System-Programmierung 0: Einführung

Ablauf heute

⅓ Vorlesung,

⅔ Hands-on,

Feedback.

Slides & Hands-on: tmb.gr/syspr-0

Hallo

Thomas Amberg (@tamberg), Software Ingenieur.

Neu an der FHNW als Prof. für Internet of Things.

Gründer von Yaler, "sicherer Fernzugriff für IoT".

Organisator IoT Meetup, Maker Faire in Zürich.

thomas.amberg@fhnw.ch

Ausgangslage

Betriebssysteme (bsys)?

System-Administration (sysad)?

Java, C, andere Programmiersprachen?

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

Aufbau Modul *syspr*

15 * 3 = 45 Stunden Unterricht:

⅓ Kontaktunterricht, ⅔ Übungen.

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
24.09.	Erste Schritte in C
01.10.	Funktionen
08.10.	Datei In-/Output
15.10.	Prozesse und Signale
22.10.	IPC mit Pipes
29.10.	Vorbereitung
05.11.	Assessment I
12.11.	Shared Memory
19.11.	Message Queues
26.11.	(Projektwoche)
03.12.	Semaphore
10.12.	RPC
17.12.	Sockets
	Ferien
07.01.	Vorbereitung
14.01.	Assessment II

Lernzielüberprüfung

Zwei obligatorische Assessments von je 60 Minuten.

Fliessen zu je 50% in die Gesamtbewertung ein.

Die Schlussnote wird auf Zehntel gerundet.

Es gibt keine Modulschlussprüfung.

In der Prüfung

Eine (mehrseitige) C-Referenzkarte ist erlaubt.

Die Karte soll keinen Beispielcode enthalten.

Weitere Unterlagen sind nicht erlaubt.

Im Unterricht

Handy auf Lautlos / Vibration.

Laptop mit Administrator-Rechten.

Slides und Übungen als PDF, mit Links.

Source Code auf Slides ist oft nur ein Auszug, der komplette Code ist jeweils vom Slide aus verlinkt.

In der Übungsstunde

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

Google (DDG.co, …) nutzen um Fehler zu beheben:
C program "command not found" [1] => ./hello [2]

Blind kopieren bringt keine neuen Einsichten.

Fremden, guten Code lesen hingegen schon.

Ablage Slides, Hands-on & Beispiele

http://tmb.gr/syspr →

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

- 01/
  - Syspr01ErsteSchritteInC.pdf
  - Syspr01HandsOnCCompilieren.pdf
  - hello.c
- …

Abgabe Übungs-Resultate via GitHub

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

/uebung-01	Repository Vorlage.
/uebung-01-USER_NAME	Generiert, 1 pro User.
my_result.c	Public, bis Ende syspr.

Übungen zählen zum Prüfungsstoff, GitHub nicht.
Abgabe ist freiwillig, falls Feedback erwünscht.

Kommunikation mit Slack

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

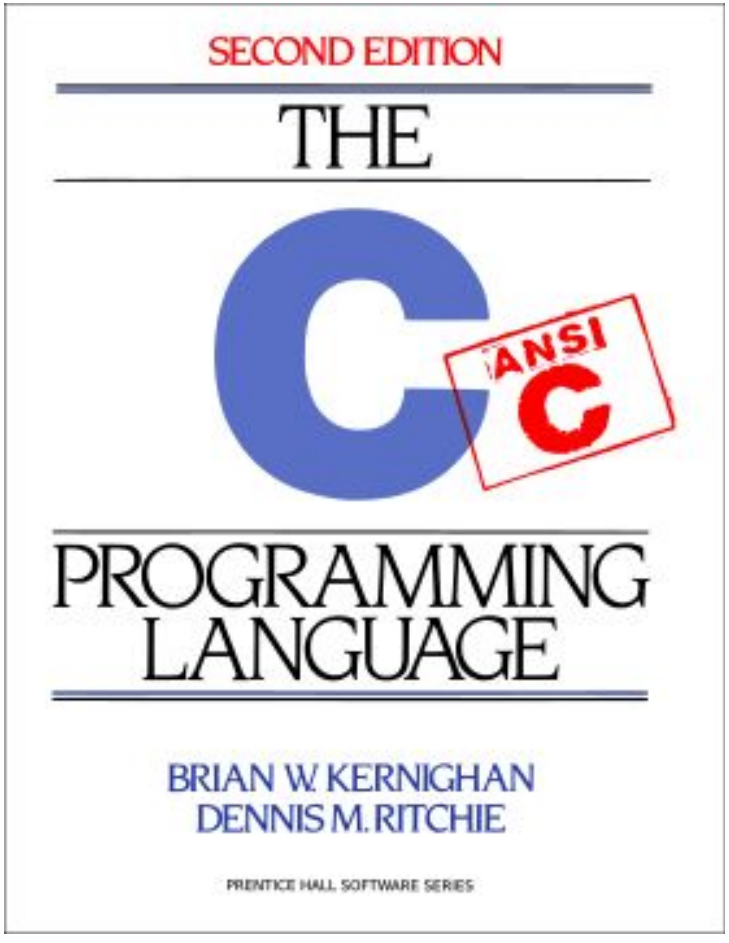
#general	Messages die alle sehen.
#github	Notifications bei Github-Updates.
• tamberg	Messages an eine Person, "privat".

