

Campus
Velbert/Heiligenhaus

Vertiefung Systemtechnik Eingebettete Systeme

Hochschule Bochum
Bochum University
of Applied Sciences



Prof. Dr. rer. nat. Peter Gerwinski
12. Oktober 2017

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

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

- 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

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

Was sind eingebettete Systeme?

Der Ausch
Rechner
(eingebe

- keine
- in de



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

- 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

- 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

- 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

- 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)

Vertiefung Systemtechnik

... für Fortgeschrittene



Vertiefung Systemtechnik

... für Fortgeschrittene

Vertiefung Systemtechnik

The screenshot shows the German Wikipedia page for 'Systemtechnik'. At the top, there are navigation tabs for 'Artikel', 'Diskussion', 'Lesen', 'Bearbeiten', and 'Versionsgeschichte'. The main text defines 'Systemtechnik' as a branch of engineering that deals with various construction and connection techniques, often involving the connection of individual modules of a system. Below this, there is a section titled 'Studium' (Study) which lists several disciplines: Mechatronik und Robotik, Photonik, Ingenieurinformatik, Echtzeitsysteme, Elektronik, and Regelungstechnik. The page also includes a sidebar with navigation options like 'Hauptseite', 'Themenportale', and 'Zufälliger Artikel', and a footer with the URL 'http://de.wikipedia.org/wiki/Mikrosystemtechnik'.

Benutzerkonto anlegen Anmelden

Artikel Diskussion Lesen Bearbeiten Versionsgeschichte

Systemtechnik

Als **Systemtechnik** bezeichnet man verschiedene Aufbau- und Verbindungstechniken, aber auch eine Fachrichtung der Ingenieurwissenschaften. Zumeist wird der Begriff in Verbindung mit der [Mikrosystemtechnik](#) genannt. Er bedeutet in der Unterscheidung zu den [Mikrotechnologien](#) die Verbindung verschiedener einzelner Module eines Systems und deren Konzeption.

Studium [[Bearbeiten](#)]

Als Fachrichtung der Ingenieurwissenschaften baut das Studium auf einem fundierten mathematisch-naturwissenschaftlichen Grundwissen auf. Zum Studium gehören alle oder einige der folgenden Disziplinen:

- [Mechatronik und Robotik](#)
- [Photonik](#)
- [Ingenieurinformatik](#)
- [Echtzeitsysteme](#)
- [Elektronik](#)
- [Regelungstechnik](#)

Die Systemtechnik versucht, mit einem ganzheitlichen Ansatz an den Entwurf komplexer Systeme heranzugehen, im Unterschied zu den Spezialisten, die sich auf den Entwurf der Teilsysteme konzentrieren.

http://de.wikipedia.org/wiki/Mikrosystemtechnik

... für Fortgeschrittene



Vertiefung Systemtechnik



Echtzeitsysteme

... für Fortgeschrittene

```
graph TD; A[Geräte aller Art] --> B[Echtzeitsysteme]; B --> C[Vertiefung Systemtechnik]; C --- D[... für Fortgeschrittene]
```

Vertiefung Systemtechnik

Echtzeitsysteme

Geräte aller Art

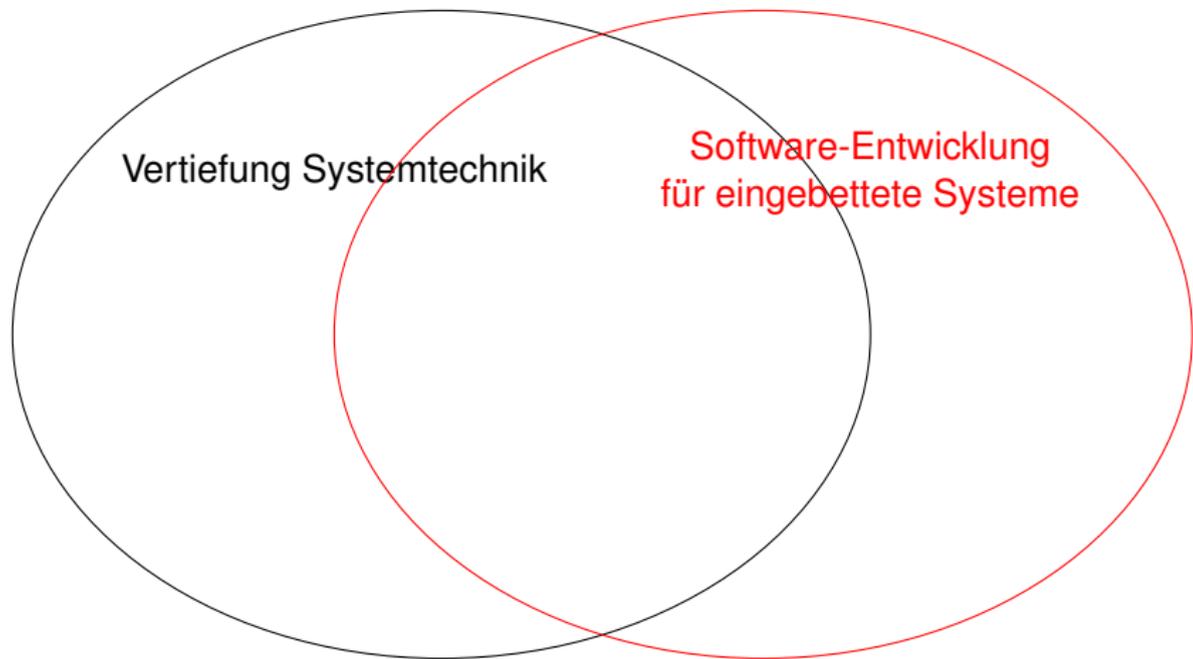
... für Fortgeschrittene
↓

Vertiefung Systemtechnik

Echtzeitsysteme

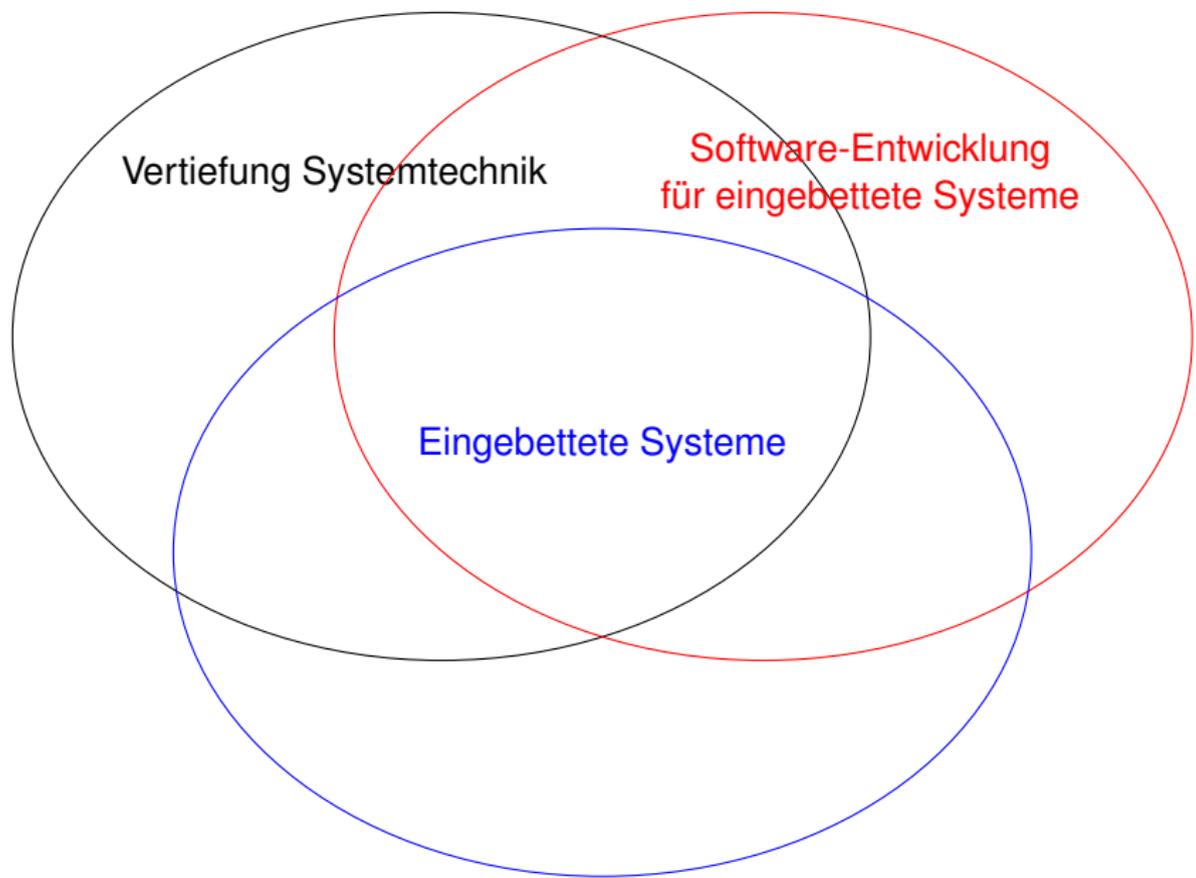
Thema dieser
Lehrveranstaltung:
Echtzeit

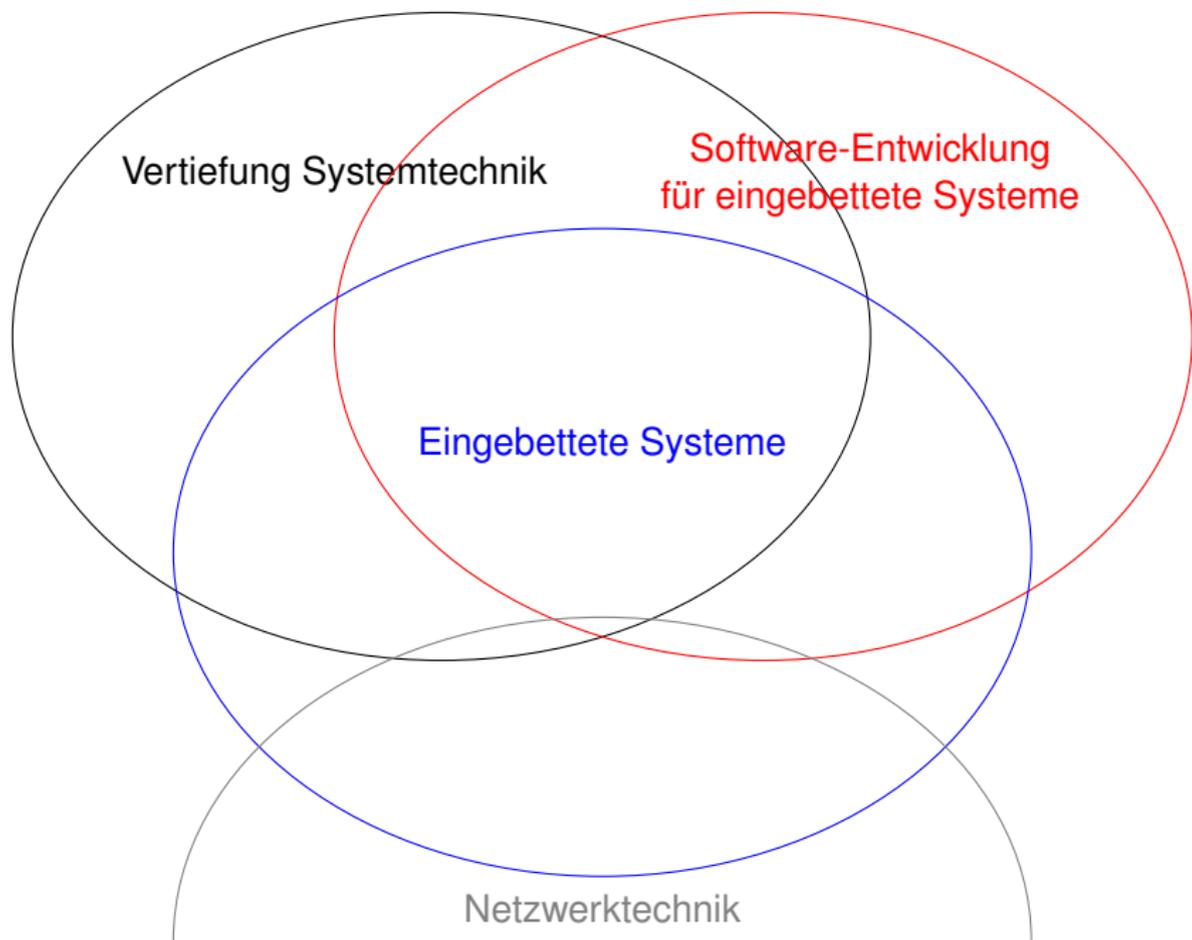
Geräte aller Art



Vertiefung Systemtechnik

Software-Entwicklung
für eingebettete Systeme





In dieser Lehrveranstaltung

- Einführung in Unix
- TCP/IP und Bus-Systeme in der Praxis
- C-Programmierung für Fortgeschrittene
- Theorie und Praxis der Echtzeitprogrammierung
- Prüfungsleistung: Projektaufgabe
Eingebettetes System bzw. Echtzeit-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

Vertiefung Systemtechnik – 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

- Grundfunktionen: `libc`
- 3d-Grafik – Kernfunktion: `libGL`
- 3d-Grafik – Utilities: `libGLU`
- 3d-Grafik – Utility Toolkit: `libglut`
- ...

