 0x80484EF: main (drawingprogramsimple.c:59)
==7766== Address 0x416709B is 0 bytes after a block of size 3 alloc'd
==7766==    at 0x40054BB: realloc (vg_replace_malloc.c:306)
==7766==    by 0x8048649: getInputCommand (drawingprogramsimple.c:109)
==7766==    by 0x80484EF: main (drawingprogramsimple.c:59)
```

```
==7766==
```

```
==7766== Invalid read of size 1
```

```
==7766==    at 0x804859B: main (drawingprogramsimple.c:69)
==7766== Address 0x416709B is 0 bytes after a block of size 3 alloc'd
==7766==    at 0x40054BB: realloc (vg_replace_malloc.c:306)
==7766==    by 0x8048649: getInputCommand (drawingprogramsimple.c:109)
==7766==    by 0x80484EF: main (drawingprogramsimple.c:59)
```



Valgrind

```
==8684== Conditional jump or move depends on uninitialised value(s)
==8684== at 0x9F30EE: _IO_file_overflow@@GLIBC_2.1 (in /lib/libc-2.5.so)
==8684== by 0x9F5B32: __overflow (in /lib/libc-2.5.so)
==8684== by 0x9EB875: putchar (in /lib/libc-2.5.so)
==8684== by 0x80484F6: write (ex1.c:64)
==8684== by 0x8048696: all (ex1.c:173)
==8684== by 0x8048867: main (ex1.c:279)

==8684== Syscall param write(buf) points to uninitialised byte(s)
==8684== at 0xA51AC3: __write_nocancel (in /lib/libc-2.5.so)
==8684== by 0x9F25B4: new_do_write (in /lib/libc-2.5.so)
==8684== by 0x9F289E: _IO_do_write@@GLIBC_2.1 (in /lib/libc-2.5.so)
==8684== by 0x9F31A7: _IO_file_overflow@@GLIBC_2.1 (in /lib/libc-2.5.so)
==8684== by 0x9F5B32: __overflow (in /lib/libc-2.5.so)
==8684== by 0x9EB875: putchar (in /lib/libc-2.5.so)
==8684== by 0x8048510: write (ex1.c:68)
==8684== by 0x8048696: all (ex1.c:173)
==8684== by 0x8048867: main (ex1.c:279)
==8684== Address 0x400A000 is not stack'd, malloc'd or (recently) free'd
```



structs

structreadwrite.c

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

- typedef struct
==> eigener Datentyp

```
//define a new datatype
typedef struct
{
    int color;
    int x_pos_start;
    int y_pos_start;
    int x_pos_end;
    int y_pos_end;
} Line;
```



Command Line Args

commandline.c

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

- Uebergabe von Kommandos an das Programm:

```
int main(int argc, char *argv[])
```

Argument

Anzahl Argumente: 0 ... N

Standardmaessig ist das argv[0] der Programmname mit komplettem Aufrufpfad



SVG

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="400" height="400"
  xmlns="http://www.w3.org/2000/svg">
  <title>SVG</title>
  <style type="text/css"><![CDATA[
    text {font-size:60px; text-anchor:middle;}
  ]]></style>
  <rect x="0" y="0" height="400" width="400"
    fill="white" stroke="blue"
    stroke-width="100" />
  <rect x="160" y="80" width="80"
    height="240" fill="red" />
  <rect x="80" y="160" width="240"
    height="80" fill="red" />
  <text x="200" y="220">Arzt</text>
</svg>
```

