

# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

- *keine Aussage über die Größe*

# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

- keine Aussage über die Größe
- in der Praxis: so klein wie möglich

# Was sind eingebettete Systeme?

Der Auswurf  
Rechner  
(eingebettete

- keine
- in der



m

# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

- keine Aussage über die Größe
- in der Praxis: so klein wie möglich, z. B.:
  - mehrere Rechnerschränke
  - Industrie-PC
  - Einplatinencomputer
  - Mikro-Controller

# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

- keine Aussage über die Größe
- in der Praxis: so klein wie möglich, z. B.:
  - mehrere Rechnerschränke
  - Industrie-PC
  - Einplatinencomputer
  - Mikro-Controller
- Bedienung über spezielle Ein-/Ausgabegeräte (z. B. Flugsimulator, Telefon)

# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

- keine Aussage über die Größe
- in der Praxis: so klein wie möglich, z. B.:
  - mehrere Rechnerschränke
  - Industrie-PC
  - Einplatinencomputer
  - Mikro-Controller
- Bedienung über spezielle Ein-/Ausgabegeräte (z. B. Flugsimulator, Telefon)
- Wartung über speziellen Administrator-Zugang
  - Bus-Schnittstelle (RS-232, CAN-BUS)
  - Netzwerk (TCP/IP, Ethernet oder WLAN)



# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

- keine Aussage über die Größe
- in der Praxis: so klein wie möglich, z. B.:
  - mehrere Rechnerschränke
  - Industrie-PC
  - Einplatinencomputer
  - Mikro-Controller
- Bedienung über spezielle Ein-/Ausgabegeräte (z. B. Flugsimulator, Telefon)
- Wartung über speziellen Administrator-Zugang
  - Bus-Schnittstelle (RS-232, CAN-BUS)
  - Netzwerk (TCP/IP, Ethernet oder WLAN)
- Programmierung oft außerhalb des Systems (Cross-Entwicklungswerkzeuge)

# In dieser Lehrveranstaltung

- Einführung in Unix
- TCP/IP und Bus-Systeme in der Praxis
- C-Programmierung für Fortgeschrittene
- Einführung in die Echtzeitprogrammierung
- Prüfungsleistung: Projektaufgabe  
Eingebettetes System eigener Wahl zum Laufen bringen

→ **Projektaufgabe überlegen**

Weitere Ideen:

- Exkursion: TFC – Simulatoren und Technik GmbH, Velbert-Nierenhof
- Einführung in die GUI-Programmierung
- Web-Interfaces



Änderungen  
vorbehalten

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

- 1.1 Was sind eingebettete Systeme?
- 1.2 Vertiefung Systemtechnik
- 1.3 In dieser Lehrveranstaltung

## 2 Einführung in Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

...



Änderungen  
vorbehalten

## 2 Einführung in Unix

### 2.1 Grundkonzepte

- 1965** Vorgänger: Multics (Multiplexed Information and Computing Service)  
„überladen“
- 1970** Unix: Einfachheit als Grundkonzept
- 1972** Umstellung auf neu entwickelte Programmiersprache C
- 1975** AT&T: Unix inkl. Quelltext für Universitäten
- 1977** Berkeley Software Distribution (BSD)
- 1983** GNU-Projekt
- 1987** Minix
- 1991** Linux
- 1993** FreeBSD, NetBSD
- 1994** OpenBSD
- 2000** Darwin (Mac OS X, BSD-basiert)
- 2008** Android (Linux-basiert)

## 2 Einführung in Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Hauptprogramm  
= „normale“ Funktion

```
int main (int argc, char **argv)
{
    printf ("Hello, _world!\n");
    return 0;
}
```

Unix: übergeordnetes Verzeichnis = „normales“ Verzeichnis

```
cassini/home/peter/foo> ls -la
insgesamt 24
drwxr-xr-x  2 peter peter  4096 Okt  6 13:30 .
drwxr-xr-x 172 peter peter 20480 Okt  6 13:30 ..
cassini/home/peter/foo> cd ..
cassini/home/peter>
```

## 2 Einführung in Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Bibliotheken

z. B.: `printf()` = „normale“ Funktion  
aus eine Bibliothek (`libc`)

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
| xargs grep -l "PBM-Datei"  
./2014ws/ainf/20150130.0/ainf-klausur-20150130.tex  
./2016ws/hp/20170920.0/klausur.tex  
./2016ws/hp/20170206.0/klausur.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
./2012ws/klausuren-gerwinski/rarch-klausur-20120322.tex  
./2013ws/ainf/20140918.0/ainf-klausur-20140918.tex  
./2017ws/hp/20180213.k1/klausur.tex  
./2017ws/hp/20180205/klausur.tex  
./2015ws/ainf/20160913/ainf-klausur-20160913.tex
```

## 2.2 Die Kommandozeile: Grundlagen

- Programm aufrufen: Namen eingeben, z. B.: `ls`
- Optionen: `ls -l`
- Lange Optionen (GNU-Konvention): `ls --help`
- Text schreiben: `echo "Hello, world!"`
- (String-)Variable setzen: `FOO=bar`
- Variable abrufen: `echo $FOO`

## 2.2 Die Kommandozeile: Grundlagen

- Programm aufrufen: Namen eingeben, z. B.: `ls`
- Optionen: `ls -l`
- Lange Optionen (GNU-Konvention): `ls --help`
- Text schreiben: `echo "Hello, world!"`
- (String-)Variable setzen: `FOO=bar`
- Variable abrufen: `echo $FOO`

