Aus dem Nähkästchen geplaudert: Daemons

Die Geister, die ich rief

Die Liste laufender Prozesse eines scheinbar untätigen Unix-Systems ist für frisch gebackene Administratoren überraschend. Klar, dass jeder Dienst und jeder Prozess eine Aufgabe erfüllt. Diese Folge des Admin-Workshops erklärt, wofür die einzelnen Daemons zuständig sind. Marc André Selig

Der Workshop in Heft 9/04 [1] stellte Methoden vor, mit denen der interessierte Administrator die aktuell laufenden Dienste aufspürt. Damit weiß er aber noch nicht, wozu diese Dienste gut sind. Neben spannenden Entdeckungen in den Tiefen der Linux-Maschinerie beschert das Wissen über die wichtigen Daemons auch einen Sicherheitsvorteil: Die Kenntnis der Software, die auf einem typischen Linux-Server läuft, erlaubt das Erkennen fremder und potenziell gefährlicher Software.

Abbildung 1 zeigt ein Beispiel für die zahlreichen Prozesse, die Linux-Nutzer auf ihrem Rechner (hier eine nicht mehr ganz taufrische Suse-8.2-Installation) antreffen. Das Beispielsystem ist eine abgespeckte Workstation, bei der die meisten Netzwerkdienste deaktiviert sind – in einem voll ausgebauten Server spuken deutlich mehr Dämonen.

Die nach dem Booten anzutreffenden Prozesse gehören zwei Gruppen an: Unter Linux sind die so genannten Kernel-Threads von User-Prozessen zu unterscheiden. Erstere arbeiten im Kernelspace, im geheiligten Adressraum von Linux. Kernel-Threads entstehen in der Regel bei der Initialisierung von Modulen. User-Prozesse hingegen sind gewöhnliche Benutzerprozesse, die zum Beispiel durch den Aufruf eines Programms oder durch »fork()« entstehen.

Aller Anfang ist Init

Der »init«-Prozess (PID 1) ist die Mutter aller Prozesse. Wie in [2] beschrieben sorgt er dafür, dass ein Linux-System stets in einem angemessenen Betriebszustand arbeitet. Dazu startet und beendet Init andere Programme gemäß den Angaben in der Konfigurationsdatei »/etc/inittab«.

Kernel-Threads [4] sind an den eckigen Klammern um den Prozessnamen zu erkennen. Sie kümmern sich um essenzielle Arbeiten für den Kernel. Typische Beispiele für einen Linux-Kernel der 2.4er Reihe sind in Tabelle 1 zu sehen. Weitere Kernel-Threads sorgen für die Verwaltung von logischen Geräten (»mdrecoveryd« oder »raid*d«) oder spezieller Hardware (»ahd*« und »scsi*« oder »khubd«).

Kernel-Threads entstehen, wenn sich die entsprechende Stelle des Kernels initialisiert, in der Regel also recht früh während des Bootvorgangs. Ausnahmen sind Module, die erst wesentlich später nachgeladen werden, zum Beispiel als Reaktion auf ein neu eingestecktes USB- oder Cardbus-Gerät. Im Gegensatz dazu werden die User-Prozesse erst nach und nach über die von »init« gestartete Programmkaskade aufgerufen. Das kann einige Minuten dauern und führt fast ausnahmslos zu höheren Prozess-IDs.

Ab ins Netz

Computer in professionellen Netzwerken müssen als eine der ersten Funktionen ihren Netzzugang einrichten. Meist liegen die erforderlichen Konfigurationsdaten nicht auf dem Endgerät, sondern eine Zentrale verteilt die Parameter über einen Mechanismus wie DHCP (Dynamic Host Configuration Protocol). In diesem Fall ist einer der frühesten User-Prozesse »dhcpcd« (DHCP Client Daemon), er hat auf dem Testsystem die PID 615. Achtung: Das zweite c vor dem d unterscheidet den Client- vom Server-Daemon »dhcpd«.