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
  | grep -v "fig" | less  
./2013ss/net/script/slides/net-probeklausur-20120712.tex  
./2013ss/net/20130924.0/net-klausur-20130924.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
...  
./2012ss/hs/20130318.0/hs-klausur-20130318.tex  
./2012ss/hs/20120715.0/hs-probeklausur-20120715.tex  
./2012ss/hs/20120720.0/hs-klausur-20120720.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 ↑, ↓
- Befehl bearbeiten: Pfeiltasten ←, → 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: `.`

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

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/sd-card/  
cassini/home/peter> mount /media/sd-card  
cassini/home/peter> ls /media/sd-card/  
DCIM  NIKON001.DSC  
cassini/home/peter> umount /media/sd-card  
cassini/home/peter> ls /media/sd-card/  
cassini/home/peter>
```

2.3 Dateisysteme

- *Zugriffsrechte*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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.

(aus Anwendersicht eher selten, daher hier nicht ausführlich)

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-hochschu
```

```
-rw-r--r-- 1 peter peter 31581 Dez 26 2011 logo-hochschu
```

```
-rw-r--r-- 1 peter peter 8538 Okt  2 2012 NPN_transisto
```

```
-rw-r--r-- 1 peter peter 6753149 Okt  1 22:59 rtech-2013100
```

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
```

Vertiefung Systemtechnik Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

19. Oktober 2017

Vertiefung Systemtechnik – 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

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

- 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)

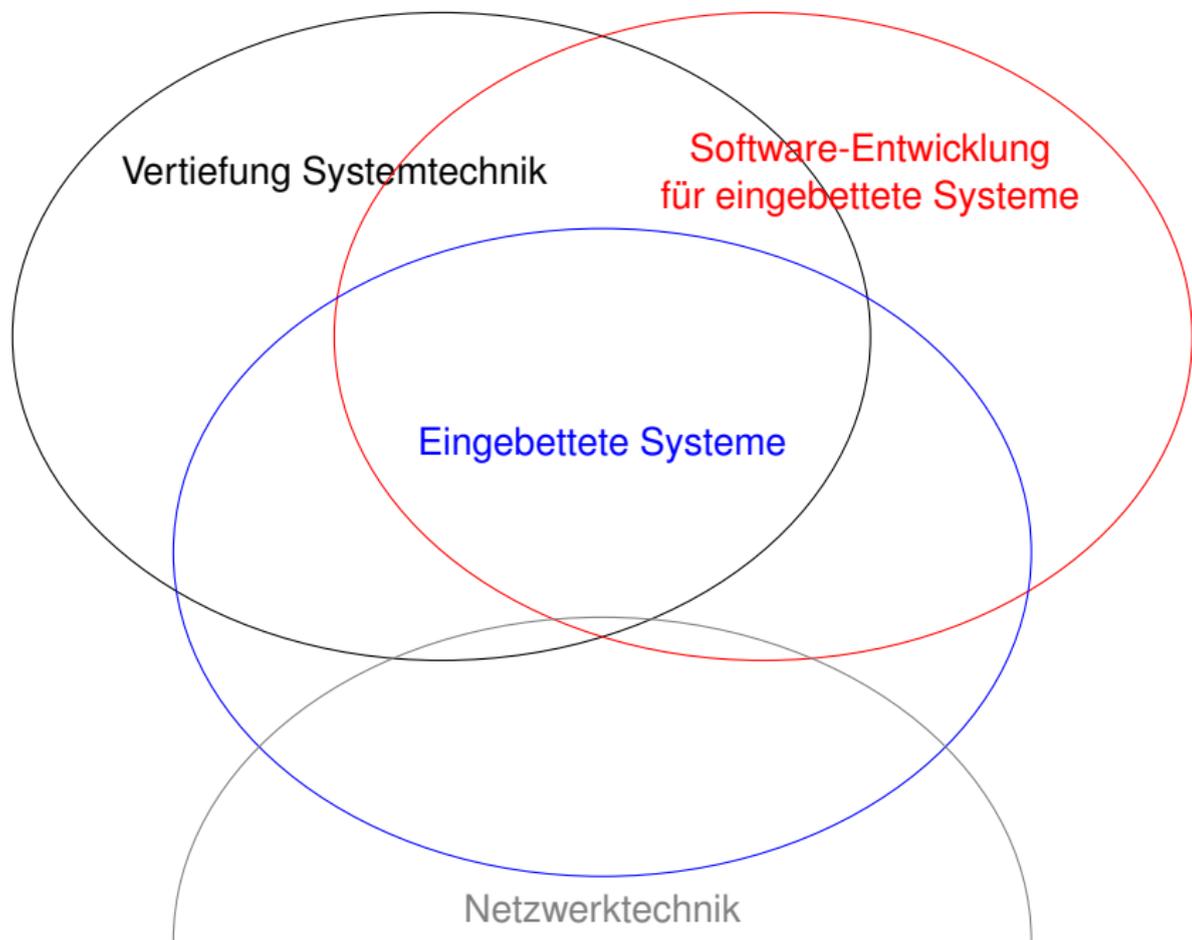
... für Fortgeschrittene

Vertiefung Systemtechnik

Echtzeitsysteme

Thema dieser
Lehrveranstaltung:
Echtzeit

Geräte aller Art



In dieser Lehrveranstaltung

- Einführung in Unix
- TCP/IP und Bus-Systeme in der Praxis
- Web-Interfaces
- C-Programmierung für Fortgeschrittene
- Theorie und Praxis der Echtzeitprogrammierung
- Prüfungsleistung: Projektaufgabe
Eingebettetes System bzw. Echtzeit-System eigener Wahl
zum Laufen bringen

→ **Projektaufgabe überlegen**

Weitere Ideen:

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



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

- Grundfunktionen: `libc`
- 3d-Grafik – Kernfunktion: `libGL`
- 3d-Grafik – Utilities: `libGLU`
- 3d-Grafik – Utility Toolkit: `libglut`
- ...

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
  | grep -v "fig" | less  
./2013ss/net/script/slides/net-probeklausur-20120712.tex  
./2013ss/net/20130924.0/net-klausur-20130924.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
...  
./2012ss/hs/20130318.0/hs-klausur-20130318.tex  
./2012ss/hs/20120715.0/hs-probeklausur-20120715.tex  
./2012ss/hs/20120720.0/hs-klausur-20120720.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 ↑, ↓
- Befehl bearbeiten: Pfeiltasten ←, → 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 –
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

- `grep`: Dateien durchsuchen

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

2.3 Dateisysteme

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/sd-card/  
cassini/home/peter> mount /media/sd-card  
cassini/home/peter> ls /media/sd-card/  
DCIM  NIKON001.DSC  
cassini/home/peter> umount /media/sd-card  
cassini/home/peter> ls /media/sd-card/  
cassini/home/peter>
```

2.3 Dateisysteme

- *Zugriffsrechte*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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*

```
phoenix/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.

(aus Anwendersicht eher selten, daher hier nicht ausführlich)

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.
```

- 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-hochschu
```

```
-rw-r--r-- 1 peter peter 31581 Dez 26 2011 logo-hochschu
```

```
-rw-r--r-- 1 peter peter 8538 Okt  2 2012 NPN_transisto
```

```
-rw-r--r-- 1 peter peter 6753149 Okt  1 22:59 rtech-2013100
```

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
```

Vertiefung Systemtechnik – 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

Vertiefung Systemtechnik Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

26. Oktober 2017

Vertiefung Systemtechnik – Eingebettete Systeme

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

...



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>`

3.2 TCP- und UDP-Ports

- nc

3.3 TCP-Protokolle

- HTTP
- SMTP
- ...

3.4 Routing

- `ip route`

3.5 Netzwerkanalyse

- `tcpdump`

Vertiefung Systemtechnik – Eingebettete Systeme

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

...



Änderungen
vorbehalten

Vertiefung Systemtechnik Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

2. November 2017

Vertiefung Systemtechnik – 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

...



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>`

Beispiel: `ip addr add 192.168.2.197/24`

The chalkboard is divided into two columns. The left column contains text explaining network layers and examples. The right column contains a list of IP addresses with their corresponding hostnames and netmasks.

Anwendung: HTTP, SMTP, ...	PM 192.168.2.142 /24
Transport: TCP- und UDP-Ports	EM .102
Internet: IP-Adresse	EB .177
Netzwerkzugang: Hardware-/MAC-Adresse	MD .166
	PG .198
	DG .126
	AT .7
	DD .99
	SD .123
	DS .77
	AB .160

Examples on the left:
huygens: 192.168.1.198 /24
Cassini: 192.168.1.197 /24
Netzmaske: 255.255.255.0 (32 Bit, davon 24 fest, die restl. 8 variabel)
24 Einsen 8 Nullen

- `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 `<Port>` auf Rechner `<IP>` 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)

Routing

192.168.2.0/24
192.168.2.126
192.168.2.7

10.42.75.0/24
10.42.75.177
10.42.75.102
10.42.75.166
10.42.75.197
192.168.2.197

Router: IP-Forwarding einschalten:
echo 1 > /proc/sys/net/ipv4/ip_forward

`ip link set dev <Device> up`
Netzwerkadapter einschalten

`route add <Netz> via <IP-Adresse>`
↑
„Router“ oder „Gateway“

WLAN-Router
über Kabel: IP-Adresse automatisch beziehen (DHCP)

`ssh root@192.168.1.1`
→ Shell auf dem Router

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

Vertiefung Systemtechnik – 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

...



Änderungen
vorbehalten

Vertiefung Systemtechnik Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

9. November 2017

Vertiefung Systemtechnik – 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 I²C (TWI)

5.4 SPI

5.5 Beispiel: Benutzung des I²C-Busses

6 Echtzeit

5.1 Was ist Echtzeit?

5.2 Echtzeitprogrammierung

5.3 Multitasking

5.4 Ressourcen

5.5 Prioritäten

...



Änderungen
vorbehalten

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

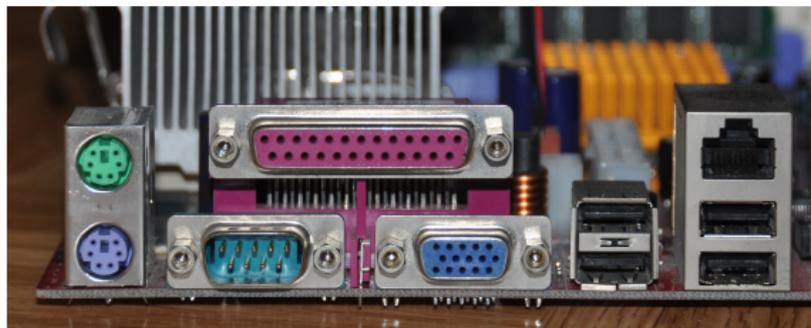
4.1 Was sind Bus-Systeme?

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
- I²C (TWI)
- SPI



4 Bus-Systeme

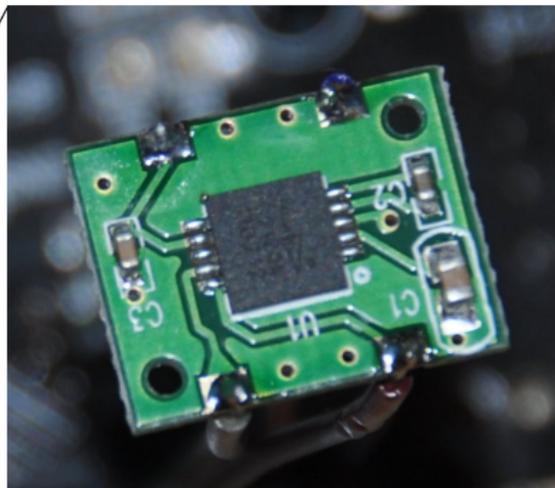
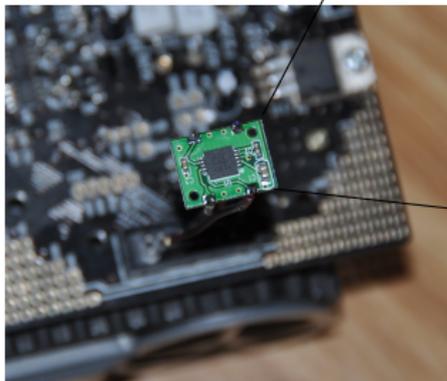
4.1 Was sind Bus-Systeme?