```
cassini/home/peter/bo> FOO=ls
cassini/home/peter/bo> echo $FOO
ls
cassini/home/peter/bo> $FOO
2011ws  2012ws  2013ws  doc          misc  projekte
2012ss  2013ss  briefe  material    orga
cassini/home/peter/bo>
```



## 2.2 Die Kommandozeile: Grundlagen

- Befehl zurückholen: Pfeiltasten  $\uparrow$ ,  $\downarrow$
- Befehl bearbeiten: Pfeiltasten  $\leftarrow$ ,  $\rightarrow$  usw.
- Befehl vervollständigen: TAB
- Befehl rückwärts suchen: Ctrl+R
- Bildschirm löschen: Ctrl+L
- Befehl abbrechen: Ctrl+C

## 2.2 Die Kommandozeile: Grundlagen

- Befehl zurückholen: Pfeiltasten  $\uparrow$ ,  $\downarrow$
- Befehl bearbeiten: Pfeiltasten  $\leftarrow$ ,  $\rightarrow$  usw.
- Befehl vervollständigen: TAB
- Befehl rückwärts suchen: Ctrl+R
- Bildschirm löschen: Ctrl+L
- Befehl abbrechen: Ctrl+C
  
- Hilfe-Option: `ls --help`
- Unix-Handbuch – *manual*: `man ls`  
(Beenden mit `q`)

## 2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

**MS-DOS:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen  
*oder* sich im aktuellen Verzeichnis befinden.

**Unix:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen.  
—→ Vermeiden von Ausnahmen

## 2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

**MS-DOS:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen  
oder sich im aktuellen Verzeichnis befinden.

**Unix:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen.

—> Vermeiden von Ausnahmen

Das aktuelle Verzeichnis ( `.` ) *kann* im `PATH` stehen,  
muß dies aber nicht –  
und sollte es aus Sicherheitsgründen auch nicht.

## 2.3 Dateisysteme

- Dateien listen: `ls`  
langes Listenformat: `ls -l`  
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`

## 2.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`  
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

## 2.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`  
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```

## 2.3 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after 'foo/test.txt'
Try 'cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`



# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

- 1.1 Was sind eingebettete Systeme?
- 1.2 Vertiefung Systemtechnik
- 1.3 In dieser Lehrveranstaltung

## 2 Einführung in Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

...



Änderungen  
vorbehalten

# Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

18. Oktober 2018

# Was sind eingebettete Systeme?

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

[https://de.wikipedia.org/wiki/Eingebettetes\\_System](https://de.wikipedia.org/wiki/Eingebettetes_System)

- keine Aussage über die Größe
- in der Praxis: so klein wie möglich, z. B.:
  - mehrere Rechnerschränke
  - Industrie-PC
  - Einplatinencomputer
  - Mikro-Controller
- Bedienung über spezielle Ein-/Ausgabegeräte (z. B. Flugsimulator, Telefon)
- Wartung über speziellen Administrator-Zugang
  - Bus-Schnittstelle (RS-232, CAN-BUS)
  - Netzwerk (TCP/IP, Ethernet oder WLAN)
- Programmierung oft außerhalb des Systems (Cross-Entwicklungswerkzeuge)

# In dieser Lehrveranstaltung

- Einführung in Unix
- TCP/IP und Bus-Systeme in der Praxis
- C-Programmierung für Fortgeschrittene
- Einführung in die Echtzeitprogrammierung
- Prüfungsleistung: Projektaufgabe  
Eingebettetes System eigener Wahl zum Laufen bringen

→ Projektaufgabe überlegen

Weitere Ideen:

- Exkursion: TFC – Simulatoren und Technik GmbH, Velbert-Nierenhof
- Einführung in die GUI-Programmierung
- Web-Interfaces



Änderungen  
vorbehalten

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

- 1.1 Was sind eingebettete Systeme?
- 1.2 Vertiefung Systemtechnik
- 1.3 In dieser Lehrveranstaltung

## 2 Einführung in Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

...



Änderungen  
vorbehalten

## 2 Einführung in Unix

### 2.1 Grundkonzepte

- 1965** Vorgänger: Multics (Multiplexed Information and Computing Service)  
„überladen“
- 1970** Unix: Einfachheit als Grundkonzept
- 1972** Umstellung auf neu entwickelte Programmiersprache C
- 1975** AT&T: Unix inkl. Quelltext für Universitäten
- 1977** Berkeley Software Distribution (BSD)
- 1983** GNU-Projekt
- 1987** Minix
- 1991** Linux
- 1993** FreeBSD, NetBSD
- 1994** OpenBSD
- 2000** Darwin (Mac OS X, BSD-basiert)
- 2008** Android (Linux-basiert)

## 2 Einführung in Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Hauptprogramm  
= „normale“ Funktion

```
int main (int argc, char **argv)
{
    printf ("Hello, _world!\n");
    return 0;
}
```

Unix: übergeordnetes Verzeichnis = „normales“ Verzeichnis