Der DHCP-Client besorgt essenzielle Daten wie die eigene IP-Adresse, Informationen über die Netzmaske sowie das Gateway und trägt diese Daten in die Routingtabellen des Kernels ein. Der Client aktualisiert die Parameter auch, wenn der DHCP-Server eine Änderung

verlangt. Statt »dhcpcd« ist manchmal noch »pump« zu finden, die ältere Version eines DHCP-Clients mit fast identischen Aufgaben.

Privat genutzte Geräte, deren DSL- oder Kabel-Zugang ins Internet über einen separaten Router läuft, benötigen normalerweise ebenfalls einen »dhcpcd«. Wer ein manuell konfiguriertes LAN verwendet, sieht dagegen keinen eigenen DHCP-Prozess, da er die Konfigurationsdaten lokal speichert.

Modem- oder ISDN-Verbindungen nutzen heute fast immer das Point-to-Point-Protokoll mit einem Daemon namens »pppd«, den viele Anwender aus Kostengründen nur manuell starten. Direkte DSL-Anbindungen, bei denen der Linux-Computer das DSL-Modem über seine Ethernet-Karte ansteuert, verwenden entweder »dhcpcd« oder – in Deutschland häufiger – »pppd« in Verbindung mit »pppoe« (PPP over Ethernet).

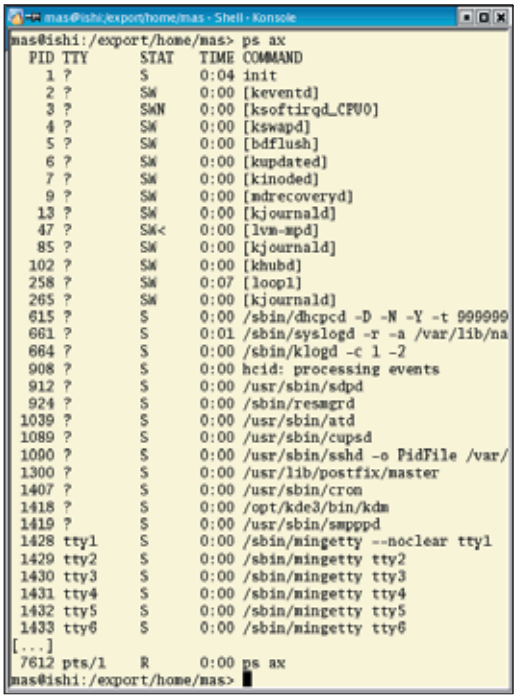

Abbildung 1: Das Kommando »ps ax« zeigt die laufenden Prozesse einer frisch gebooteten Suse 8.2 (hier gekürzt). Kernel-Threads sind an den eckigen Klammern erkennbar.

Protokoll führen

Gleich als nächstes Daemon-Paar folgen in Abbildung 1 die besten Freunde jedes Admin: »syslogd« und »klogd« [3] sorgen für die zentrale Verwaltung wichtiger Protokollmeldungen. Der »syslogd« nimmt die Syslog-Meldungen der User-Prozesse entgegen und schreibt sie gemäß der Konfiguration aus »/etc/syslogd.conf« (oder »/etc/syslog.conf«) in Dateien, auf den Bildschirm oder auf die Konsole. Der Kernel Log Daemon »klogd« verwandelt Kernelmeldungen in Syslog-taugliche Protokolleinträge. Weitere Daemons kümmert sich um lokale Geräte. Unix-Workstations stehen als Mehrbenutzer-Betriebssystem nicht nur einem User zur Verfügung, sondern gleich einer ganzen Reihe – beispielsweise im Computerpool einer Bibliothek, in einem CIP-Raum an der Universität (Computer-Investitionsprogramm) oder in einer Cyber-Cafeteria.

