FAQ
Basics zum Linux-Kernel, Teil 3

Antworten auf die häufigsten Fragen

Von Thorsten Leemhuis

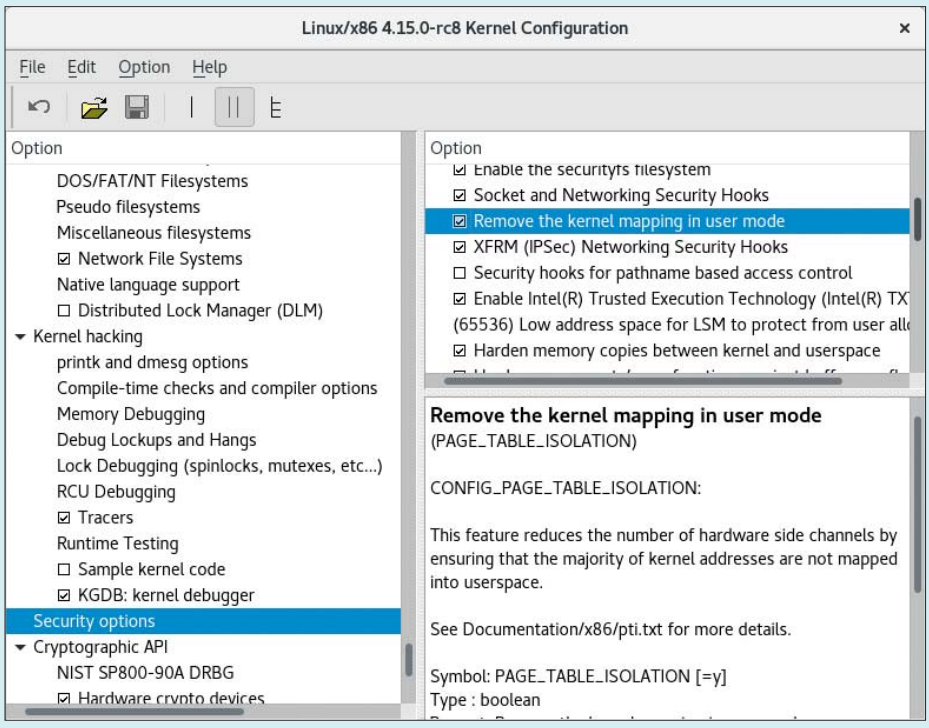
Aufwand für eigenen Kernel

? Wie aufwendig ist es, aus den Quellen von Linux ein eigenes Kernel-Image zu bauen?

! Das kommt auf Ihr Vorwissen und das Ziel an, das Sie verfolgen. Mit entsprechendem Know-how können Sie ein eigenes Image in ein paar Minuten konfigurieren, kompilieren und installieren. Selbst Experten brauchen aber Stunden oder Tage, um einen maßgeschneiderten Kernel zu erzeugen, der genau zu einem bestimmten System oder Einsatzzweck passt – kein Wunder, schließlich gibt es über achttausend Optionen, über die Sie die Fähigkeiten des Kernels festlegen müssen, bevor der Compiler anläuft.

Die langwierige Konfiguration samt einiger Fallen beim Kernel-Bau können Sie unter anderem mit `make localmodconfig` vermeiden. Dieses Make-Target sucht die Konfigurationsdatei des gerade laufenden Linux-Kernels, um sie als Ausgangsbasis für die Konfiguration Ihres Kernels einzurichten; anschließend deaktiviert es die Optionen für Module, die auf Ihrem System im jeweiligen Moment nicht geladen sind. So kommt man schnell zu einer recht gut zum jeweiligen System passenden Konfiguration, die dem Compiler viel Arbeit und Ihnen viel Zeit spart. Aber Vorsicht: Falls Sie beispielsweise bis zum Aufruf des Make-Targets keinerlei USB-Datenträger angesprochen haben, ist das dafür zuständige Modul noch nicht geladen. Daher wird es in der Konfigurationsdatei deaktiviert, sodass Ihr Kernel später keine USB-Datenträger ansprechen kann. Die gleiche Falle lauert auch an anderen Stellen, etwa bei Modulen für externe Peripherie, VPNs oder selten genutzte Dateisysteme.

