



**Linux
Professional
Institute**

4.3 Wo Daten gespeichert werden

Referenz zu den LPI-Lernzielen

Linux Essentials version 1.6, Exam 010, Objective 4.3

Gewichtung

3

Hauptwissensgebiete

- Programme und Konfiguration
- Prozesse
- Speicheradressen
- Systembenachrichtigungen
- Protokollierung

Auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme

- `ps`, `top`, `free`
- `syslog`, `dmesg`
- `/etc/`, `/var/log/`
- `/boot/`, `/proc/`, `/dev/`, `/sys/`



4.3 Lektion 1

| | |
|--------------------|---------------------------------|
| Zertifikat: | Linux Essentials |
| Version: | 1.6 |
| Thema: | 4 Das Linux-Betriebssystem |
| Lernziel: | 4.3 Wo Daten gespeichert werden |
| Lektion: | 1 von 2 |

Einführung

Für ein Betriebssystem ist alles Daten. Für Linux ist alles eine Datei: Programme, normale Dateien, Verzeichnisse, Blockgeräte bzw. Block Devices (z.B. Festplatten), zeichenorientierte Geräte bzw. Character Devices (z.B. Konsolen), Kernelprozesse, Sockets, Partitionen, Links etc. Die Linux-Verzeichnisstruktur, beginnend mit dem Wurzelverzeichnis (auch *root* genannt und symbolisiert durch `/`), ist eine Sammlung von Dateien mit Daten. Dass alles eine Datei ist, ist ein mächtiges Feature von Linux, da sich so praktisch jeder Bereich des Systems optimieren lässt.

In dieser Lektion werden wir die verschiedenen Orte besprechen, an denen wichtige Daten gespeichert werden, wie sie der Filesystem Hierarchy Standard (FHS) festlegt. Einige dieser Orte sind echte Verzeichnisse, die Daten dauerhaft auf Festplatten speichern, während andere Pseudodateisysteme sind, die in den Speicher geladen werden und uns Zugang zu Daten des Kernel-Subsystems wie etwa laufende Prozesse, den Speicherverbrauch, die Hardware-Konfiguration usw. geben. Die in diesen virtuellen Verzeichnissen gespeicherten Daten werden von einer Reihe von Befehlen verwendet, die es uns ermöglichen, sie zu überwachen und zu verwalten.

Programme und ihre Konfiguration

Wichtige Daten auf einem Linux-System sind zweifellos seine Programme und deren Konfigurationsdateien. Erstere sind ausführbare Dateien mit Anweisungen, die vom Prozessor des Computers ausgeführt werden sollen, während letztere in der Regel Textdokumente sind, die den Betrieb eines Programms steuern. Ausführbare Dateien können entweder Binärdateien oder Textdateien sein. Ausführbare Textdateien werden als Skripte bezeichnet. Konfigurationsdaten unter Linux werden traditionell auch in Textdateien gespeichert, obwohl es verschiedene Arten der Darstellung von Konfigurationsdaten gibt.

Wo Binärdateien gespeichert sind

Wie alle anderen Dateien liegen ausführbare Dateien in Verzeichnissen unterhalb von `/`. Genauer gesagt, werden Programme über eine dreistufige Struktur verteilt: Die erste Schicht (`/`) enthält Programme, die im Einzelbenutzermodus notwendig sein können, die zweite Schicht (`/usr`) enthält die meisten Mehrbenutzerprogramme und die dritte Schicht (`/usr/local`) wird verwendet, um Software zu speichern, die nicht von der Distribution bereitgestellt und lokal kompiliert wurde.

Typische Orte für Programme sind:

`/sbin`

enthält wichtige Binärdateien für die Systemadministration wie `parted` oder `ip`.

`/bin`

enthält essentielle Binärdateien für alle Benutzer wie `ls`, `mv` oder `mkdir`.

`/usr/sbin`

enthält Binärdateien für die Systemadministration wie `deluser` oder `groupadd`.

`/usr/bin`

enthält die meisten ausführbaren Dateien — wie `free`, `pstree`, `sudo` oder `man --`, die von allen Benutzern verwendet werden können.

`/usr/local/sbin`

wird verwendet, um lokal installierte Programme für die Systemadministration zu speichern, die nicht vom Paketmanager des Systems verwaltet werden.

`/usr/local/bin`

dient dem gleichen Zweck wie `/usr/local/sbin`, jedoch für normale Benutzerprogramme.

Einige Distributionen sind dazu übergegangen, `/bin` und `/sbin` durch symbolische Links zu `/usr/bin` und `/usr/sbin` zu ersetzen.

NOTE

Das Verzeichnis `/opt` wird manchmal zur Ablage optionaler Anwendungen von Drittanbietern verwendet.

Abgesehen von diesen Verzeichnissen können normale Benutzer ihre eigenen Programme in den folgenden haben:

- `/home/$USER/bin`
- `/home/$USER/.local/bin`

TIP

Sie sehen, aus welchen Verzeichnissen heraus Sie Binärdateien ausführen können, indem Sie den Variableninhalt von `PATH` mit dem Kommando `echo $PATH` ausgeben. Weitere Informationen zu `PATH` finden Sie in den Lektionen über Variablen und Shell-Anpassung.

Den Standort eines Programms liefert der Befehl `which`:

```
$ which git
/usr/bin/git
```

Wo Konfigurationsdateien liegen

Das Verzeichnis `/etc`

In den frühen Tagen von Unix gab es für jeden Datentyp einen Ordner, wie z.B. `/bin` für Binärdateien und `/boot` für den/die Kernel. `/etc` (für “et cetera”) wurde als Catch-All-Verzeichnis angelegt, um alle Dateien zu speichern, die nicht zu den anderen Kategorien gehörten, und die meisten dieser Dateien waren eben Konfigurationsdateien. Mit der Zeit wurden immer mehr Konfigurationsdateien hinzugefügt, so dass `/etc` zum Hauptordner für Konfigurationsdateien von Programmen wurde. Wie bereits erwähnt, ist eine Konfigurationsdatei in der Regel eine lokale Textdatei (im Gegensatz zu einer binären), die den Betrieb eines Programms steuert.

In `/etc` finden wir verschiedene Muster zur Benennung von Konfigurationsdateien:

- Dateien mit einer *ad hoc* Erweiterung oder gar keiner Erweiterung, z.B.

group

Systemgruppen-Datenbank

hostname

Name des Host-Computers

hosts

Liste der IP-Adressen und deren Hostnamen-Übersetzungen

passwd

Systembenutzer-Datenbank — bestehend aus sieben Feldern, die durch Doppelpunkte getrennt sind und Informationen über den Benutzer liefern

profile

Systemweite Konfigurationsdatei für Bash

shadow

Verschlüsselte Datei für Benutzerpasswörter

- Initialisierungsdateien mit der Endung **rc**:

bash.bashrc

Systemweite **.bashrc** Datei für interaktive Bash Shells

nanorc

Beispiel-Initialisierungsdatei für GNU nano (ein einfacher Texteditor, der normalerweise mit jeder Distribution ausgeliefert wird)

- Dateien mit der Endung **.conf**:

resolv.conf

Config-Datei für den Resolver, der den Zugriff auf das Internet Domain Name System (DNS) ermöglicht

sysctl.conf

Config-Datei zum Setzen von Systemvariablen für den Kernel

- Verzeichnisse mit dem Suffix **.d**:

Einige Programme mit einer eindeutigen Konfigurationsdatei (***.conf** oder ähnlich) haben sich zu einem dedizierten Verzeichnis ***.d** entwickelt, das den Aufbau modularer, robusterer Konfigurationen ermöglicht. Zum Beispiel finden Sie bei der Konfiguration von logrotate **logrotate.conf**, aber auch **logrotate.d** Verzeichnisse.

Dieser Ansatz ist besonders nützlich, wenn verschiedene Anwendungen Konfigurationen für denselben spezifischen Dienst benötigen. Wenn beispielsweise ein Webserver-Paket eine logrotate-Konfiguration enthält, kann diese Konfiguration nun in eine eigene Datei im Verzeichnis `logrotate.d` abgelegt werden. Diese Datei kann vom Webserver-Paket aktualisiert werden, ohne die übrige logrotate-Konfiguration zu beeinträchtigen. Ebenso können Pakete bestimmte Aufgaben hinzufügen, indem sie Dateien in das Verzeichnis `/etc/cron.d` legen, anstatt `/etc/crontab` zu ändern.

