# Zusammenfassung Tag 12

## Die Kernel-Verzeichnisse /proc/ und /sys/

- Es gibt zahlreiche virtuelle Dateisysteme für die verschiedenen Zwecke
    - Das Dateisystem `proc`, wird unter `/proc` gemountet
    - Das Dateisystem `sysfs` wird unter `/sys` gemountet
- `/sys` und `/proc` stellen virtuelle Kernel-Verzeichnisse bereit
- Sie existieren nur im Arbeitsspeicher und werden vom Kernel dynamisch erstellt
- Die Daten darin werden beim Herunterfahren nicht gespeichert
- Wichtige Dateien in `/proc`
    - `cpuinfo`: Zeigt z.B. Informationen zum Prozessor
    - `interrupts`: Zeigt Anfragen für Datenaustausch mit dem Prozessor
    - `ioports`: Zeigt die Input-Output-Adressen der Hardware-Komponenten. Über sie ist ein Datenaustausch zwischen der CPU und dem jeweiligen Gerät möglich
    - `meminfo`: Zeigt die Verwendung des Arbeitsspeichers
    - `swaps`: Hier wird der verwendete Swap-Space aufgeführt
    - `version`: Zeigt die Linux-Version des Systems an
    - Unter `/proc` befindet sich das Unterverzeichnis `/sys` mit weiteren Unterverzeichnissen für verschiedene Aspekte des Systems. Das umfasst Hardware, wie `dev`, `fs` oder `net`, aber auch Software, wie z.B. `users`
    - Die Unterverzeichnisse mit den Ziffern entsprechen den Prozess-IDs und beinhalten Dateien zu diesem jeweiligen Prozess
- Die Dateien unter `/proc` können auch zur Manipulation und zum Tuning des Systems angepasst werden
    - Beispiel: Unter `/proc/sys/net/ipv4` diverse Tuning-Parameter für den TCP/IP-Stack unter IPv4
- Im Verzeichnis `/sys` befinden sich wie unter `/proc` Informationen über Geräte, Kernel-Module, Dateisystem und andere Kernel-Komponenten
    - Auch hier sehen wir diverse Unterverzeichnisse. Unter denen sich Dateien und weitere Unterverzeichnisse befinden
- Während `/sys` sich mehr auf die Kernel-bezogenen Informationen konzentriert, bezieht sich `/proc` auf die Prozesse des Systems
- Beim System-Tuning arbeiten wir meistens im `/proc`-Verzeichnis

## Das Geräteverzeichnis /dev unter der Lupe

- Im Geräteverzeichnis `/dev` befinden sich aktive Komponenten, und auch Geräte die nicht permanent aktiv sind:
    - Character Devices, die nur seriell angesprochen werden können, also bit für bit hintereinander lesend und schreibend.
      Zum Beispiel die virtuellen Terminals tty xy
    - Block Devices, auf die wahlfrei und blockweise zugegriffen wird. Das bedeutet, dass nicht erst die ersten 1000 Bytes des Datenträgers ausgelesen werden müssen, bevor die Daten ab dem 1001. Byte gelesen werden können. Stattdessen kann gezielt auf die gewünschten Speicherblöcke zugegriffen werden.
      Zum Beispiel Festplatten und andere Speichergeräte
    - Die drei Ein- und Ausgabekanäle `Stdin`, `stdout` und `stderr` verweisen auf `/proc/self/fd/`
    - `cdrom` und `dvd` zeigen auf `sr0`. Dies ist die Gerätedatei für optische Laufwerke
    - Hinter `/dev/random` steckt ein hochwertigen Zufallsgenerator
    - Hinter `/dev/urandom` steckt ein nicht so hochwertigen Zufallsgenerator
    - `/dev/zero` liefert die geforderte Anzahl an Null-Bytes zurück
    - Bei `/dev/null` handelt es sich um den virtuellen Mülleimer. Alles was hier hin geschrieben wird ist unwiederbringlich verloren
- Das Verzeichnis `/dev` ist Mountpoint für eine virtuelle Komponente namens `udev` und das verwendete Pseudo-Dateisystem nennt sich `devtmpfs`. Dieses Verzeichnis wird dynamisch vom Kernel und seinen Komponenten gefüllt und die hier aufgeführten Dateien sind nur virtuell vorhanden
- Auch die Unterverzeichnisse unter `/dev` sind dynamisch erzeugte Objekte, die zur besseren Verwaltung dienen und meistens Symlinks bereitstellen. Denn wenn sich die Gerätebezeichnungen ändern, weil z.B. Festplatten intern umgesteckt werden, so werden hier einfach die Symlinks korrigiert und es passt wieder