Das ist nur eine der vielen Tücken, die bei Konfiguration und Bau eigener Kernel lauern. Der erfordert daher nicht nur viel Zeit, sondern auch einiges an Linux-Know-how. Wer sich da überschätzt, deaktiviert bei der Konfiguration womöglich unscheinbar wirkende Funktionen, die zum Booten oder für optimale Performance essenziell sind. Außerdem kann eine unachtsam gesetzte Option auch leicht Features lahmlegen, die für die Sicherheit Ihres Systems überaus wichtig sind.



Die händische Konfiguration eines eigenen Kernel-Images ist äußerst aufwendig und tückisch, daher vermeidet man sie besser.

Pflege eines Kernels

? Welcher Wartungsaufwand erwartet mich beim Einsatz eigener Kernel-Images?

! Liegt Ihnen die Sicherheit Ihres Systems am Herzen? Dann stellen Sie sich darauf ein, alle paar Tage einen neuen Kernel bauen zu müssen. Im dümmsten Fall halt auch vom Urlaub aus – und auch da kann es zügig nötig sein, falls just dann eine gravierende Sicherheitslücke bekannt wird, die schnell gestopft werden will. Solche als „kritisch“ eingestuften Lücken gibt es alle ein oder zwei Jahre. Ferner gibt es praktisch jeden Monat einige von hoher Wichtigkeit; welche von mittlerer oder geringer Bedeutung sogar alle paar Tage.

Letztlich erscheinen daher typischerweise fast jede Woche neue Versionen der gepflegten Stable- und Longterm-Kernel-Linien (siehe 2. Teil der FAQ). In manchen Wochen erscheinen auch mal zwei oder drei neue Ausgaben, falls kurz nacheinander mehrere Sicherheitslücken publik werden.

Nicht jede neue Version der Stable- oder Longterm-Linien beseitigt Sicherheitslücken; außerdem betrifft längst nicht jede Schwachstelle alle Systeme. Sie sollten trotzdem immer auf die jeweils neueste Version der eingesetzten Versionslinie wechseln, um auf der sicheren Seite zu sein: Sie bekommen nämlich nicht ohne Weiteres raus, ob oder welche Lücken ein neuer Stable- oder Longterm-Kernel beseitigt.

Ähnlich wie viele große Software-Schmieden korrigieren die Linux-Entwickler manche Fehler nämlich stillschweigend: Sie wollen Bösewichte nicht mit der Nase auf Schwachstellen stoßen, die Malware oder Angriffen dienlich sein könnten. Selbst bei öffentlich bereits diskutierten Schwachstellen gibt es typischerweise keinen expliziten Hinweis,

wenn ein neuer Stable- oder Longterm-Kernel diese beseitigt. Anwender würden sonst neue Versionen womöglich überspringen, denen ein Hinweis auf beseitigte Sicherheitslücken fehlt, obwohl welche stillschweigend behoben wurden. Daher ist auch der gelegentlich zu hörenden Tipp nicht zielführend, aus Security Advisories oder Fehlerdatenbanken von Linux-Distributioren abzuleiten, ob neue Stable- oder Longterm-Kernel irgendwelche Lücken stopfen; dieser Ansatz kostet ohnehin schnell mehr Zeit als der Bau einer aktualisierten Kernel-Version, denn für diesen Fall sollen Sie sich ohnehin rüsten.

Geschwindigkeitsgewinn durch eigenen Kernel

? Läuft mein System schneller, wenn ich mir einen eigenen, genau auf meine Anforderungen zugeschnittenen Linux-Kernel baue? Also einen, bei dem ich die Konfiguration genau auf meine Hardware und Anforderungen abstimme? Und zugleich auch den Compiler anweise, den Code für meinen Prozessor zu optimieren?

! Der Aufwand rechnet sich definitiv, wenn Sie mehr über die Funktionsweise von Linux lernen wollen. Stellen Sie sich aber darauf ein, dass das Ganze viel Zeit und Arbeit kostet. In anderen Fällen sollten Sie sich daher gut überlegen, ob sich der Aufwand lohnt.