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
- I²C (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>

- I²C: 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

synchron

- *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

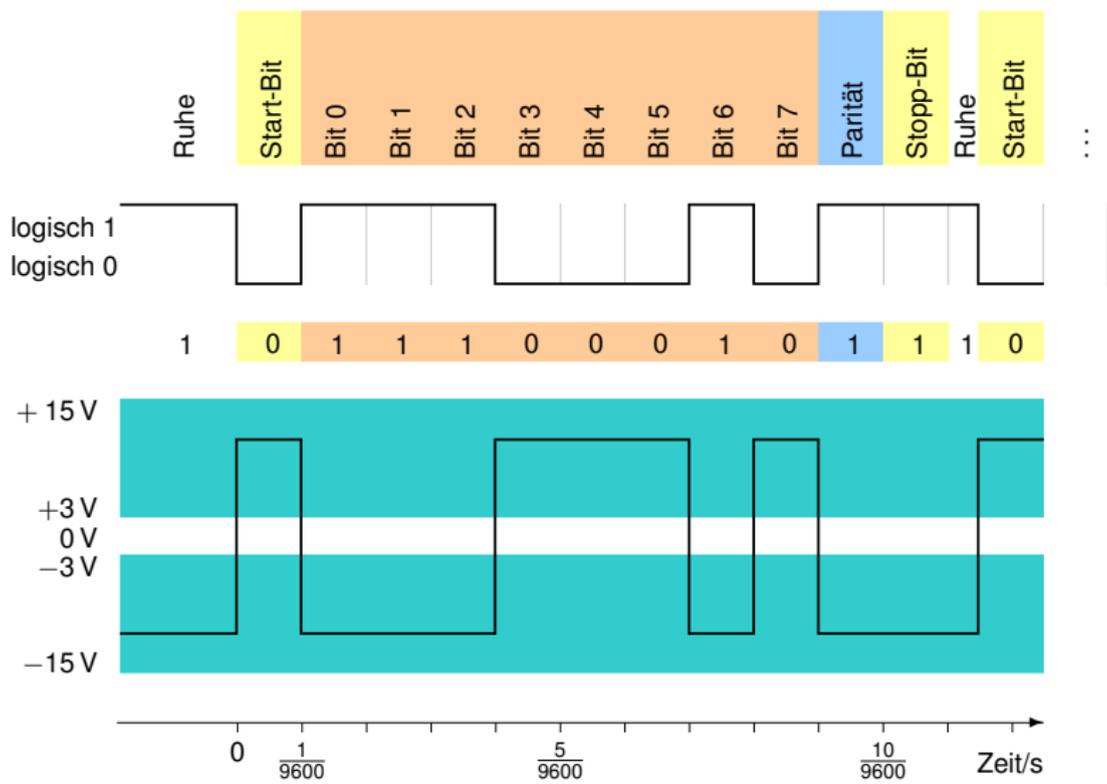
9600 Baud, 8 Daten-Bits, ungerade Parität, 1 Stopp-Bit

Daten

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

Check

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



4 Bus-Systeme

4.3 I²C (TWI)

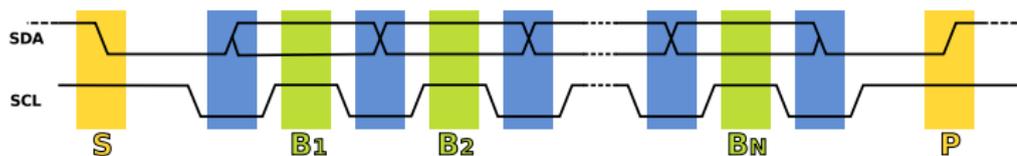
I²C = Inter-IC; TWI = Two-Wire-Interface

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

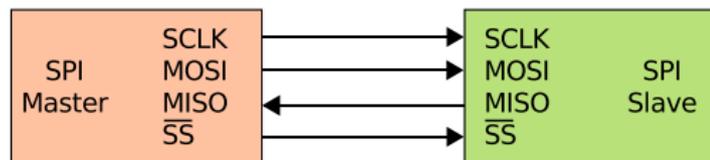
4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

- *MISO*: Master Out, Slave In
- *MOSI*: 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

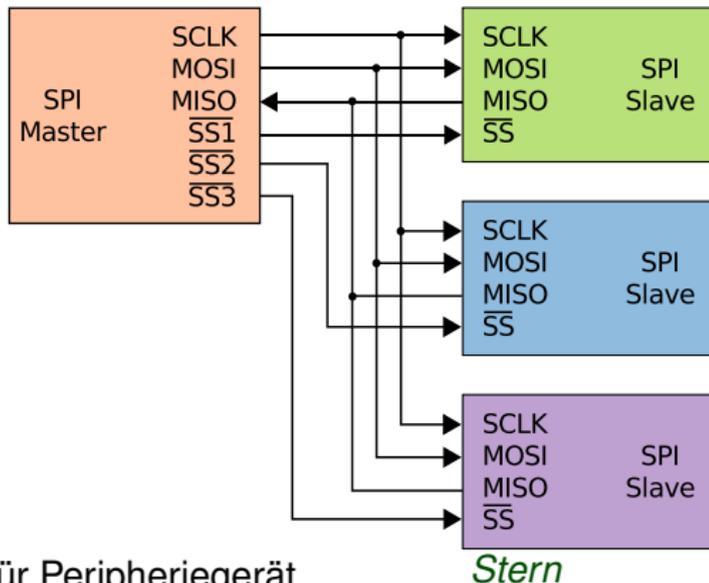
- *MISO*: Master Out, Slave In
- *MOSI*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

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

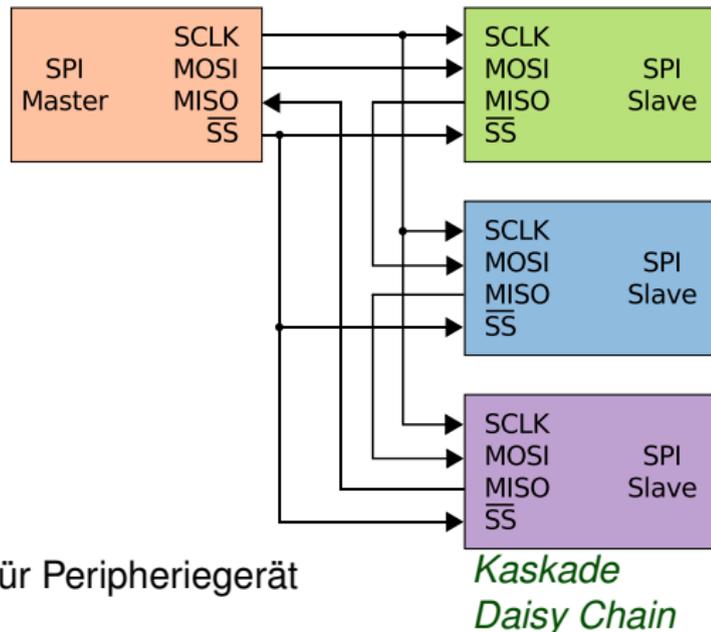
- *MISO*: Master Out, Slave In
- *MOSI*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

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

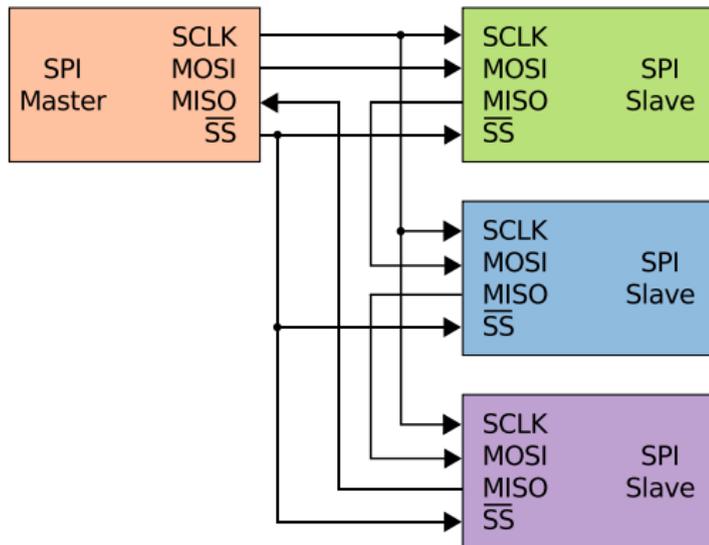
- *MISO*: Master Out, Slave In
- *MOSI*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

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

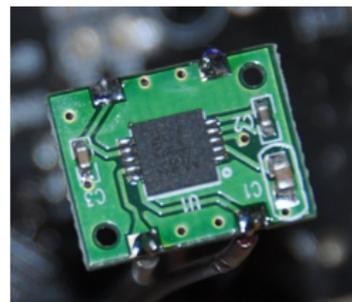


*Kaskade
Daisy Chain*

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

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses



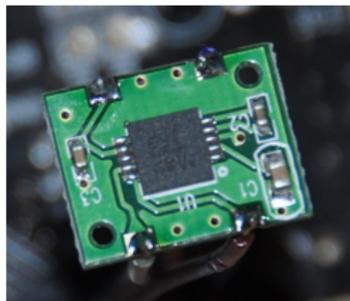
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{ ... }
```

```
int main (void)
{
    uint16_t compass_x, compass_y;
    read_compass (&compass_x, &compass_y);
    writeInteger (compass_x, DEC);
    writeChar ('_');
    writeInteger (compass_y, DEC);
    writeChar ('\n');
    return 0;
}
```

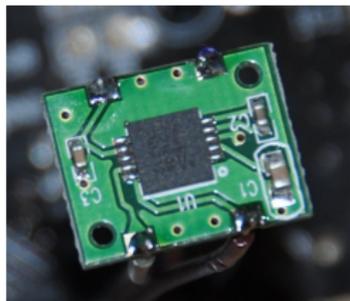


4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses

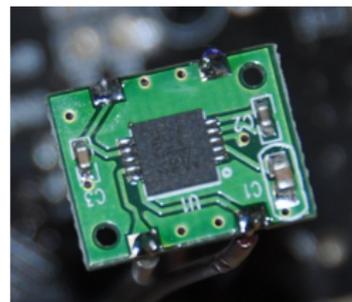
```
void read_compass (uint16_t *x, uint16_t *y)
{ ... }
```

```
int main (void)
{
    uint16_t compass_x, compass_y;
    read_compass (&compass_x, &compass_y);
    writeInteger (compass_x, DEC);
    writeChar ('_');
    writeInteger (compass_y, DEC);
    writeChar ('\n');
    return 0;
}
```



4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
```

```
{
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
```

```
    mSleep (1);
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
```

```
    mSleep (5);
```

```
    uint8_t result[5];
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
```

```
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
```

```
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
```

```
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
```

```
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
```

```
    result[2] &= 0b00001111;
```

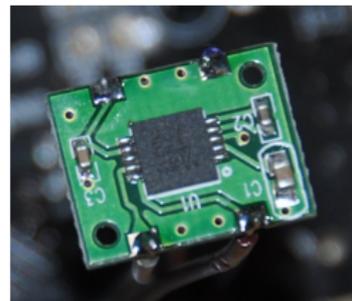
```
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
```

```
    *y = (result[2] << 8) + result[3];
```

```
}
```

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
```

```
{
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
```

```
    mSleep (1);
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
```

```
    mSleep (5);
```

```
    uint8_t result[5];
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
```

```
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
```

```
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
```

```
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
```

```
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
```

```
    result[2] &= 0b00001111;
```

```
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
```

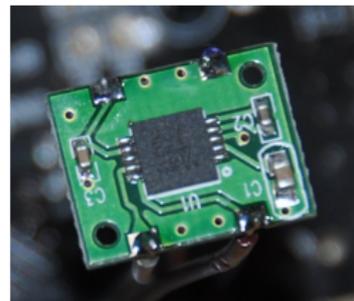
```
    *y = (result[2] << 8) + result[3];
```

```
}
```

Gerät adressieren:
Schreib-Adresse

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
```

```
{
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
```

```
    mSleep (1);
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
```

```
    mSleep (5);
```

```
    uint8_t result[5];
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
```

```
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
```

```
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
```

```
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
```

```
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
```

```
    result[2] &= 0b00001111;
```

```
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
```

```
    *y = (result[2] << 8) + result[3];
```

```
}
```

