BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

4

OOP – Linux

Ausgewählte Themen der System-administration

mit Skriptmaterial von Dr.-Ing. M. Feldmann

Prof. Dr.-Ing. Tenshi Hara
tenshi.hara@ba-dresden.de

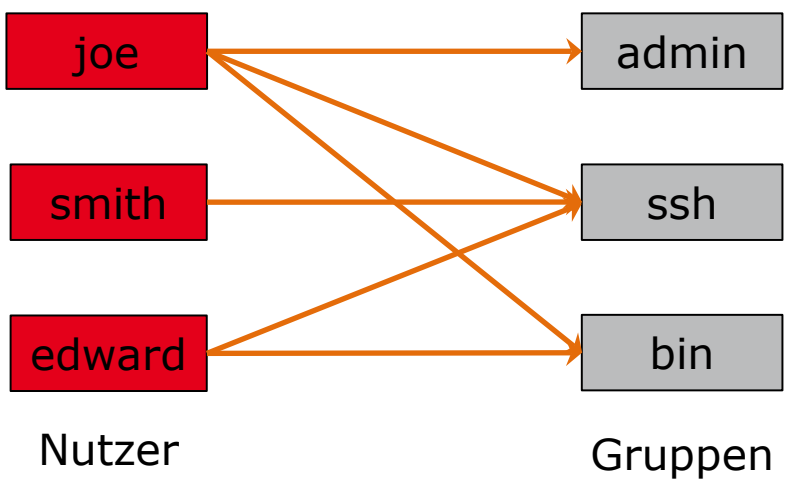
Gliederung der Vorlesung

Inhalte

- Nutzerverwaltung / Gruppen / Passwörter
- Rechtemanagement
- Softwareinstallation
- Cron
- Syslog

Nutzerverwaltung

- Linux = Mehrbenutzersystem
  - eine Liste mit Nutzern wird verwaltet
  - jeder Nutzer kann Rechte an verschiedenen Dingen (Dateien, Programmen, …) halten
- Linux = Gruppensystem
  - jeder Nutzer kann einer oder mehreren Gruppen zugeordnet werden
  - Standardgruppe mit Rechten für Gäste („everybody“)



Nutzerverwaltung – Ausgewählte Kommandos

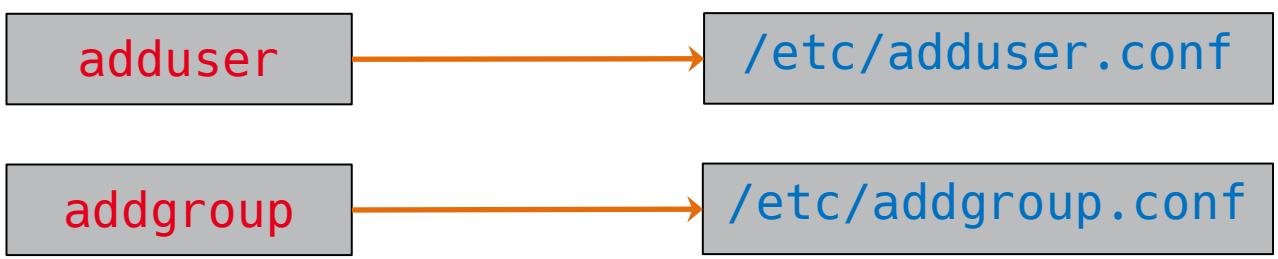
- `useradd`/`adduser`: Einrichten eines neuen Nutzers
- `groupadd`/`addgroup`: Einrichten einer neuen Gruppe
- `userdel`/`deluser`: Löscht einen Nutzer
- `groupdel`/`delgroup`: Löscht eine Gruppe
- `id`: zeigt Nutzer- und Gruppen-Id des aktuellen Nutzers an
- `groups`: zeigt Gruppen des aktuellen Nutzers an
- `passwd`: ändert das Passwort eines Nutzers

(Allgemein/Debian-artige):

- die zuerst genannten Systembefehle gibt es in allen Linux-artigen Systemen
- die Aliase sind Perl-Skripte in Debian-artigen Systemen, die mehr Interaktivität bieten

Nutzerverwaltung – Ausgewählte Kommandos

- Unterschiede zwischen Linux-Kommandos und Debian-spezifischen (und damit von Ubuntu und anderen abgeleiteten Distributionen) Kommandos:
  - `adduser` und `addgroup` berücksichtigen Einstellungen in `/etc/adduser.conf` beziehungsweise `/etc/addgroup.conf`
  - von regulären Linux-Kommandos (`useradd`, `groupadd`) werden diese nicht berücksichtigt