```
cassini/home/peter/foo> ls -la
insgesamt 24
drwxr-xr-x  2 peter peter  4096 Okt  6 13:30 .
drwxr-xr-x 172 peter peter 20480 Okt  6 13:30 ..
cassini/home/peter/foo> cd ..
cassini/home/peter>
```

## 2 Einführung in Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Bibliotheken

z. B.: `printf()` = „normale“ Funktion  
aus eine Bibliothek (`libc`)

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
| xargs grep -l "PBM-Datei"  
./2014ws/ainf/20150130.0/ainf-klausur-20150130.tex  
./2016ws/hp/20170920.0/klausur.tex  
./2016ws/hp/20170206.0/klausur.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
./2012ws/klausuren-gerwinski/rarch-klausur-20120322.tex  
./2013ws/ainf/20140918.0/ainf-klausur-20140918.tex  
./2017ws/hp/20180213.k1/klausur.tex  
./2017ws/hp/20180205/klausur.tex  
./2015ws/ainf/20160913/ainf-klausur-20160913.tex
```



## 2.2 Die Kommandozeile: Grundlagen

- Programm aufrufen: Namen eingeben, z. B.: `ls`
- Optionen: `ls -l`
- Lange Optionen (GNU-Konvention): `ls --help`
- Text schreiben: `echo "Hello, world!"`
- (String-)Variable setzen: `FOO=bar`
- Variable abrufen: `echo $FOO`

```
cassini/home/peter/bo> FOO=ls
cassini/home/peter/bo> echo $FOO
ls
cassini/home/peter/bo> $FOO
2011ws  2012ws  2013ws  doc          misc  projekte
2012ss  2013ss  briefe  material    orga
cassini/home/peter/bo>
```

## 2.2 Die Kommandozeile: Grundlagen

- Befehl zurückholen: Pfeiltasten  $\uparrow$ ,  $\downarrow$
- Befehl bearbeiten: Pfeiltasten  $\leftarrow$ ,  $\rightarrow$  usw.
- Befehl vervollständigen: TAB
- Befehl rückwärts suchen: Ctrl+R
- Bildschirm löschen: Ctrl+L
- Befehl abbrechen: Ctrl+C
  
- Hilfe-Option: `ls --help`
- Unix-Handbuch – *manual*: `man ls`  
(Beenden mit `q`)

## 2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

**MS-DOS:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen  
oder sich im aktuellen Verzeichnis befinden.

**Unix:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen.

—→ Vermeiden von Ausnahmen

Das aktuelle Verzeichnis ( `.` ) *kann* im `PATH` stehen,  
muß dies aber nicht.

## 2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

**MS-DOS:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen  
oder sich im aktuellen Verzeichnis befinden.

**Unix:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen.

—> Vermeiden von Ausnahmen

Das aktuelle Verzeichnis ( `.` ) *kann* im `PATH` stehen,  
muß dies aber nicht –  
und sollte es aus Sicherheitsgründen auch nicht.

## 2.3 Dateisysteme

- Dateien listen: `ls`  
langes Listenformat: `ls -l`  
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`

## 2.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`  
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```

## 2.3 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after 'foo/test.txt'
Try 'cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`

## 2.3 Dateisysteme

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/usb1/
cassini/home/peter> mount /media/usb1
cassini/home/peter> ls /media/usb1/
es-20181018.pdf  hello.c  hexapode  KIS-Bericht.pdf
cassini/home/peter> umount /media/usb1
cassini/home/peter> ls /media/usb1/
cassini/home/peter>
```



## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```

## 2.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```



Benutzer (u – *user*) darf lesen und schreiben


## 2.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```



Gruppe (g – *group*) darf lesen

## 2.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```



alle anderen (o – *other*) dürfen lesen

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```

- Zugriffsrechte ändern:

`chmod o-r orbit-1x.c` – Lesezugriff entziehen

`chmod g+w orbit-1x.c` – Schreibzugriff gewähren

`chmod 640 orbit-1x.c` – auf `-rw-r-----` setzen

## 2.3 Dateisysteme

- *Zugriffsrechte*


```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```

- Zugriffsrechte ändern:

`chmod o-r orbit-1x.c` – Lesezugriff entziehen

`chmod g+w orbit-1x.c` – Schreibzugriff gewähren

`chmod 640 orbit-1x.c` – auf `-rw-r-----` setzen

  
6     4     0

## 2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2013ws/systech/20131008.0> cat test
ls -l systech-20131008.*
cassini/home/peter/bo/2013ws/systech/20131008.0> chmod +x test
cassini/home/peter/bo/2013ws/systech/20131008.0> ls -l test
-rwxr-xr-x 1 peter peter  25 Okt 6 16:45 test
cassini/home/peter/bo/2013ws/systech/20131008.0> ./test
-rw-r--r-- 1 peter peter 4120 Okt 6 16:44 systech-20131008.aux
...
```

## 2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2013ws/systech/20131008.0> cat test
ls -l systech-20131008.*
cassini/home/peter/bo/2013ws/systech/20131008.0> chmod +x test
cassini/home/peter/bo/2013ws/systech/20131008.0> ls -l test
-rwxr-xr-x 1 peter peter  25 Okt 6 16:45 test
cassini/home/peter/bo/2013ws/systech/20131008.0> ./test
-rw-r--r-- 1 peter peter 4120 Okt 6 16:44 systech-20131008.aux
...
```

