

# Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

15. Dezember 2020

# Eingebettete Systeme

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

## 1 Einführung

## 2 Einführung in Unix

## 3 TCP/IP in der Praxis

### 3.0 Vorbereitungen

### 3.1 IP-Adressen

### 3.2 MAC-Adressen

### 3.3 TCP- und UDP-Ports

### 3.4 TCP-Protokolle

### 3.5 Routing

### 3.6 Netzwerkanalyse

### 3.7 SSH

### 3.8 X11 und VNC

### 3.9 Programmierung

## 4 Versionsverwaltungssysteme

## 5 Bus-Systeme

...

## 3.7 SSH

- Mit `nc -p 2345 -l -c /bin/bash` ermögliche ich Shell-Zugriff auf den Rechner über Port 2345.
- Achtung: Es erfolgt keine Verschlüsselung und noch nicht einmal eine Passwort-Abfrage!

Richtige Lösung mit Verschlüsselung und Authentifizierung: SSH

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

Authentifizierung

- Ich hinterlege den öffentlichen Schlüssel auf dem Ziel-Rechner.
- Wer dann im Besitz des privaten Schlüssels ist, kann sich dann auf dem Ziel-Rechner einloggen.
- Diese Methode ist sicherer als die Abfrage eines Passworts.

## 3.8 X11 und VNC

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

## 3.8 X11 und VNC

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`: X11-Forwarding
  
- VNC = Virtual Network Computing
- VNC-Server stellt Bildschirminhalt zur Verfügung
  - entweder: eigener, virtueller X11-Server
  - oder: ruft Inhalt von anderem (X11-) Bildschirm ab
- VNC-Client ruft Bildschirminhalt ab und stellt ihn dar
  - z. B. per X11
  - z. B. per Web-Interface: noVNC

## 3.9 Programmierung

- Shell-Skripte: nc (traditional oder OpenBSD)
- C: Sockets, select()
- C++: Callbacks

## 3.9 Programmierung

- Shell-Skripte: `nc` (`traditional` oder `OpenBSD`)
- C: Sockets, `select()`
- C++: Callbacks

Aktuelles Beispiel: Programmierung eines VNC-Servers per Web-Interface

- Grundlagen: RFC 6143
- Praxis: Untersuchung mit Wireshark
- Implementation: JavaScript, WebSockets, `websocketify`, Callbacks

# 4 Versionsverwaltungssysteme

## 4.1 Historisches

Bekannte Versionsverwaltungswerkzeuge:

**1974** diff

**1982** RCS – Revision Control System

**1985** patch

**1990** CVS – Concurrent Versions System

**2000** SVN – Subversion

**2005** Git

**2005** Mercurial

... und *vielen* weiteren!



# 4 Versionsverwaltungssysteme

## 4.2 Git

- Name: „git“ (engl.) = „Idiot“
- Initiator: Linus Torvalds  
*„I'm an egotistical bastard, and I name all my projects after myself.  
First ,Linux', now ,Git'.“*

# 4 Versionsverwaltungssysteme

## 4.2 Git

- Name: „git“ (engl.) = „Idiot“
- Initiator: Linus Torvalds  
*„I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'Git'.“*
- **Repository** – „Behälter“, in dem alle Dateien liegen  
– einschließlich sämtlicher historischer Versionsn
- bei Git: dezentral
- **Branches** – parallele Versionen desselben Repository
- bei Git: leistungsfähige Werkzeuge, um Branches wieder zusammenzuführen

# 4 Versionsverwaltungssysteme

## 4.2 Git

- *commit* – Änderung in das Repository übernehmen
- *stage* – für *commit* vorgemerkte Änderungen
- Unterverzeichnis *.git*: Konfiguration und Repository

Befehle:

- *git status* – Status anzeigen  
Welche Dateien sind in der *stage*? Welche sind neu?
- *git add* – Änderungen vormerken
- *git commit* – Änderungen in Repository übernehmen
- *git checkout -- <Datei>* – Datei aus Repository zurückholen
- *git push* – Änderungen in anderes Repository übertragen
- *git clean* – temporäre Dateien löschen
- *git init* – Verzeichnis für Arbeit mit Git vorbereiten