Slack App wird empfohlen, mobile oder Desktop.

Literatur (optional)

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

Klassiker, 270 Seiten.

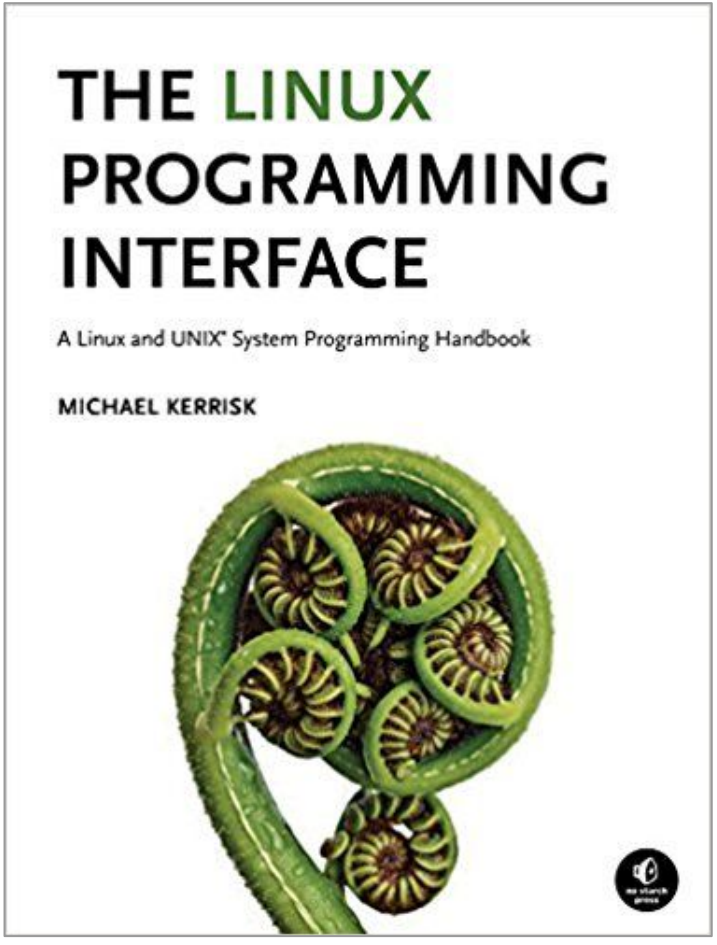


Literatur (optional)

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

Nachschlagwerk,

1500+ Seiten.



Tools

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

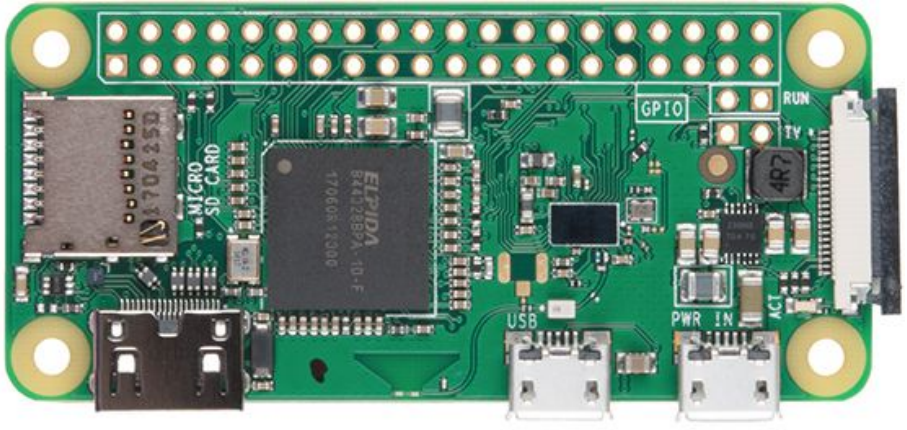
Text-Editor, z.B. *nano* oder VS Code.

