



Inhalt

1. Was sind eingebettete Systeme?
2. Arrays und Pointer für Fortgeschrittene
3. Einführung in Unix
4. TCP/IP-Netzwerke in der Praxis
5. Bus-Systeme

... und *viel* praktisches Arbeiten!

1 Was sind eingebettete Systeme?

*Der Ausdruck **eingebettetes System** (auch englisch **embedded system**) bezeichnet einen elektronischen Rechner oder auch Computer, der in einen technischen Kontext eingebunden (eingebettet) ist.*

Wikipedia

1 Was sind eingebettete Systeme?

*Der Ausdruck **eingebettetes System** (auch englisch **embedded system**) bezeichnet einen elektronischen Rechner oder auch Computer, der in einen technischen Kontext eingebunden (eingebettet) ist.*

Wikipedia

—→ Keine Aussage über die Größe.

1 Was sind eingebettete Systeme?

- Projektaufgabe:
Eingebettetes System eigener Wahl zum Laufen bringen

→ Projektaufgabe überlegen

Beispiele:

- Rasenmährobotersteuerung
- Weiterarbeit an Smartphone-Prototyp
- Verbesserungen an Flugzeugkabinensimulatoren
- Spülmaschinensteuerung
- Sampler-Grabber für MIDI
- ...

Aufgaben

Aufgabe 1: Sieb des Eratosthenes
möglichst viele Primzahlen berechnen

Aufgabe 2: Kalender
Jahreszahl eingeben,
Kalender mit Feiertagen ausgeben
Hinweis: „Osterformel“

2 Arrays und Zeiger für Fortgeschrittene

Array:

```
char a[] = "Test";
```

Zeiger:

```
char *p = "Test";
```

- In beiden Fällen wird ein Array von ganzen Zahlen (5 **char**-Variable mit den Werten 84, 101, 115, 116 und 0) im Speicher angelegt.
- Links heißt das Array **a**; rechts ist es „anonym“.
- Rechts wird zusätzlich ein Zeiger **p** im Speicher angelegt, der auf das (anonyme) Array zeigt.
- **&a** ist dasselbe wie **a**, nämlich die Adresse des Arrays.
- **&p** ist die Adresse des Zeigers.
- **p** ist der Wert des Zeigers, momentan also die Adresse des (anonymen) Arrays.

2 Arrays und Zeiger für Fortgeschrittene

Array:

```
char a[] = "Test";
```

Zeiger:

```
char *p = "Test";
```

- In beiden Fällen wird ein Array von ganzen Zahlen (5 **char**-Variable mit den Werten 84, 101, 115, 116 und 0) im Speicher angelegt.
- Links heißt das Array **a**; rechts ist es „anonym“.
- Rechts wird zusätzlich ein Zeiger **p** im Speicher angelegt, der auf das (anonyme) Array zeigt.
- **&a** ist **fast** dasselbe wie **a**, nämlich die Adresse des Arrays **bzw. das Array selbst, das zuweisungskompatibel zu einem Zeiger auf Elemente des Arrays ist. & bewirkt hier eine (nicht explizite!) Typumwandlung.**
- **&p** ist die Adresse des Zeigers.
- **p** ist der Wert des Zeigers, momentan also die Adresse des (anonymen) Arrays.

2 Arrays und Zeiger für Fortgeschrittene

Array:

```
char *a[] = { "Dies", "ist", "ein", "Test" };
```

Zeiger:

```
char **p = a;
```

- Array von Zeigern auf **char**-Variable
- Zeiger auf das Array = Zeiger auf Zeiger auf **char**-Variable
- Schleife durch äußeres Array mit **p++** möglich

2 Arrays und Zeiger für Fortgeschrittene

Array:

```
char a[][5] = { "Dies", "ist", "ein", "Test" };
```

Zeiger:

```
char *p = a[0];
```

- zweidimensionales Array von **char**-Variablen
- Zeiger auf Array-Komponente
 - = Zeiger auf eindimensionales Array
 - = Zeiger auf **char**-Variable
- Schleife durch äußeres Array mit Zeiger-Arithmetik ~~nicht~~ möglich


nur mit Trick: `p += 5`

2 Arrays und Zeiger für Fortgeschrittene

```
typedef char string5[5];  
string5 a[] = { "Dies", "ist", "ein", "Test" };  
string5 *p = a;
```

- Array von Array von **char**-Variablen
= zweidimensionales Array von **char**-Variablen
- Zeiger auf zweidimensionales Array
- Schleife durch äußeres Array mit **p++** möglich

→ Fazit:
Ein Hoch auf **typedef**!

2 Arrays und Zeiger für Fortgeschrittene

```
typedef char string5[5];  
string5 *p = { "Dies", "ist", "ein", "Test" };
```

- anonymes Array von Array von **char**-Variablen
= anonymes zweidimensionales Array von **char**-Variablen
- Zeiger auf zweidimensionales Array
- Schleife durch äußeres Array mit **p++** möglich

2 Arrays und Zeiger für Fortgeschrittene

```
typedef char string5[5];  
string5 *p = { "Dies", "ist", "ein", "Test" };
```

- ~~• anonymes Array von Array von **char**-Variablen
= anonymes zweidimensionales Array von **char**-Variablen~~
- ~~• Zeiger auf zweidimensionales Array~~
- ~~• Schleife durch äußeres Array mit **p++** möglich~~

Das Konstrukt { "Dies", "ist", "ein", "Test" }
steht für ein Array von 4 Zeigern auf **char**-Variable.

string5 *p hingegen erwartet einen Zeiger auf ein Array von 5 **char**-Variablen.

Es bekommt die Adresse von "Dies" zugewiesen.

Durch das Erhöhen von **p** (um 5) zeigt es danach *zufällig* auf das "ist".
Bei nochmaligem Erhöhen zeigt es auf das "in" von "ein".

(Auch ohne Optimierung werden die Strings "ist", "ein" und "Test"
u. U. wegoptimiert.)