

# System-Programmierung o: Einführung

CC BY-SA, Thomas Amberg, FHNW  
(soweit nicht anders vermerkt)



## Ablauf heute

⅓ Vorlesung,  
⅔ Hands-on,  
Feedback.

Slides, Code & Hands-on: [tmb.gr/syspr-o](https://tmb.gr/syspr-o)



## Hallo

Thomas Amberg (@[tamberg](https://twitter.com/tamberg)), Software Ingenieur.  
Neu an der FHNW als Prof. für Internet of Things.  
Gründer von [Yaler](https://yaler.ch), "sicherer Fernzugriff für IoT".  
Organisator [IoT Meetup](https://iotschweiz.ch), [Maker Faire](https://makerfaire.ch) in Zürich.

[thomas.amborg@fhnw.ch](mailto:thomas.amborg@fhnw.ch)



## Ausgangslage

Betriebssysteme (*bsys*)?

System-Administration (*sysad*)?

Java, C, andere Programmiersprachen?

Wer benutzt MacOS, Windows (10, 8, 7), Linux?



## Aufbau Modul *syspr*

15 \* 3 = 45 Stunden Unterricht:

⅓ Vorlesung plus ⅔ Hands-on.

Dazu ca. 45 Stunden Selbststudium.

Total 90 Stunden, d.h. 3 ECTS Punkte.



## Lernziele Modul *syspr*

Programmierung in C, da der Unix/Linux-Kern und Basisanwendungen in der Sprache geschrieben sind.

Praktische Nutzung der System-Call Schnittstelle von Unix/Linux lernen anhand von Beispielprogrammen.

Kommunikation zwischen Prozessen (IPC) und deren Synchronisation verstehen und einsetzen lernen.



## Termine 2018/19

17.09. Einführung	12.11. IPC mit Pipes
24.09. Erste Schritte in C	19.11. Sockets
01.10. Funktionen	26.11. (Projektwoche)
08.10. File In-/Output	03.12. Weitere Arten von I/O
15.10. Prozesse und Signale	10.12. Terminals
22.10. Prozess-Lebenszyklus	17.12. POSIX IPC
29.10. Threads und Synchr.	07.01. Zeitmessung
05.11. Assessment I	14.01. Assessment II

n|w

## Lernzielüberprüfung

Zwei obligatorische Assessments von je 2 Stunden.

Fliessen zu je 50% in die Gesamtbewertung ein.

Die Schlussnote wird auf Zehntel gerundet.

Es gibt keine Modulschlussprüfung.

n|w

## Während den Assessments

Eine (mehreseitige) **C-Referenzkarte** ist erlaubt.

Die Karte soll keinen Beispielcode enthalten.

Weitere Unterlagen sind nicht erlaubt.

Das Assessment ist schriftlich.

n|w

## Im Unterricht

Sie brauchen einen Computer mit Admin-Rechten.

Prüfungsstoff = Inhalt von Slides und Hands-on.

Slides und Hands-on als PDF, mit vielen Links.

Code-Beispiele sind von Slides aus verlinkt.

n|w

## Während Hands-on Sessions

"Be excellent to each other", Fragen / Helfen ist OK.

Google ([DDG.co](http://DDG.co), ...) nutzen um Fehler zu beheben.

Blind kopieren bringt keine neuen Einsichten.

Fremden, guten Code lesen hingegen schon.

n|w

## Ablage Slides, Code & Hands-on

<http://tmb.gr/syspr> →

<https://github.com/tamberg/fhnw-syspr>

```
01/  
  hello.c  
  README.md → Slides, Hands-on  
02/  
...
```

n|w

## Abgabe Hands-on Resultate via GitHub

<https://github.com/fhnw-syspr-3ia> bzw. [-3ib](https://github.com/fhnw-syspr-3ib)

fhnw-syspr-work-01	Repo Vorlage mit Link
fhnw-syspr-work-01-USER	Repo Kopie pro User
README.md	Hands-on Aufgaben
my_result.c	"Privat", Dozent & User

Hands-on zählt zum Prüfungsstoff, GitHub nicht.  
Abgabe ist freiwillig, falls Feedback erwünscht.