NUTZERVERWALTUNG

Superuser alias Systemadministrator (admin) alias root

- Besitzt meist den Namen root
- Hat uneingeschränkte Rechte
- Nutzeraccount sollte nicht zu normalen Nutzungszwecken eingesetzt werden

Reguläre Nutzer

- Für die normale Arbeit mit Linux eingesetzt
- Besitzen uneingeschränkte Rechte auf eigene Dateien, aber eingeschränkte auf den Rest des Systems

Systembenutzer für Dämonen und Server-Dienste

- Benutzeraccounts, die nicht für Interaktion eines Menschen mit dem System vorgesehen sind, sondern zur Ausführung von Programmen
- Dient Erhöhung der Systemsicherheit
- Beispiel: Nutzer *httpd*

Nutzerverwaltung - /etc/passwd

- Liste aller Benutzer mit wichtigen Konfigurationsinformationen befindet sich in `/etc/passwd`
- Aufbau der Datei:
  - Zeilenweise Angabe von Nutzeraccounts
  - Login:Passwort:UID:GID:Name:Heimverzeichnis:Kommando

Eindeutige Nutzer-ID Eindeutige Gruppen-ID

- Beispiel: Root-UID/GID = 0

```
root:x:0:0:Administrator:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
lp:x:4:7:lp daemon:/var/spool/lpd:/bin/bash
...
luke:x:502:100:Lukas Nutzermann:/home/luke:/bin/bash
```

Passwörter

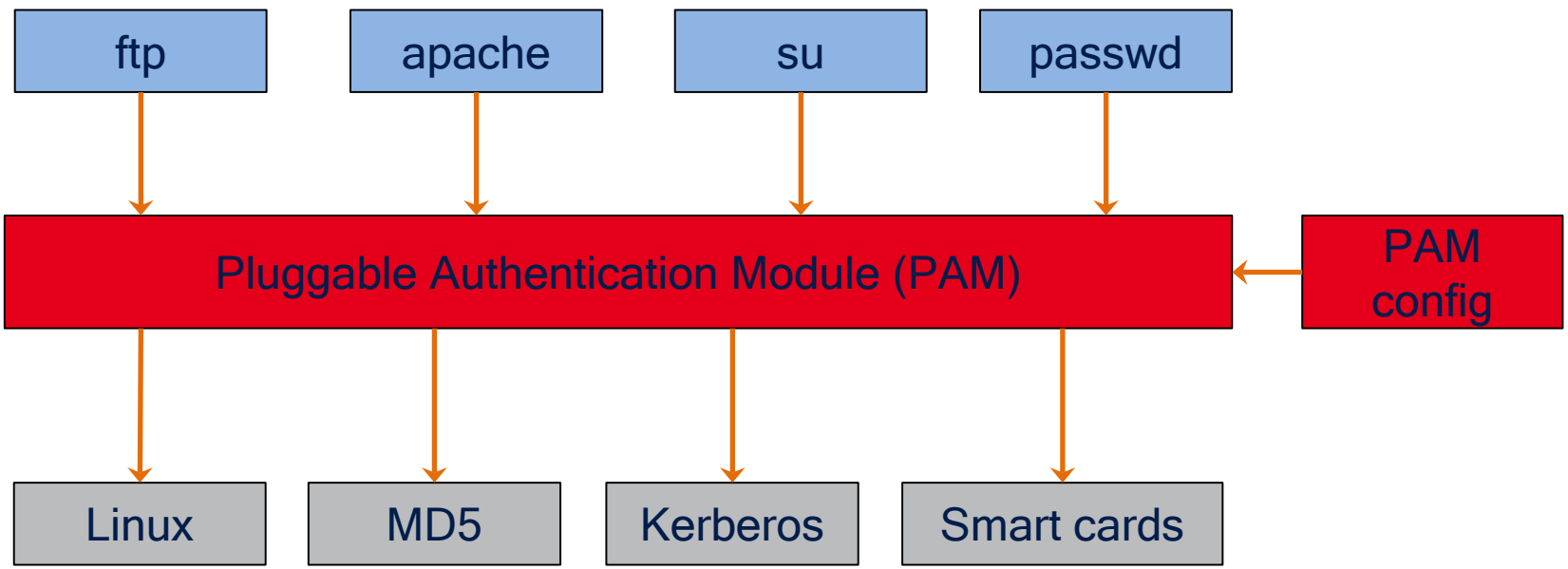
- dürfen nur aus ASCII-Zeichen bestehen, sollten Sonderzeichen beinhalten, damit sie nicht leicht ermittelt werden können
- i.d.R. beliebig lang
- Speicherung in `/etc/shadow` in Form von Hashes (z.B. mit SHA512 Hash-Algorithmus)
- wichtige Parameter für Login befinden sich in `/etc/login.defs`
- Shadow-Datei im Gegensatz zu `/etc/passwd` und `/etc/group` nicht für normale Nutzer lesbar
- Aufbau der Datei:
  `Login:Password(weitere Infos):d1:d2:d3:d4:d5:d6:reserved`

