

# Betriebssystem-Entwicklung mit Literate Programming



TECHNISCHE HOCHSCHULE NÜRNBERG  
GEORG SIMON OHM

Wintersemester 2015/16

Dr. Hans-Georg Eßer

h.g.esser@gmx.de  
<http://ohm.hgesser.de/>

v1.2, 04.10.2015

Hans-Georg Eßer, TH Nürnberg  
BS-Entwicklung mit Literate Programming, WS 2015/16

Foliensatz 1: Einleitung  
Folie 1

## Hans-Georg Eßer

- Dipl.-Math. (RWTH Aachen, 1997)  
Dipl.-Inform. (RWTH Aachen, 2005)  
Dr.-Ing. (FAU Erlangen-Nürnberg, 2015)
- Chefredakteur einer Linux-Zeitschrift (seit 2000) und  
Autor diverser Linux-Bücher
- seit 2006 Lehraufträge an verschiedenen Hochschulen:  
Betriebssysteme, Rechnerarchitektur, Informatik-Grundlagen,  
Systemprogrammierung, Betriebssystem-Entwicklung u. a.
- 2010–2015 Doktorand an der Univ. Erlangen-Nürnberg –  
Thema: „Design, Implementation and Evaluation of the ULIX  
Operating System“
- **Betreuung von Abschlussarbeiten (für LBAs nicht mehr...)**

Hans-Georg Eßer, TH Nürnberg  
BS-Entwicklung mit Literate Programming, WS 2015/16

Foliensatz 1: Einleitung  
Folie 2

## Zur Vorlesung (1)

### Termine BS-Entwicklung

Vorlesung: Do 15:45–17:15 Uhr HQ 104  
Praktikum: Do 17:30–19:00 Uhr HQ 104

### Webseite

<http://ohm.hgesser.de/>

(Folien, Videos, Termine,  
sonstige Informationen)

Hans-Georg Eßer, TH Nürnberg  
BS-Entwicklung mit Literate Programming, WS 2015/16

Foliensatz 1: Einleitung  
Folie 3

## Zur Vorlesung (2)

### Hilfreiche Vorkenntnisse:

- **Vorlesung Betriebssysteme**
- **C** – Grundlagen der Programmierung in C  
(oder C++, C#, Java)  
→ C-Crashkurs (im Praktikumsteil)
- **Vorlesung Systemprogrammierung Linux**
- **Unix-Shell** – Benutzung der Standard-Shell *bash*  
unter Linux → Bash-Crashkurs

Hans-Georg Eßer, TH Nürnberg  
BS-Entwicklung mit Literate Programming, WS 2015/16

Foliensatz 1: Einleitung  
Folie 4

# Zur Vorlesung (3)

# Kontakt

## Termine im Wintersemester 2015/16

4 SWS, d. h.

- Vorlesung/Praktikum: 13 x 180 min.  
Raum HQ 104

## Service / Web-Seite

- Folien und Praktikumsaufgaben
- Vorlesungs-Videos

Sprechstunde: –

## Fragen:

- direkt in der Vorlesung
- oder danach
- oder per E-Mail: [h.g.esser@gmx.de](mailto:h.g.esser@gmx.de)

# Zur Vorlesung (4)

**Aufwand: 5 ECTS (1 ECTS-Punkt ≈ 25–30 h)**

**5 x 25 = 125 Stunden, davon:**

- 52 Stunden Präsenz (Vorlesung + Praktikum)
- ≤ 13 Stunden: Durcharbeiten der Aufgaben und Videos zu den Themen C-Programmierung und Bash-Shell (entfällt für Hörer der „Systemprogrammierung“)
- ≤ 15 Stunden: Selbständige Einarbeitung in LaTeX
- ≤ 20 Stunden: Lesen von Artikeln und Fachbüchern
- ≤ 25 Stunden: abschließende Bearbeitung der Übungsaufgaben mit Fertigstellung der Dokumentation (nach Vorlesungsende)
- Veranstaltung etwas aufwendiger als die Systemprogrammierung

```
Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from :ffff:87.234.201.207 port 61557
Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > 30d")
Sep 20 02:00:01 amd64 sshd[6516]: Accepted rsa for esser from :ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6699]: Accepted rsa for esser from :ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from :ffff:87.234.201.207 port 62514
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from :ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:37:11 amd64 sshd[10102]: Accepted rsa for esser from :ffff:87.234.201.207 port 63375
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[1410]: Accepted rsa for esser from :ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[1768]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[1768]: (root) CMD (/sbin/evlogmgr -c "age > 30d")
Sep 21 17:43:26 amd64 sshd[1763]: Accepted rsa for esser from :ffff:87.234.201.207 port 61897
Sep 21 17:43:26 amd64 sshd[1763]: Accepted rsa for esser from :ffff:87.234.201.207 port 61897
Sep 21 17:53:39 amd64 sshd[31263]: Accepted rsa for esser from :ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[4674]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[5499]: (root) CMD (/sbin/evlogmgr -c "age > 30d")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[2555]: (root) CMD (/sbin/evlogmgr -c "age > 30d")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6554]: Accepted publicity for esser from :ffff:192.168.1.5 port 59771 ssh2
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6696]: Accepted rsa for esser from :ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13251]: (root) CMD (/sbin/evlogmgr -c "age > 30d")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[20998]: Accepted rsa for esser from :ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[23197]: Accepted rsa for esser from :ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_mid_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from :ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[6621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[14841]: (root) CMD (/sbin/evlogmgr -c "age > 30d")
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from :ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from :ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[12930]: Accepted rsa for esser from :ffff:87.234.201.207 port 62029
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from :ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from :ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11601]: Accepted rsa for esser from :ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from :ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from :ffff:87.234.201.207 port 62778
```