Zwar kann ein genau für Ihr System konfigurierter und kompilierter Kernel manchmal etwas mehr Geschwindigkeit aus der Hardware kitzeln, als es universelle Linux-Kernel der Distributionen vermögen. Auf Desktop-PCs und Notebooks sind die Vorteile typischerweise aber meist so gering, dass Sie den mit einem eigenen Kernel einhergegangenen Aufwand nicht aufwiegen.

Es gibt aber durchaus Umgebungsbedingungen, wo ein eigener Kernel die Effizienz so stark steigert, dass sich die Investition lohnt. Bei Desktop-PCs und Notebooks kann das etwa der Fall sein, wenn der Kernel der Distribution gerade auf Ihrer Hardware ein Problem hat, das Sie mit einem neueren, modifizierten oder anders konfigurierten Kernel vermeiden können. Auch in großen Rechenzentren oder beim High Performance Computing (HPC) kann sich der Aufwand lohnen, wenn die Vorteile die Kosten wettmachen, die der Arbeitsaufwand auslöst.

So oder so: Bevor Sie sich an den Kernel machen, sollten Sie klären, ob Sie viel einfacher umsetzbare Optimierungen schon ausgeschöpft haben. Bei Mehrprozessorsystemen hat beispielsweise die in BIOS-Setup und Linux-Distribution festgelegte NUMA-Konfiguration einen großen Einfluss auf die Geschwindigkeit – Feintuning an diesen und anderen Stellen kann daher viel mehr bringen als ein maßgeschneiderter Kernel.

Performance-Optimierung mit Patch-Sammlungen

? Mir wurde in einem Forum empfohlen, meinen Kernel mit dem von Con Kolivas entwickelten Prozess-Scheduler zu patchen, weil der die Geschwindigkeit erheblich steigern soll. Stimmt das? Gibt es noch andere Patch-Sammlungen, mit denen ich die Performance verbessern kann?

! Die Kurzantwort ist: Es geistern ständig Sammlungen von Kernel-Patches durch die Welt, die versprechen, Systeme schneller zu machen – manchmal sogar deutlich. In einigen Fällen sind solche Änderungen den Aufwand und die Risiken wert; in einigen sind sie aber kontraproduktiv.

Der seit vielen Jahren von Kolivas entwickelte Prozess-Scheduler entlockt der Hardware bei bestimmten Umgebungsbedingungen tatsächlich etwas mehr Leistung. In anderen Fällen arbeitet er hingegen viel schlechter. Nach unserer Erfahrung ist der Unterschied am Ende nicht der Rede wert, sofern man nicht just ein System hat, wo der Scheduler des offiziellen Kernels aufgrund eines Fehlers oder einer Fehlkonfiguration etwas gehörig falsch macht. Das passiert aber nur selten.

Andere Patchsets können mehr bewirken; von 2014 bis Ende 2017 zirkulieren beispielsweise ALPM-Patches von Matthew Garrett, die die Leistungsaufnahme mancher Notebooks mit SATA-Datenträgern um ein oder zwei Watt reduzieren. Das kann die Akkulaufzeit spürbar verlängern. Aber: Die Entwickler des offiziellen Kernels hatten Gründe, warum sie diese Patches nicht sofort in Linux integrierten. Von Garretts Patches war beispielsweise bekannt, dass diese unter bestimmten Umgebungsbedingungen zu Systemabstürzen führen; teilweise sollen sie sogar Datenverfälschungen ausgelöst haben. Das passiert zwar nur selten – im dümmsten Fall fällt es

Patch-Sammlungen, die einen Performance-Zuwachs versprechen, sollte man mit einer gewissen Skepsis begegnen.

aber erst auf, nachdem monatelang Dateien beschädigt wurden, sodass dann auch Backups nichts mehr taugen.

Letztlich muss man daher immer im Einzellfall evaluieren, ob die Vorteile einer Patch-Sammlung die Risiken und den Aufwand wert sind, die mit ihr einhergehen. Für die ALPM-Patches ist das in Zukunft nicht mehr nötig, denn Linux 4.15 bringt eine überarbeitete Variante, die das Stromsparpotenzial freilegt.