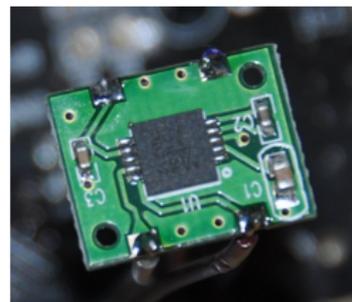
Gerät adressieren:
Schreib-Adresse

Register-Nr.

innerhalb des Geräts

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
```

```
{
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
```

```
    mSleep (1);
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
```

```
    mSleep (5);
```

```
    uint8_t result[5];
```

```
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
```

```
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
```

```
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
```

```
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
```

```
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
```

```
    result[2] &= 0b00001111;
```

```
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
```

```
    *y = (result[2] << 8) + result[3];
```

```
}
```

Gerät adressieren:
Schreib-Adresse

Register-Nr.

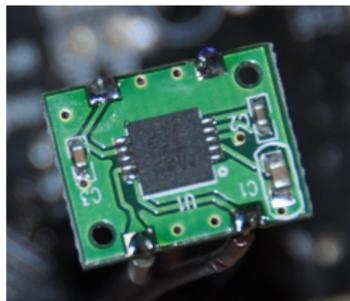
Befehl: Spule an

innerhalb des Geräts

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses

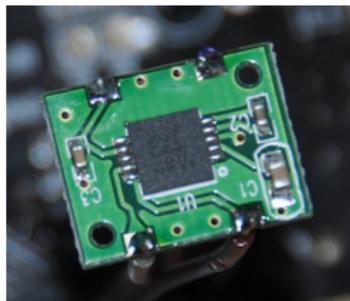
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```



4 Bus-Systeme

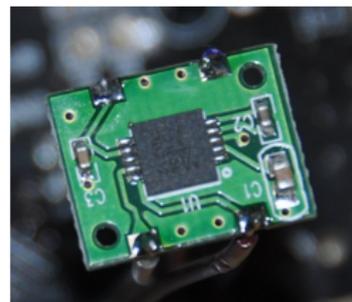
4.5 Beispiel: Benutzung des I²C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```



4 Bus-Systeme

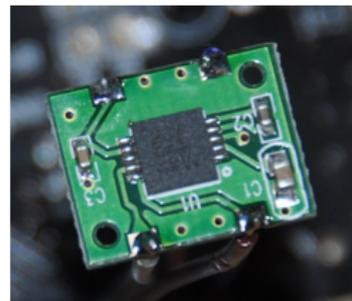
4.5 Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

4 Bus-Systeme

4.5 Beispiel: Benutzung des I²C-Busses

Messung:

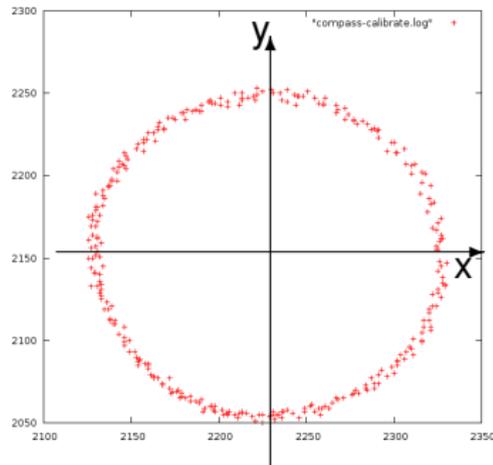
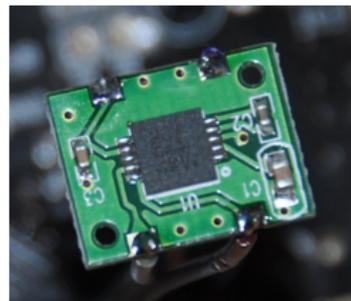
Roboter im Kreis drehen lassen,
x- und y-Werte aufzeichnen
und gegeneinander auftragen

Ergebnis: Ellipse

Anwendung:

Roboter in eine Himmelsrichtung ausrichten

- Achsen normieren:
Mittelwert subtrahieren
- Nulldurchgang einer Achse
markiert eine Himmelsrichtung
- Vorzeichen der anderen Achse
sagt aus, welche Himmelsrichtung



Beispiel: nach Norden ausrichten
Grob drehen, bis y positiv ist, fein drehen, bis $x = 0$ ist

Vertiefung Systemtechnik – 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 I²C (TWI)

5.4 SPI

5.5 Beispiel: Benutzung des I²C-Busses

6 Echtzeit

5.1 Was ist Echtzeit?

5.2 Echtzeitprogrammierung

5.3 Multitasking

5.4 Ressourcen

5.5 Prioritäten

...



Änderungen
vorbehalten

5 Echtzeit

5.1 Was ist Echtzeit?

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.
- Datenverarbeitung in Echtzeit:
Die Daten werden so schnell verarbeitet, wie sie anfallen.

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.
- Datenverarbeitung in Echtzeit:
Die Daten werden so schnell verarbeitet, wie sie anfallen.
- speziell: Echtzeit-Steuerung von Maschinen:
Die Berechnung kann mit den physikalischen Vorgängen schritthalten.

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.
- Datenverarbeitung in Echtzeit:
Die Daten werden so schnell verarbeitet, wie sie anfallen.
- speziell: Echtzeit-Steuerung von Maschinen:
Die Berechnung kann mit den physikalischen Vorgängen schritthalten.

→ „Schnell genug.“

5.1 Was ist Echtzeit?

„Schnell genug.“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

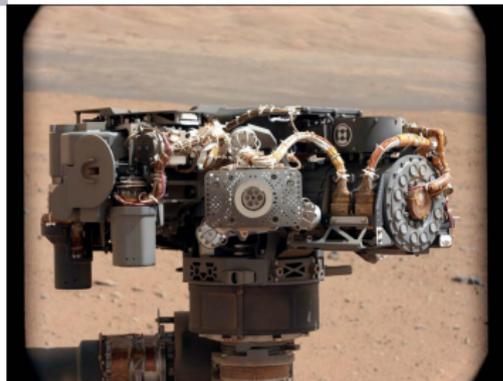
- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*
verspätetes Ergebnis qualitätsmindernd
 - verwenden und Verzögerung in Kauf nehmen
 - verwerfen und Ausfall in Kauf nehmen

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*
verspätetes Ergebnis qualitätsmindernd
 - verwenden und Verzögerung in Kauf nehmen
 - verwerfen und Ausfall in Kauf nehmen
- „Es gibt keinen festen Termin. Möglichst schnell halt.“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*
verspätetes Ergebnis qualitätsmindernd
 - verwenden und Verzögerung in Kauf nehmen
 - verwerfen und Ausfall in Kauf nehmen
- „Es gibt keinen festen Termin. Möglichst schnell halt.“
→ *keine Echtzeit*

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000 \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10 \mu\text{s}$ dauert.

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000 \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10 \mu\text{s}$ dauert.
- Entweder: Der Steuer-Computer macht noch andere Dinge.
→ Risiko der Zeitüberschreitung

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000 \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10 \mu\text{s}$ dauert.
- Entweder: Der Steuer-Computer macht noch andere Dinge.
→ Risiko der Zeitüberschreitung
- Oder: Der Steuer-Computer macht nichts anderes.
→ Verschwendung von Rechenzeit

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000 \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10 \mu\text{s}$ dauert.
- Entweder: Der Steuer-Computer macht noch andere Dinge.
→ Risiko der Zeitüberschreitung
- Oder: Der Steuer-Computer macht nichts anderes.
→ Verschwendung von Rechenzeit
→ Na und?

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

Wertvolle Ressourcen

- Fähigkeiten einer Raumsonde optimieren



5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

Wertvolle Ressourcen

- Fähigkeiten einer Raumsonde optimieren
- Implantat: Platz- und Stromverbrauch minimieren



5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

Wertvolle Ressourcen

- Fähigkeiten einer Raumsonde optimieren
- Implantat: Platz- und Stromverbrauch minimieren

→ **Echtzeitprogrammierung**

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Aber wie?

5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware

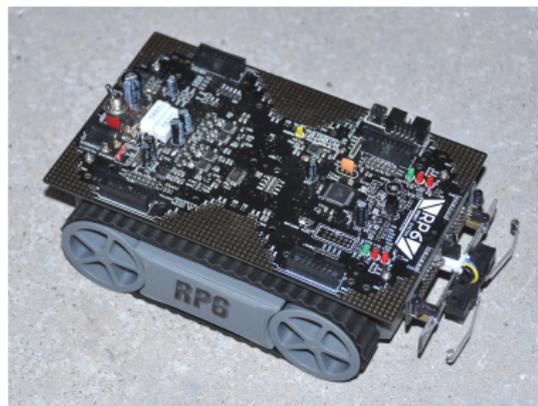


5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware
- RP6:
Motorsteuerung vs. Anwender-Software
→ spezielle Software



5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware
- RP6:
Motorsteuerung vs. Anwender-Software
→ spezielle Software
- Quadrocopter:
Motorsteuerung vs. Sensoren-Abfrage
vs. Funk-Fernsteuerung ...
→ spezielle Software



5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware
- RP6:
Motorsteuerung vs. Anwender-Software
→ spezielle Software
- Quadrocopter:
Motorsteuerung vs. Sensoren-Abfrage
vs. Funk-Fernsteuerung ...
→ spezielle Software
- Flugzeugkabinensimulatortür:
Türsteuerung vs. Bedienung
→ Echtzeitbetriebssystem



5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
Durch Zähler im Interrupt-Handler: verschiedene Taktraten
1000mal pro Sekunde: Stopwatches
5mal pro Sekunde: Blinkende Power-On-LED
1000mal pro Sekunde: Bumper, ACS, PWM zur Motorsteuerung
Geschwindigkeitsmessung durch Zählen der Ticks in 0.2 s
Anpassung der Motorkraft in ± 1 -Schritten

5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
Durch Zähler im Interrupt-Handler: verschiedene Taktraten
1000mal pro Sekunde: Stopwatches
5mal pro Sekunde: Blinkende Power-On-LED
1000mal pro Sekunde: Bumper, ACS, PWM zur Motorsteuerung
Geschwindigkeitsmessung durch Zählen der Ticks in 0.2 s
Anpassung der Motorkraft in ± 1 -Schritten
- Nebenbei: Benutzerprogramm

5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
Durch Zähler im Interrupt-Handler: verschiedene Taktraten
1000mal pro Sekunde: Stopwatches
5mal pro Sekunde: Blinkende Power-On-LED
1000mal pro Sekunde: Bumper, ACS, PWM zur Motorsteuerung
Geschwindigkeitsmessung durch Zählen der Ticks in 0.2 s
Anpassung der Motorkraft in ± 1 -Schritten
- Nebenbei: 1 Benutzerprogramm

5.3 Multitasking

- *Kooperatives Multitasking*
Prozesse geben freiwillig Rechenzeit ab
- *Präemptives Multitasking*
Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)
- *Scheduler*
Steuerprogramm, das Prozessen Rechenzeit zuteilt
- *Kontextwechsel*
Umschalten zwischen zwei Prozessen
- *Round-Robin-Verfahren (Rundlauf)*
Zuteilung von *Zeitschlitz*en auf einer *Zeitscheibe* an die Prozesse

5.3 Multitasking

- *Kooperatives Multitasking*
Prozesse geben freiwillig Rechenzeit ab
- *Präemptives Multitasking*
Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)
- *Scheduler*
Steuerprogramm, das Prozessen Rechenzeit zuteilt
- *Kontextwechsel*
Umschalten zwischen zwei Prozessen
- *Round-Robin-Verfahren (Rundlauf)*
Zuteilung von *Zeitschlitz*en auf einer *Zeitscheibe* an die Prozesse
- Ausblick: Zuteilung von Rechenzeit = wichtiger Spezialfall
allgemein: Zuteilung von Ressourcen

Beispiele für Multitasking

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm

Linux 0.01

- Timer-Interrupt:
Task mit höchstem Zähler bekommt Rechenzeit.
- Wenn es keinen lafbereiten Task mit positivem Zähler gibt, bekommen alle Tasks gemäß ihrer Priorität neue Zähler zugewiesen.
- *keine* harte Echtzeit

Beispiele für Multitasking

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm

Linux 0.01

- Timer-Interrupt: Zähler des aktuellen Tasks wird dekrementiert; Task mit höchstem Zähler bekommt Rechenzeit.
- Wenn es keinen lafbereiten Task mit positivem Zähler gibt, bekommen alle Tasks gemäß ihrer Priorität neue Zähler zugewiesen.
- *keine* harte Echtzeit

Zombies

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“),
bekommt keine Rechenzeit mehr („seiner Seele beraubt“),
hat alle belegten Ressourcen wieder freigegeben („willenlos“),
wird aber noch in der Prozeßliste geführt („untot“).

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“),
bekommt keine Rechenzeit mehr („seiner Seele beraubt“),
hat alle belegten Ressourcen wieder freigegeben („willenlos“),
wird aber noch in der Prozeßliste geführt („untot“).

- Warum? → Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“),
bekommt keine Rechenzeit mehr („seiner Seele beraubt“),
hat alle belegten Ressourcen wieder freigegeben („willenlos“),
wird aber noch in der Prozeßliste geführt („untot“).

- Warum? → Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.
- Schadet das? → Nein.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“),
bekommt keine Rechenzeit mehr („seiner Seele beraubt“),
hat alle belegten Ressourcen wieder freigegeben („willenlos“),
wird aber noch in der Prozeßliste geführt („untot“).

- Warum? → Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.
- Schadet das? → Nein.
- Aber? → Wenn sich Zombie-Prozesse anhäufen, deutet dies auf einen Prozeß hin, der andere Prozesse erzeugt und anschließend „hängt“.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“), bekommt keine Rechenzeit mehr („seiner Seele beraubt“), hat alle belegten Ressourcen wieder freigegeben („willenlos“), wird aber noch in der Prozeßliste geführt („untot“).

- Warum? → Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.
- Schadet das? → Nein.
- Aber? → Wenn sich Zombie-Prozesse anhäufen, deutet dies auf einen Prozeß hin, der andere Prozesse erzeugt und anschließend „hängt“.
- Beispiel: Datenträger entfernt, zugreifender Prozeß „hängt“.
→ Tochterprozesse werden zu Zombies.

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter*
vs. *Durchsatz*

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter*
vs. *Durchsatz*
 - *Latenz*: interaktive Anwendungen
 - *Jitter*: Echtzeitanwendungen
 - *Durchsatz*: Stapelverarbeitung

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter*
vs. *Durchsatz*
- Ressourcen reservieren:
Mutexe

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter*
vs. *Durchsatz*
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter*
vs. *Durchsatz*
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)
→ kommt gleich

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter*
vs. *Durchsatz*
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)
→ kommt gleich
- Verschiedene Methoden
der Priorisierung
→ später

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits
(IPv4) bzw. Traffic-Class-Bits (IPv6)
im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.3 Multitasking

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter*
vs. *Durchsatz*
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)
→ kommt gleich
- Verschiedene Methoden
der Priorisierung
→ später
- Umgehung der Probleme durch
speziell geschriebene Software
(MultiWii, RP6, ...)

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits
(IPv4) bzw. Traffic-Class-Bits (IPv6)
im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

5 Echtzeit

5.4 Ressourcen

Ressourcen reservieren

- *Semaphor*
gemeinsame Variable mehrerer Prozesse
zur Regelung des Zugriffs auf eine Ressource
Ressource belegt → Kontextwechsel

griechisch: *sema* – Zeichen, *pherein* – tragen
„Eisenbahnsignal“

5 Echtzeit

5.4 Ressourcen

Ressourcen reservieren

- *Semaphor*
gemeinsame Variable mehrerer Prozesse
zur Regelung des Zugriffs auf eine Ressource
Ressource belegt → Kontextwechsel
- *Mutex*
Mechanismus, damit immer nur ein Prozeß gleichzeitig
auf eine Ressource zugreifen kann

englisch: *mutual exclusion* – wechselseitiger Ausschluß
spezieller binärer Semaphor: nur „Besitzer“ darf freigeben

5 Echtzeit

5.4 Ressourcen

Ressourcen reservieren

- *Semaphor*
gemeinsame Variable mehrerer Prozesse
zur Regelung des Zugriffs auf eine Ressource
Ressource belegt \rightarrow Kontextwechsel
- *Mutex*
Mechanismus, damit immer nur ein Prozeß gleichzeitig
auf eine Ressource zugreifen kann
- *Spinlock (busy waiting)*
leichtgewichtige Alternative zu Kontextwechsel
englisch: *spin* – rotieren, *lock* Sperre
busy waiting auf etwas Schnelles, z. B. auf einen Semaphor
Hardware-Unterstützung: Prüfen, ob Variable bestimmten Wert hat;
wenn ja, auf anderen Wert setzen; andere Prozessoren solange anhalten

5 Echtzeit

5.4 Ressourcen

Ressourcen reservieren

- *Semaphor*
gemeinsame Variable mehrerer Prozesse
zur Regelung des Zugriffs auf eine Ressource
Ressource belegt → Kontextwechsel
- *Mutex*
Mechanismus, damit immer nur ein Prozeß gleichzeitig
auf eine Ressource zugreifen kann
- *Spinlock (busy waiting)*
leichtgewichtige Alternative zu Kontextwechsel
- *Kritischer Abschnitt – critical section*
Programmabschnitt zwischen Reservierung
und Freigabe einer Ressource
→ sollte immer so kurz wie möglich sein

5 Echtzeit

5.4 Ressourcen

Ressourcen reservieren – Beispiel: `linux-3.7rc1`

- *Semaphor*
kernel/semaphor.c
drivers/usb/core/file.c
- *Mutex*
kernel/mutex.c
drivers/usb/serial/usb-serial.c
- *Spinlock*
kernel/spinlock.c
kernel/semaphor.c, kernel/mutex.c

Beispiel: `usb_serial_get_by_index()` – serielle Schnittstelle reservieren
Datei `linux-3.7-rc1/drivers/usb/serial/usb-serial.c`, ab Zeile 62