Verfügt ein Computer über eine Soundkarte, sollen die legitimen Nutzer des Rechners diese auch benutzen dürfen. In Unix bedeutet das: Die User benötigen Zugriffsrechte für die entsprechenden Gerätedateien in »/dev«. Aber Vorsicht: Hein Jansson hat auf den fiktiven PCs in

Tabelle 1: Kernel-Threads

Thread	Erläuterung
keventd	Hilft bei der Verwaltung von Kernel-Threads selbst.
kapmd, kacpid	Kümmern sich um die Interaktion mit dem Powermanagement des Computers (Advanced Power Management oder Advanced Configuration and Power Interface).
ksoftirqd	Sorgt für die Performance-freundliche Abarbeitung von Interrupts (Soft-IRQs). Möglicherweise existiert dieser Prozess unter verschiedenen Namen für die einzelnen Prozessoren.
pdflush, bdflush, kupdated, kjournald	Stellen sicher, dass auf Festplatte geschriebene Daten auch tatsächlich auf der Magnetschicht ankommen.
kswapd, kscand	Verlagern die zum jeweiligen Zeitpunkt nicht benötigten Daten aus dem Hauptspeicher auf die Festplatte.

der Cafeteria ebenso Zugriff wie sein Chef. Angenommen Hein sei an der Konsole angemeldet, sein Vorgesetzter hingegen über das Netzwerk. Wenn beide Zugriff auf die Soundkarte hätten, könnte der Chef seinen Mitarbeiter über das Mikrofon abhören. Solche Spionagetätigkeiten gilt es zu unterbinden.

Lokaler Zugriff

Zur Abhilfe verfügen moderne Linux-Distributionen über Mechanismen, die Zugriffsrechte für lokale Geräte nur an den jeweils an der Konsole angemeldeten Nutzer zuweisen. Solange jemand nur remote über das Netzwerk eingeloggt ist, darf er diese Devices nicht verwenden. So bleibt die Privatsphäre des Users an der Konsole ebenso gewahrt wie seine Zugriffsmöglichkeit auf die angeschlossenen Geräte. Als lokale Geräte gelten Soundkarte, Brenner, Scanner sowie einige USB-Geräte, meist auch das Bluetooth-PAN (Personal Area Network) sowie manchmal das Modem.

Der Resource Manager Daemon »resmgrd« (PID 924 in Abbildung 1) aktualisiert die Zugriffsrechte für Device-Files je nachdem, wer den Computer gerade benutzt. In »/etc/resmgr.conf« (siehe Listing 1) definiert der Admin Klassen von Geräten und bestimmt, unter welchen Voraussetzungen ein Benutzer auf sie zugreifen darf.

Bei der Interaktion mit Bluetooth-Geräten helfen »hcid« (Host Controller Interface Daemon) und »sdpd« (Service Discovery Protocol). Eine Suse-Erfindung ist der »smpppd« (Suse Meta PPP Daemon). Er gibt Konsolennutzern Zugriff auf Modem, ISDN und DSL, ohne den nativen PPP-Daemon »pppd« einsetzen zu müssen. Er vereinfacht auch die Nutzung an der grafischen Oberfläche.

Listing 1: Auszug aus »resmgr.conf«

```
01 # Desktop-User:
02 class desktop
03 # Audio:
04 add /dev/audio      desktop
05 add /dev/mixer      desktop
06 add /dev/dsp        desktop
07 add /dev/sequencer  desktop
08 # Modem:
09 add /dev/modem      desktop
10 [...]
11 # Lokale User haben Zugriff:
12 allow desktop       tty=/dev/tty[1-9]* || tty=tty[1-
   9]* || tty=:0
```

In eine andere Klasse von Daemons gehört »cron«, der den Aufräumjob zeitgesteuert startet. Er benutzt dazu systemweite Tabellen aus »/etc/cron*« und benutzerspezifische in »/var/spool/cron«. Eng verwandt ist der Batch-Daemon »atd«. Während »cron« für regelmäßig wiederkehrende Aufgaben zuständig ist, startet »atd« zu einem frei wählbaren Zeitpunkt einzelne Jobs.