- ausführbare Textdateien: *Skripte*



## 2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2013ws/systech/20131008.0> cat test
ls -l systech-20131008.*
cassini/home/peter/bo/2013ws/systech/20131008.0> chmod +x test
cassini/home/peter/bo/2013ws/systech/20131008.0> ls -l test
-rwxr-xr-x 1 peter peter  25 Okt 6 16:45 test
cassini/home/peter/bo/2013ws/systech/20131008.0> ./test
-rw-r--r-- 1 peter peter 4120 Okt 6 16:44 systech-20131008.aux
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen  
(ohne spezielle Kennung): Shell-Skript

## 2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2013ws/systech/20131008.0> cat test
ls -l systech-20131008.*
cassini/home/peter/bo/2013ws/systech/20131008.0> chmod +x test
cassini/home/peter/bo/2013ws/systech/20131008.0> ls -l test
-rwxr-xr-x 1 peter peter  25 Okt 6 16:45 test
cassini/home/peter/bo/2013ws/systech/20131008.0> ./test
-rw-r--r-- 1 peter peter 4120 Okt 6 16:44 systech-20131008.aux
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen  
(ohne spezielle Kennung): Shell-Skript

Kennung: 1. Zeile enthält `#!` und den Interpreter,  
z. B. `#!/bin/bash`

## 2.3 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`

(Richtung: wie bei `cp`)

Beispiel: `ln -s opengel-magic-double.c opengl-magic.c`

- *Harte Verknüpfungen – hard links*

Dieselben Daten auf dem Datenträger

sind unter mehreren Namen verfügbar.

→ Wenn man einen löscht, sind die Daten noch da.

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

## 2 Einführung in Unix

### 2.1 Grundkonzepte

### 2.2 Die Kommandozeile: Grundlagen

### 2.3 Dateisysteme

### 2.4 Ein- und Ausgabeströme

### 2.5 Pipes

### 2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

...



Änderungen  
vorbehalten

# Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

25. Oktober 2018

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

## 2 Einführung in Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

...



Änderungen  
vorbehalten

## 2.3 Dateisysteme

- Dateien listen: `ls`  
langes Listenformat: `ls -l`  
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`

## 2.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`  
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```



## 2.3 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after 'foo/test.txt'
Try 'cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`

## 2.3 Dateisysteme

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/usb1/  
cassini/home/peter> mount /media/usb1  
cassini/home/peter> ls /media/usb1/  
es-20181018.pdf  hello.c  hexapode  KIS-Bericht.pdf  
cassini/home/peter> umount /media/usb1  
cassini/home/peter> ls /media/usb1/  
cassini/home/peter>
```

## 2.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```



Benutzer (u – *user*) darf lesen und schreiben


## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```



Gruppe (g – *group*) darf lesen

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter    1539 Nov 29  2012 orbit-x1.c
```



alle anderen (o – *other*) dürfen lesen

## 2.3 Dateisysteme

- *Zugriffsrechte*


```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter      1539 Nov 29  2012 orbit-x1.c
```

- Zugriffsrechte ändern:

`chmod o-r orbit-1x.c` – Lesezugriff entziehen

`chmod g+w orbit-1x.c` – Schreibzugriff gewähren

`chmod 640 orbit-1x.c` – auf `-rw-r-----` setzen

  
6     4     0

## 2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2013ws/systech/20131008.0> cat test
ls -l systech-20131008.*
cassini/home/peter/bo/2013ws/systech/20131008.0> chmod +x test
cassini/home/peter/bo/2013ws/systech/20131008.0> ls -l test
-rwxr-xr-x 1 peter peter  25 Okt 6 16:45 test
cassini/home/peter/bo/2013ws/systech/20131008.0> ./test
-rw-r--r-- 1 peter peter 4120 Okt 6 16:44 systech-20131008.aux
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen  
(ohne spezielle Kennung): Shell-Skript

Kennung: 1. Zeile enthält `#!` und den Interpreter,  
z. B. `#!/bin/bash`

## 2.3 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`  
(Richtung: wie bei `cp`)

Beispiel: `ln -s opengel-magic-double.c opengl-magic.c`

- *Harte Verknüpfungen – hard links*

Dieselben Daten auf dem Datenträger  
sind unter mehreren Namen verfügbar.

→ Wenn man einen löscht, sind die Daten noch da.

```
cassini/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r--  1 peter peter    1539 Nov 29  2012 orbit-x1.c
drwxr-x---  2 peter peter    4096 Nov 30  2012 test
```



Anzahl der („harten“) Links auf diese Datei / dieses Verzeichnis



## 2.3 Dateisysteme

- `grep`: Dateien durchsuchen

```
cassini/.../ainf/20131031.0> grep printf *.c  
philosophy.c:  printf ("The answer is %d.\n", answer ());
```

## 2.3 Dateisysteme

- **find**: Dateien anhand ihrer Eigenschaften suchen

```
$ find . -name "*orbit-x*"
./20131031.0/orbit-x.c
./20131031.0/orbit-x1.c
./20131031.0/orbit-x
./20131107.0/orbit-x.c
./20131107.0/orbit-x1.c
./20131107.0/orbit-x
$ find . -name "*orbit-x*" -perm /u+x
./20131031.0/orbit-x
./20131107.0/orbit-x
$ find . -name "*orbit-x*" -perm /u+x -exec ls -l {} \;
-rwxr-xr-x 1 peter peter 15831 Okt 31 13:19 ./20131031.0/orbit-x
-rwxr-xr-x 1 peter peter 15831 Okt 31 13:19 ./20131107.0/orbit-x
```