Nutzerverwaltung – /etc/shadow

- historisch: `/etc/passwd` beinhaltete auch das Nutzerpasswort, heute meist in `/etc/shadow`
- Inhalte:
  - **Login**: sollte aus ASCII-Minuskeln bestehen und nicht länger als 8 Zeichen sein (Kompatibilität für bestimmte Legacy-Programme)
  - **UID**: dient der internen Identifikation des Nutzers
  - **GID**: spezifiziert, zu welcher primären Gruppe der Nutzer gehört
  - **Heimverzeichnis**: Speicherort für private Dateien und oft auch nutzerspezifische Programmkonfigurationen; i.d.R. Ort, an den Benutzer nach Login platziert wird; Verzeichnis enthält weiterhin wichtige Konfigurationsinformationen
  - **Kommando**: nach Login auszuführendes Kommando

Pluggable Authentification Modules (PAM)

- Bibliothek, deren Funktionen bei Authentifizierungsaufgaben helfen
- PAM verwaltet Zugriff auf `/etc/passwd` bzw. `/etc/shadow`
- sollen weitere Authentifizierungsformen verwendet werden (z.B. LDAP), muss PAM entsprechend konfiguriert werden
  → Werkzeug (Ubuntu): pam-auth-update



Gruppen

- Gruppen ermöglichen grobgranulare Rechteverwaltung
- jedem Nutzer ist eine primäre Gruppe zugeordnet (in `/etc/passwd`)
- Zugehörigkeit zu weiteren Gruppen über die primäre Gruppe hinaus wird in `/etc/group` festgelegt
- Aufbau der Datei: `Gruppenname:Passwort:GID:Benutzerliste`
- Beispiel:

```
root:*:0:root
bin:*:1:root,bin,daemon
daemon:*:2:
...
admin:x:119:user
```

Zugriffsrechte auf Dateien

- als Mehrbenutzersystem benötigt Linux Mechanismen, um zu bestimmen, wer welche Rechte an welchen Dateien hat
- zwei Formen von Rechteverwaltung:
  - in der traditionelle Form werden mit jeder Datei Informationen für den Zugriff gespeichert:
    - Besitzer der Datei
    - Gruppe, der die Datei zugeordnet ist
    - **neun** Zugriffsbits für den Besitzer, für alle Gruppenmitglieder und für den Rest der Welt
    - Zusatzbits mit Spezialfunktionen
  - Access Control Lists (ACL)

Zugriffsrechte auf Dateien

- Zugriffsbits beschreiben Lese-, Schreib- und Ausführungsberechtigung
- Angabe kann als Abfolge von 9 Zeichen erfolgen, z.B.: `rwxrw-r--`
  - Position 1-3: Rechte des Besitzers
    → im Beispiel `rwx` = Read / Write / Execute
  - Position 4-6: Rechte der zugehörigen Gruppe
    → im Beispiel `rw-` = Read / Write / keine Ausführungsrechte
  - Position 7-9: Rechte aller anderen
    → im Beispiel: `r--` = Read / keine Schreibrechte / keine Ausführungsrechte
- Rechte können auch oktal geschrieben werden: `r = 4`, `w = 2`, `x = 1`, `- = 0`
  → Superposition führt zu eigentlichen Rechten
  - `6` = Lese- und Schreibrechte
  - `5` = Lese- und Ausführungsrechte

Zugriffsrechte auf Dateien

Neben den Standardrechten existieren auch Spezialbits:

- Suid-Bit: bewirkt, dass Programm so ausgeführt wird, als hätte Besitzer der Datei es ausgeführt (wird durch `s` statt `x` angezeigt)
- Sticky-Bit: verhindert, dass in Verzeichnissen, auf die alle Schreibrechte haben, Benutzer Dateien von anderen Benutzern löschen können (wird durch `t` statt `x` angezeigt)
  → bspw. sinnvoll für Verzeichnis `/tmp`
- Kann durch `chmod +s` bzw. `chmod +t` gesetzt werden

Zugriffsrechte auf Dateien

- `chmod`: ändert die neun Zugriffsbits von Dateien
  - Entweder Angabe der numerischen Werte oder der zu ergänzenden/entfernenden Rechte (z.B. `+x` oder `-x`)
  - Ermöglicht das Setzen der Spezialbits; z.B. setuid:

```
user@linux$ chmod +s DATEINAME
```

- `chown`: Ändert den Besitzer oder optional die Gruppenzugehörigkeit einer Datei
- `chgrp`: Ändert die Gruppenzugehörigkeit von Dateien

Access Control Lists (ACL)

- alternatives Konzept zu bewährtem Mechanismus
- ermöglichen sehr feingranulare Definition, wer welche Rechte auf Datei besitzt und wer keine Rechte besitzt
- können via `getfacl`/`setfacl` angefragt und verändert werden

```
user@linux$ sudo setfacl -m user:grp:r-- myfile.txt
```

- Achtung: Dateisystem muss ACL unterstützen
  → ext-Dateisysteme muss mit mount-Option `acl` gemountet werden

Dateiverwaltung – Symbolische Links

- Symbolische Links = Verweise auf Dateien
- auf eine physische Datei kann von mehreren Orten aus dem Dateisystem heraus verwiesen werden
- vermeidet Redundanzen
- zwei Formen von Links:
  - feste Links (`ln datei link`) → Referenz auf einen INODE (=Eintrag in Dateisystem mit Metadaten zu Datei)
  - Symbolische Links / Softlink / Symlink (`ln -s datei link`) → Referenz auf einen PFAD

Dateiverwaltung – Grundlegende Befehle

`cd`	Aktuelles Verzeichnis wechseln
`cp`	Kopieren von Dateien (behält Original)
`less`/`more`	Seitenweise Anzeige von Textdateien
`ls`	Zeigt alle Dateien eines Verzeichnisses an
`mkdir`	Anlegen eines neuen Verzeichnisses
`mv`	Verschiebt eine Datei / ein Verzeichnis bzw. ändert Namen
`touch`	Datei anlegen
`rm`	Entfernen von Dateien
`rmdir`	Entfernen von Verzeichnissen

Software-Installation

- Software kann häufig nach einem Download über ein mitgeliefertes `make`-Skript konfiguriert und installiert werden
  - Details dann im Kapitel zu C-Programmierung unter Linux
  - Hier: Betrachtung von Paketverwaltungssoftware
- Distributionen besitzen Software zur Installation und Verwaltung von Programmen
  - Software distributionsabhängig
  - Mächtigkeit der Paketverwaltung distributionsabhängig
  - verfügbare Programme distributionsabhängig
  - Fokus auf APT (Debian, Ubuntu, …)

APT

- Advanced Packaging Tool ermöglicht komfortabel von der Kommandozeile Software
  - zu suchen,
  - von entfernten Quellen herunterzuladen,
  - zu konfigurieren,
  - zu installieren,
  - zu updaten und
  - zu deinstallieren
- Quellen, von denen Software geladen werden kann, werden in `/etc/apt/sources.list` angegeben
- Aufbau von `/etc/apt/sources.list`:
  - Pakettyp URI distribution [komponente1] [komponente2] …
  - Pakettyp: deb = Standardpaket, deb-src: Quellcodepaket
  - URI: Quelle, von der heruntergeladen wird

```
deb http://archive.ubuntu.com/ubuntu/ zesty-updates main restricted
deb-src http://archive.ubuntu.com/ubuntu/ zesty-updates main restricted
```

APT

- eigentliche Paketverwaltung wird mit `apt-get` bzw. `aptitude` durchgeführt
- Arbeit mit `apt-get`:

```
user@linux$ sudo apt-get update
```

Aktualisierung von Informationen über Pakete durch Anfrage bei Quellen

```
user@linux$ sudo apt-get install apache2
```