Gefahr für Daten und Hardware?

? Kann ein selbst gebauter Kernel meine Daten zerstören? Oder gar die Hardware kaputt machen?

! Sofern Sie keine Entwicklerversionen einsetzen, ist das so unwahrscheinlich, dass Sie sich darum keine großen Gedanken zu machen brauchen. Aber: Vollständig auszuschließen ist Datenverlust nie, schließlich ist auch der Linux-Kernel nur ein Stück Software und hat Fehler. Manchmal halt auch welche, die zu Datenverlust führen. Und manche davon treten womöglich nur unter sehr bestimmten Umgebungsbedingungen hervor. Diese werden daher machmal bei Feldtests nicht gefunden und zeigen sich erst bei Ihnen.

Diese Gefahr ist einer der Gründe für die immer wieder gehörte Empfehlung, regelmäßige Backups anzulegen. Auch

Distributions-Kernel sind vor Datenverlust nicht gefeit, schließlich gibt es unendlich viel Hardware. Hinzu kommen noch gigantisch viele Möglichkeiten, wie man Linux konfigurieren und nutzen kann. Kein Testlabor dieser Welt kann all diese Konfigurationen testen.

Die Gefahr von Problemen steigt indes, je weiter sich Hardware und Konfiguration Ihres Linux-Systems von typischen Installationen unterscheidet. Das gilt insbesondere, wenn sich Kernel-seitig niemand um von Ihnen eingesetzte Techniken kümmert. Linux 4.14 hatte etwa einen Fehler, der beim Einsatz der SSD-Caching-Technik Bcache zu Datenverlust führte. Während der Entwicklung von 4.14 war niemandem dieses Problem aufgefallen, weil die Technik nicht sonderlich verbreitet ist und der Bcache-Code des Kernels zu der Zeit als verwaist galt. Bei 4.14.2 haben die Entwickler den Fehler korrigiert; kurz zuvor hatte sich auch wieder jemand bereit erklärt, den Bcache-Code zu pflegen.

Hardware-Schäden durch den Linux-Einsatz sind noch viel unwahrscheinlicher, wenn man nicht selbst Linux-Treiber programmiert oder in für Entwickler gedachte Untiefen des Kernels vordringt. 2003 hatte beispielsweise ein Fehler in einem Kernel der Distribution Mandrake einige CD-Laufwerke von LG zerstört, weil Letztere ein von Linux abgesetztes Kommando versehentlich als Aufforderung zum Schreiben einer neuen Firmware interpretiert haben. Seitdem gab es noch einige ähnlich gelagerte Fälle, die aber nie eine größere Zahl von Nutzern betroffen hat; die Chance auf so einen Bug zu stoßen ist letztlich um Längen kleiner als die Aussicht auf einen Sechser mit Zusatzzahl im Lotto.

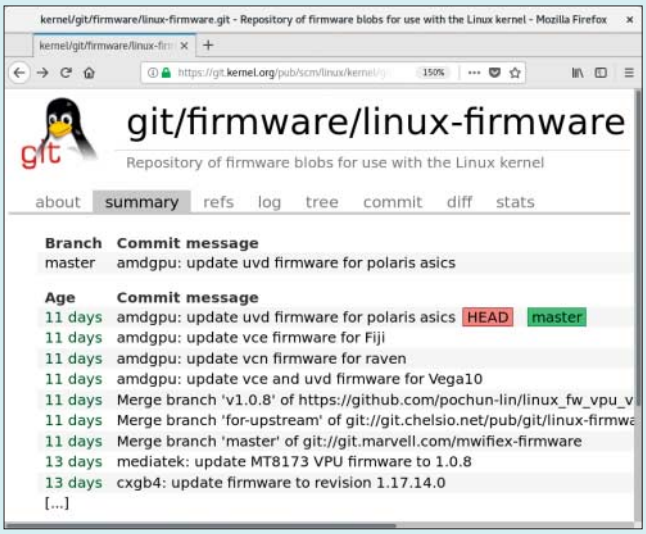
Einsatzbereite Kernel

? Eine neue Linux-Version bietet eine Funktion und einen Treiber, die ich dringend brauche. Komme ich an diese auch ran, ohne mir einen eigenen Kernel zu kompilieren?