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```

## 2.4 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

## 2.4 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```

## 2.5 Pipes

Standard-Ausgabe von Programm A  
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

## 2.5 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```



## 2.5 Pipes

- **grep**: Standard-Eingabe durchsuchen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v fig
```

```
logo-hochschule-bochum-cvh-text.pdf
```

```
logo-hochschule-bochum.pdf
```

```
NPN_transistor_basic_operation.pdf
```

```
rtech-20131002.pdf
```

```
$ ls -l $(ls *.pdf | grep -v fig)
```

```
-rw-r--r-- 1 peter peter 14488 Sep  2 21:02 logo-hochschule-bochum-cvh-text.pdf
```

```
-rw-r--r-- 1 peter peter 31581 Dez 26 2011 logo-hochschule-bochum.pdf
```

```
-rw-r--r-- 1 peter peter 8538 Okt  2 2012 NPN_transistor_basic_operation.pdf
```

```
-rw-r--r-- 1 peter peter 6753149 Okt  1 22:59 rtech-20131002.pdf
```

## 2.6 Verzweigungen und Schleifen

```
$ if grep Pipes test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

## 2 Einführung in Unix

### 2.1 Grundkonzepte

### 2.2 Die Kommandozeile: Grundlagen

### 2.3 Dateisysteme

### 2.4 Ein- und Ausgabeströme

### 2.5 Pipes

### 2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

...

## Aufgabe

Schreiben Sie ein Shell-Skript, das regelmäßig aktualisierte Informationen aus dem WWW für Sie herunterlädt und speichert.



Änderungen  
vorbehalten

# Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

8. November 2018

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

## 2 Einführung in Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

...



Änderungen  
vorbehalten

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```

## 2.4 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

## 2.4 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```



## 2.5 Pipes

Standard-Ausgabe von Programm A  
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

## 2.5 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```

## 2.5 Pipes

- **grep**: Standard-Eingabe durchsuchen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v fig
```

```
logo-hochschule-bochum-cvh-text.pdf
```

```
logo-hochschule-bochum.pdf
```

```
NPN_transistor_basic_operation.pdf
```

```
rtech-20131002.pdf
```

```
$ ls -l $(ls *.pdf | grep -v fig)
```

```
-rw-r--r-- 1 peter peter 14488 Sep  2 21:02 logo-hochschule-bochum-cvh-text.pdf
```

```
-rw-r--r-- 1 peter peter 31581 Dez 26 2011 logo-hochschule-bochum.pdf
```

```
-rw-r--r-- 1 peter peter 8538 Okt  2 2012 NPN_transistor_basic_operation.pdf
```

```
-rw-r--r-- 1 peter peter 6753149 Okt  1 22:59 rtech-20131002.pdf
```

## 2.6 Verzweigungen und Schleifen

```
$ if grep Pipes test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

## 2 Einführung in Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 TCP/IP in der Praxis

- 3.0 Vorbereitungen
- 3.1 IP-Adressen
- 3.2 TCP- und UDP-Ports
- 3.3 TCP-Protokolle
- 3.4 Routing
- 3.5 Netzwerkanalyse

...

...

## Aufgabe

Schreiben Sie ein Shell-Skript, das regelmäßig aktualisierte Informationen aus dem WWW für Sie herunterlädt und speichert.



Änderungen  
vorbehalten

## 3.0 Vorbereitungen

- Verkabelung: Twisted-Pair-Kabel, Switches
- Automatismen abschalten

## 3.1 IP-Adressen

- `ip addr` (Linux)  
  `ifconfig` (Unix allgemein)  
  `ipconfig` (MS Windows)
- `ip addr add <Netz>`
- `ip link`
- `ping <IP-Adresse>`

## 3.1 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>



## 3.2 TCP- und UDP-Ports

- `nc <IP> <Port>`  
Verbindung zu Programm  $\langle \text{Port} \rangle$  auf Rechner  $\langle \text{IP} \rangle$  aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`  
auf eingehende Verbindungen warten („lauschen“)
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:  
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

## 3.3 TCP-Protokolle

- **HTTP**

GET / HTTP/1.1

Host: www.hs-bochum.de

(Leerzeile)

## 3.3 TCP-Protokolle

- **HTTP**

GET / HTTP/1.1

Host: www.hs-bochum.de

(Leerzeile)

URL: Schema://Benutzer:Passwort@Rechner:port/Pfad?Query#Fragment

## 3.3 TCP-Protokolle

- **HTTP**

```
GET / HTTP/1.1
```

```
Host: www.hs-bochum.de
```

```
(Leerzeile)
```

- **SMTP**

```
HELO cassini
```

```
MAIL FROM: <example@example.com>
```

```
RCPT TO: <beispiel@example.de>
```

```
(E-Mail-Header – Teil der Nutzdaten)
```

```
From: Eddie Example <example@example.com>
```

```
To: Bert Beispiel <beispiel@example.de>
```

```
Subject: Hello, world!
```

```
(Leerzeile)
```

```
Hi, there!
```

```
.
```

- Protokolle „mal eben“ selbst schreiben: [inetd](#)