# 1. Einführung / Motivation

## Warum BS-Entwicklung?

- logische Fortsetzung der Veranstaltungen Betriebssysteme und Systemprogrammierung
- verstehen,
  - wie Programme, Bibliotheken und Betriebssystem zusammen arbeiten
  - wie Aufgaben im BS gelöst werden
  - wie das BS (mehreren) Prozessen den Zugriff auf die Hardware ermöglicht
- und das anhand von konkretem BS-Code

## Eigenschaften von ULIX

- 32-Bit-System (x86)
- virtueller Speicher für BS und Prozesse, separate Adressräume (Paging)
- System Call Interface mit zugeh. Library für User-Mode-Anwendungen (Ulix-Programme)
- virtuelles Dateisystem (Disketten und Festplatten, Treiber für Minix v2)
- Mini-Shell mit integrierten Tools (pwd, cd, ls, cat, head, cp, mv, rm, ln, diff, hexdump, ps, kill u. a.)
- Implementation mit → Literate Programming

## Warum Unix?

- lange Tradition (Unix: 1969; C: 1972)
- Unix-Internas (Original-Unix oder modernere Varianten wie Linux, \*BSD etc.) gut dokumentiert bzw. Code frei verfügbar
- Übersichtliches Set an BS-Features (in Form von System Calls)
- kleine Unix-Systeme lassen sich mit wenig Code implementieren
  - z. B.: keine grafische Oberfläche

## Warum Literate Programming

- spezielle Form der Code-Entwicklung und Code-Dokumentation
- Literate Programs sind gut geeignet für die Präsentation von komplexem Code
  - unterstützt Bottom-Up- und Top-Down-Entwicklung
  - kann auch dokumentieren, wie der Code entstanden ist (kreativer Prozess)
- Einstieg in Literate Programming ist leicht



## 2. Crashkurse C und Bash

### Gliederung

1. Einleitung
2. C und Bash
3. Unix-Features
4. LaTeX / LitProg
5. Booten, Prozesse, Speicher
6. Interr., Syscalls
7. fork ()
8. Dateisysteme
9. Projekt

- Wenn Sie noch keine Kenntnisse in C und/oder in der Unix-Shell (bash) haben, nutzen Sie (außerhalb der Präsenztermine) die angebotenen Crash-Kurse und Videos.
- Ich beantworte (per Mail) Fragen zu C / bash, die bei der Bearbeitung der Aufgaben auftreten.

## 4. LaTeX und Literate Programming

### Gliederung

1. Einleitung
2. C und Bash
3. Unix-Features
4. LaTeX / LitProg
5. Booten, Prozesse, Speicher
6. Interr., Syscalls
7. fork ()
8. Dateisysteme
9. Projekt

- (Ultra-) Kurz-Einführung in LaTeX
- Prinzip des Literate Programming
- Kleine Beispiele mit LaTeX und HTML
- Übungen mit NoWeb (notangle, noweave)

## 3. Features eines Unix-Systems

### Gliederung

1. Einleitung
2. C und Bash
3. Unix-Features
4. LaTeX / LitProg
5. Booten, Prozesse, Speicher
6. Interr., Syscalls
7. fork ()
8. Dateisysteme
9. Projekt

- Überblick Unix / ULIX
- Prozesse
- Speicher (für BS und Prozesse)
- Dateisystem, VFS
- Interrupt-Behandlung
- System Calls
- Shell
- Scheduler