! Entwickler oder Fans von Mainstream-Distributionen bieten meist Paket-Repositories an, die einsatzbereite Kernel der Mainline- oder Stable-Linien bereithalten. Bei Ubuntu gibt es solche etwa im „Kernel-PPA“. Das lässt sich allerdings nicht wie ein normales PPA (Personal Package Archives) einbinden, daher ist die Handhabung umständlich; hier helfen Skripte wie das Ubuntu Kernel Upgrade Utility (UKUU), die die Einrichtung von Kerneln aus dem PPA automatisieren.

Überlegen Sie sich aber noch gründlicher als bei anderen Paket-Repositories, ob Sie dem Anbieter der Pakete trauen können: Der Linux-Kernel ist schließlich das zentrale Element Ihrer Linux-Distribution, das auf alles Zugriff hat. Damit ist er natürlich ein idealer Kandidat für ein Trojanisches Pferd, das Ihnen nicht nur Schadcode oder Backdoor unterjubelt, sondern diese zugleich auch komplett vor Ihren Augen verbirgt.

Parallel zum Kernel muss man manchmal auch Firmware, VirtualBox oder Nvidias Grafiktreiber aktualisieren.



Abhängigkeiten zu Programmen

? Muss ich beim Wechsel auf einen neuen Linux-Kernel auch irgendwelche Userland-Software aktualisieren, damit eine moderne Kernel-Version alles Nötige findet?

! Normalerweise nicht: Die Kernel-Entwickler achten sehr stark auf Abwärtskompatibilität. Damit wollen sie erreichen, dass neue Kernel-Versionen auch auf älteren Distributionen laufen. Hin und wieder stehen aber dennoch Updates an.

Moderne Kernel harmonieren beispielsweise nur schlecht oder gar nicht mit dem Userland von zehn oder fünfzehn Jahre alten Distributionen: In der Zwischenzeit hat sich einfach zu viel verändert. Daran können Sicherheitslücken schuld sein, denn die lassen sich manchmal nicht in abwärtskompatibler Weise stopfen. Vielfach sind es auch einfach Unachtsamkeiten bei Umbauten am Kernel, die zu Inkompatibilitäten mit alter Userland-Software führen.

Mit einem lediglich wenige Jahre alten Userland oder einer halbwegs modernen Distribution hat man meist Glück. Häufigste Problemquelle hier: Den im Kernel enthaltenen Treibern sind die in /lib/firmware/ liegenden Firmware-Dateien zu alt. Die Treiber funktionieren dann oft gar nicht und weisen auf die Problematik in den Log-Meldungen hin, die `dmesg` ausgibt. Neuere Dateien liefert die bei Kernel.org gepflegte Firmware-Sammlung „linux-firmware“, die viele Distributionen über ein gleichnamiges Paket verteilen. Der einfachste Weg zu neuen Firmware-Dateien: Stibitzen Sie neue Firmware-Pakete aus der Entwickler-Version der eingesetzten Distribution.

Keine Abwärtskompatibilität gibt es indes bei den Schnittstellen, an denen Kernel-eigene Treiber andocken. Beim Einsatz von Software, die eigene Kernel-Module mitbringt, können daher letztlich doch größere Updates ins Haus stehen, selbst wenn Sie nur auf eine minimal neuere Kernel-Version wechseln. Das passiert etwa mit VirtualBox oder den proprietären Grafiktreibern von AMD und Nvidia häufiger. Wenn die zugehörigen Module mit dem neuen Kernel nicht harmonieren, muss man oft eine neue Version der Software einspielen. Manchmal erscheint die aber erst Wochen oder Monate nach einem Kernel, der Änderungen erforderlich macht. Im Internet finden sich daher oft Patches, die den Module-Quellcode anpassen, damit er zu neueren Kerneln passt.*(thl@ct.de)*

Basiswissen zum Linux-Kernel, Teil 1 & 2: ***ct.de/ysmf***