Quelle: Wikipedia - <http://de.wikipedia.org/wiki/SVG>



ASCII vs. Binary

Fragen

Pipes

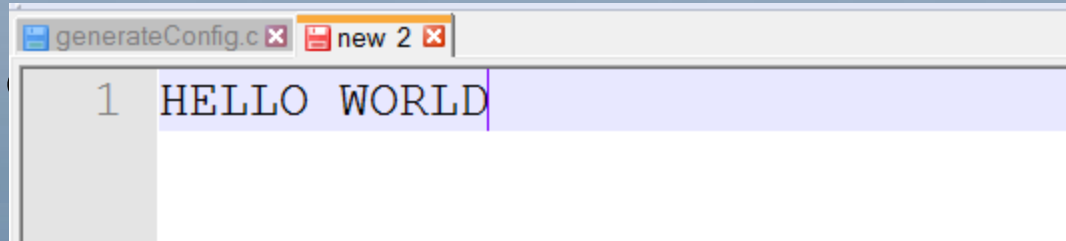
valgrind

```
[altihar@pluto esp]$ hexdump tc1.cfg
00000000 6568 6c6c 006f 0000 000a
0000 0064 0000
```

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000000	68	65	6c	6c	6f	00	00	00	0d	0a	00	00	00	64	00	00	hello.....d..
00000010	00	e8	03	00	00è.....
00000020