## 5. Booten, Protected Mode, Prozesse, Speicher

### Gliederung

1. Einleitung
2. C und Bash
3. Unix-Features
4. LaTeX / LitProg
5. Booten, Prozesse, Speicher
6. Interr., Syscalls
7. fork ()
8. Dateisysteme
9. Projekt

- Wie bootet ein (x86-) PC?
- x86: Real Mode vs. Protected Mode
- Prozesse
  - Process Control Block
  - Prozess-Erzeugung
  - Scheduling / Context Switch
- Speicher
  - Paging: Adressbereiche, Seitentabellen

# 6. System Calls in UNIX

## Gliederung

- 1. Einleitung
- 2. C und Bash
- 3. Unix-Features
- 4. LaTeX / LitProg
- 5. Booten, Prozesse, Speicher
- 6. Interr., Syscalls
- 7. fork ()
- 8. Dateisysteme
- 9. Projekt

- Interrupts auf x86 (im Prot. Mode)
- UNIX: Interrupt Handler
- Beispiel: Timer-Handler
- Software-Interrupt (int 0x80)
- UNIX: System-Call-Interface
- Beispiel: Syscall für read ()

# 8. Dateisysteme

## Gliederung

- 1. Einleitung
- 2. C und Bash
- 3. Unix-Features
- 4. LaTeX / LitProg
- 5. Booten, Prozesse, Speicher
- 6. Interr., Syscalls
- 7. fork ()
- 8. Dateisysteme
- 9. Projekt

- Mehrere Schichten
  - Hardware: Zugriff auf Disketten und Festplatten (readblock, writeblock)
  - Logisch: Minix v2 Dateisystem
    - Superblock, Maps, Inodes, Dateien, Verzeichnisse (mit Inhalten aus der Vorlesung Systemprogrammierung; Lektüre)
    - Indirektionsblöcke
  - Virtuelles Dateisystem
  - User-Mode-Funktionen (open, read, write etc.)

# 7. Prozess-Verdopplung mit fork()

## Gliederung

- 1. Einleitung
- 2. C und Bash
- 3. Unix-Features
- 4. LaTeX / LitProg
- 5. Booten, Prozesse, Speicher
- 6. Interr., Syscalls
- 7. fork ()
- 8. Dateisysteme
- 9. Projekt

- fork () auf allen Ebenen:
  - fork ()-Implementierung im UNIX-Kernel
    - Was wird kopiert?
    - Was wird geändert?
    - Wie aktiviert UNIX den neuen Prozess?
  - fork () als Bibliotheksfunktion
  - fork () im Programm

# 9. Projekt

## Gliederung

- 1. Einleitung
- 2. C und Bash
- 3. Unix-Features
- 4. LaTeX / LitProg
- 5. Booten, Prozesse, Speicher
- 6. Interr., Syscalls
- 7. fork ()
- 8. Dateisysteme
- 9. Projekt

- Abhängig vom Tempo, mit dem wir durch die vorherigen Themen kommen:
  - Start im laufenden Semester oder als Abschlussaufgabe
- Mittelgroßes Implementierungsprojekt
- Dokumentation im Literate-Programming-Stil

# Literatur

- H.-G. Eßer, F. C. Freiling: The Design and Implementation of the UNIX Operating System, 2015 (verfügbar über Kurswebseite)
- F. Mittelbach, M. Goossens: Der LaTeX-Begleiter, 2010, ISBN: 386894088X
- D. E. Knuth: Literate Programming, The Computer Journal, 27(2), pp. 97-111, 1984, <http://literateprogramming.com/knuthweb.pdf>
- N. Ramsey: Literate Programming Simplified, IEEE Software, 11(5):97-105, 1994, <http://literateprogramming.com/lpsimp.pdf>
- N. Ramsey: A One-Page Guide to Using noweb with LaTeX, <http://literateprogramming.com/nowebpg1.pdf>
- A. S. Tanenbaum, A. S. Woodhull: Operating Systems Design and Implementation, 3rd edition, ISBN 0131429388 bzw. 0135053765 (TB)
- W. R. Stevens, S. A. Rago: Advanced Programming in the Unix Environment, 2013, ISBN 0321637739
- Helmut Herold: Linux/Unix Systemprogrammierung, 2004, ISBN: 3827321603
- Elias Fischer: Das C-Tutorial, <http://www.c-howto.de/>
- Ted Jensen: A Tutorial on Pointers and Arrays in C, 2003, <http://home.earthlink.net/~momotuk/pointers.pdf>