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/

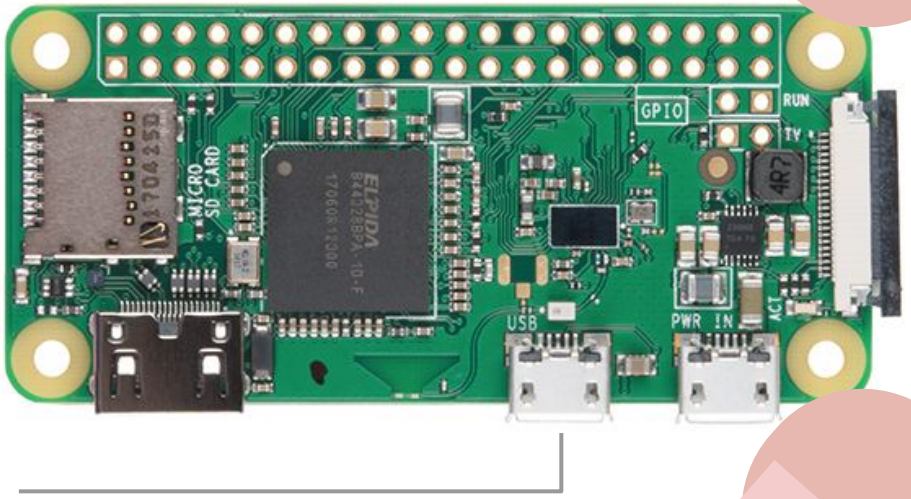
1GHz, single core ARM CPU, 512 MB RAM,
Mini HDMI, USB On-The-Go, Wi-Fi, Bluetooth, etc.

Leihweise, inklusive USB Kabel, gegen CHF 20 Depot.

Raspberry Pi Setup

Raspbian "Stretch Lite"
Linux IMG auf SD Card.

Internet-Sharing via USB.



Getestet auf MacOS und Windows 10.

Wieso Raspberry Pi?

Günstige Hardware.

Interessante Schnittstellen.

Separates System => "Sandbox".

SD Card neu schreiben => "Factory reset".

Embedded Linux Systeme sind relevant für IoT.

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
```

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 …
```

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 > SHIFT-click Wi-Fi and RNDIS Ethernet adapter > right-click > Bridge
```

Wi-Fi Konfiguration via SSH (optional)

Datei *wpa_supplicant.conf* ergänzen, mit:

```
$ sudo nano /etc/wpa_supplicant/wpa_supplican\
t.conf
```

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

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
```

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!

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).

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
```

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
```

Datei runterladen auf den Raspberry Pi

Datei runterladen mit *wget*:

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

```
$ wget -O hello.c https://raw.githubuser\
content.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
```

Git Versionierung auf dem Raspberry Pi

Account erstellen auf GitHub.

```
=> USER_NAME, USER_EMAIL
```

Git installieren mit *apt-get*:

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

Versionierung = Backup.

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
```

GitHub konfigurieren

Raspberry Pi SSH Key eintragen auf GitHub:

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

Auf Raspberry Pi, Passphrase cachen mit *keychain*:

```
$ sudo apt-get install keychain
$ keychain ~/.ssh/id_rsa
```

Bei Reboot wird der *keychain* Cache gelöscht.

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
```

Git verwenden auf dem Raspberry Pi

Geänderte Dateien anzeigen:

```
$ git status
```

Änderungen committen:

```
$ git commit -a -m "changed everything"
```

Änderungen pushen:

```
$ git push
```

Mehr zu *git* hier.

Hands-on

Raspberry Pi Setup via USB zum eigenen Computer.

Textdatei erstellen, kopieren und runterladen.

C Tools installieren auf Raspberry Pi.

"Hello World" in C kompilieren.

Code auf GitHub speichern.

Bonus

Ken Thompson and Dennis Ritchie Explain UNIX:
https://www.youtube.com/watch?v=JoVQTPbD6UY

Feedback?

Gerne jederzeit an

thomas.amberg@fhnw.ch

Slides & Hands-on: tmb.gr/syspr-0