```
struct usb_serial *usb_serial_get_by_index (unsigned index)
```

```
{
```

```
    struct usb_serial *serial;
```

```
    mutex_lock (&table_lock); ← exklusiven Zugriff auf Tabelle sichern
```

```
    serial = serial_table[index];
```

```
    if (serial)
```

```
    {
```

```
        mutex_lock (&serial->disc_mutex);
```

```
        if (serial->disconnected)
```

```
        {
```

```
            mutex_unlock (&serial->disc_mutex);
```

```
            serial = NULL;
```

```
        }
```

```
        else
```

```
            kref_get (&serial->kref);
```

```
    }
```

```
    mutex_unlock (&table_lock); ← exklusiven Zugriff auf Tabelle
```

```
    return serial; wieder freigeben
```

```
}
```

```

mutex_lock) - Ressourcenbesucher, nicht alle warten
Datei linux-3.7.rc1/drivers/usb/net/usb-net.c, ab Zeile 62
void __sched mutex_lock (struct mutex *lock)
{
    high_sleep ();
    __mutex_lockpath (lock->count, __mutex_lock_sleepfn);
    mutex_set_owner (lock);
}

Datei linux-3.7.rc1/arch/86/include/asm/mutex_32.h, ab Zeile 24
Macro Definition for __mutex_lockpath (expanded)
Assembler:
lock die: lock ->count
jnb 1
call __mutex_lock_sleepfn
t:

Datei linux-3.7.rc1/kernel/mutex.c, ab Zeile 206
static inline noinline void __sched
__mutex_lock_sleepfn (atomic_t *lock_count)
{
    struct mutex_lock = container_of (lock_count, struct mutex, count);
    __mutex_lock_common (lock, TASK_UNINTERRUPTIBLE, 0,
        NULL, _RET_IP_);
}

Datei linux-3.7.rc1/kernel/mutex.c, ab Zeile 132
static inline int __sched
__mutex_lock_common (struct mutex *lock, long state, unsigned int subclass,
    struct lockdep_map *nest_lock, unsigned long ip)
{
    struct task_struct *task = current;
    struct mutex_walker waiter;
    unsigned long flags;

    preempt_disable ();
    mutex_acquire_nest (lock->dep_map, subclass, 0, nest_lock, ip);
    A ... /

    spin_lock_mutex (lock->wait_lock, flags);
    debug_mutex_lock_common (lock, &waiter);
    debug_mutex_add_waiter (lock, &waiter, task, thread_info (task));
    A add waiting task to the end of the waitqueue (FIFO) -> /
    list_add_tail (&waiter, list, lock->wait_list);
    waiter.task = task;

    #if (atomic_achg (lock->count, -1) == 1)
    goto done;
    #endif

    lock_contended (lock->dep_map, ip);
    for (;;)
    {
        A
        A Lets try to take the lock again - this is needed even if
        A we get here for the first time (shortly after failing to
        A acquire the lock, to make sure that we get a wakeup once
        A it's unlocked. Later on, if we sleep, this is the
        A operation that gives us the lock. We sleep if it == -1, as
        A as that when we release the lock, we properly wake up the
        A other waiters:
        A
        #if (atomic_achg (lock->count, -1) == 1)
        break;
        #endif

        A got a signal? (This code gets eliminated in the
        A TASK_UNINTERRUPTIBLE case.)
        A
        #if (unlikely (signal_pending_state (state, task)))
        {
            mutex_remove_waiter (lock, &waiter, task, thread_info (task));
            mutex_release (lock->dep_map, 1, ip);
            spin_unlock_mutex (lock->wait_lock, flags);
            debug_mutex_free_waiter (&waiter);
            preempt_enable ();
            return -EINTR;
        }
        A
        A _list_wait_state (lock, state);
        A didn't get the lock, go to sleep:
        spin_unlock_mutex (lock->wait_lock, flags);
        schedule_preempt_disabled ();
        spin_lock_mutex (lock->wait_lock, flags);
    }

done:
lock_acquire (lock->dep_map, ip);
A got the lock - rejoice!
mutex_remove_waiter (lock, &waiter, current, thread_info ());
mutex_set_owner (lock);

A set it to 0 if there are no waiters left:
A (likely list_empty (lock->wait_list))
atomic_set (lock->count, 0);

spin_unlock_mutex (lock->wait_lock, flags);
debug_mutex_free_waiter (&waiter);
preempt_enable ();

return 0;
}

```

→ Evaluation Zugriff auf Mutex sichern

→ Evaluation Zugriff auf Mutex wieder freigeben

```

spin_lock_mutex() - Mutex besprochen, mittels busy waiting
Datei linux-3.7-rc1/kernel/mutex.c, ab Zeile 12
#define spin_lock_mutex(lock, flags)
do {
    {
        spin_lock(lock);
        (void) (flags);
    }
} while (0)
Datei linux-3.7-rc1/kernel/spinlock.h, ab Zeile 283
static inline void spin_lock(spinlock_t *lock)
{
    raw_spin_lock(&lock->rlock);
}
Datei linux-3.7-rc1/kernel/spinlock.h, Zeile 170
#define raw_spin_lock(lock) _raw_spin_lock(lock)
Datei linux-3.7-rc1/include/linux/spinlock_api_smp.h, Zeile 47
#define _raw_spin_lock(lock)
Datei linux-3.7-rc1/kernel/spinlock.c, ab Zeile 46 (expandiert):
void __lockfunc__raw_spin_lock(spinlock_t *lock)
{
    for (;;)
    {
        preempt_disable();
        if (likely(!do_raw_spin_trylock(lock)))
            break;
        preempt_enable();
        if (!lock) --break_lock
            (lock) --break; lock = 1;
        while (raw_spin_can_lock(lock) && (lock) --break; lock)
            arch_spin_relax(&lock->raw_lock);
        (lock) --break; lock = 0;
    }
}
Datei linux-3.7-rc1/include/linux/spinlock.h, ab Zeile 150:
static inline int do_raw_spin_trylock(raw_spinlock_t *lock)
{
    return arch_spin_trylock(&lock->raw_lock);
}
Datei arch/x86/include/asm/spinlock.h, ab Zeile 116:
static __always_inline int arch_spin_trylock(arch_spinlock_t *lock)
{
    return __sckat_spin_trylock(lock);
}
Datei arch/x86/include/asm/spinlock.h, ab Zeile 65:
static __always_inline int __sckat_spin_trylock(arch_spinlock_t *lock)
{
    arch_spinlock_t old, new;

    old.tickets = ACCESS_ONCE(lock->tickets);
    if (old.tickets.head != old.tickets.tail)
        return 0;

    new.head_tail = old.head_tail + (1 << TICKET_SHIFT);