In Debian — und Debian-Derivaten — wurde dieser Ansatz auf die Liste der vertrauenswürdigen Quellen angewendet, die vom Paketverwaltungswerkzeug `apt` gelesen werden: Abgesehen vom klassischen `/etc/apt/sources.list` finden wir jetzt das Verzeichnis `/etc/apt/sources.list.d`:

```
$ ls /etc/apt/sources*  
/etc/apt/sources.list  
/etc/apt/sources.list.d:
```

Konfigurationsdateien im Home-Verzeichnis (Dotfiles)

Auf Benutzerebene speichern Programme ihre Konfigurationen und Einstellungen in versteckten Dateien im Heimatverzeichnis des Benutzers (dargestellt als `~`), wobei versteckte Dateien mit einem Punkt (`.`) beginnen — daher ihr Name: *Dotfiles*.

Einige dieser Dotfiles sind Bash-Skripte, die die Shell-Sitzung des Benutzers anpassen und ausgelesen werden, sobald sich der Benutzer am System anmeldet:

.bash_history

speichert den Verlauf der Kommandozeile

.bash_logout

enthält Befehle, die beim Verlassen der Login-Shell ausgeführt werden sollen

.bashrc

Initialisierungsskript der Bash für Nicht-Login-Shells

.profile

Initialisierungsskript der Bash für Login-Shells

NOTE

Lesen Sie die Lektion über “Grundlagen der Befehlszeile”, um mehr über Bash und seine Init-Dateien zu erfahren.

Andere benutzerspezifische Programmkonfigurationsdateien werden beim Start der jeweiligen Programme ausgewertet: `.gitconfig`, `.emacs.d`, `.ssh` etc.

Der Linux-Kernel

Bevor ein Prozess ausgeführt werden kann, muss der Kernel in einen geschützten Speicherbereich geladen werden. Danach löst der Prozess mit PID 1 (heute meist `systemd`) die Prozesskette aus, d.h. ein Prozess startet einen anderen Prozess usw. Sobald die Prozesse aktiv sind, kann ihnen der Linux-Kernel Ressourcen (Tastatur, Maus, Festplatten, Speicher, Netzwerkschnittstellen usw.) zuweisen.

NOTE

Vor `systemd` war `/sbin/init` immer der erste Prozess in einem Linux-System als Teil des *System V Init* System Managers. Tatsächlich finden Sie `/sbin/init` immer noch, aber in der Regel als Link zu `/lib/systemd/systemd`.

Wo Kernel gespeichert sind: `/boot`

Der Kernel befindet sich in `/boot` — zusammen mit anderen boot-bezogenen Dateien, von denen die meisten die Teile der Kernel-Versionsnummer im Namen haben (Kernel-Version, Major-Revision, Minor-Revision und Patch-Nummer).

Das Verzeichnis `/boot` enthält die folgenden Dateitypen, deren Namen der jeweiligen Kernelversion entsprechen:

`config-4.9.0-9-amd64`

Konfigurationseinstellungen für den Kernel wie Optionen und Module, die zusammen mit dem Kernel kompiliert wurden.

`initrd.img-4.9.0-9-amd64`

Initiales RAM-Disk-Image, das beim Start hilft, indem es ein temporäres Root-Dateisystem in den Speicher lädt.

`System-map-4.9.0-9-amd64`

Die Datei `System-map` (auf einigen Systemen auch `System.map`) enthält Speicheradressorte für Kernel-Symbolnamen. Jedes Mal, wenn ein Kernel neu gebaut wird, ändert sich der Inhalt der Datei, da die Speicherorte unterschiedlich sein können. Der Kernel benutzt diese Datei, um Speicheradressorte für ein bestimmtes Kernel-Symbol nachzuschlagen, oder umgekehrt.

`vmlinuz-4.9.0-9-amd64`

Der Kernel selbst in einem selbstextrahierenden, platzsparenden, komprimierten Format (dafür steht das `z` in `vmlinuz`; `vm` steht für virtuellen Speicher (virtual memory) und wurde

verwendet, als der Kernel zum ersten Mal Unterstützung für virtuellen Speicher erhielt).

grub

Konfigurationsverzeichnis für den grub2 Bootloader.

TIP

Da es sich um ein kritisches Merkmal des Betriebssystems handelt, werden mehr als ein Kernel und die zugehörigen Dateien in `/boot` aufbewahrt, falls die Standardversion fehlerhaft wird und wir auf eine frühere Version zurückgreifen müssen, um wenigstens das System starten und reparieren zu können.

Das Verzeichnis `/proc`

Das Verzeichnis `/proc` ist eines der sogenannten virtuellen oder Pseudo-Dateisysteme, da sein Inhalt nicht auf die Festplatte geschrieben, sondern in den Arbeitsspeicher geladen wird. Es wird bei jedem Hochfahren des Computers dynamisch gefüllt und spiegelt ständig den aktuellen Zustand des Systems wider. `/proc` enthält Informationen über:

- Laufende Prozesse
- Kernelkonfiguration
- Systemhardware

Neben allen Daten zu Prozessen, die wir in der nächsten Lektion sehen werden, speichert dieses Verzeichnis auch Dateien mit Informationen über die Hardware des Systems und die Konfigurationseinstellungen des Kernels, darunter einige dieser Dateien:

`/proc/cpuinfo`

speichert Informationen über die CPU des Systems:

```
$ cat /proc/cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 158
model name    : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
stepping      : 10
cpu MHz       : 3696.000
cache size    : 12288 KB
(...)
```


/proc/cmdline

speichert die Zeichenketten, die beim Booten an den Kernel übergeben werden:

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro
quiet
```

/proc/modules

zeigt die Liste der in den Kernel geladenen Module an:

```
$ cat /proc/modules
nls_utf8 16384 1 - Live 0xffffffffc0644000
isofs 40960 1 - Live 0xffffffffc0635000
udf 90112 0 - Live 0xffffffffc061e000
crc_itu_t 16384 1 udf, Live 0xffffffffc04be000
fuse 98304 3 - Live 0xffffffffc0605000
vboxsf 45056 0 - Live 0xffffffffc05f9000 (0)
joydev 20480 0 - Live 0xffffffffc056e000
vboxguest 327680 5 vboxsf, Live 0xffffffffc05a8000 (0)
hid_generic 16384 0 - Live 0xffffffffc0569000
(...)
```

Das Verzeichnis /proc/sys

Dieses Verzeichnis enthält Kernelkonfigurationseinstellungen in Dateien, die in Kategorien pro Unterverzeichnis eingeteilt sind:

```
$ ls /proc/sys
abi debug dev fs kernel net user vm
```

Die meisten dieser Dateien verhalten sich wie ein Schalter und enthalten daher nur einen der beiden möglichen Werte: 0 oder 1 ("on" oder "off"), zum Beispiel:

/proc/sys/net/ipv4/ip_forward

Der Wert, der unsere Maschine aktiviert oder deaktiviert, um als Router zu fungieren (d.h. Pakete weiterleiten zu können):

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

Es gibt jedoch einige Ausnahmen:

/proc/sys/kernel/pid_max

Die maximal zulässige PID:

```
$ cat /proc/sys/kernel/pid_max
32768
```

WARNING

Seien Sie besonders vorsichtig, wenn Sie die Kernel-Einstellungen ändern, da ein falscher Wert zu einem instabilen System führen kann.

Hardware-Geräte

Denken Sie daran: Unter Linux “ist alles eine Datei”. Auch Hardware-Geräteinformationen und die eigenen Konfigurationseinstellungen des Kernels werden in speziellen Dateien gespeichert, die sich in virtuellen Verzeichnissen befinden.

Das Verzeichnis /dev

Das Geräteverzeichnis `/dev` (“device”) enthält Gerätedateien (*Nodes* oder Knoten) für alle angeschlossenen Hardware-Geräte. Diese Gerätedateien bilden die Schnittstelle zwischen den Geräten und den sie verwendenden Prozessen. Jede dieser Dateien gehört zu einer von zwei Kategorien:

Blockgeräte oder Block Devices

Sind solche, bei denen Daten in Blöcken gelesen und geschrieben werden, die individuell adressiert werden können, z.B. Festplatten (und deren Partitionen, wie `/dev/sda1`), USB-Sticks, CDs, DVDs etc.

Zeichenorientierte Geräte oder Character Devices

Sind solche, bei denen Daten nacheinander zeichenweise gelesen und geschrieben werden, z.B. Tastaturen, die Textkonsole (`/dev/console`), serielle Schnittstellen (wie `/dev/ttyS0` und so weiter) etc.

Wenn Sie Gerätedateien auflisten, stellen Sie sicher, dass Sie `ls` mit dem Schalter `-l` verwenden, um zwischen den beiden zu unterscheiden. Wir können zum Beispiel nach Festplatten und Partitionen suchen:

```
# ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0 may 25 17:02 /dev/sda
```

```
brw-rw---- 1 root disk 8, 1 may 25 17:02 /dev/sda1
brw-rw---- 1 root disk 8, 2 may 25 17:02 /dev/sda2
(...)
```

Oder nach seriellen Terminals (TeleTYpewriter):

```
# ls -l /dev/tty*
crw-rw-rw- 1 root tty      5,  0 may 25 17:26 /dev/tty
crw--w---- 1 root tty      4,  0 may 25 17:26 /dev/tty0
crw--w---- 1 root tty      4,  1 may 25 17:26 /dev/tty1
(...)
```

Beachten Sie, wie das erste Zeichen **b** Block Devices bzw. **c** Character Devices kennzeichnet.

TIP

Das Sternchen (*) ist ein Globbing-Zeichen, das 0 oder mehr Zeichen repräsentiert. Daher ist es in den obigen Befehlen `ls -l /dev/sd*` und `ls -l /dev/tty*` wichtig. Um mehr über diese Sonderzeichen zu erfahren, lesen Sie die Lektion über Globbing.

Außerdem enthält `/dev` einige spezielle Dateien, die für verschiedene Programmierzwecke sehr nützlich sind:

`/dev/zero`

stellt so viele Nullzeichen wie gewünscht bereit.

`/dev/null`

auch “Datenmülleimer” oder “bit bucket” genannt, verwirft alle Informationen, die dorthin gesendet werden.

`/dev/urandom`

erzeugt Pseudozufallszahlen.

Das Verzeichnis `/sys`

Das `sys`-Dateisystem (`sysfs`) ist auf `/sys` gemountet. Es wurde mit Kernel 2.6 eingeführt und bedeutete eine große Verbesserung gegenüber `/proc/sys`.

Prozesse müssen mit den Geräten in `/dev` interagieren, und so benötigt der Kernel ein Verzeichnis, das Informationen über diese Hardware-Geräte enthält. Dieses Verzeichnis ist `/sys`, und seine Daten sind nach Kategorien geordnet. Um beispielsweise die MAC-Adresse Ihrer Netzwerkkarte (`enp0s3`) zu überprüfen, würden Sie die folgende Datei mit `cat` anzeigen:

```
$ cat /sys/class/net/enp0s3/address
08:00:27:02:b2:74
```

Speicher und Speichertypen

Damit ein Programm ausgeführt werden kann, muss es grundsätzlich in den Speicher geladen werden. Im Allgemeinen beziehen wir uns beim Thema Speicher auf *Random Access Memory* (RAM); verglichen mit mechanischen Festplatten hat er den Vorteil, viel schneller zu sein. Auf der anderen Seite ist er volatil, d.h. wenn der Computer heruntergefahren wird, sind die Daten weg.

Wenn es um Speicher geht, unterscheiden wir zwei Haupttypen in einem Linux-System:

Physischen Speicher

auch bekannt als *RAM*, hat die Form von Chips, die aus integrierten Schaltungen mit Millionen von Transistoren und Kondensatoren bestehen. Diese wiederum bilden Speicherzellen (der Grundbaustein des Computerspeichers), von denen jeder ein hexadezimaler Code (eine Speicheradresse) zugeordnet ist, so dass er bei Bedarf referenziert werden kann.

Swap

auch *Swap Space* genannt, ist der Teil des virtuellen Speichers, der auf der Festplatte lebt und verwendet wird, wenn kein RAM mehr verfügbar ist.

Auf der anderen Seite gibt es das Konzept des *virtuellen Speichers*, eine Abstraktion der Gesamtmenge an nutzbarem, adressierendem Speicher (RAM, aber auch Festplattenspeicher) aus Sicht der Anwendungen.

`free` parst `/proc/meminfo` und zeigt die Menge an freiem und verbrauchtem Speicher im System sehr übersichtlich an:

```
$ free
```

| | total | used | free | shared | buff/cache | available |
|-------|---------|---------|---------|--------|------------|-----------|
| Mem: | 4050960 | 1474960 | 1482260 | 96900 | 1093740 | 2246372 |
| Swap: | 4192252 | 0 | 4192252 | | | |

Klären wir die Bedeutung der einzelnen Spalten:

total

Gesamtmenge des installierten physischen und Swap-Speichers.

used

Menge des aktuell verwendeten physischen und Swap-Speichers.

free

Menge des physischen und Swap-Speichers, die aktuell nicht verwendet wird.

shared

Menge des (meist) von `tmpfs` verwendeten physikalischen Speichers.

buff/cache

Menge des physischen Speichers, der aktuell von Kernelpuffern, dem Seitencache und den Slabs verwendet wird.

available

schätzt, wie viel physischer Speicher für neue Prozesse zur Verfügung steht.

Standardmäßig zeigt `free` Werte in Kibibytes an, ermöglicht aber eine Vielzahl von Schaltern, um die Ergebnisse in verschiedenen Maßeinheiten anzuzeigen, darunter folgende:

-b

Bytes

-m

Mebibytes

-g

Gibibytes

-h

für Menschen lesbares Format

`-h` ist immer angenehm zu lesen:

```
$ free -h
```

| | total | used | free | shared | buff/cache | available |
|-------|-------|------|------|--------|------------|-----------|
| Mem: | 3,9G | 1,4G | 1,5G | 75M | 1,0G | 2,2G |
| Swap: | 4,0G | 0B | 4,0G | | | |

NOTE

Ein Kibibyte (KiB) entspricht 1.024 Byte, während ein Kilobyte (KB) 1000 Byte entspricht. Dasselbe gilt jeweils für Mebibytes, Gibibytes, etc.

Geführte Übungen

1. Verwenden Sie den Befehl `which`, um die Position der folgenden Programme herauszufinden und die Tabelle zu vervollständigen:

| Programm | <code>which</code> Befehl | Pfad zur ausführbaren Datei (Ausgabe) | Benutzer benötigt root-Rechte? |
|----------------------|---------------------------|---------------------------------------|--------------------------------|
| <code>swapon</code> | | | |
| <code>kill</code> | | | |
| <code>cut</code> | | | |
| <code>usermod</code> | | | |
| <code>cron</code> | | | |
| <code>ps</code> | | | |

2. Wo sind die folgenden Dateien zu finden?

| Datei | /etc | ~ |
|--------------------------|------|---|
| <code>.bashrc</code> | | |
| <code>bash.bashrc</code> | | |
| <code>passwd</code> | | |
| <code>.profile</code> | | |
| <code>resolv.conf</code> | | |
| <code>sysctl.conf</code> | | |

3. Erklären Sie die Bedeutung der Zahlenelemente für die Kerneldatei `vmlinuz-4.15.0-50-generic` in `/boot`:

| Zahlenelement | Bedeutung |
|---------------|-----------|
| 4 | |
| 15 | |
| 0 | |
| 50 | |

4. Welchen Befehl würden Sie verwenden, um alle Festplatten und Partitionen in `/dev` aufzulisten?

Offene Übungen

1. Gerätedateien für Festplatten werden auf der Grundlage der Controller dargestellt, die sie verwenden. Wir haben `/dev/sd*` für Laufwerke mit SCSI (Small Computer System Interface) und SATA (Serial Advanced Technology Attachment) gesehen, aber

- Wie wurden alte IDE (Integrated Drive Electronics) Laufwerke dargestellt?

- Und moderne NVMe (Non-Volatile Memory Express) Laufwerke?

2. Werfen Sie einen Blick auf die Datei `/proc/meminfo`. Vergleichen Sie den Inhalt dieser Datei mit der Ausgabe des Befehls `free` und identifizieren Sie, welcher Schlüssel aus `/proc/meminfo` den folgenden Feldern in der Ausgabe von `free` entspricht:

| free Ausgabe | /proc/meminfo Feld |
|--------------|--------------------|
| total | |
| free | |
| shared | |
| buff/cache | |
| available | |

Zusammenfassung

In dieser Lektion haben Sie sich mit dem Speicherort von Programmen und deren Konfigurationsdateien in einem Linux-System vertraut gemacht. Wichtige Fakten, die Sie beachten sollten:

- Grundsätzlich sind Programme in einer dreistufigen Verzeichnisstruktur zu finden: `/`, `/usr` und `/usr/local`. Jede dieser Ebenen kann `bin` und `sbin` Verzeichnisse enthalten.
- Konfigurationsdateien werden in `/etc` und `~` gespeichert.
- Punktdateien sind versteckte Dateien, die mit einem Punkt (.) beginnen.

Wir haben auch über den Linux-Kernel gesprochen. Wichtige Fakten sind:

- Für Linux ist alles eine Datei.
- Der Linux-Kernel lebt in `/boot` zusammen mit anderen boot-bezogenen Dateien.
- Damit Prozesse mit der Ausführung beginnen können, muss der Kernel zuerst in einen geschützten Speicherbereich geladen werden.
- Aufgabe des Kernels ist es, Systemressourcen den Prozessen zuzuweisen.
- Das virtuelle (oder Pseudo-)Dateisystem `/proc` speichert wichtige Kernel- und Systemdaten auf flüchtige Weise.

Ebenso haben wir uns mit Hardware-Geräten beschäftigt und folgendes gelernt:

- Das Verzeichnis `/dev` speichert spezielle Dateien (auch bekannt als Knoten) für alle angeschlossenen Hardware-Geräte: *Block Devices* oder *Character Devices*. Die ersten übertragen Daten in Blöcken, letztere zeichenweise.
- Das Verzeichnis `/dev` enthält auch andere spezielle Dateien wie `/dev/zero`, `/dev/null` oder `/dev/urandom`.
- Das Verzeichnis `/sys` speichert Informationen über Hardwaregeräte, die Kategorien zugewiesen sind.

Schließlich haben wir Speicher behandelt und gelernt:

- Ein Programm wird ausgeführt, wenn es in den Speicher geladen wird.
- Was RAM (Random Access Memory) ist.
- Was Swap ist.
- Wie man die Speichernutzung anzeigt.

Befehle, die in dieser Lektion verwendet wurden:

cat

Dateiinhalte verketteten/ausgeben.

free

Zeigt die Menge an freiem und verbrauchtem Speicher im System an.

ls

Listet Verzeichnisinhalte auf.

which

Zeigt den Standort des Programms an.

Lösungen zu den geführten Übungen

1. Verwenden Sie den Befehl `which`, um die Position der folgenden Programme herauszufinden und die Tabelle zu vervollständigen:

| Programm | <code>which</code> Befehl | Pfad zur ausführbaren Datei (Ausgabe) | Benutzer benötigt root-Rechte? |
|----------------------|----------------------------|---------------------------------------|--------------------------------|
| <code>swapon</code> | <code>which swapon</code> | <code>/sbin/swapon</code> | Ja |
| <code>kill</code> | <code>which kill</code> | <code>/bin/kill</code> | Nein |
| <code>cut</code> | <code>which cut</code> | <code>/usr/bin/cut</code> | Nein |
| <code>usermod</code> | <code>which usermod</code> | <code>/usr/sbin/usermod</code> | Ja |
| <code>cron</code> | <code>which cron</code> | <code>/usr/sbin/cron</code> | Ja |
| <code>ps</code> | <code>which ps</code> | <code>/bin/ps</code> | Nein |

2. Wo sind die folgenden Dateien zu finden?

| Datei | <code>/etc</code> | <code>~</code> |
|--------------------------|-------------------|----------------|
| <code>.bashrc</code> | Nein | Ja |
| <code>bash.bashrc</code> | Ja | Nein |
| <code>passwd</code> | Ja | Nein |
| <code>.profile</code> | Nein | Ja |
| <code>resolv.conf</code> | Ja | Nein |
| <code>sysctl.conf</code> | Ja | Nein |

3. Erklären Sie die Bedeutung der Zahlenelemente für die Kerneldatei `vmlinuz-4.15.0-50-generic` in `/boot`:

| Zahlenelement | Bedeutung |
|---------------|----------------|
| 4 | Kernel-Version |
| 15 | Major-Revision |
| 0 | Minor-Revision |
| 50 | Patch-Nummer |

4. Welchen Befehl würden Sie verwenden, um alle Festplatten und Partitionen in `/dev` aufzulisten?

```
ls /dev/sd*
```

Lösungen zu den offenen Übungen

1. Gerätedateien für Festplatten werden auf der Grundlage der Controller dargestellt, die sie verwenden. Wir haben `/dev/sd*` für Laufwerke mit SCSI (Small Computer System Interface) und SATA (Serial Advanced Technology Attachment) gesehen, aber

- Wie wurden alte IDE (Integrated Drive Electronics) Laufwerke dargestellt?

`/dev/hd*`

- Und moderne NVMe (Non-Volatile Memory Express) Laufwerke?

`/dev/nvme*`

2. Werfen Sie einen Blick auf die Datei `/proc/meminfo`. Vergleichen Sie den Inhalt dieser Datei mit der Ausgabe des Befehls `free` und identifizieren Sie, welcher Schlüssel aus `/proc/meminfo` den folgenden Feldern in der Ausgabe von `free` entspricht:

| free Ausgabe | /proc/meminfo Feld |
|--------------|----------------------------------|
| total | MemTotal / SwapTotal |
| free | MemFree / SwapFree |
| shared | Shmem |
| buff/cache | Buffers, Cached und SReclaimable |
| available | MemAvailable |



4.3 Lektion 2

| | |
|--------------------|---------------------------------|
| Zertifikat: | Linux Essentials |
| Version: | 1.6 |
| Thema: | 4 Das Linux-Betriebssystem |
| Lernziel: | 4.3 Wo Daten gespeichert werden |
| Lektion: | 2 von 2 |

Einführung

Nachdem wir uns Programme und deren Konfigurationsdateien genauer angesehen haben, lernen wir in dieser Lektion, wie Befehle als Prozesse ausgeführt werden. Darber hinaus geht es um Systemmeldungen, die Verwendung des Kernel Ring Buffers und wie `systemd` und seine Journal-Daemon (`journald`) die bis dahin übliche Systemprotokollierung verändert haben.

Prozesse

Jedes Mal, wenn ein Benutzer einen Befehl ausführt, werden ein Programm ausgeführt und ein oder mehrere Prozesse generiert.

Prozesse sind hierarchisch geordnet. Nachdem der Kernel beim Booten in den Speicher geladen wurde, wird der erste Prozess gestartet, der wiederum andere Prozesse startet, die wiederum andere Prozesse starten können. Jeder Prozess hat eine eindeutige Kennung (PID) und eine Kennung des Elternprozesses (Parent Process, PPID) — positive ganze, fortlaufende Zahlen.

Prozesse dynamisch untersuchen: top

Mit dem Befehl `top` erhalten Sie eine dynamische Liste aller laufenden Prozesse:

```
$ top
```

```
top - 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14
Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache
KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem
```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|-----|-------|----|-----|-------|------|------|---|------|------|---------|---------------|
| 436 | carol | 20 | 0 | 42696 | 3624 | 3060 | R | 0,7 | 0,4 | 0:00.30 | top |
| 4 | root | 20 | 0 | 0 | 0 | 0 | S | 0,3 | 0,0 | 0:00.12 | kworker/0:0 |
| 399 | root | 20 | 0 | 95204 | 6748 | 5780 | S | 0,3 | 0,7 | 0:00.22 | sshd |
| 1 | root | 20 | 0 | 56872 | 6596 | 5208 | S | 0,0 | 0,6 | 0:01.29 | systemd |
| 2 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | kthreadd |
| 3 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.02 | ksoftirqd/0 |
| 5 | root | 0 | -20 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | kworker/0:0H |
| 6 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | kworker/u2:0 |
| 7 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.08 | rcu_sched |
| 8 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | rcu_bh |
| 9 | root | rt | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | migration/0 |
| 10 | root | 0 | -20 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | lru-add-drain |

(...)

Wie wir oben sehen, liefert `top` auch Informationen über den Speicher- und CPU-Verbrauch des Gesamtsystems sowie für jeden Prozess.

`top` erlaubt dem Benutzer eine gewisse Interaktion.

Standardmäßig ist die Ausgabe nach dem Prozentsatz der von jedem Prozess verbrauchten CPU-Zeit in absteigender Reihenfolge sortiert, was durch Drücken der folgenden Tasten innerhalb von `top` zu ändern ist:

M

Sortieren nach Speicherverbrauch.

N

Sortieren nach Prozess-ID.

T

Sortieren nach Laufzeit.

P

Sortieren nach Prozentsatz der CPU-Auslastung.

Um zwischen absteigender/aufsteigender Reihenfolge zu wechseln, drücken Sie einfach R.

TIP

Eine hübschere und benutzerfreundlichere Version von `top` ist `htop`. Eine weitere — vielleicht schon zu ausführliche — Alternative ist `atop`. Wenn nicht bereits in Ihrem System installiert, nutzen Sie den Paketmanager, um beide zu installieren und auszuprobieren.

Eine Momentaufnahme von Prozessen: `ps`

Ein weiterer sehr nützlicher Befehl für Informationen über Prozesse ist `ps`. Während `top` dynamische Informationen liefert, ist `ps` statisch.

Ohne Optionen aufgerufen, ist die Ausgabe von `ps` ziemlich übersichtlich und bezieht sich nur auf die Prozesse innerhalb der aktuellen Shell:

```
$ ps
  PID TTY          TIME CMD
 2318 pts/0    00:00:00 bash
 2443 pts/0    00:00:00 ps
```

Die angezeigten Informationen umfassen die Prozesskennung (PID), das Terminal, in dem der Prozess ausgeführt wird (TTY), die vom Prozess benötigte CPU-Zeit (TIME) und den Befehl zum Starten des Prozesses (CMD).

Ein nützlicher Schalter für `ps` ist `-f`, der eine vollständige Liste anzeigt:

```
$ ps -f
  UID          PID    PPID  C STIME TTY          TIME CMD
  carol        2318    1682  0  08:38 pts/1    00:00:00 bash
  carol        2443    2318  0  08:46 pts/1    00:00:00 ps -f
```

In Verbindung mit anderen Schaltern zeigt `-f` die Beziehung zwischen Eltern- und Kindprozessen an:


```
$ ps -uf
```

```
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
carol    2318  0.0  0.1  21336  5140 pts/1    Ss   08:38   0:00 bash
carol    2492  0.0  0.0  38304  3332 pts/1    R+   08:51   0:00 \_ ps -uf
carol    1780  0.0  0.1  21440  5412 pts/0    Ss   08:28   0:00 bash
carol    2291  0.0  0.7 305352 28736 pts/0    Sl+  08:35   0:00 \_ emacs index.en.adoc
-nw
(...)
```

Ebenso kann `ps` den Prozentsatz des Speicherverbrauchs anzeigen, wenn er mit dem Schalter `-v` aufgerufen wird:

```
$ ps -v
```

```
PID TTY      STAT   TIME  MAJFL   TRS   DRS   RSS %MEM COMMAND
1163 tty2    Ssl+   0:00    1    67 201224 5576  0.1 /usr/lib/gdm3/gdm-x-session (...)
(...)
```

NOTE

Ein weiterer optisch ansprechender Befehl, der die Hierarchie der Prozesse darstellt, ist `pstree`, den alle gängigen Distributionen enthalten.

Prozessinformationen im Verzeichnis `/proc`

Wir haben das Dateisystem `/proc` bereits kennengelernt. `/proc` enthält ein nummeriertes Unterverzeichnis für jeden laufenden Prozess im System (die Nummer ist die `PID` des Prozesses):

```
carol@debian:~# ls /proc
```

```
1    108 13 17 21 27 354 41 665 8 9
10   109 14 173 22 28 355 42 7 804 915
103  11 140 18 23 29 356 428 749 810 918
104  111 148 181 24 3 367 432 75 811
105  112 149 19 244 349 370 433 768 83
106  115 15 195 25 350 371 5 797 838
107  12 16 2 26 353 404 507 798 899
(...)
```

Sämtliche Informationen zu einem bestimmten Prozess liegen also in einem Verzeichnis. Lassen Sie uns den Inhalt des ersten Prozesses auflisten, dessen `PID` gleich `1` ist (die Ausgabe wurde zur besseren Lesbarkeit abgeschnitten):

```
# ls /proc/1/
```

| | | | | | |
|-----------|-----------------|---------|----------|-----------|-----------|
| attr | cmdline | environ | io | mem | ns |
| autogroup | comm | exe | limits | mountinfo | numa_maps |
| auxv | coredump_filter | fd | loginuid | mounts | oom_adj |
| ... | | | | | |

Sie können z.B. die ausführbare Datei des Prozesses überprüfen:

```
# cat /proc/1/cmdline; echo
/sbin/init
```

Wie Sie sehen, ist die Binärdatei, die die Hierarchie der Prozesse gestartet hat, `/sbin/init`.

NOTE

Befehle können mit einem Semikolon (;) verknüpft werden. Der Grund für die Verwendung des Befehls `echo` ist die Bereitstellung einer neuen Zeile. Führen Sie einfach `cat /proc/1/cmdline` aus, um den Unterschied zu sehen.

Die Systemlast

Jeder Prozess in einem System verbraucht potenziell Systemressourcen. Die so genannte *Systemlast* (*System Load*) versucht, die Gesamtlast des Systems zu einem einzigen numerischen Indikator zu aggregieren, den Sie mit dem Befehl `uptime` sehen:

```
$ uptime
22:12:54 up 13 days, 20:26, 1 user, load average: 2.91, 1.59, 0.39
```

Die drei letzten Ziffern zeigen die durchschnittliche Load des Systems für die letzte Minute (2.91), die letzten fünf Minuten (1.59) und die letzten fünfzehn Minuten (0.39).

Jede dieser Zahlen gibt an, wie viele Prozesse entweder auf CPU-Ressourcen oder auf den Abschluss von Ein-/Ausgabevorgängen gewartet haben, d.h. diese Prozesse waren ausführbereit, wenn sie die entsprechenden Ressourcen bekommen hätten.

Systemprotokoll und Systemmeldungen

Sobald der Kernel und die Prozesse mit der Ausführung und Kommunikation untereinander beginnen, entstehen viele Informationen. Die meisten werden an Dateien gesendet - die sogenannten *Logfiles* oder einfach *Logs*.

Ohne Logging wäre die Suche nach Ereignissen auf einem Server für Systemadministratoren überaus schwierig. Darum ist es wichtig, alle Systemereignisse standardisiert und zentralisiert zu

erfassen und im Auge zu behalten. Protokolle sind entscheidend, wenn es um Fehlersuche und Sicherheit geht, aber auch um verlässliche Datenquellen zum Verständnis von Systemstatistiken und zur Vorhersage weiterer Entwicklungen.

Logging mit dem syslog-Daemon

Traditionell werden Systemmeldungen von dem Logging-Tool `syslog` oder einem der davon abgeleiteten Tools wie `syslog-ng` oder `rsyslog` verwaltet. Der Logging-Daemon sammelt Nachrichten von anderen Diensten und Programmen und speichert sie in Protokolldateien, typischerweise unter `/var/log`. Einige Dienste kümmern sich jedoch um ihre eigenen Protokolle (z.B. der Apache HTTPD Webserver), ebenso wie der Linux-Kernel einen In-Memory-Ringpuffer zur Speicherung seiner Log-Nachrichten verwendet.

Log-Dateien in `/var/log`

Da es sich bei Logs um Daten handelt, die sich im Laufe der Zeit verändern, finden Sie sie normalerweise in `/var/log`.

Wenn Sie sich `/var/log` ansehen, werden Sie feststellen, dass die Namen der Protokolle — bis zu einem gewissen Grad — selbsterklärend sind. Hier einige Beispiele:

`/var/log/auth.log`

Speichert Informationen zur Authentifizierung.

`/var/log/kern.log`

Speichert Kernel-Informationen.

`/var/log/syslog`

Speichert Systeminformationen.

`/var/log/messages`

Speichert System- und Anwendungsdaten.

NOTE

Der genaue Name und Inhalt der Protokolldateien kann je nach Linux-Distribution variieren.

Zugriff auf Protokolldateien

Denken Sie beim Durchsuchen von Protokolldateien daran, `root` zu sein (wenn Sie keine Leseberechtigung haben) und einen Pager wie `less` zu verwenden:

less /var/log/messages