Installation des Pakets apache2

```
user@linux$ sudo apt-get upgrade
```

Aktualisierung installierter Software

```
user@linux$ sudo apt-get dist-upgrade
```

Upgrade der Distribution (`/etc/sources.list` muss zunächst aktualisiert werden)

- für grundlegende Kommandos auch Verwendung von `apt`

APT

- zur Ermittlung von Informationen über Pakete und zur Suche wird `apt-cache` eingesetzt
- Arbeit mit `apt-cache`:

```
user@linux$ apt-cache show apache2
```

Zeigt Meta-Informationen zum Paket an

```
user@linux$ apt-cache search apache2
```

Liefert Suchergebnisse zum übergebenen Begriff (hier: apache2)

- Durch `cron` kann regelmäßig auf erforderliche Updates geprüft werden → siehe dazu `/etc/apt/apt.conf.d/50unattended-upgrades` o.ä.

Cron

- Cron-Daemon ist eine Jobsteuerung für Unix-artige Systeme
- Cron-Jobs können von Nutzern oder systemweit definiert werden

```
user@linux$ crontab -e
```

Öffnet die Konfiguration der Cron-Jobs für den aktuellen Nutzer

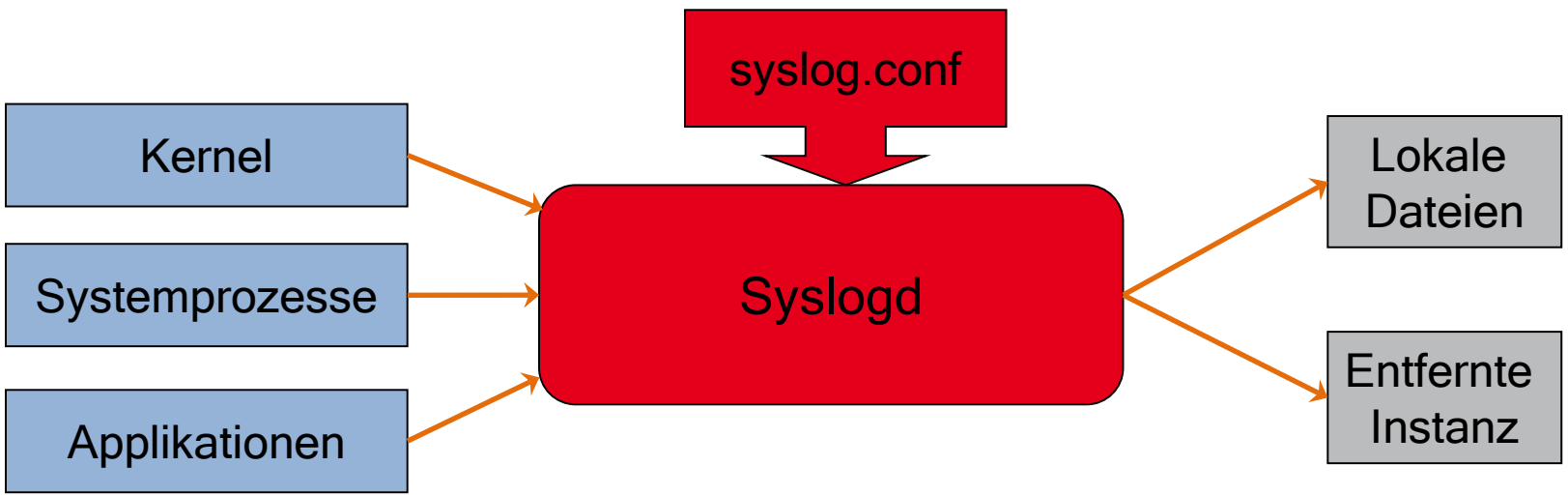
- Systemweit definierte Cronjobs sind in `/etc/crontab` beziehungsweise in den Dateien in `/etc/cron.d` definiert
- Aufbau der Datei `/etc/crontab`:

```
min hour day month weekday user command
```

- Bei der Verwendung von `crontab` werden die Dateien beim Speichern semantisch geprüft; Dateien können aber auch mit jedem anderen Editor editiert werden (ohne Überprüfung)