    /* cmpchg is a full barrier, so nothing can move before it */
    return cmpchg(&lock->head_tail, old.head_tail, new.head_tail) == old.head;
}
Datei arch/x86/include/asm/compchg.h, ab Zeile 147:
#define compchg(ptr, old, new)
__compchg(ptr, old, new, sizeof(ptr))
Datei arch/x86/include/asm/compchg.h, ab Zeile 131:
#define __compchg(ptr, old, new, size)
__raw_compchg((ptr), (old), (new), (size), LOCK_PREFIX)
Datei arch/x86/include/asm/compchg.h, ab Zeile 110:
asm volatile (lock "cmpchgl,%2,%1"
: "a"(_new), "m"(_old)
: "r"(_new), "r"(_old)
:"memory");

```

atomarer und exklusiver
Zugriff auf Spinlock
durch Hardware-Unterstützung

5 Echtzeit

5.4 Ressourcen

Ressourcen reservieren

- *Semaphor*
gemeinsame Variable mehrerer Prozesse
zur Regelung des Zugriffs auf eine Ressource
Ressource belegt → Kontextwechsel
- *Mutex*
Mechanismus, damit immer nur ein Prozeß gleichzeitig
auf eine Ressource zugreifen kann
- *Spinlock (busy waiting)*
leichtgewichtige Alternative zu Kontextwechsel
- *Kritischer Abschnitt – critical section*
Programmabschnitt zwischen Reservierung
und Freigabe einer Ressource
→ sollte immer so kurz wie möglich sein

5 Echtzeit

5.4 Ressourcen

Verklemmungen: Gegenseitiges Blockieren von Ressourcen

- *Deadlock*: Prozeß wartet
- *Livelock*: Prozeß macht andere Dinge
(z. B. *busy waiting*)

5 Echtzeit

5.4 Ressourcen

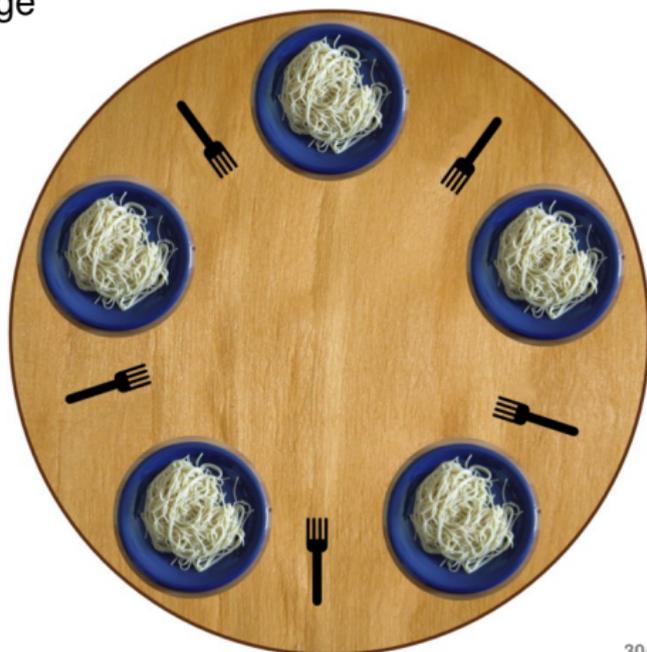
Verklemmungen: Gegenseitiges Blockieren von Ressourcen

- **Deadlock:** Prozeß wartet
- **Livelock:** Prozeß macht andere Dinge (z. B. *busy waiting*)

Beispiel: Philosophenproblem

- 5 Philosophen, 5 Gabeln
- 2 Gabeln zum Essen notwendig
- Wer essen will, nimmt eine Gabel und wartet notfalls auf die zweite.
- Keiner legt eine einzelne Gabel wieder zurück.

Jeder hält 1 Gabel → **Verklemmung**



5 Echtzeit

5.4 Ressourcen

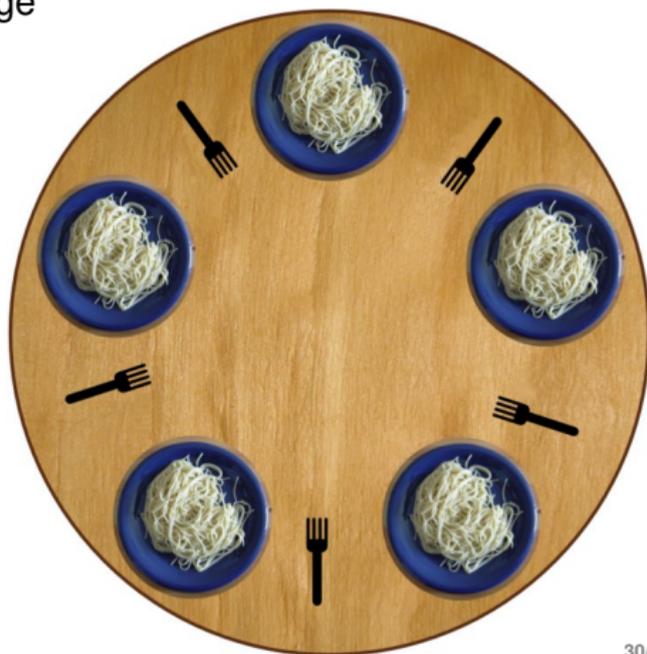
Verklemmungen: Gegenseitiges Blockieren von Ressourcen

- *Deadlock*: Prozeß wartet
- *Livelock*: Prozeß macht andere Dinge (z. B. *busy waiting*)

Beispiel: Philosophenproblem

- 5 Philosophen, 5 Gabeln
- 2 Gabeln zum Essen notwendig
- Wer essen will, nimmt eine Gabel und wartet notfalls auf die zweite.
- Keiner legt eine einzelne Gabel wieder zurück.

Jeder hält 1 Gabel → *Verklemmung*
schweigen → *Deadlock*
philosophieren weiter → *Livelock*



5 Echtzeit

5.4 Ressourcen

Verklemmungen: Gegenseitiges Blockieren von Ressourcen

- *Deadlock*: Prozeß wartet
- *Livelock*: Prozeß macht andere Dinge
(z. B. *busy waiting*)

Beispiel: Philosophenproblem

Bedingungen für Verklemmungen:

- Exklusivität
- *hold and wait*
- Entzug nicht möglich
- zirkuläre Blockade

5 Echtzeit

5.4 Ressourcen

Verklemmungen: Gegenseitiges Blockieren von Ressourcen

- *Deadlock*: Prozeß wartet
- *Livelock*: Prozeß macht andere Dinge
(z. B. *busy waiting*)

Beispiel: Philosophenproblem

Bedingungen für Verklemmungen:

- Exklusivität → Spooling