```
Jun  4 18:22:48 debian liblogging-stdlog: [origin software="rsyslogd" swVersion="8.24.0" x-
pid="285" x-info="http://www.rsyslog.com"] rsyslogd was HUPed
Jun 29 16:57:10 debian kernel: [    0.000000] Linux version 4.9.0-8-amd64 (debian-
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP
Debian 4.9.130-2 (2018-10-27)
Jun 29 16:57:10 debian kernel: [    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-
8-amd64 root=/dev/sda1 ro quiet
```

Alternativ können Sie **tail** mit dem Schalter **-f** verwenden, um die neuesten Nachrichten der Datei zu lesen und dynamisch neue Zeilen anzuzeigen, während sie angehängt werden:

tail -f /var/log/messages

```
Jul  9 18:39:37 debian kernel: [    2.350572] RAPL PMU: hw unit of domain psys 2^-0 Joules
Jul  9 18:39:37 debian kernel: [    2.512802] input: VirtualBox USB Tablet as
/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/0003:80EE:0021.0001/input/input7
Jul  9 18:39:37 debian kernel: [    2.513861] Adding 1046524k swap on /dev/sda5.  Priority:-
1 extents:1 across:1046524k FS
Jul  9 18:39:37 debian kernel: [    2.519301] hid-generic 0003:80EE:0021.0001:
input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
Jul  9 18:39:37 debian kernel: [    2.623947] snd_intel8x0 0000:00:05.0: white list rate for
1028:0177 is 48000
Jul  9 18:39:37 debian kernel: [    2.914805] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not
ready
Jul  9 18:39:39 debian kernel: [    4.937283] e1000: enp0s3 NIC Link is Up 1000 Mbps Full
Duplex, Flow Control: RX
Jul  9 18:39:39 debian kernel: [    4.938493] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link
becomes ready
Jul  9 18:39:40 debian kernel: [    5.315603] random: crng init done
Jul  9 18:39:40 debian kernel: [    5.315608] random: 7 urandom warning(s) missed due to
ratelimiting
```

Sie finden die Ausgabe im folgenden Format:

- Zeitstempel
- Hostname, von dem die Nachricht kam
- Name des Programms/Dienstes, der die Nachricht erzeugte
- PID des Programms, das die Nachricht erzeugte
- Beschreibung der Aktion, die stattgefunden hat

Die meisten Logfiles sind Klartextdateien, aber einige wenige können binäre Daten enthalten, wie z.B. `/var/log/wtmp` mit Daten, die für eine erfolgreiche Anmeldungen relevant sind. Der Befehl `file` gibt Aufschluss:

```
$ file /var/log/wtmp
/var/log/wtmp: dBase III DBT, version number 0, next free block index 8
```

Solche Dateien werden normalerweise mit speziellen Befehlen gelesen. `last` wird verwendet, um die Daten in `/var/log/wtmp` zu interpretieren:

```
$ last
carol    tty2      :0          Thu May 30 10:53  still logged in
reboot   system boot 4.9.0-9-amd64 Thu May 30 10:52  still running
carol    tty2      :0          Thu May 30 10:47 - crash (00:05)
reboot   system boot 4.9.0-9-amd64 Thu May 30 09:11  still running
carol    tty2      :0          Tue May 28 08:28 - 14:11 (05:42)
reboot   system boot 4.9.0-9-amd64 Tue May 28 08:27 - 14:11 (05:43)
carol    tty2      :0          Mon May 27 19:40 - 19:52 (00:11)
reboot   system boot 4.9.0-9-amd64 Mon May 27 19:38 - 19:52 (00:13)
carol    tty2      :0          Mon May 27 19:35 - down  (00:03)
reboot   system boot 4.9.0-9-amd64 Mon May 27 19:34 - 19:38 (00:04)
```

NOTE

Ähnlich wie bei `/var/log/wtmp` speichert `/var/log/btmp` Informationen über fehlgeschlagene Anmeldeversuche, und der spezielle Befehl zum Lesen des Inhalts ist `lastb`.

Log Rotation

Protokolldateien können über Wochen oder Monate hinweg stark wachsen und den gesamten freien Festplattenspeicher beanspruchen. Hier hilft `logrotate`, das eine Log Rotation oder einen Zyklus implementiert, so dass Log-Dateien umbenannt, archiviert und/oder komprimiert, manchmal per E-Mail an den Systemadministrator gesendet und schließlich gelöscht werden, wenn sie ein bestimmtes Alter erreicht haben. Die Konventionen zur Benennung dieser rotierten Logfiles sind vielfältig (etwa das Anfügen eines Suffixes mit Datum), aber das einfache Hinzufügen eines Suffixes mit einer ganzen Zahl ist üblich:

```
# ls /var/log/apache2/
access.log  error.log  error.log.1  error.log.2.gz  other_vhosts_access.log
```

Beachten Sie, dass `error.log.2.gz` bereits mit `gzip` komprimiert wurde (daher das Suffix `.gz`).

Der Kernel Ring Buffer

Der Kernel Ring Buffer ist eine Datenstruktur fester Größe, die Kernel-Meldungen sowohl beim Boot-Prozess als auch live aufzeichnet. Eine wichtige Funktion besteht darin, alle beim Booten erzeugten Kernel-Meldungen zu protokollieren, solange `syslog` noch nicht verfügbar ist. Der Befehl `dmesg` gibt den Kernel Ring Buffer aus (der früher auch in `/var/log/dmesg` gespeichert war). Aufgrund der Erweiterung des Ringspeichers wird dieser Befehl normalerweise in Kombination mit dem Textfilterprogramm `grep` oder einem Pager wie `less` verwendet. Um etwa nach Boot-Meldungen zu suchen:

```
$ dmesg | grep boot
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[ 0.000000] smpboot: Allowing 1 CPUs, 0 hotplug CPUs
[ 0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[ 0.144986] AppArmor: AppArmor disabled by boot time parameter
(...)
```

NOTE

Wenn der Kernel Ring Buffer stetig mit neuen Nachrichten wächst, verschwinden die ältesten.

Das System-Journal: systemd-journald

Seit 2015 ersetzt `systemd` `SysV Init` als *de facto* System- und Servicemanager in den meisten großen Linux-Distributionen, so dass der Journal-Daemon (`journald`) zur Standard-Log-Komponente geworden ist und `Syslog` weitestgehend ablöst. Die Daten werden nicht mehr im Klartext, sondern in Binärform gespeichert, so dass das Dienstprogramm `journalctl` zum Lesen der Protokolle erforderlich ist. Darüber hinaus ist `journald` `syslog`-kompatibel und kann in `syslog` integriert werden.

`journalctl` ist das Dienstprogramm zum Lesen und Abfragen der Journal-Datenbank von `systemd`. Ohne Optionen aufgerufen, gibt es das gesamte Journal aus:

```
# journalctl
-- Logs begin at Tue 2019-06-04 17:49:40 CEST, end at Tue 2019-06-04 18:13:10 CEST. --
jun 04 17:49:40 debian systemd-journald[339]: Runtime journal (/run/log/journal/) is 8.0M,
max 159.6M, 151.6M free.
jun 04 17:49:40 debian kernel: microcode: microcode updated early to revision 0xcc, date =
2019-04-01
Jun 04 17:49:40 debian kernel: Linux version 4.9.0-8-amd64 (debian-kernel@lists.debian.org)
```

```
(gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) )
Jun 04 17:49:40 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-8-amd64
root=/dev/sda1 ro quiet
(...)
```

Wenn Sie es mit den Schaltern `-k` oder `--dmesg` aufrufen, entspricht es dem Aufruf des Befehls `dmesg`:

```
# journalctl -k
[    0.000000] Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (gcc version
6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP Debian 4.9.168-1+deb9u2 (2019-05-13)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-
441f-b8f5-8061c0034c74 ro quiet
(...)
```

Weitere interessante Optionen für `journalctl` sind:

-b, --boot

Zeigt Boot-Informationen an.

-u

Zeigt Meldungen über eine bestimmte Einheit an, wobei eine Einheit grob als jede vom System verwaltete Ressource definiert werden kann. So wird `journalctl -u apache2.service` verwendet, um Meldungen über den apache2 Webserver zu lesen.

-f

Zeigt die neuesten Journal-Einträge an und gibt immer wieder neue Einträge aus, sobald sie an das Journal angehängt werden — ähnlich `tail -f`.

Geführte Übungen

1. Werfen Sie einen Blick auf die folgende Ausgabe von `top` und beantworten Sie die folgenden Fragen:

```
carol@debian:~$ top
```

```
top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 1020332 total,  698700 free,  170664 used,  150968 buff/cache
KiB Swap: 1046524 total, 1046524 free,      0 used.  710956 avail Mem
```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|-----|----------|----|----|---------|--------|-------|---|------|------|---------|-------------|
| 605 | nobody | 20 | 0 | 1137620 | 132424 | 34256 | S | 6.3 | 13.0 | 1:47.24 | ntopng |
| 444 | www-data | 20 | 0 | 364780 | 4132 | 2572 | S | 0.3 | 0.4 | 0:00.44 | apache2 |
| 734 | root | 20 | 0 | 95212 | 7004 | 6036 | S | 0.3 | 0.7 | 0:00.36 | sshd |
| 887 | carol | 20 | 0 | 46608 | 3680 | 3104 | R | 0.3 | 0.4 | 0:00.03 | top |
| 1 | root | 20 | 0 | 56988 | 6688 | 5240 | S | 0.0 | 0.7 | 0:00.42 | systemd |
| 2 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | kthreadd |
| 3 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.09 | ksoftirqd/0 |
| 4 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.87 | kworker/0:0 |

(...)

- Welche Prozesse wurden vom Benutzer `carol` gestartet?
- Welches virtuelle Verzeichnis von `/proc` sollten Sie besuchen, um nach Daten des Befehls `top` zu suchen?
- Welcher Prozess wurde zuerst ausgeführt? Woher wissen Sie das?
- Vervollständigen Sie die Tabelle und geben Sie an, in welchem Bereich der `top`-Ausgabe die folgenden Informationen zu finden sind:

| Information zu | Übersichtsbereich | Aufgabenbereich |
|----------------|-------------------|-----------------|
| Speicher | | |
| Swap | | |

| Information zu | Übersichtsbereich | Aufgabenbereich |
|----------------|-------------------|-----------------|
| PID | | |
| CPU-Zeit | | |
| Befehle | | |

2. Mit welchem Befehl werden die folgenden binären Protokolle gelesen?

- `/var/log/wtmp`

- `/var/log/btmp`

- `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

3. Welche Befehle würden Sie in Kombination mit `grep` verwenden, um die folgenden Informationen über Ihr Linux-System zu erhalten?

- Wann wurde das System zuletzt neu gestartet (`wtmp`)?

- Welche Festplatten sind installiert (`kern.log`)?

- Wenn erfolgte die letzte Anmeldung (`auth.log`)?

4. Welche zwei Befehle würden Sie verwenden, um den Kernel Ring Buffer anzuzeigen?

5. Geben Sie an, wo die folgenden Protokollmeldungen hingehören:

- `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

| | |
|--------------------------------|--|
| <code>/var/log/auth.log</code> | |
| <code>/var/log/kern.log</code> | |
| <code>/var/log/syslog</code> | |
| <code>/var/log/messages</code> | |

- Jul 10 11:23:58 debian kernel: [1.923349] usbhid: USB HID core driver

| | |
|-------------------|--|
| /var/log/auth.log | |
| /var/log/kern.log | |
| /var/log/syslog | |
| /var/log/messages | |

- Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)

| | |
|-------------------|--|
| /var/log/auth.log | |
| /var/log/kern.log | |
| /var/log/syslog | |
| /var/log/messages | |

- Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting...

| | |
|-------------------|--|
| /var/log/auth.log | |
| /var/log/kern.log | |
| /var/log/syslog | |
| /var/log/messages | |

6. Hat journalctl Informationen über die folgenden Einheiten?

| Einheit | Befehl |
|------------|--------|
| ssh | |
| networking | |
| rsyslog | |
| cron | |

Offene Übungen

1. Betrachten Sie die Ausgabe von `top` in der geführten Übung und beantworten Sie die folgenden Fragen:

- Welche zwei Schritte würden folgen, um den *apache* Webserver zu töten?

- Wie könnten Sie im Übersichtsbereich die Informationen über den physischen Speicher und den Swap mit Fortschrittsbalken anzeigen?

- Sortieren Sie nun die Prozesse nach Speicherverbrauch:

- Da nun Speicherinformationen in Fortschrittsbalken angezeigt werden und die Prozesse nach Speichernutzung sortiert sind, speichern Sie diese Konfiguration, so dass Sie sie beim nächsten Aufruf von `top` als Standard verwendet wird:

- Welche Datei speichert die Konfigurationseinstellungen von `top`? Wo liegt sie? Wie kann prüfen, dass es sie gibt?

2. Machen Sie sich mit dem Befehl `exec` in Bash vertraut. Starten Sie eine Bash-Sitzung, finden Sie den Bash-Prozess mit `ps`, führen Sie anschließend `exec /bin/sh` aus und suchen Sie dann erneut nach dem Prozess mit derselben PID.

3. Folgen Sie diesen Schritten, um Kernel-Ereignisse und die dynamische Verwaltung von Geräten durch `udev` zu untersuchen:

- Schließen Sie ein USB-Laufwerk direkt an Ihren Computer an (Hotplug). Führen Sie `dmesg` aus und achten Sie auf die letzten Zeilen. Wie lautet ist die jüngste Zeile?

- Führen Sie unter Berücksichtigung der Ausgabe des vorherigen Befehls `ls /dev/sd*` aus und stellen Sie sicher, dass Ihr USB-Stick in der Liste erscheint. Wie lautet die Ausgabe?

- Entfernen Sie nun das USB-Laufwerk und führen Sie `dmesg` erneut aus. Wie lautet die letzte Zeile?

- Führen Sie `ls /dev/sd*` erneut aus und stellen Sie sicher, dass Ihr Gerät aus der Liste verschwunden ist. Was lautet die Ausgabe?

Zusammenfassung

Im Zusammenhang mit der Datenspeicherung wurden in dieser Lektion folgende Themen behandelt: Prozessmanagement sowie Systemprotokollierung und -benachrichtigungen.

Was das Prozessmanagement betrifft, so haben wir folgendes gelernt:

- Programme erzeugen Prozesse und Prozesse existieren in einer Hierarchie.
- Jeder Prozess hat eine eindeutige Kennung (PID) und eine übergeordnete Prozesskennung (PPID).
- `top` ist ein sehr nützlicher Befehl, um dynamisch und interaktiv die laufenden Prozesse des Systems zu untersuchen.
- `ps` liefert eine Momentaufnahme der aktuellen laufenden Prozesse im System.
- Das Verzeichnis `/proc` enthält Verzeichnisse für jeden laufenden Prozess im System, benannt nach der jeweiligen PID.
- Das Konzept der durchschnittlichen Systemlast — was sehr nützlich ist, um die CPU-Auslastung zu überprüfen.

In Bezug auf das System-Logging sollten Sie sich merken:

- Ein Log ist eine Datei, in der Systemereignisse aufgezeichnet werden. Logs sind für die Fehlerbehebung unverzichtbar.
- Das Logging wurde traditionell von speziellen Diensten wie `syslog`, `syslog-ng` oder `rsyslog` durchgeführt. Dennoch verwenden einige Programme ihre eigenen Logging-Daemonen.
- Da Logs variable Daten sind, werden sie in `/var` abgelegt. Manchmal geben ihre Namen einen Hinweis auf den Inhalt (`kern.log`, `auth.log`, etc.)
- Die meisten Logs sind Klartextdateien und können mit jedem Texteditor gelesen werden, solange Sie die notwendigen Berechtigungen haben. Einige davon sind jedoch binär und müssen mit speziellen Befehlen gelesen werden.
- Um Probleme mit dem Festplattenspeicher zu vermeiden, wird die Log Rotation vom Programm `logrotate` durchgeführt.
- Der Kernel nutzt eine ringförmige Datenstruktur, den Ring Buffer, in dem Boot-Meldungen gespeichert werden (alte Nachrichten verschwinden mit der Zeit).
- Der System- und Servicemanagersystemd hat System V init in praktisch allen Distributionen abgelöst, wobei `journald` zum Standard-Logging-Service wurde.
- Um das Journal von `systemd` zu lesen, wird das Programm `journalctl` benötigt.

Befehle, die in dieser Lektion verwendet wurden:

cat

Dateiinhalt verketteten/ausgeben.

dmesg

Gibt den Kernel Ring Buffer aus.

echo

Zeigt eine Textzeile oder eine neue Zeile an.

file

Bestimmt den Dateityp.

grep

Gibt Zeilen aus, die einem Muster entsprechen.

last

Gibt eine Liste der zuletzt angemeldeten Benutzer aus.

less

Zeigt den Inhalt einer Datei seitenweise an.

ls

Listet Verzeichnisinhalte auf.

journalctl

Fragt das `systemd`-Journal ab.

tail

Zeigt die letzten Zeilen einer Datei an.

Lösungen zu den geführten Übungen

1. Werfen Sie einen Blick auf die folgende Ausgabe von `top` und beantworten Sie die folgenden Fragen:

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks: 73 total,   2 running, 71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,   0.4 sy,   0.0 ni, 98.6 id,   0.0 wa,   0.0 hi,   0.0 si,   0.0 st
KiB Mem : 1020332 total,  698700 free,   170664 used,   150968 buff/cache
KiB Swap: 1046524 total, 1046524 free,        0 used.  710956 avail Mem

  PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
  605 nobody    20   0 1137620 132424 34256 S   6.3  13.0   1:47.24 ntopng
  444 www-data   20   0 364780   4132   2572 S   0.3   0.4   0:00.44 apache2
  734 root       20   0   95212   7004   6036 S   0.3   0.7   0:00.36 sshd
  887 carol     20   0   46608   3680   3104 R   0.3   0.4   0:00.03 top
    1 root       20   0   56988   6688   5240 S   0.0   0.7   0:00.42 systemd
    2 root       20   0        0        0        0 S   0.0   0.0   0:00.00 kthreadd
    3 root       20   0        0        0        0 S   0.0   0.0   0:00.09 ksoftirqd/0
    4 root       20   0        0        0        0 S   0.0   0.0   0:00.87 kworker/0:0
(...)
```