## 3.4 Routing

- `ip route` (Linux)  
`route` (MS-Windows, Unix)  
`netstat -nr` (MacOS)

## 3.5 Netzwerkanalyse

- tcpdump
- wireshark
- ettercap

## 3.6 SSH

- SSH <Rechner>
- -C: Komprimierung
- -L: lokalen Port auf Remote-Port umleiten
- -R: Remote-Port auf lokalen Port umleiten

## 3.7 X11

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`: X11-Forwarding



## 3.8 GNU screen

- Text-Bildschirm und Eingabegeräte über's Netz
- `Ctrl+A c`: neues Fenster (create)
- `Ctrl+A Leertaste`: nächstes Fenster
- `Ctrl+A 3`: Fenster Nr. 3
- `Ctrl+A ESC`: hochscrollen, suchen, markieren (copy)
- `Ctrl+A Ctrl+]`: einfügen (paste)
- `Ctrl+A d`: von `screen` ablösen (detach)
- `screen -x`: an laufenden `screen` andocken
- ähnliche Funktionalität für Grafik: `x2go`

## 3.9 Programmierung

- `server.c`: auf Port 1234 lauschen, „Hello, world!“ senden
- `client.c`: „Hello, world!“ an Rechner `localhost`, Port 1234 senden

# Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

22. November 2018

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

- 1 Einführung**
- 2 Einführung in Unix**
- 3 TCP/IP in der Praxis**
  - 3.0 Vorbereitungen
  - 3.1 IP-Adressen
  - 3.2 TCP- und UDP-Ports
  - 3.3 TCP-Protokolle
  - 3.4 Routing
  - 3.5 Netzwerkanalyse
  - 3.6 SSH
  - 3.7 X11
  - 3.8 GNU screen
  - 3.9 Programmierung
- 4 Bus-Systeme**
- 6 Echtzeit**

...



Änderungen  
vorbehalten

## 3.0 Vorbereitungen

- Verkabelung: Twisted-Pair-Kabel, Switches
- Automatismen abschalten

## 3.1 IP-Adressen

- IP-Adresse zuweisen:

`ip addr` (Linux)

`ifconfig` (Unix allgemein)

`ipconfig` (MS Windows)

- Beispiel:

`ip addr add 192.168.42.197/24`

- Verbindung prüfen:

`ping <IP-Adresse>`

## 3.1 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

## 3.2 TCP- und UDP-Ports

- `nc <IP> <Port>`  
Verbindung zu Programm  $\langle \text{Port} \rangle$  auf Rechner  $\langle \text{IP} \rangle$  aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`  
auf eingehende Verbindungen warten („lauschen“)
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:  
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse



## 3.3 TCP-Protokolle

- HTTP

```
$ nc -C <Rechner> 80  
GET / HTTP/1.1  
Host: www.hs-bochum.de  
(Leerzeile)
```

## 3.3 TCP-Protokolle

- HTTP

```
$ nc -C <Rechner> 80
```

```
GET / HTTP/1.1
```

```
Host: www.hs-bochum.de
```

```
(Leerzeile)
```

URL: Schema://Benutzer:Passwort@Rechner:port/Pfad?Query#Fragment

## 3.3 TCP-Protokolle

- **HTTP**

```
$ nc -C <Rechner> 80
GET / HTTP/1.1
Host: www.hs-bochum.de
(Leerzeile)
```

- **SMTP**

```
HELO cassini
MAIL FROM: <example@example.com>
RCPT TO: <beispiel@example.de>
(E-Mail-Header – Teil der Nutzdaten)
From: Eddie Example <example@example.com>
To: Bert Beispiel <beispiel@example.de>
Subject: Hello, world!
(Leerzeile)
Hi, there!
.
```

- Protokolle „mal eben“ selbst schreiben: [inetd](#)

## 3.4 Routing

- `ip route` (Linux)  
`route` (MS-Windows, Unix)  
`netstat -nr` (MacOS)

192.168.42.x 21 BW gateway-192.168.7.27

80 Dimi

240 Skymaster

25 Heisenberg

„default“ = 0.0.0.0/0

192.168.42.66

192.168.42...

192.168.42.21 — ... — 195.37.15.82

↑  
iptables -t nat -A POSTROUTING  
-o <interface> -j MASQUERADE

Internet

Ⓢ Network Address  
Translation

7.27 <sup>USB-</sup> 192.168.7.23  
<sub>Netz-  
werk</sub> EL Burto

7 192.168.43.23

...43.7 ...43.707

141.1.1.1

route add 192.168.42.0/24

gw ...