- *hold and wait* → simultane Zuteilung
- Entzug nicht möglich → Prozesse suspendieren, beenden, *Rollback*
- zirkuläre Blockade → Reihenfolge abhängig von Ressourcen

5 Echtzeit

5.5 Prioritäten

Linux 0.01

- Timer-Interrupt: Zähler des aktuellen Prozesses wird dekrementiert; Prozeß mit höchstem Zähler bekommt Rechenzeit.
- Wenn es keinen lafbereiten Prozeß mit positivem Zähler gibt, bekommen alle Prozesse gemäß ihrer *Priorität* neue Zähler zugewiesen.
- *keine* harte Echtzeit

→ *dynamische Prioritätenvergabe*:
Rechenzeit hängt vom Verhalten des Prozesses ab

Echtzeitbetriebssysteme

- Prozesse können einen festen Anteil an Rechenzeit bekommen.
- Bei Ereignissen können Prozesse hoher Priorität Prozesse niedriger Priorität unterbrechen, aber nicht umgekehrt.

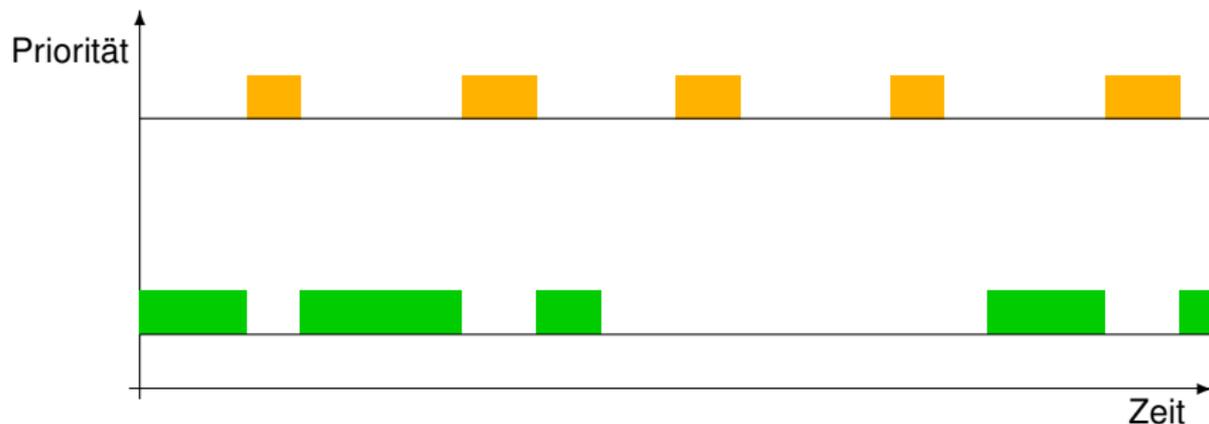
→ *statische Prioritätenvergabe*

5 Echtzeit

5.5 Prioritäten

Echtzeitbetriebssysteme

- Prozesse können einen festen Anteil an Rechenzeit bekommen.
- Bei Ereignissen können Prozesse hoher Priorität Prozesse niedriger Priorität unterbrechen, aber nicht umgekehrt.

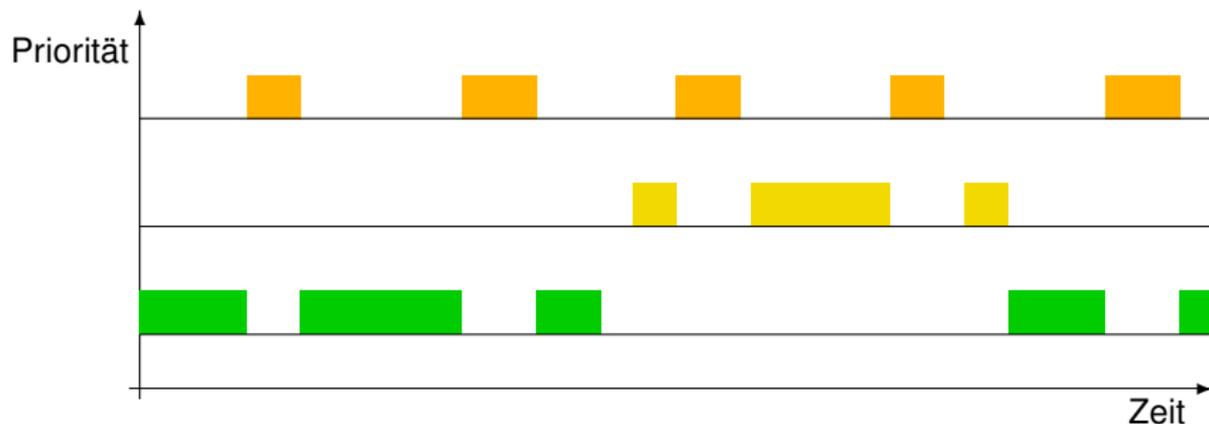


5 Echtzeit

5.5 Prioritäten

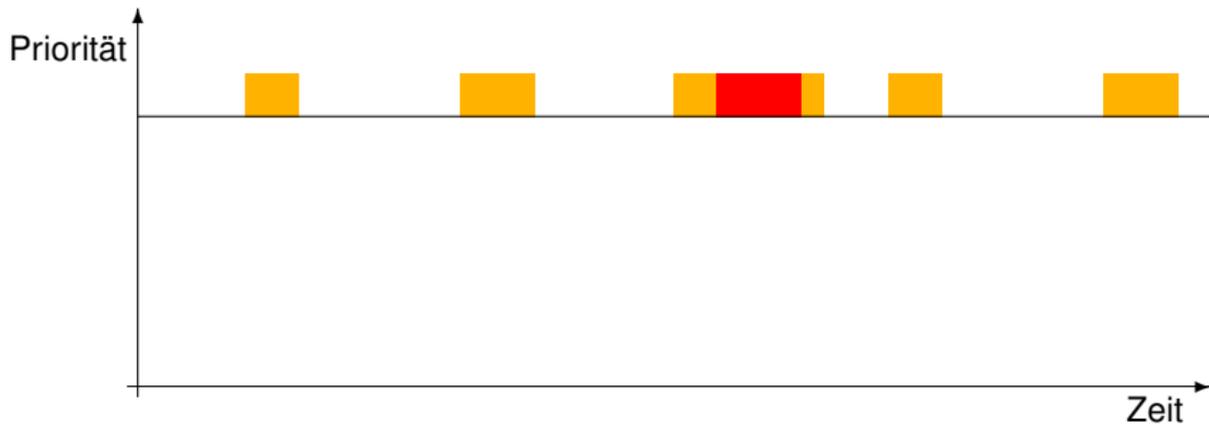
Echtzeitbetriebssysteme

- Prozesse können einen festen Anteil an Rechenzeit bekommen.
- Bei Ereignissen können Prozesse hoher Priorität Prozesse niedriger Priorität unterbrechen, aber nicht umgekehrt.



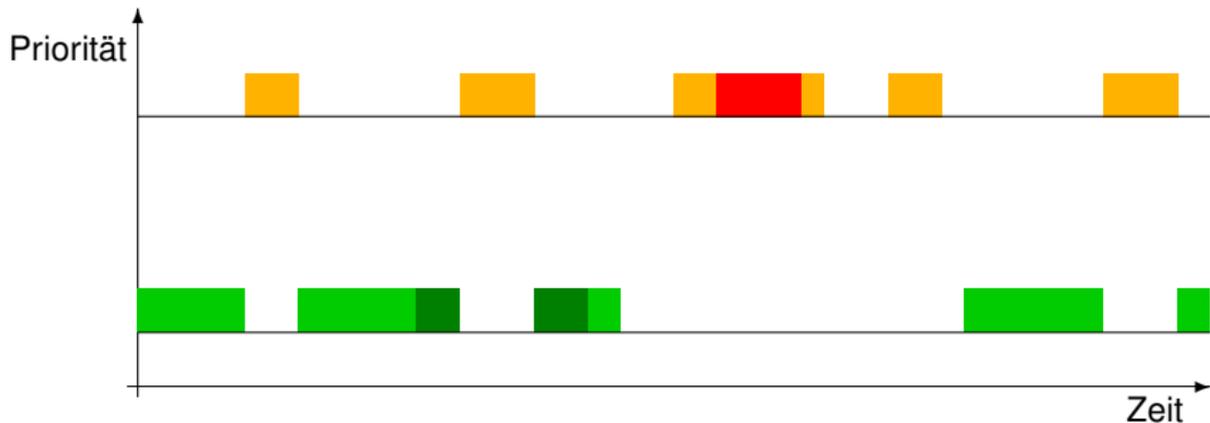
5 Echtzeit

5.5 Prioritäten und Ressourcen



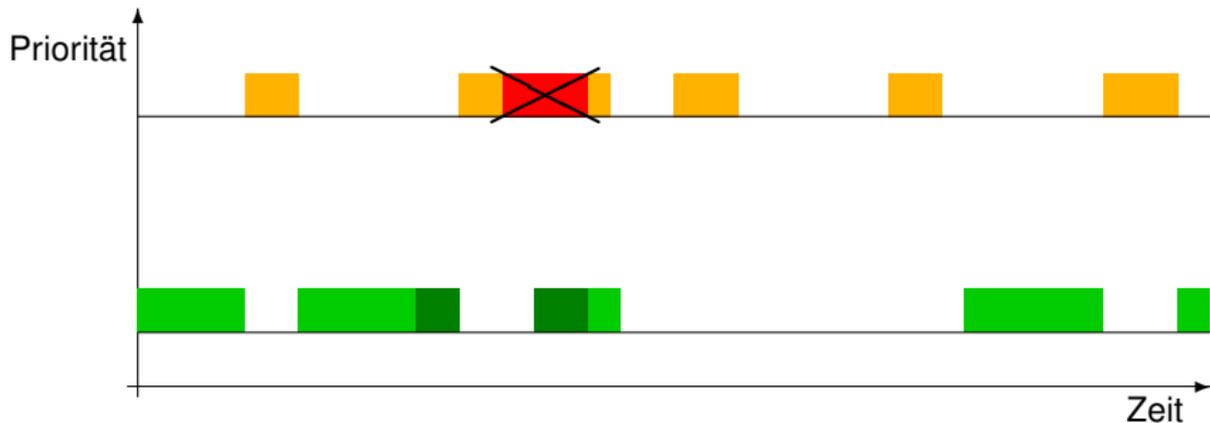
5 Echtzeit

5.5 Prioritäten und Ressourcen



5 Echtzeit

5.5 Prioritäten und Ressourcen



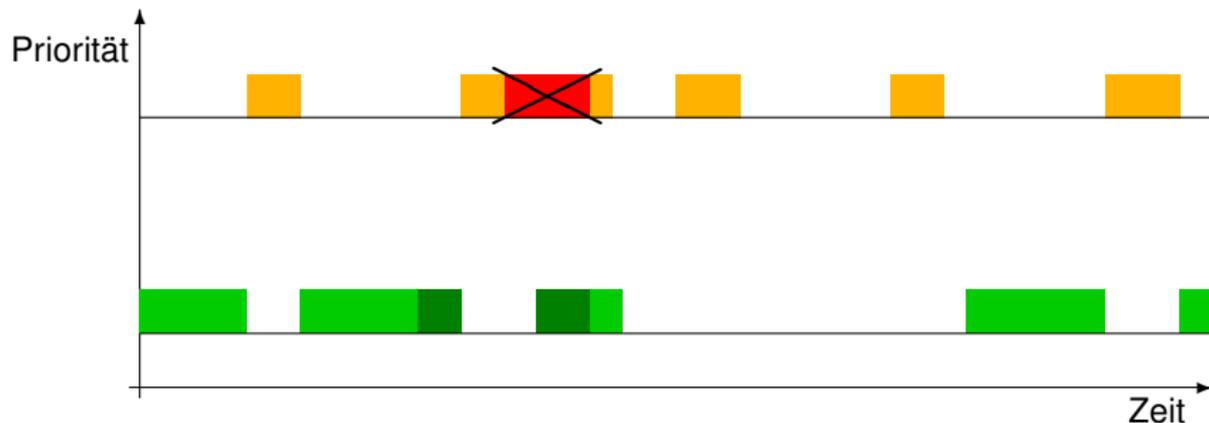
5 Echtzeit

5.5 Prioritäten und Ressourcen

Der höher priorisierte Prozeß bewirkt selbst, daß er eine Ressource verspätet bekommt.

→ *begrenzte Prioritätsinversion*

maximale Verzögerung: Länge des kritischen Bereichs



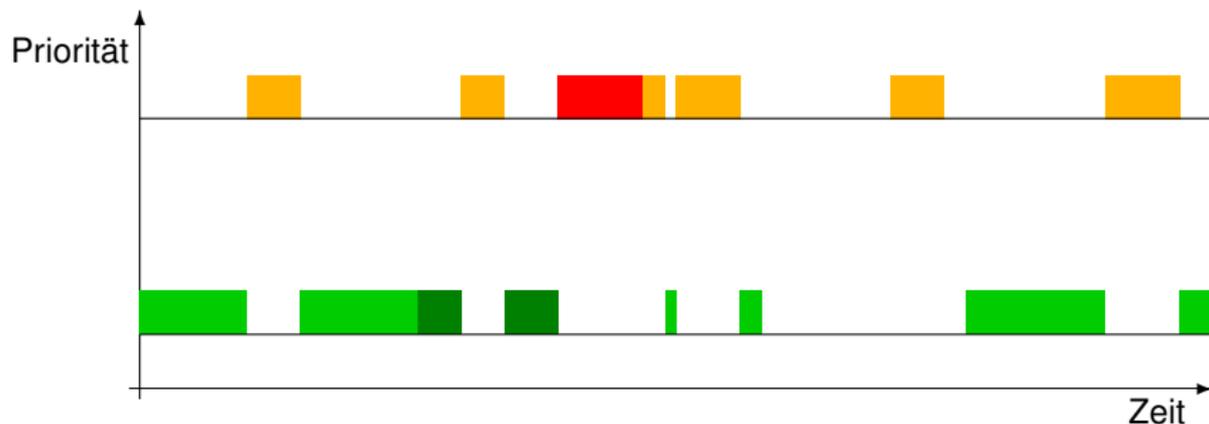
5 Echtzeit

5.5 Prioritäten und Ressourcen

Der höher priorisierte Prozeß bewirkt selbst, daß er eine Ressource verspätet bekommt.

→ *begrenzte Prioritätsinversion*

maximale Verzögerung: Länge des kritischen Bereichs



5 Echtzeit

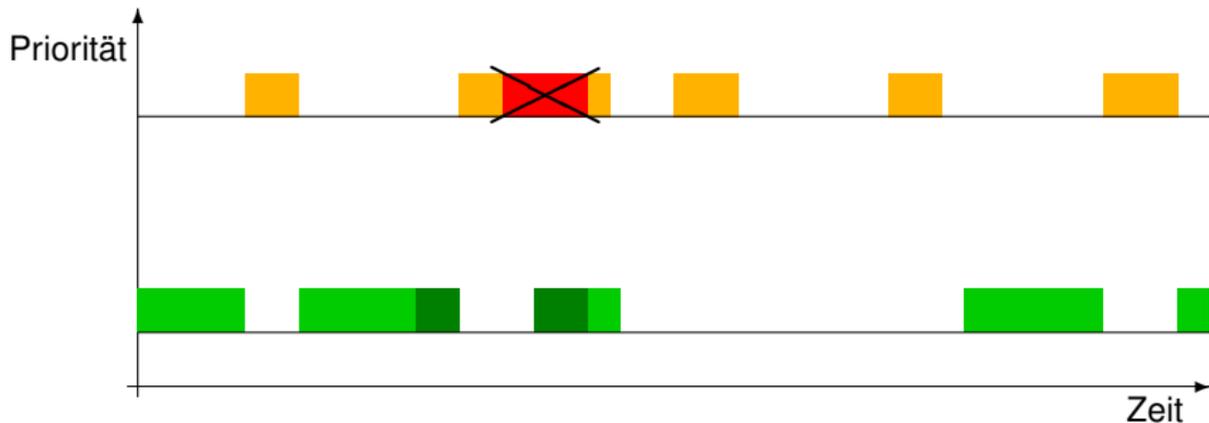
5.5 Prioritäten und Ressourcen

unbegrenzte Prioritätsinversion

5 Echtzeit

5.5 Prioritäten und Ressourcen

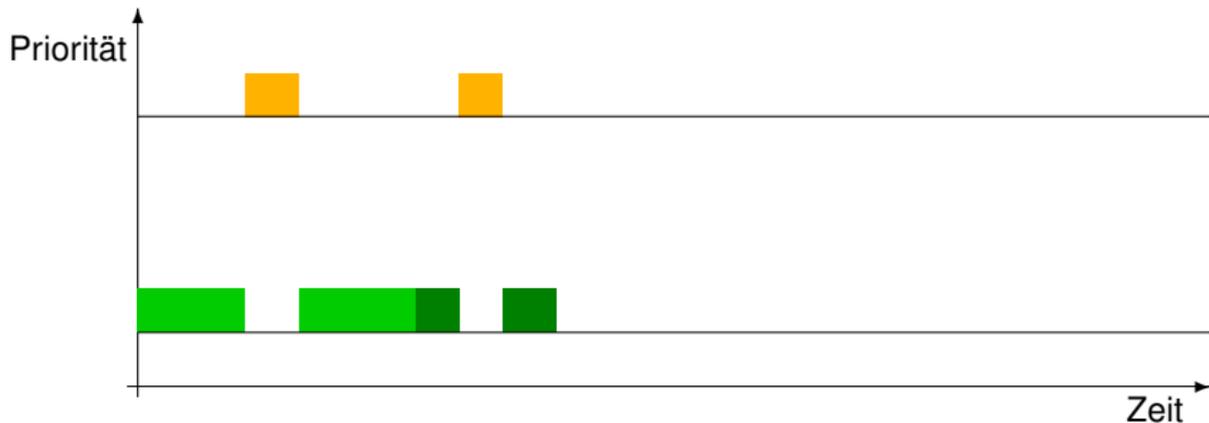
unbegrenzte Prioritätsinversion



5 Echtzeit

5.5 Prioritäten und Ressourcen

unbegrenzte Prioritätsinversion

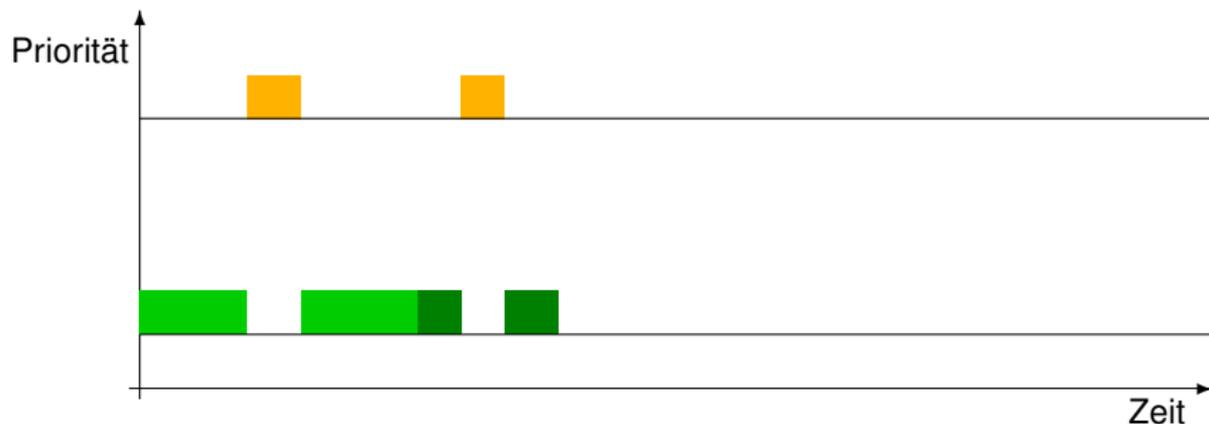


5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

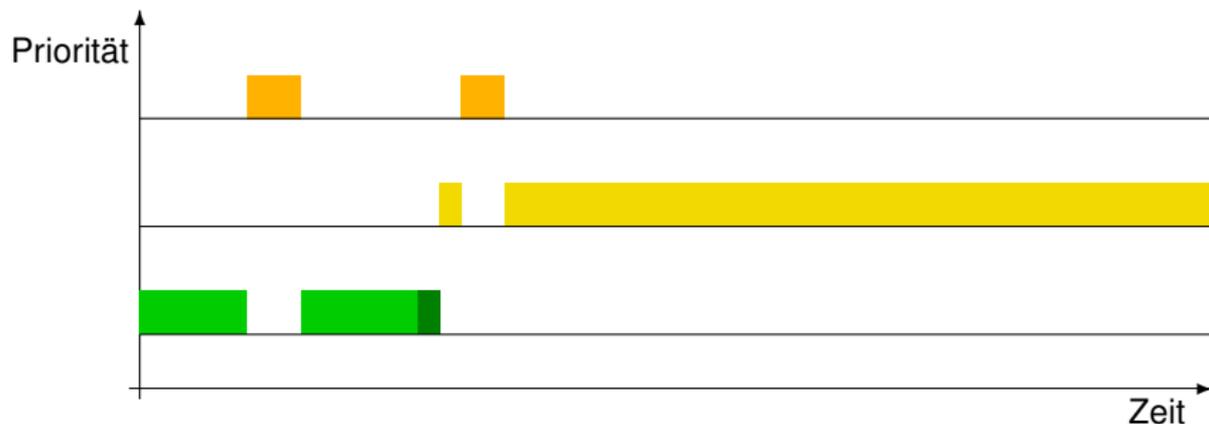


5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*



5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

—→ *unbegrenzte Prioritätsinversion*

Beispiel: Beinahe-Verlust der Marssonde *Pathfinder* im Juli 1997



5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