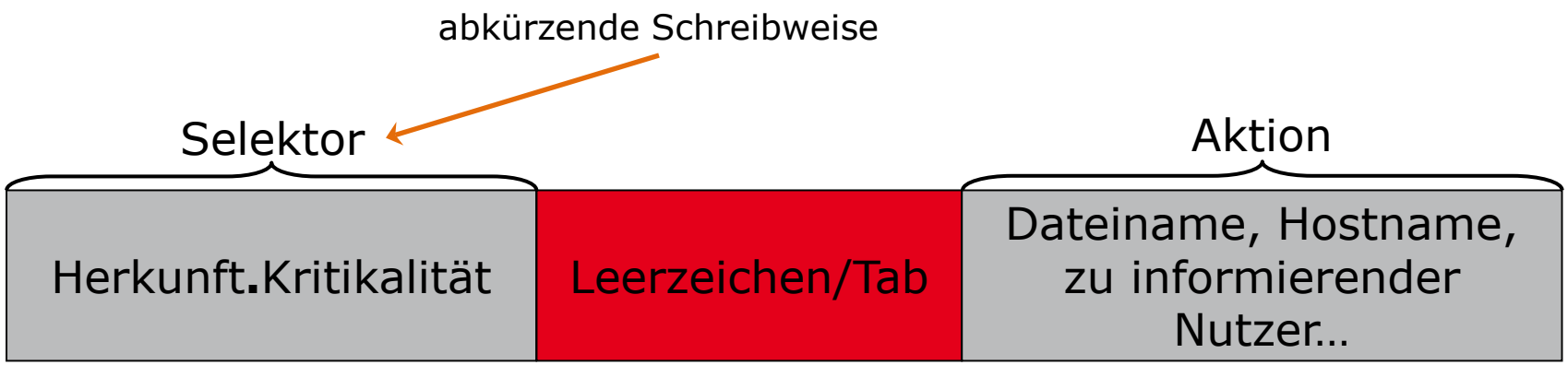
SysLog

- kurze Meldungen über Ereignisse, Fehler, etc. in Systemen (vergleichbar mit Logbuch-Einträgen)
  - Speicherung in `/var/log`
  - `Aug 24 23:19:01 shadow su:(to root) master on /dev/pts/2`
- erzeugt von: Kernel, Systemprozessen, Anwendungen, …
- Ziel:
  - Abweichen von Normalverhalten aufspüren, Fehlersuche
  - Frühwarnsystem bei Fehlern / Angriffen
  - Spurensuche nach Angriffen



SysLog

- Zeilenweise Angabe von Nachrichten, die geloggt werden sollen und wohin sie geschrieben/gesendet werden
- Syntaktischer Aufbau:



- Ziele im Netzwerk (z.B. via Hostname) werden mit vorangestelltem @ angegeben

SysLog

- Problem: `/var/log` wird schnell sehr groß
- zur Lösung bietet sich `logrotate` an:
  - wird z.B. täglich durch `cron`-Skript aufgerufen
    → `/etc/cron.daily/logrotate`
  - verarbeitet alle Logdateien, die in `/etc/logrotate.d/` beschrieben sind
  - Vorgehen:
    - benenne vorhandene Logarchive um, z.B. aus `name.6.gz` wird `name.7.gz`
    - wenn maximale Zahl von Archiven erreicht ist: lösche Archiv mit höchstem Index
    - Benne aktuelle Logdatei um – z.B. `name` zu `name.0`
    - komprimiere umbenannte Datei zu `name.0.gz`
    - erzeuge neue, leere Logdatei

SysLog

Beispiel /etc/logrotate.d/apache2

```
/var/log/apache/*.log
{
   weekly
   ...
   rotate 52
   compress
   delaycompress
   create 640 root adm
   postrotate
   if [ -f /var/run/apache.pid ]; then
      /etc/init.d/apache restart > /dev/null
   fi
   endscript
}
```

Aufgaben

1. Erstellen Sie Cron-Jobs, die das in Kapitel 2 entwickelte Backup-Skript dienstags bis samstags um 1.45 Uhr und montags um 6 Uhr aufrufen.

2. Erstellen Sie ein Programm `helloworld.sh`, das bei Aufruf den String „Hello World!“ ausgibt. Paketieren Sie das Programm in ein .deb-Paket. Durch Aufruf von `dpkg --install PACKETNAME` soll das Programm automatisch im System installiert werden können. Dabei soll es im Verzeichnis `/usr/bin` abgelegt werden.
   Einen Überblick über das Erstellen von .deb-Paketen erhalten Sie via:
   https://blog.serverdensity.com/how-to-create-a-debian-deb-package/