Rechnerpraxis

05 Prozess- & Userverwaltung

Hochschule
für Technik
Stuttgart

SS19 | INF 1

Ben Lebherz
benjamin.lebherz@hft-stuttgart.de

Vorlesungsplan

Datum	Thema
19. März	Motivation & Organisation
2. April	Grundlagen Unix/Linux
11. April	Dateisystem und Rechteverwaltung
18. April	dotfiles & Linux Installation
25. April	**Prozesse und Benutzer**
2. Mai	Shell Scripting
9. Mai	VIM Editor
16. Mai	Suchen und Finden
23. Mai	???
30. Mai	Wiederholung und PVL-Vorbereitung
6. Juni	**PVL Termin**
13. Juni	*Pfingstferien*
20. Juni	Zusammenfassung & Feedback
27. Juni	Bonus: Wunschthema, Docker, AWS, ...

Prüfung / PVL

Am 6. Juni 2019 um XX:YY

Anmeldung in LSF *und* Moodle

LSF bis **Montag, 29. April 2019** | Moodle bis **Sonntag, 5. Mai 2019**

KEIN Internet!

Moodle Dateien/Folien erlaubt (keine Links)

Terminal/Shell erlaubt

Wiederholung

Betriebssysteme

- Abstraktionsschicht zwischen Hardware und Userspace/Apps
- Teilt CPU, RAM, ... zwischen Prozessen auf (**Scheduling**)
- Diverse Scheduling Algorithmen verfügbar

Kurztest 🎉

https://moodle.hft-stuttgart.de/mod/quiz/view.php?id=137229

- Zeitlimit: **25 Minuten**
- Befehle immer so angeben, dass man sie **direkt im Terminal verwenden könnte**!

`man` Befehl nutzen!

Viel Erfolg!

Kurztest 🎉

https://moodle.hft-stuttgart.de/mod/quiz/view.php?id=137229

- Zeitlimit: **25 Minuten**
- Befehle immer so angeben, dass man sie **direkt im Terminal verwenden könnte**!

`man` Befehl nutzen!

Passwort: Stadtbienen

Viel Erfolg!

Rechnerpraxis

05 Prozess- & Userverwaltung

Hochschule
für Technik
Stuttgart

SS19 | INF 1

Ben Lebherz
benjamin.lebherz@hft-stuttgart.de

(virtualisierte) CPU

- Single-Tasking: max. 1 Prozess ausführbar (Microcontroller)
- Multi-Tasking: mehrere Prozesse *gleichzeitig* ausführbar
  - Kooperatives Multi-Tasking: Prozesse *können* CPU abgeben

---

  - **Preemptives Multi-Tasking**: Prozesse *müssen* CPU abgeben **Scheduler** des **OS** teilt CPU den Prozessen zu (via **Time-Slicing**)

Time Slice Scheduling

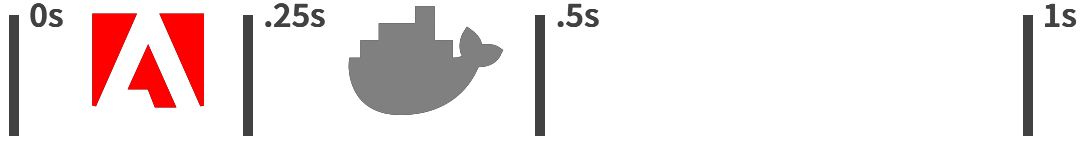
Vier Apps, jede einen *Slot pro Sekunde*