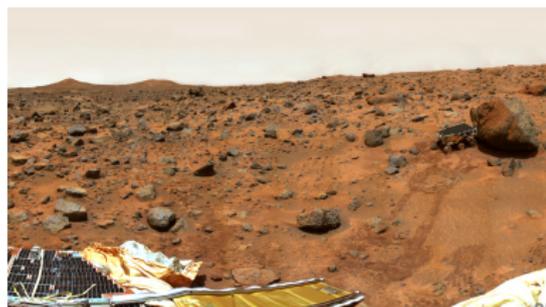
—→ *unbegrenzte Prioritätsinversion*

Beispiel: Beinahe-Verlust der Marssonde *Pathfinder* im Juli 1997

Gegenmaßnahmen

- *Priority Inheritance – Prioritätsvererbung*
Der Besitzer des Mutex erbt die Priorität des Prozesses, der auf den Mutex wartet.

http://research.microsoft.com/en-us/\um/people/mbj/Mars_Pathfinder/



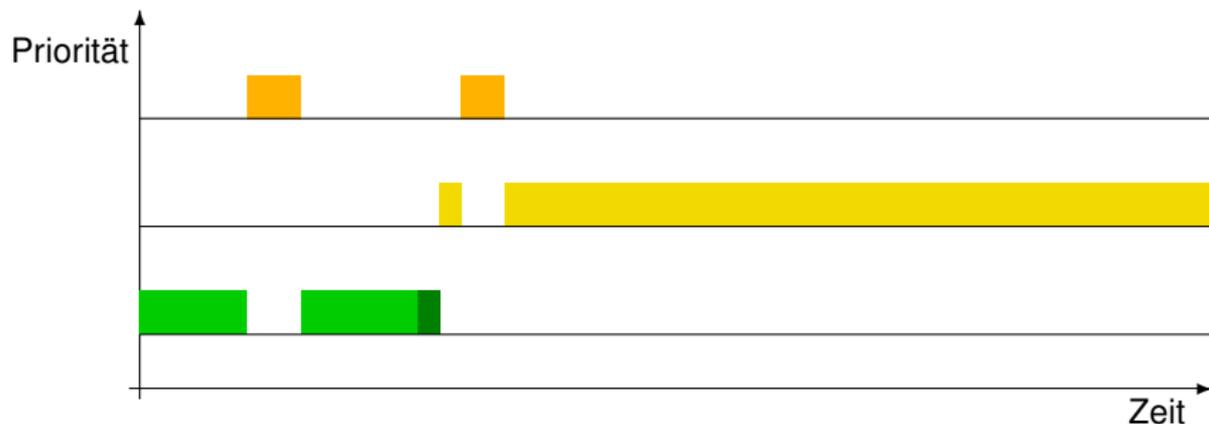
5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

Gegenmaßnahme: *Priority Inheritance – Prioritätsvererbung*



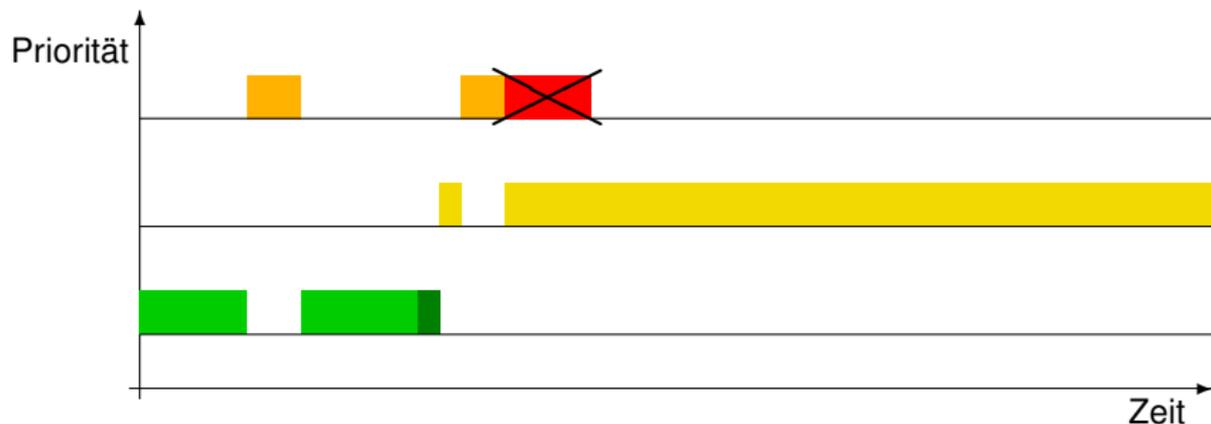
5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

Gegenmaßnahme: *Priority Inheritance – Prioritätsvererbung*



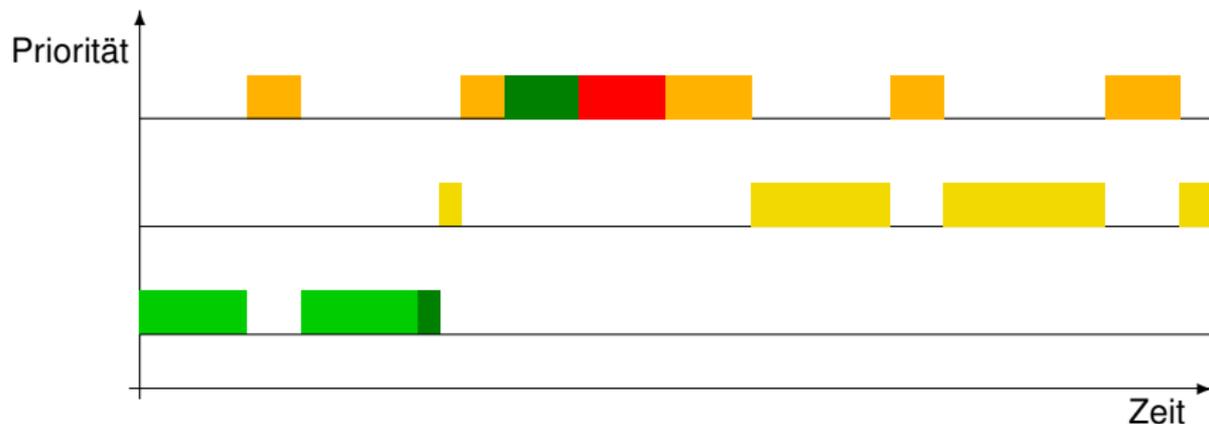
5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

Gegenmaßnahme: *Priority Inheritance – Prioritätsvererbung*



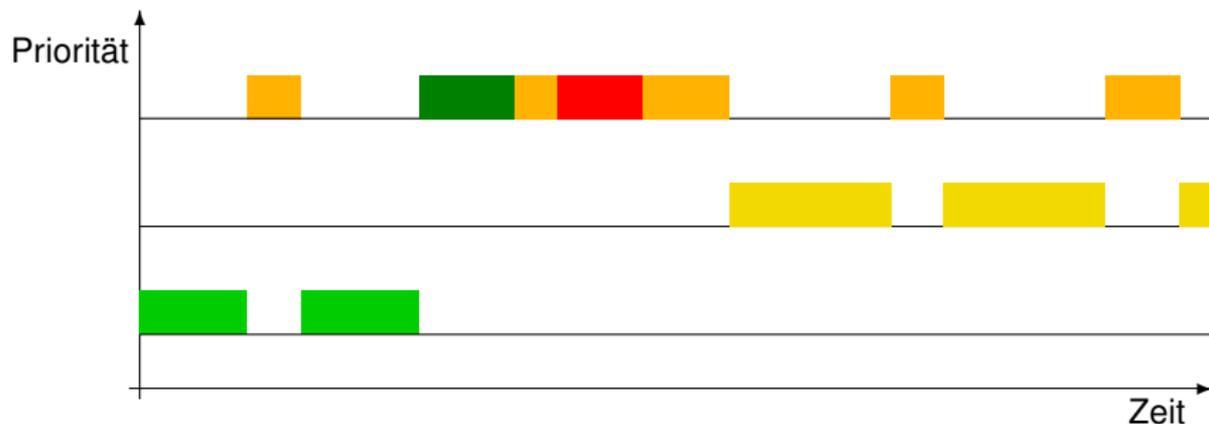
5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

Gegenmaßnahme: *Priority Ceiling – Prioritätsobergrenze*



5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

Beispiel: Beinahe-Verlust der Marssonde *Pathfinder* im Juli 1997

Gegenmaßnahmen

- *Priority Inheritance – Prioritätsvererbung*
Der Besitzer des Mutex erbt die Priorität des Prozesses, der auf den Mutex wartet.
- *Priority Ceiling – Prioritätsobergrenze*
Der Besitzer des Mutex bekommt sofort die Priorität des höchstmöglichen Prozesses, der evtl. den Mutex benötigen könnte.

5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

Beispiel: Beinahe-Verlust der Marssonde *Pathfinder* im Juli 1997

Gegenmaßnahmen

- *Priority Inheritance – Prioritätsvererbung*
Der Besitzer des Mutex erbt die Priorität des Prozesses, der auf den Mutex wartet.
- *Priority Ceiling – Prioritätsobergrenze*
Der Besitzer des Mutex bekommt sofort die Priorität des höchstmöglichen Prozesses, der evtl. den Mutex benötigen könnte.

} nur möglich, wenn
Mutexe im Spiel sind

5 Echtzeit

5.5 Prioritäten und Ressourcen

Ein Prozeß mit mittlerer Priorität bewirkt, daß ein Prozeß mit hoher Priorität eine Ressource überhaupt nicht bekommt.

→ *unbegrenzte Prioritätsinversion*

Beispiel: Beinahe-Verlust der Marssonde *Pathfinder* im Juli 1997

Gegenmaßnahmen

- *Priority Inheritance – Prioritätsvererbung*

Der Besitzer des Mutex erbt die Priorität des Prozesses, der auf den Mutex wartet.

- *Priority Ceiling – Prioritätsobergrenze*

Der Besitzer des Mutex bekommt sofort die Priorität des höchstmöglichen Prozesses, der evtl. den Mutex benötigen könnte.

- *Priority Aging*

Die Priorität wächst mit der Wartezeit.

} nur möglich, wenn
Mutexe im Spiel sind

Vertiefung Systemtechnik – 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 I²C (TWI)

5.4 SPI

5.5 Beispiel: Benutzung des I²C-Busses

6 Echtzeit

5.1 Was ist Echtzeit?

5.2 Echtzeitprogrammierung

5.3 Multitasking

5.4 Ressourcen

5.5 Prioritäten

...



Änderungen
vorbehalten