## Kommunikation mit Slack

<https://fhnw-syspr.slack.com/>

#general	Allg. Fragen und Ankündigungen.
#random	Eher Unwichtiges, Zufälliges.
#c-lang	C spezifische Fragen.
#...	Weitere Channels.
• tamberg	Messages an eine Person, "privat".

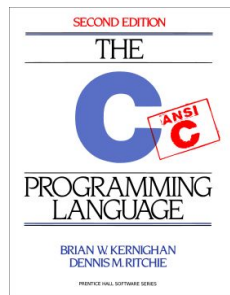
[Slack App](#) wird empfohlen, mobile oder Desktop.



## Literatur

<https://ddg.co/?q=the+c+programming+language+kernighan+ritchie>

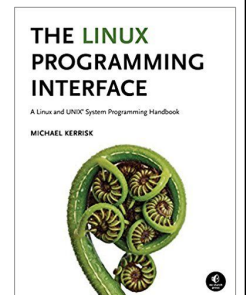
Klassiker, 270 Seiten.



## Literatur (optional)

<https://ddg.co/?q=the+linux+programming+interface>

Nachschlagwerk,  
1500+ Seiten.



## Tools

*Terminal* (MacOS) bzw. *cmd* (Windows).

Text-Editor, z.B. *nano* oder [VS Code](#).

C Compiler, *gcc* / Debugger, *gdb*.

Code Versionierung mit *git*.

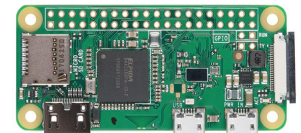
Einfache Tools, ohne "Magie" => Verständnis.



## Raspberry Pi

Einplatinencomputer

<https://raspberrypi.org/products/raspberry-pi-zero-w/>



1GHz, single core ARM CPU, 512 MB RAM,  
Mini HDMI, USB On-The-Go, Wi-Fi, Bluetooth, etc.

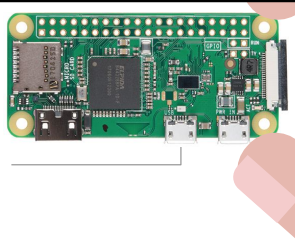
Leihweise, inklusive USB Kabel, gegen Unterschrift.



## Raspberry Pi Setup

[Raspbian](#) "Stretch Lite"  
Linux IMG auf SD Card.

Internet-Sharing via USB.



Getestet auf MacOS und Windows 10.

n|w

## Wieso Raspberry Pi?

Günstige Hardware.

Interessante Schnittstellen.

Separates System => "Sandbox".

SD Card neu schreiben => "Factory reset".

Embedded Linux Systeme sind relevant für IoT.

n|w

## Raspberry Pi SD Card erstellen

[Etcher](#) Tool installieren, [Raspbian](#) "Stretch Lite" IMG  
runterladen und mit Etcher auf leere SD Card spielen.

(IMG Datei auf SD Card kopieren geht nur mit Tool.)

Um SSH einzuschalten, leere Datei *ssh* erstellen:

MacOS, Linux:

```
$ cd /Volumes/boot
```

```
$ touch ssh
```

Windows:

```
C:\> E:
```

```
E:\> type nul > ssh
```

n|w

## Raspberry Pi Zero W als USB Gadget

Auf SD Card in *config.txt* neue Zeile *dtoverlay=dwc2*:

```
$ open config.txt
```

```
...
```

```
dtoverlay=dwc2
```

In *cmdline.txt* nach *rootwait* diesen Text einfügen:

```
$ open cmdline.txt
```

```
... rootwait modules-load=dwc2,g_ether ...
```

(Windows: *open* durch *notepad* ersetzen.)