## 3.5 Netzwerkanalyse

- tcpdump
- wireshark
- ettercap

## 3.6 SSH

- SSH <Rechner>
- -C: Komprimierung
- -L: lokalen Port auf Remote-Port umleiten
- -R: Remote-Port auf lokalen Port umleiten

## 3.7 X11

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`: X11-Forwarding

## 3.8 GNU screen

- Text-Bildschirm und Eingabegeräte über's Netz
- `Ctrl+A c`: neues Fenster (create)
- `Ctrl+A Leertaste`: nächstes Fenster
- `Ctrl+A 3`: Fenster Nr. 3
- `Ctrl+A ESC`: hochscrollen, suchen, markieren (copy)
- `Ctrl+A Ctrl+]`: einfügen (paste)
- `Ctrl+A d`: von `screen` ablösen (detach)
- `screen -x`: an laufenden `screen` andocken
- ähnliche Funktionalität für Grafik: `x2go`



## 3.9 Programmierung

- `tcpip-server-0.c`: auf Port 1234 lauschen, „Hello, world!“ senden
- `tcpip-client-0.c`: Webseite <http://ngc224.gerwinski.de> abrufen

## 4 Bus-Systeme

### 4.1 Was sind Bus-Systeme?

*Ein Bus ist ein System zur Datenübertragung zwischen mehreren Teilnehmern über einen gemeinsamen Übertragungsweg.*

[https://de.wikipedia.org/wiki/Bus\\_\(Datenverarbeitung\)](https://de.wikipedia.org/wiki/Bus_(Datenverarbeitung))

Beispiele:

- Computer kommuniziert mit Peripherie
- Computer kommunizieren (direkt) miteinander
- Prozessor kommuniziert mit externem Speicher
- Teile eines Prozessors kommunizieren miteinander

# 4 Bus-Systeme

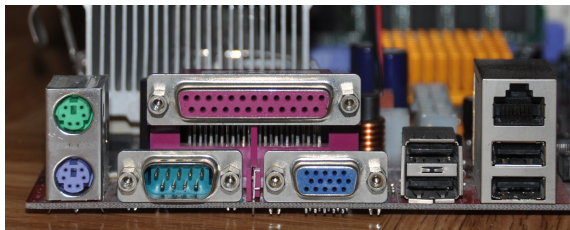
## 4.1 Was sind Bus-Systeme?

Standard-Computer:

- Einsteckkarten: PCI (und Vorgänger)
- Festplatten: SATA (und Vorgänger)
- USB, FireWire, ...
- Ethernet, CAN-Bus, ...
- WLAN, BlueTooth, IR, ...
- PS/2, RS-232, Centronics

Minimal-Hardware:

- RS-232
- I<sup>2</sup>C (TWI)
- SPI



# 4 Bus-Systeme

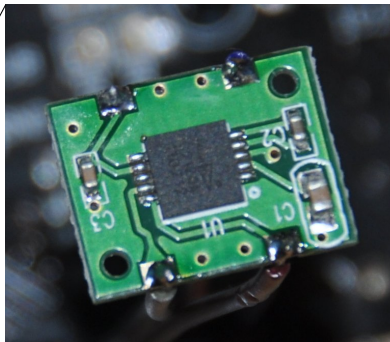
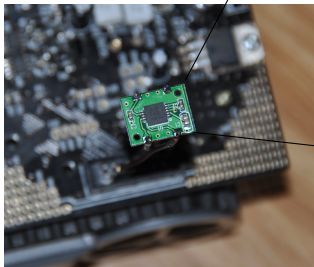
## 4.1 Was sind Bus-Systeme?

Standard-Computer:

- Einsteckkarten: PCI (und Vorgänger)
- Festplatten: SATA (und Vorgänger)
- USB, FireWire, ...
- Ethernet, CAN-Bus, ...
- WLAN, BlueTooth, IR, ...
- PS/2, RS-232, Centronics

Minimal-Hardware:

- RS-232
- I<sup>2</sup>C (TWI)
- SPI



## 4 Bus-Systeme

### 4.1 Was sind Bus-Systeme?

<i>seriell</i>	jedes Bit einzeln übertragen
<i>parallel</i>	mehrere Bits gleichzeitig
<i>synchron</i>	Abgleich über Steuerleitung: <i>Takt</i>
<i>asynchron</i>	Abgleich über Zeitvereinbarungen
<i>Punkt-zu-Punkt</i>	genau zwei Teilnehmer
<i>busfähig</i>	mehrere Teilnehmer, mit <i>Adressierung</i>

- I<sup>2</sup>C: seriell, synchron, mit Adressierung
- RS-232: seriell, asynchron, Punkt-zu-Punkt
- RS-485, USB, CAN: seriell, asynchron, mit Adressierung
- SPI: seriell, synchron, Punkt-zu-Punkt oder mit Adressierung

# 4 Bus-Systeme

## 4.2 RS-232

seriell

- *TX*: 1 Leitung für Daten
- *RX*: ggf. 1 Leitung für Daten in der anderen Richtung
- *GND*: gemeinsame *Masse*
- evtl. zusätzliche Steuerleitungen

asynchron

- *keine* Taktleitung für Abgleich, wann Daten anliegen
- Stattdessen: Abgleich über Zeitvereinbarungen

→ Jeder Teilnehmer braucht eine eigene Zeitbasis.

