Programmieren in C

Speichergefummel

thoto

/dev/tal e.V.

20. April 2013

Agenda für Heute

1. Speichergefummel
   - Strings lesen
2. Typen und Typumwandlungen
   - Typumwandlung (Casts)
   - Schöner: Union!
   - Strukturen
   - Enum
   - typedef
3. Erweiterter Umgang mit Speicher
   - Zugriff
   - Umgang mit Strukturen

Strings einlesen

getc()

- getchar() liest einzelnes Zeichen

getc()

- getchar() liest einzelnes Zeichen
- Vorteil: Genau ein Zeichen

getc()

- getchar() liest einzelnes Zeichen
- Vorteil: Genau ein Zeichen
- Nachteil: Enter

getc()

- getchar() liest einzelnes Zeichen
- Vorteil: Genau ein Zeichen
- Nachteil: Enter

$\Rightarrow$ Ziel: Begrenzte Zahl Zeichen lesen.

getc()

- getchar() liest einzelnes Zeichen
- Vorteil: Genau ein Zeichen
- Nachteil: Enter

⇒ Ziel: Begrenzte Zahl Zeichen lesen.

⇒ Lösung: fgets()

Typumwandlungen

Zeichentabelle selber bauen

- Wir wollen:
- Eigene ASCII-Tabelle
- Iteration über Zeilen

Zeichentabelle selber bauen

- Wir wollen:
- Eigene ASCII-Tabelle
- Iteration über Zeilen
- Iteration über Spalten

Casting

- begrenzte Typumwandlung

Casting

- begrenzte Typumwandlung
- normalerweise Umwandlung

Casting

- begrenzte Typumwandlung
- normalerweise Umwandlung
- nicht bei Pointern

Casting

- begrenzte Typumwandlung
- normalerweise Umwandlung
- nicht bei Pointern

clean

```
char foo='a';
int bar=(int) foo;
```

Casting

- begrenzte Typumwandlung
- normalerweise Umwandlung
- nicht bei Pointern
- Vorsicht bei Referenzen! (Pointer muss Pointer bleiben!)

clean

```
char foo='a';
int bar=(int) foo;
```

ASCII-Tabelle

```
int main(int argc, char** argv){
  int i,j;
  for(i=32;i<128;i+=16){
    for(j=i;j<(i+16);j++){
      printf("%d:%c", j, (char) j);
    }
  putchar('\n');
  }
  return 0;
})
```

Unions

- Vereinigung von Typen
- Typ X ist gleichzeitig int und char
- verschiedene Größen im gleichen Speicher.

Unions

- Vereinigung von Typen
- Typ X ist gleichzeitig int und char
- verschiedene Größen im gleichen Speicher.

Unions

- Vereinigung von Typen
- Typ X ist gleichzeitig int und char
- verschiedene Größen im gleichen Speicher.

Unions

- Vereinigung von Typen
- Typ X ist gleichzeitig int und char
- verschiedene Größen im gleichen Speicher.

clean

```
union foo {
int* integer;
char* charakter;
};
```

ASCII-Tabelle mit Union

```
int main(int argc, char** argv){
  int i,j;
  union charout {
    int* integer;
    char* charakter;
  } charout;

  charout.integer=(int*) malloc(sizeof(integer));

  for(i=32;i<128;i+=16){
    for(j=i;j<(i+16);j++){
      *(charout.integer)=j;
      printf("%d:%c"   *charout.integer,*charout.cha
```

Vorsicht: Ausrichtung!

Vorsicht!

Ausrichtung nicht an erstes Byte der Struktur gebunden!
(Grund: Prozessorspezifisch performanter)
Hier klappt's aber: KEINENFALLS drauf vertrauen!

Strukturen

- Wie Union nur hintereinander

Strukturen

- Wie Union nur hintereinander
- mehrere Daten zusammen speichern

Strukturen

- Wie Union nur hintereinander
- mehrere Daten zusammen speichern

struct

```
struct foo{
char* foo;
int bar;
} datum;

datum.foo="hallo welt";
datum.bar=123;
```

Enum

- Auswahl aus mehreren Konstanten

enum

```
enum Ampel { ROT=1, GRUEN, BLAU } ampelphase;
```

Typedef

- Definition eines Types wie `int`
- Kein nerviges struct foo ...

```
typedef struct MyKreuzung {
  int ampelzahl;
  enum phase { ROT=1, GELB, GRUEN } *phase;
} Kreuzung;
```

Speicher kopieren

memcpy();

Speicher initialisieren

memset();