Time Slice Scheduling

Vier Apps, jede einen *Slot pro Sekunde*

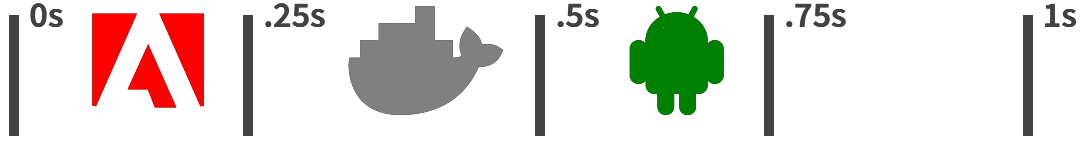
Time Slice Scheduling

Vier Apps, jede einen *Slot pro Sekunde*



Time Slice Scheduling

Vier Apps, jede einen *Slot pro Sekunde*



Time Slice Scheduling

Vier Apps, jede einen *Slot pro Sekunde*

Time Slice Scheduling

Vier Apps, jede einen *Slot pro Sekunde*

Fair Scheduling Algorithmus

Time Slice Scheduling

Zwei Apps, jede zwölf *Slots pro Sekunde*

0s

1s

Time Slice Scheduling

Zwei Apps, jede zwölf *Slots pro Sekunde*

Time Slice Scheduling

Zwei Apps, jede zwölf *Slots pro Sekunde*

Time Slice Scheduling

Zwei Apps, jede zwölf *Slots pro Sekunde*

Time Slice Scheduling

Zwei Apps, jede zwölf *Slots pro Sekunde*

Time Slice Scheduling

Zwei Apps, jede zwölf *Slots pro Sekunde*

Frequenz/Tick Rate auf echten Systemen?
10Hz? 75Hz? 100Hz?

Time Slice Scheduling

1000Hz

Time Slice Scheduling

Frequenz/Tick Rate auf echten Systemen, in **Hertz** (Hz)

```
$ cat /boot/config-4.19.0-4-amd64 | grep CONFIG_HZ

# CONFIG_HZ_PERIODIC is not set
# CONFIG_HZ_100 is not set
CONFIG_HZ_250=y
# CONFIG_HZ_300 is not set
# CONFIG_HZ_1000 is not set
CONFIG_HZ=250
```

Time Slice Scheduling

- **250Hz**: Desktop Systeme; guter Kompromiss für viele Anwendungen (früher 100Hz)
- **1000Hz**: Server Systeme mit vielen Tausend Prozessen oder speziellen Anwendungen
- **Variabel**: Laptop / System zur numerischen DV

nice

- **Scheduling-Priotität** eines Prozesses
- Dargestellt als Zahl

Befehle

```
$ nice
# und
$ renice
```

(virtualisierter) RAM

- Phys. Speicher / RAM addressiert via **`0x0000.. – 0xFFFF..`**
- unterteilt in **Pages** (Blöcke fester Größe)
- Auslagern von Pages auf Festplatte möglich (**langsam!**)
  - **Swap** / swapping auf Linux/Unix/macOS
  - **Pagefile** in Windows
- *Hugepages*?
- Virtueller Speicher **gemappt** auf physikalischen Speicher

(virtualisierter) RAM

Page Size (in Bytes) auf echten Systemen

```
$ getconf PAGE_SIZE
4096
```

Hugepages aktiviert/verfügbar?

```
$ mount | grep huge
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
```

(virtualisierter) RAM

(virtualisierter) RAM

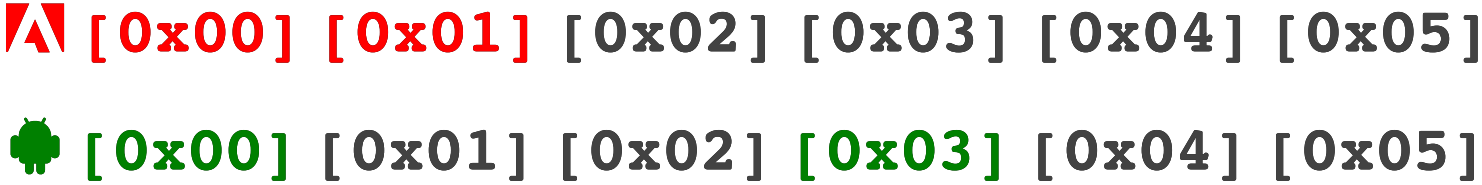
Physikalisch: `[0x00]` `[0x01]` `[0x02]` `[0x03]` `[0x04]` `[0x05]`