Punkt-zu-Punkt

- nur 2 Teilnehmer vorgesehen

## 4.2 RS-232

Synchronisation

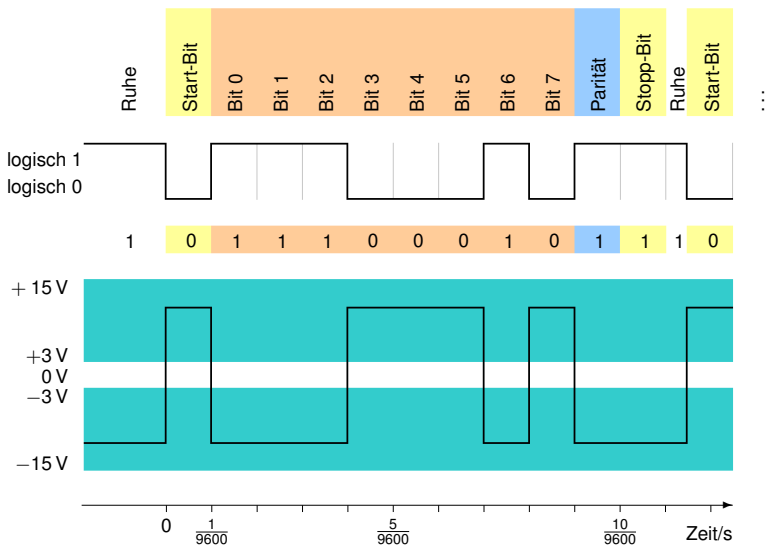
Daten

Check

9600 Baud, 8 Daten-Bits, ungerade Parität, 1 Stopp-Bit

Beispiel-Daten: ASCII „G“ = 71 = 0100 0111 binär

Übertragung der Daten von rechts (Bit 0) nach links (Bit 7)



# 4 Bus-Systeme

## 4.3 I<sup>2</sup>C (TWI)

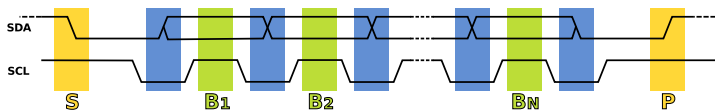
I<sup>2</sup>C = Inter-IC; TWI = Two-Wire-Interface

seriell

- *SDA*: 1 Leitung für Daten (in beiden Richtungen)
- *SCL*: Taktleitung (Clock)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung



busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- erstes gesendetes Byte: *Adresse* des Teilnehmers
- 2 Adressen pro Teilnehmer: Lesen/Schreiben



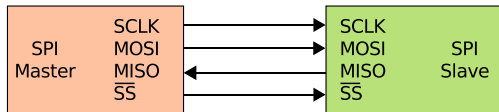
# 4 Bus-Systeme

## 4.4 SPI

### Serial Peripheral Interface

seriell

- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät



synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt

# 4 Bus-Systeme

## 4.4 SPI

### Serial Peripheral Interface

seriell

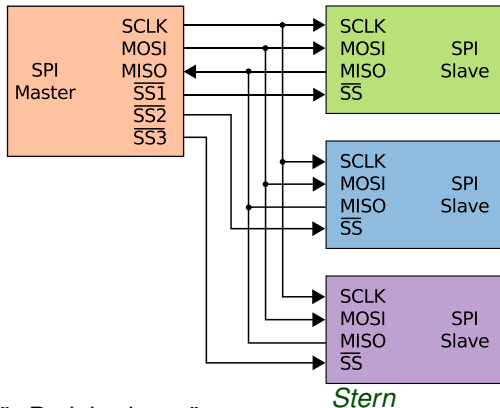
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



# 4 Bus-Systeme

## 4.4 SPI

Serial Peripheral Interface

seriell

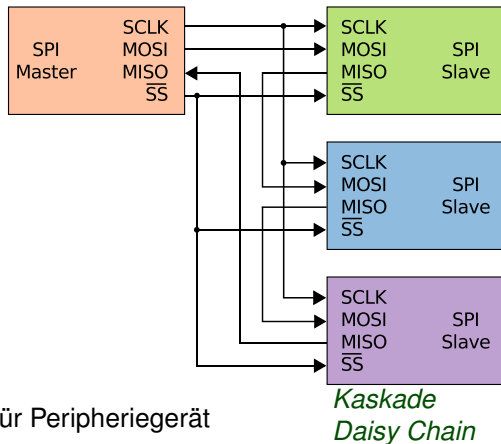
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



# 4 Bus-Systeme

## 4.4 SPI

Serial Peripheral Interface

seriell

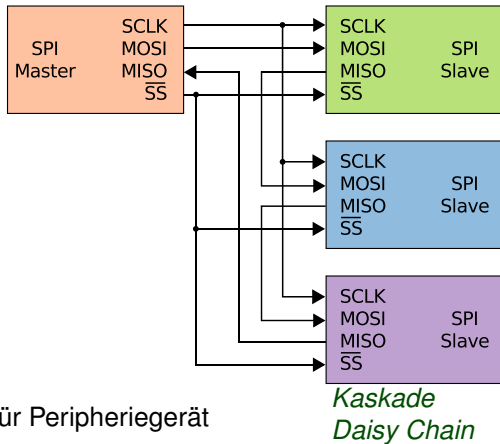
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



Slave gibt MOSI-Input um 1 Takt verzögert an MISO aus  $\rightarrow$  Master setzt „im richtigen Moment“  $\overline{SS}$

# Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

## 1 Einführung

## 2 Einführung in Unix

## 3 TCP/IP in der Praxis

## 4 Bus-Systeme

### 5.1 Was sind Bus-Systeme?

### 5.2 RS-232

### 5.3 I<sup>2</sup>C (TWI)

### 5.4 SPI

### 5.5 PWM

## 6 Echtzeit

### 5.1 Was ist Echtzeit?

### 5.2 Echtzeitprogrammierung

### 5.3 Multitasking

### 5.4 Ressourcen

### 5.5 Prioritäten

...



Änderungen  
vorbehalten