Grundlegende Netzdienste

Für die Interaktion mit dem lokal oder über das Netzwerk angeschlossenen Drucker sorgen der Spooler-Daemon »cupsd« (Common Unix Printing System) oder der etwas betagte »lpd« (Line Printer Daemon). Beide nehmen Druckaufträge entgegen, wandeln sie gegebenenfalls in ein druckertaugliches Format um, fügen auf Wunsch eine Coverseite ein und leiten die Daten anschließend an den Drucker weiter.

Für Fernwartungsaufgaben ist der SSH-Daemon »sshd« zuständig. Mit ihm ist sicheres Einloggen aus der Ferne möglich. Als zentraler Bestandteil der Unix-Philosophie gehört auch ein funktionierender E-Mail-Server zu einem vollständigen System. Das bedeutet aber keinesfalls, dass jedes Unix-System von außen über das Mail-Protokoll SMTP (Simple Mail Transfer Protocol) erreichbar sein muss. Im Gegenteil: Wenn es keinen zwingenden Grund gibt, sollte jeder Admin den Dienst abschalten. Wichtig ist aber ein Programm, das lokal erzeugte Mails annimmt und weiterleitet, entweder an einen lokalen Benutzer oder an einen MTA (Mail Transfer Agent) im LAN oder im Internet. Je nach Vorlieben des Admin und der Distribution heißen die MTA-Daemons »sendmail«, »master« (Postfix) oder »qmail-*«.

Die bis hierher beschriebenen Daemons erlauben den reibungslosen Betrieb des Kernels und der Internetverbindung, kümmern sich um Protokolle und Wartungsarbeiten und warten gespannt darauf, dass sich ein Benutzer anmeldet, dem sie zu Diensten sein könnten. Fehlt nur noch die Interaktion mit diesem User: Die Konsole wird unter Linux von zwei Programmtypen bedient. Dank der virtuellen Konsolen können unterschiedliche Philosophien koexistieren.

Anhänger des Textmodus sowie alle User, die ihre Grafikkarte ausgewechselt haben ohne vorher die Konfiguration zu korrigieren, nehmen als Erstes Kontakt mit einem so genannten Getty auf (Get TTY). Dieser zeigt die Datei »/etc/issue« sowie einen Login-Prompt an. Sobald jemand als Reaktion darauf etwas eingibt, übergibt »getty« die Kontrolle an »/bin/login«, das nach Eingabe des korrekten Passworts eine Shell startet.

Traditionelle Gettys sind trickreiche Programme, die für die reibungslose Interaktion mit beliebigen Modems entwickelt wurden (die Abkürzung TTY steht für Teletype: Fernschreiber). Da die virtuelle Konsole von Linux weniger Ansprüche stellt, genügt heutzutage ein Ressourcen-schonender »agetty« (Alternative Linux Getty) oder »mingetty« (Minimal Getty for Consoles).

Weniger sparsam präsentiert sich der grafische Login. Hierfür zeichnet ein so genannter Display Manager zuständig – entweder der klassische X Display Manager »xdm« oder ein Pendant aus einer Desktop-Umgebung, etwa »gdm« von Gnome oder »kdm« von KDE.

Überblick behalten

Die wichtigsten Dienste seines Rechners kennen ist Voraussetzung für jeden Admin. Nur so kann er im Fehlerfall gezielt eingreifen. Wer sich erst mit den Prozessen beschäftigt, wenn sie die Systemlast unerwartet hochtreiben, hat es dagegen schwer. Etwas Neugierde erweist sich als nützlicher und hilfreicher Charakterzug guter Admins. *(fjl)*

Infos

[1] Marc André Selig, „Quasselpause – Netzwerkdienste finden und abschalten“: Linux-Magazin 09/04, S. 70

[2] Marc André Selig, „Startaufstellung – SysV-Init“: Linux-Magazin 06/04, S. 78

[3] Marc André Selig, „Computer-Logbuch – Syslog“: Linux-Magazin 02/04, S. 64

[4] Linux-Kernel 2.4, Internals: [http://www.faqs.org/docs/kernel_2_4/lki.html]

[5] Linux-Kernel, Analysis-Howto: [http://www.linux.org/docs/ldp/howto/KernelAnalysis-HOWTO.html]