Hinweise – C-Programmierung

Rainer Müller

Department Informatik 4
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander-Universität Erlangen-Nürnberg

WS 2014/2015

Bit-Manipulation

das n-te Bit gesetzt:	`1 << n`
Maske für alle außer dem n-ten Bit:	`~(1 << n)`
Bit setzen $\rightarrow$ Verodern:	`x |= 0x01;`
Bit löschen $\rightarrow$ Verunden:	`x &= ~0x01;`
Bit umkehren $\rightarrow$ Exklusives Oder	`x ^= 0x01;`
niederwertiges Byte:	`x & 0xff`
nur niederwertiges Byte verändern:	`x &= ~0xff; x |= b & 0xff;`

Variablen fester Größe

C garantiert keine Größe von `short int`, `int`, `long int`, aber `inttypes.h` enthält folgende Definitionen:

- `uint8_t`
- `uint16_t`
- `uint32_t`
- `uint64_t`
- `int8_t`
- `int16_t`
- `int32_t`
- `int64_t`

Format-String für inttypes

Problem: `%d` bezieht sich auf `int`, `%ld` auf `long int` usw.

Für Variablen fester Größe können die Macros `PRId8`, `PRIu16`, `PRIo32`, `PRIx64` etc. verwendet werden.

Beispiel:

```
uint32_t x;

x = 1234;
printf("x hat den Wert %" PRIu32 "\n", x);
```

Funktionspointer

- Eine Funktion ist eine Adresse im `.text`-Segment
- Diese Adresse kann in C genutzt werden (z.B. für Callbacks)
- Deklaration für Funktionspointer:
  Rückgabe-Typ (*Name)(Parameter-Typ, Parameter-Typ, ...)
- Beispiel: `void (*funptr)(int, long int);`
- Zuweisung: `funptr = myfunc;`
- Aufruf mittels: `(*funptr)(23, 42);`
- Einfacher: `funptr(23, 42);`

C-Code → Binär-Code

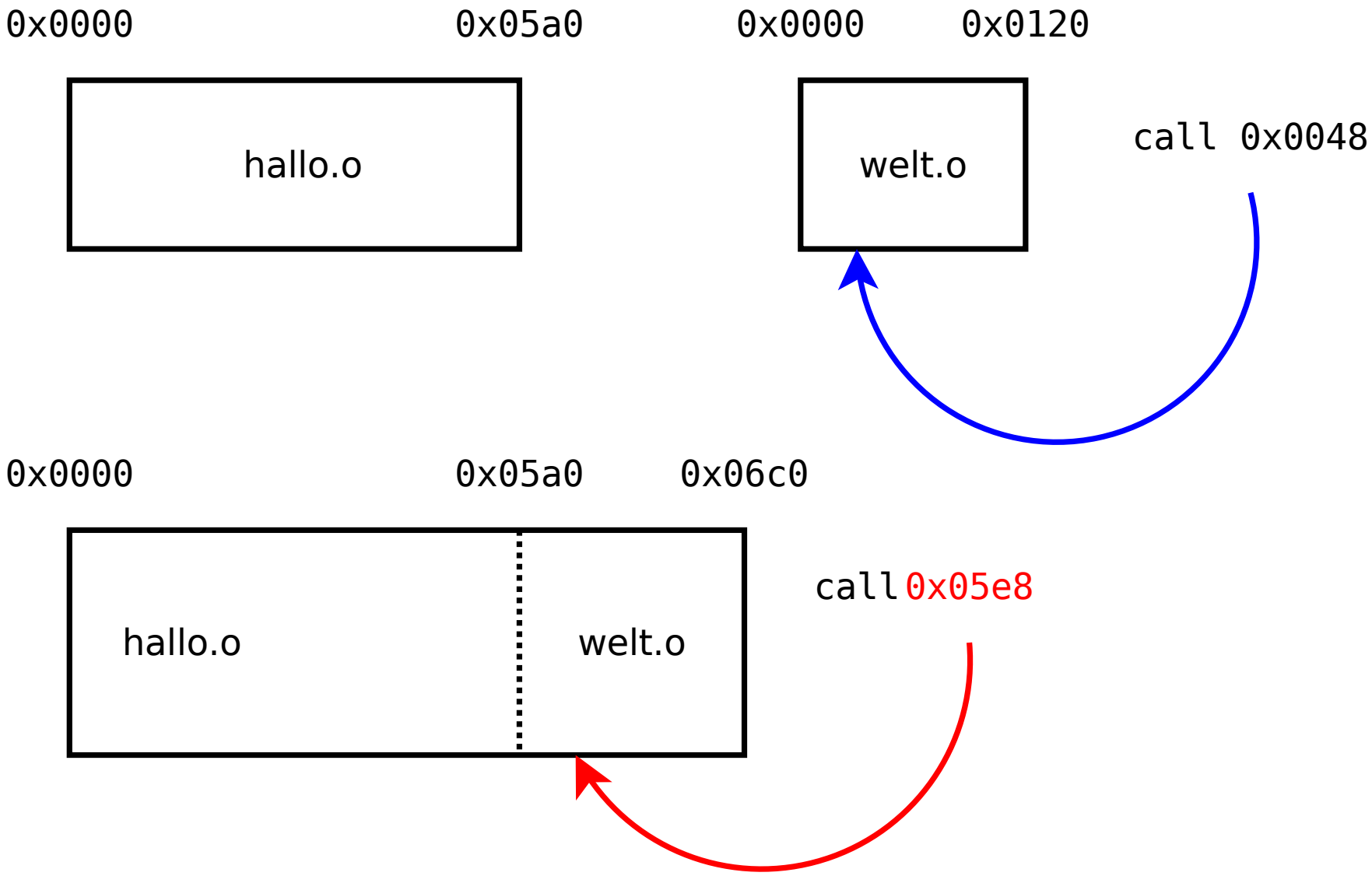
- Programmierer schreibt Programm in C
- CPU führt binäre Befehle im Speicher aus

→ Was passiert dazwischen?

Toolchain

1. Preprozessor cpp ersetzt Makros und fügt Code für inkludierte Dateien ein.
   `cpp hallo.c > hallo.pre.c`
2. Compiler `cc1` erzeugt Assembler-Code
   `cc1 [-m32] hallo.pre.c`
3. Assembler generiert Object-Code
   `as [-32] hallo.pre.s -o hallo.o`
4. Linker bindet Object-Dateien zusammen, und löst dabei Referenzen auf und führt Relokation durch
   `ld [-m elf_i386] hallo.o -o hallo.bin [...]`

Relokation beim Binden



Tools zum Inspizieren von Objects/ELFs

- Anzeigen der Relokationstabelle:
  `objdump -r <datei>`
- Assembler-Listing (mit Binärdarstellung) anzeigen:
  `objdump -D <datei>`
- Eine Sektion in Binärform in eine Datei kopieren:
  `objcopy -O binary -j <sektion> <datei> <zieldatei>`
- Weitere Tools: `readelf`, `nm`