00000010

```
[altih
hello
```



*txt, *bin

Liste

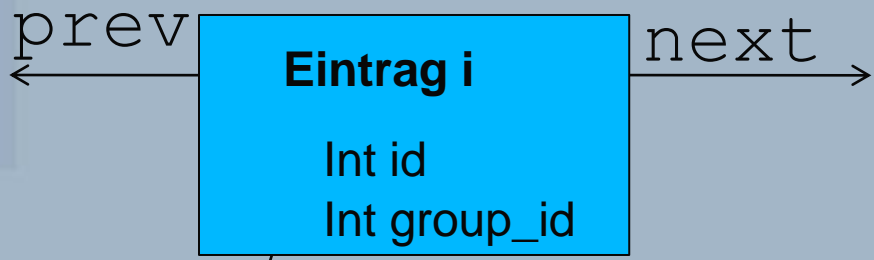
ende

```
[altihar@pluto esp]$
```

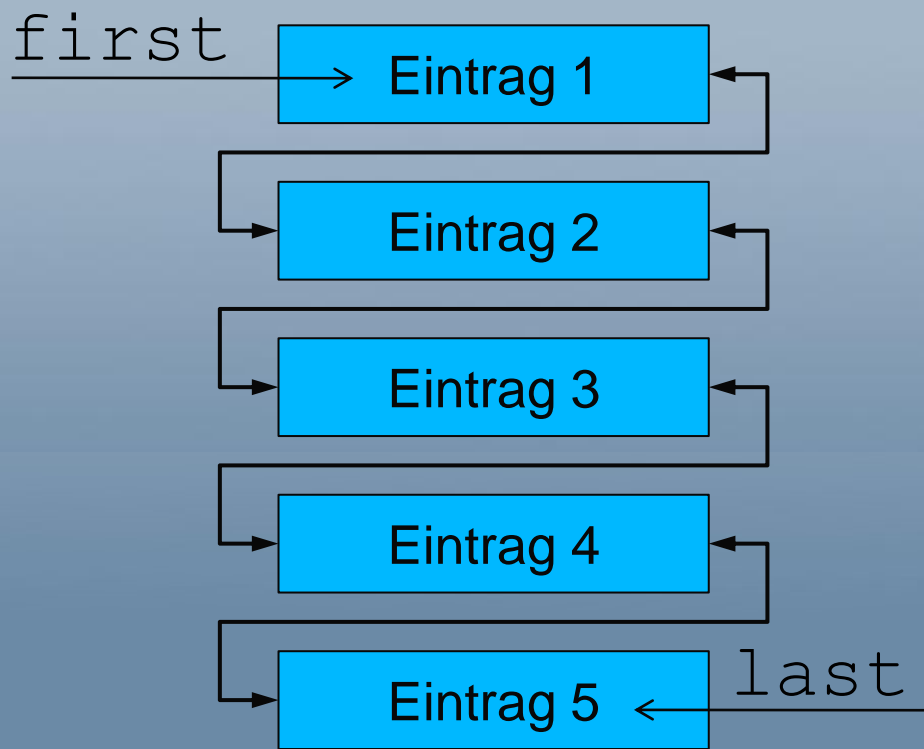


Datenstruktur

- Fragen
- Pipes
- valgrind
- Struct
- CLA, SVG
- *txt, *bin
- Liste
- ende



Objekt



Datenstruktur

Fragen

Pipes

valgrind

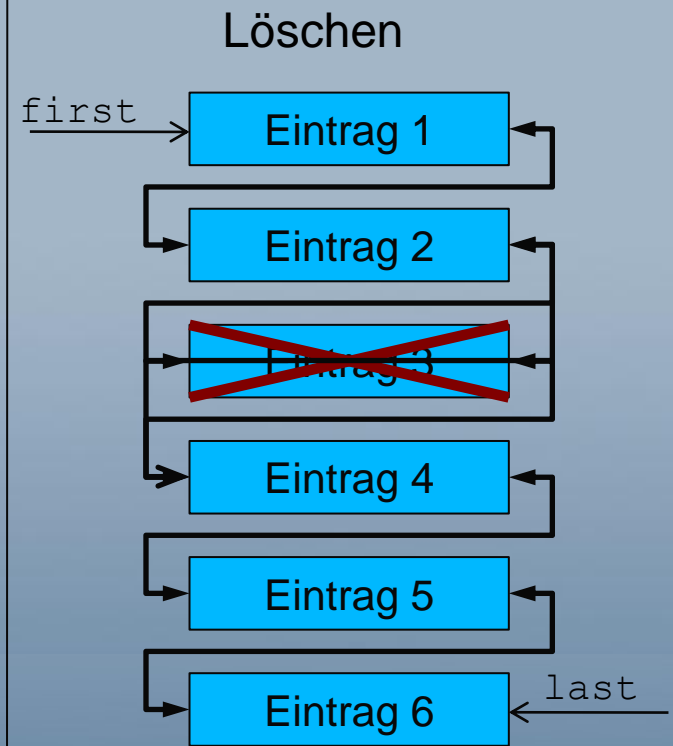
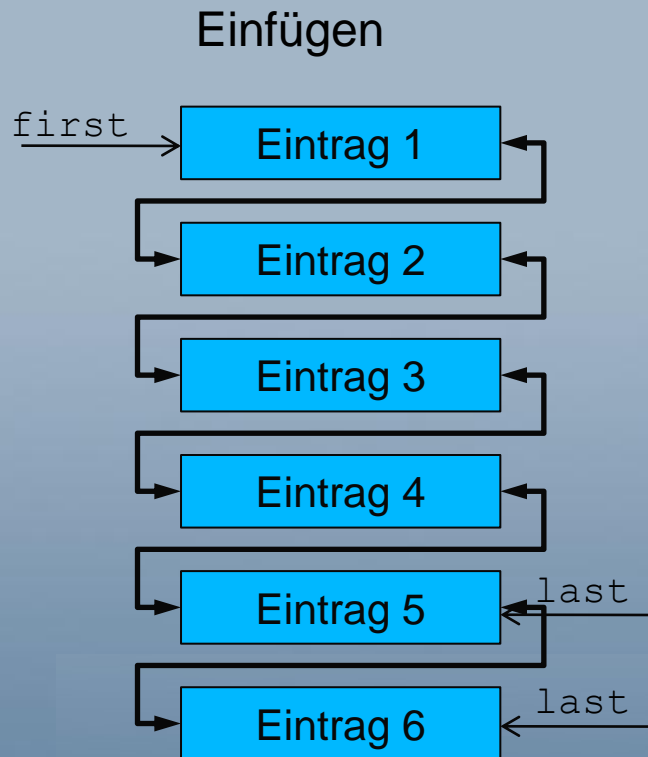
Struct

CLA, SVG

*txt, *bin

Liste

ende





Arbeit fuer danach

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

- Newsgroup lesen
- Drawingprogrammsimple
- generateConfig.c
- Coden, coden, coden



**Achtung,
kein Tutorium mehr**



Fuer den Rest der LV

Fragen

Pipes

valgrind

Struct

CLA, SVG

*txt, *bin

Liste

ende

Viel Glueck bei den Uebungen