(virtualisierter) RAM


[0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

Physikalisch: [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

(virtualisierter) RAM



Physikalisch: [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

(virtualisierter) RAM

 [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

 [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

Page Table:

Physikalisch: [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

(virtualisierter) RAM

 [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

 [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

Page Table:

 [0x00] --> [0x02]
 [0x01] --> [0x05]

Physikalisch: [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

(virtualisierter) RAM

[0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

[0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

Page Table:

[0x00] --> [0x02]
[0x01] --> [0x05]

[0x00] --> [0x01]
[0x03] --> [0x04]

Physikalisch: [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

(virtualisierter) RAM

Physikalisch: [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

[0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

[0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

Page Table:

[0x00] --> [0x02]
[0x01] --> [0x05]

[0x00] --> [0x01]
[0x03] --> [0x04]

Physikalisch: [0x00] [0x01] [0x02] [0x03] [0x04] [0x05]

(virtualisierter) RAM

Verfügbarer Speicher/Swap

```
$  free -m
              total        used        free      shared  buff/cache   av
Mem:          15951        2143         240           3       13567
Swap:          2047         138        1909
```

Prozess Management Tools

`top`

- Klassiker zum Prozesse darstellen
- überall vorhanden

```
$ top
```

`ps`

- Klassiker zum Prozesse ausgeben mit diversen Parametern
- überall vorhanden

```
$ ps
```

`kill`

- Klassiker zum Prozesse beenden oder *töten*
- Sendet *Signal* zu Prozess, kein *abschießen* (ausser –9)

```
$ kill <Prozess ID>
```

Prozess Management Tools

htop

- "schöner" 😃 mehr Übersicht, mehr Werte/Features
- meist direkt als Paket verfügbar

```
$ htop
```

gotop

- noch "schöner" 🤩 unterstützt auch Netzwerk, Sensoren, ...
- **go** (golang.org) benötigt, einfach zu installieren

```
$ gotop --statusbar --rate=1
```

glances

- auch "schön" 🤪 unterstützt auch Docker Container, ...
- meist direkt als Paket verfügbar

```
$ glances --time 2 --byte --percpu
```

Demo 🤓💻

`htop`

`gotop`

`glances`

User-Management

- User/Gruppen Konzept mit 1`:n` Beziehung
- Identifiziert über eindeutige **uid** bzw. **gid**
  (**`uid`**/**`gid`** < 1000 für System-Benutzer/Gruppen)

```
# add
$ useradd --uid 1111 --home-dir /tmp --shell /bin/bash hft
# delete
$ userdel hft
```

Hands-on!

Prozess- und Usermanagement

Prozess-Management Aufgaben

- Ermitteln Sie die erlaubten Werte für die Prozess Priorität und schauen sie sich die Prioritäten der laufenden Prozesse an
- Erkunden Sie die Parameter von `ps` und testen sie die Beispiele aus der *man Page*
- Starten Sie **`top`** und rufen die integrierte Hilfe auf. Ändern sie nun die Sortierreihenfolge der aufgelisteten Prozesse.
- Was bewirkt **`kill -9 $$`**? Wofür steht **`$$`**?
- Ändern sie die Priotität eines CPU-intensiven Prozesses auf die "schlechteste" mögliche Priorität. Was stellen sie fest?

User-Management Aufgaben

- Ermitteln Sie ihre User-ID (`uid`)
- Ermitteln Sie die **`uid`** ihrer beiden Sitznachbarn
- Ermitteln Sie in welchen Gruppen sie sind
- Welche Informationen liefern ihnen folgende Befehle
  - **`whoami`**
  - **`who`**
  - **`w`**
  - **`last`**