n|w

## Internet-Sharing von Wi-Fi zu USB

MacOS

System Preferences > Sharing > [✓] Internet  
Sharing > Share your connection from: Wi-Fi  
to computers using RNDIS Ethernet Gadget

Windows (vorher [Bonjour](#) installieren)

```
WINDOWS-R > ncpa.cpl > CTRL-Klick Wi-Fi und  
RNDIS Ethernet adapter > Rechtsklick > Bridge
```

Oder Wi-Fi > Properties > Sharing > [✓] Allow

## Wi-Fi Konfiguration zu Hause (optional)

Auf Raspi, Datei *wpa\_supplicant.conf* ergänzen, mit:

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

```
...
```

```
network={  
    ssid="WIFI_SSID"  
    psk="WIFI_PASSWORD"  
    key_mgmt=WPA-PSK  
}
```

Oder via SD Card, in */boot/wpa\_supplicant.conf*

n|w

## Wi-Fi Konfiguration für fhnw-private

```
$ echo -n 'PASSWORD' | iconv -t utf16le | openssl md4  
=> PW_HASH, e.g. 62f6e1dc44a0eac6784f134e1c2c2b03  
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

```
network={  
    ssid="fhnw-private"  
    scan_ssid=1  
    priority=1  
    proto=RSN  
    key_mgmt=WPA-EAP  
    pairwise=CCMP  
    auth_alg=OPEN  
    eap=PEAP  
    identity="FHNW_EMAIL"  
    password=hash:PW_HASH  
    phase1="peaplabel=0"  
    phase2="auth=MSCHAPV2"  
}
```

n|w

## Zugriff auf den Raspberry Pi mit SSH

Auf Windows mit dem PuTTY Tool:

Host: raspberrypi.local, Port: 22, User: pi

Auf MacOS und Linux mit ssh:

```
$ ssh pi@raspberrypi.local
```

Oder ssh mit IP Adresse, z.B.

```
$ ssh pi@192.168.0.42  
pi@192.168.0.42's password: raspberry
```

n|w

## Raspberry Pi finden im lokalen Netzwerk

IP Adresse finden, auf MacOS und Linux mit dns-sd:

```
$ dns-sd -G v4 raspberrypi.local
```

Oder mit ifconfig (bzw. ipconfig) und nmap:

```
$ ifconfig  
en0: ... inet 192.168.0.23
```

```
$ nmap 192.168.0.0-255 -p 22  
Nmap scan report for 192.168.0.42  
22/tcp open  ssh
```

Achtung:  
Keine Port  
Scans an  
der FHNW!

n|w

## Linux/Unix Shell Kommandos

\$ ls	Directory auflisten
\$ mkdir my_directory	Directory erstellen
\$ cd my_directory	Directory öffnen
\$ echo "my file" > my_file	Datei erstellen
\$ cat my_file	Datei anzeigen
\$ rm my_file	Datei löschen
\$ man rm	Doku zu rm anzeigen

Mehr hier oder auf tldr.sh (auch als PDF).

n|w

## Textdatei erstellen auf dem Raspberry Pi

Copy & Paste in eine neue Datei hello.c:

```
$ nano hello.c {Text einfügen}
```

Speichern und nano beenden:

```
CTRL-X Y ENTER
```

Anzeigen der Datei:

```
$ cat hello.c
```

n|w

## Datei kopieren zum/vom Raspberry Pi

Auf Windows mit dem WinSCP Tool.

Auf MacOS oder Linux mit FileZilla oder scp.

Datei vom Computer zum Raspberry Pi kopieren:

```
$ scp -P 22 LOCAL_FILE pi@RASPI_IP:RASPI_PATH
```

Bzw. vom Raspberry Pi auf den Computer kopieren:

```
$ scp -P 22 pi@RASPI_IP:RASPI_FILE LOCAL_PATH
```

n|w

## Datei runterladen auf den Raspberry Pi

Datei runterladen mit *wget*:

```
$ wget -O LOCAL_PATH REMOTE_URL
```

```
$ wget -O hello.c https://raw.githubusercontent.com/leachim6/hello-world/master/c/c.c
```

Oder, wenn der Ziel-Dateiname identisch ist:

```
$ wget https://raw.githubusercontent.com/antirez/kilo/master/kilo.c
```

n|w

## Hands-on, 1h: Raspberry Pi Grundlage für das ganze Modul syspr.

Raspberry Pi Setup via USB zum eigenen Computer.

"Hello World" in C auf Raspberry Pi speichern.

Den C Source Code mit *gcc* kompilieren.

```
$ gcc -o hello hello.c
```

```
$ ./hello
```

Fertig? Bitte Nachbarn helfen.

n|w

## Source Code Versionierung mit Git

Account erstellen auf [GitHub.com](https://github.com).

```
=> USER_NAME, USER_EMAIL
```

Auf dem Raspberry Pi, *git* installieren mit *apt-get*:

```
$ sudo apt-get update
```

```
$ sudo apt-get install git
```

Installation prüfen:

```
$ git
```

n|w

## Git konfigurieren auf dem Raspberry Pi

User konfigurieren:

```
$ git config --global user.email "USER_EMAIL"
```

```
$ git config --global user.name "USER_NAME"
```

SSH Key erstellen:

```
$ ssh-keygen -t rsa -b 4096 -C "USER_EMAIL"
```

```
$ eval "$(ssh-agent -s)"
```

```
$ cat ~/.ssh/id_rsa.pub
```

n|w

## GitHub konfigurieren

Raspberry Pi **SSH Key eintragen** auf [GitHub](https://github.com):

User Icon > Settings > SSH and GPG keys > New SSH key > {SSH Key einfügen}

Auf Raspberry Pi, Passphrase ablegen in *keychain*:

```
$ sudo apt-get install keychain
```

```
$ keychain ~/.ssh/id_rsa
```

```
$ . ~/.keychain/$HOSTNAME-sh
```

(Bei Reboot wird der *keychain* Cache gelöscht.)

n|w

## GitHub Repository klonen

(GitHub Repository [erstellen](#).)

GitHub Repository klonen:

```
$ git clone git@github.com:USER_NAME/REPO.git
```

Neue Datei hinzufügen:

```
$ cd REPO
```

```
$ nano my.c
```

```
$ git add my.c
```

n|w

## Git verwenden auf dem Raspberry Pi

Geänderte Dateien anzeigen:

```
$ git status
```

Änderungen committen:

```
$ git commit -a -m "fixed all bugs"
```

Änderungen pushen:

```
$ git push
```

Mehr zu [git](#) [hier](#).

n|w

## Hands-on, 1h: GitHub

*Grundlage für das  
ganze Modul syspr.*

GitHub Account einrichten, falls keiner vorhanden.

Git auf Raspberry Pi installieren und konfigurieren.

<https://github.com/tamberg/fhnw-syspr> klonen.

Daneben, Übungs-Repository aus Mail\* klonen.

Code in Übungs-Repo committen, pushen.

\*) Keine Mail bekommen? Bitte melden.

n|w

## Selbststudium, 3h: Pointers and Arrays

Suchen Sie eine [C-Referenzkarte](#) als Übersicht zu C.

Als Vorbereitung auf die nächste Lektion, lesen Sie [\[K&R\] Chapter 5: Pointers and Arrays](#) bis p.126.

Die nächste Lektion fasst dann beides zusammen, ohne Selbststudium wird das Tempo eher hoch sein.

n|w

## Feedback?

Gerne im [Slack](#) oder an [thomas.amberg@fhnw.ch](mailto:thomas.amberg@fhnw.ch)

Programmierfragen am besten schriftlich.

Sprechstunde auf [Vor Anmeldung](#) :)

Slides, Code & Hands-on: [tmb.gr/syspr-o](http://tmb.gr/syspr-o)