- Welche Prozesse wurden vom Benutzer `carol` gestartet?

Lösung: Nur einer: `top`.

- Welches virtuelle Verzeichnis von `/proc` sollten Sie besuchen, um nach Daten des Befehls `top` zu suchen?

Lösung: `/proc/887`

- Welcher Prozess wurde zuerst ausgeführt? Woher wissen Sie das?

Lösung: `systemd`, weil es die PID 1 hat.

- Vervollständigen Sie die Tabelle und geben Sie an, in welchem Bereich der `top`-Ausgabe die folgenden Informationen zu finden sind:

| Information zu | Übersichtsbereich | Aufgabenbereich |
|----------------|-------------------|-----------------|
| Speicher | Ja | Ja |

| Information zu | Übersichtsbereich | Aufgabenbereich |
|----------------|-------------------|-----------------|
| Swap | Ja | Nein |
| PID | Nein | Ja |
| CPU-Zeit | Ja | Ja |
| Befehle | Nein | Ja |

2. Mit welchem Befehl werden die folgenden binären Protokolle gelesen?

- `/var/log/wtmp`

Lösung: `last`

- `/var/log/btmp`

Lösung: `lastb`

- `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

Lösung: `journalctl`

3. Welche Befehle würden Sie in Kombination mit `grep` verwenden, um die folgenden Informationen über Ihr Linux-System zu erhalten?

- Wann wurde das System zuletzt neu gestartet (`wtmp`)?

Lösung: `last`

- Welche Festplatten sind installiert (`kern.log`)?

Lösung: `less /var/log/kern.log`

- Wenn erfolgte die letzte Anmeldung (`auth.log`)?

Lösung: `less /var/log/auth.log`

4. Welche zwei Befehle würden Sie verwenden, um den Kernel Ring Buffer anzuzeigen?

`dmesg` und `journalctl -k` (oder auch `journalctl --dmesg`).

5. Geben Sie an, wo die folgenden Protokollmeldungen hingehören:

- `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

| | |
|-------------------|---|
| /var/log/auth.log | |
| /var/log/kern.log | |
| /var/log/syslog | X |
| /var/log/messages | |

- Jul 10 11:23:58 debian kernel: [1.923349] usbhid: USB HID core driver

| | |
|-------------------|---|
| /var/log/auth.log | |
| /var/log/kern.log | X |
| /var/log/syslog | |
| /var/log/messages | X |

Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)

| | |
|-------------------|---|
| /var/log/auth.log | X |
| /var/log/kern.log | |
| /var/log/syslog | |
| /var/log/messages | |

- Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting...

| | |
|-------------------|---|
| /var/log/auth.log | |
| /var/log/kern.log | |
| /var/log/syslog | |
| /var/log/messages | X |

6. Hat journalctl Informationen über die folgenden Einheiten?

| Einheit | Befehl |
|------------|----------------------------------|
| ssh | journalctl -u ssh.service |
| networking | journalctl -u networking.service |
| rsyslog | journalctl -u rsyslog.service |

| Einheit | Befehl |
|---------|----------------------------|
| cron | journalctl -u cron.service |

Lösungen zu den offenen Übungen

1. Betrachten Sie die Ausgabe von `top` in der geführten Übung und beantworten Sie die folgenden Fragen:

- Welche zwei Schritte würden folgen, um den *apache* Webserver zu töten?

Zuerst drücken Sie `k`, dann geben Sie einen `kill`-Wert ein.

- Wie könnten Sie im Übersichtsbereich die Informationen über den physischen Speicher und den Swap mit Fortschrittsbalken anzeigen?

Durch ein- oder zweimaliges Drücken von ``m`.

- Sortieren Sie nun die Prozesse nach Speicherverbrauch:

`M`

- Da nun Speicherinformationen in Fortschrittsbalken angezeigt werden und die Prozesse nach Speichernutzung sortiert sind, speichern Sie diese Konfiguration, so dass Sie sie beim nächsten Aufruf von `top` als Standard verwendet wird:

`W`

- Welche Datei speichert die Konfigurationseinstellungen von `top`? Wo liegt sie? Wie kann prüfen, dass es sie gibt?

Die Datei ist `~/.config/procps/toprc` und befindet sich im Heimatverzeichnis des Benutzers (`~`). Da es sich um eine versteckte Datei handelt (sie beginnt mit einem Punkt), können wir mit `ls -a` (Liste aller Dateien) auf ihre Existenz prüfen. Diese Datei kann durch Drücken von `Shift + W` innerhalb `top` erzeugt werden.

2. Machen Sie sich mit dem Befehl `exec` in Bash vertraut. Starten Sie eine Bash-Sitzung, finden Sie den Bash-Prozess mit `ps`, führen Sie anschließend `exec /bin/sh` aus und suchen Sie dann erneut nach dem Prozess mit derselben PID.

`exec` ersetzt einen Prozess durch einen anderen Befehl. Im folgenden Beispiel sehen wir, dass der Bash-Prozess durch `/bin/sh` ersetzt wird (anstatt `/bin/sh` zu einem Unterprozess zu werden):

```
$ echo $$
19877
$ ps auxf | grep 19877 | head -1
```

```
carol 19877 0.0 0.0 7448 3984 pts/25 Ss 21:17 0:00 \_ bash
$ exec /bin/sh
sh-5.0$ ps auxf | grep 19877 | head -1
carol 19877 0.0 0.0 7448 3896 pts/25 Ss 21:17 0:00 \_ /bin/sh
```

3. Folgen Sie diesen Schritten, um Kernel-Ereignisse und die dynamische Verwaltung von Geräten durch udev zu untersuchen:

- Schließen Sie ein USB-Laufwerk direkt an Ihren Computer an (Hotplug). Führen Sie `dmesg` aus und achten Sie auf die letzten Zeilen. Wie lautet ist die jüngste Zeile?

Sie sollten etwas wie `[1967.700468] sd 6:0:0:0: [sdb] Attached SCSI removable disk` erhalten.

- Führen Sie unter Berücksichtigung der Ausgabe des vorherigen Befehls `ls /dev/sd*` aus und stellen Sie sicher, dass Ihr USB-Stick in der Liste erscheint. Wie lautet die Ausgabe?

Abhängig von der Anzahl der an Ihr System angeschlossenen Geräte sollten Sie so etwas wie `/dev/sda /dev/sda /dev/sda1 /dev/sdb /dev/sdb1 /dev/sdb2` erhalten. In unserem Fall finden wir unseren USB-Stick (`/dev/sdb`) und seine beiden Partitionen (`/dev/sdb1` und `/dev/sdb2`).

- Entfernen Sie nun das USB-Laufwerk und führen Sie `dmesg` erneut aus. Wie lautet die letzte Zeile?

Sie sollten etwas wie `[2458.881695] usb 1-9: USB disconnect, device number 6` erhalten.

- Führen Sie `ls /dev/sd*` erneut aus und stellen Sie sicher, dass Ihr Gerät aus der Liste verschwunden ist. Was lautet die Ausgabe?

In unserem Fall: `/dev/sda /dev/sda1`.