## Gerätetreiber verwalten mit modprobe & Co.

- Voraussetzung für die Kommunikation des Linux-Kernels mit der im Computer integrierten oder angeschlossenen Hardware ist, dass der Kernel die Sprache der Komponente versteht. Hierfür ist ein Gerätetreiber notwendig
- Der Gerätetreiber kann entweder bereits im Kernel fest einkompiliert sein oder aber dynamisch beim Systemstart oder im laufenden Betrieb als Loadable Kernel Modul (kurz: LKM) nachgeladen werden
- Bei modernen Linux-Systemen erledigen die Subkomponenten des Kernels in der Regel die Hardware-Erkennung und integrieren den benötigten Treiber ganz automatisch
- Verantwortlich für die Kernelverwaltung ist eine Komponente namens `kmod`. Das steht für Linux Kernel Module Handling

- Es gibt unter Treibern diverse Anhängigkeiten die mit `kmod list` und `lsmod` angezeigt werden können
- Treiber können auch manuell mit allen Abhängigkeiten nachinstalliert werden (`modprobe`)
- Welche Abhängigkeiten vorhanden sind, zeigt der Inhalt der Datei `modules.dep`

## USB-Speichersticks

- Wird ein USB-Stick an einem Linux-System angeschlossen, so wird dieser automatisch erkannt, Treiber nachgeladen und gemountet
- Mountpoint ist die UUID des USB-Sticks
- Treiber-Module für die Unterstützung von USB:
    - OHCI, steht für Open Host Controller Interface
    - UHCI, steht für Universal Host Controller Interface und ist eine lizenzpflichtige Version von Intel. Beide dienen der Unterstützung von USB 1.1
    - EHCI, steht für Enhance HCI und stellt Unterstützung von USB 2.0 dar
    - UAS steht für USB attached SCSI und ist ein Protokoll, das für USB 3.0 eingeführt, aber abwärtskompatible ist
    - xHCI stellt ebenfalls den Support für USB 3.x sicher

## Geräteverwaltung mit Udev

- Udev dient zur Geräteverwaltung und läuft als `udevd` im Hintergrund
- Udev dient in erster Linie zur Verwaltung von Hotplugging-Geräten
- Udev generiert die Inhalte des Geräteverzeichnisses `/dev`
- Udev arbeitet nach festen, aber konfigurierbaren und erweiterbaren Regeln
    - `/etc/udev/rules.d`
    - `/lib/udev/rules.d`
- Meldungen des Kernels die bestimmte Ereignisse, wie das Einbinden eines Speichermediums, anzeigen werden als `uevents` bezeichnet
- Die uevents werden vom Udev-Daemon abgefangen und ausgewertet

## Der D-BUS

- Der D-BUS ist eine Programmbibliothek, die dazu dient, dass Prozesse untereinander über einen logischen BUS kommunizieren können
- Der D-BUS ist eine Middleware, eine Vermittlungsschicht zwischen Prozessen
- Der D-BUS besteht aus drei Komponenten:
    - Dem D-BUS-Daemon
    - Der D-BUS-Bibliothek namens `libdbus`
    - Dem D-BUS-Protokoll
- D-BUS erhält Informationen von Hotplugging-Ereignissen von Udev und sorgt im Beispiel des USB-Sticks dafür, dass diese Informationen an den Dateimanager weitergegeben werden.
- Der D-BUS muss nicht konfiguriert werden