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 `gunzip` 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		
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'`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/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`	
`/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 `journalalctl` benötigt.

Befehle, die in dieser Lektion verwendet wurden:

`cat`
: Dateiinhalt verketten/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.