Basics zum Linux-Kernel, Teil 4

Linux wird ganz anders entwickelt als kommerzielle Software, denn niemand legt fest, was programmiert wird oder mit welcher Priorität. Über Aufnahme oder Ablehnung von Änderungen entscheidet zudem letztlich immer eine einzelne Person.

Von Thorsten Leemhuis

Entwicklungsprioritäten

Ich las jüngst in Ihrem Magazin, der Linux-Kernel habe eine Infrastruktur erhalten, um den Prozessor-Cache zu partitionieren. Da hab ich mich zum wiederholten Mal gefragt: Warum arbeiten die Linux-Entwickler ständig an so exotischen Dingen? Warum konzentrieren die sich nicht auf die vielen anderen und wichtigeren Bereiche, in denen es große Defizite gibt, etwa bei der Unterstützung für aktuelle Notebooks oder Treibern für Grafik- und WLAN-Hardware?

Die Kurzantwort: weil niemand eine zentrale Marschrichtung vorgibt. Kein Wunder, schließlich gibt es keine große Organisation mit Mitarbeitern, die die Entwicklung leitet. Vielmehr arbeiten freie oder irgendwo angestellte Programmierer einfach Verbesserungen für Linux aus, an denen diese oder ihre jeweiligen Arbeitgeber ein Interesse haben. Diese Änderungen werden meist integriert, sofern keine guten Gründe gegen eine Aufnahme sprechen.

Letztlich gibt es daher keine Roadmap oder jemanden, der Prioritäten festlegt; vielmehr ergeben sich diese von selbst aus der Motivation der Entwickler. Auch wenn das vielleicht etwas paradox klingt: Die Linux-Entwicklung funktioniert seit jeher so. Hintergründe zur im Detail viel facettenreicheren Herangehensweise liefern die folgenden Fragen und Antworten.

Wer entscheidet?

Wer legt fest, welche Verbesserungen in den offiziellen Linux-Kernel einfließen oder draußen bleiben?

Letztlich macht das immer noch Linus Torvalds, der 1991 die Entwicklung von Linux gestartet hat, das heute Kernel zahlloser Betriebssysteme ist. Torvalds schaut sich aber nicht jede als Quellcode-Patch eingereichte Änderung selbst an – kein Wunder, schließlich bringen die alle neun oder zehn Wochen freigegebenen Versionen der „Mainline“ genannten Hauptentwicklungslinie meist ungefähr 13.500 große und kleine Anpassungen.

Torvalds verlässt sich bei der Arbeit vielmehr stark auf einige Dutzende Mitstreiter, die festgelegte Bereiche des Quellcodes von Linux betreuen. Diese „Maintainer“ sammeln und begutachten die für das jeweilige „Subsystem“ eingesandten Änderungen. Beim Netzwerkcode macht das etwa David S. Miller. Ihm und einer Handvoll andere langjährigen Subsystem-Maintainern vertraut Torvalds nahezu blind; bei einigen Betreuern schaut er sich Änderungen hingegen näher an, die ihm die Maintainer meist per Git-Pull-Request als Patch-Sammlung schicken, damit er sie integriert.

Linux-Erfinder Linus Torvalds entscheidet nach wie vor, welche Änderungen in Linux einfließen oder nicht.

Wie in Führungshierarchien von Firmen herrscht dabei nicht immer Einigkeit. Daher geschieht es durchaus mal, dass Torvalds auch eine von Miller eingereichte Patch-Sammlung kritisiert oder gar zurückweist; dann kann es passieren, dass er die von Miller gesammelten Änderungen am Netzwerkcode als Druckmittel außen vor lässt und erst integriert, wenn Miller für Nachbesserung gesorgt hat. Der delegiert Letzteres womöglich an den Entwickler der Änderung, die Torvalds missfällt. Ohnehin ist das Netzwerk-Subsystem so groß, dass Miller bei der Pflege auf weitere Betreuer zurückgreift, die den Code für WLAN-Treiber, Bluetooth, VPN-Techniken, Firewalls und andere Unterbereiche wie ein eigenes Subsystem warten. So was ist aber eher die Ausnahme.

Entlohnung

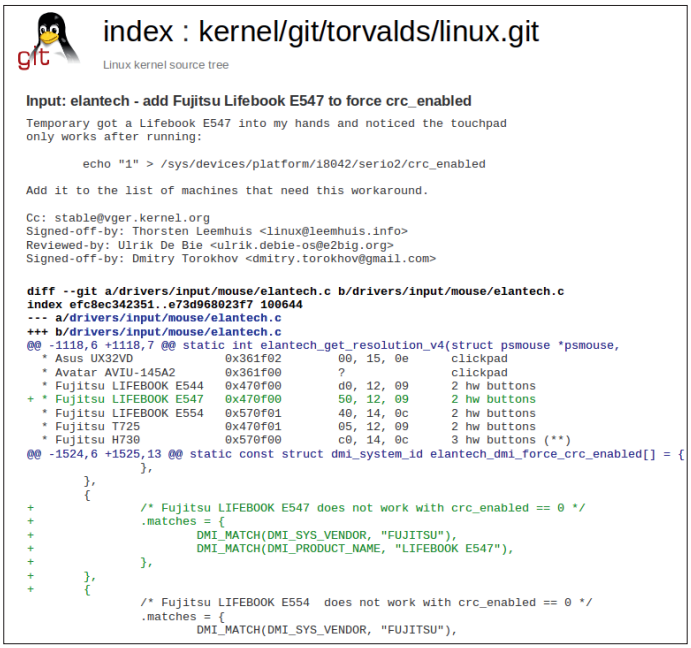
Linus Torvalds arbeitet doch nicht zum Spaß. Wer bezahlt ihn und warum werden nicht mehr Leute angeheuert?

Torvalds ist bei der Linux Foundation beschäftigt, die lediglich eine Handvoll Entwickler für die Arbeit an Linux bezahlen. Diese Leute gelten offiziell als „Fellow“ und können somit weitgehend frei entscheiden, woran sie arbeiten. Ähnlich wie andere Firmenzusammenschlüsse dieser Art vermeidet die Linux Foundation so, dass sich ihre Mitglieder über die Prioritäten der Mitarbeiter zanken können. Aus dem gleichen Grund hält die Organisation auch die Zahl der Mitarbeiter so niedrig.

Kann ich mithelfen?

Kann ich auch bei der Entwicklung des Kernels mithelfen?

Ja, suchen Sie sich einfach eine Stelle, wo es etwas zu verbessern gibt, und packen Sie an. Dazu muss man nicht mal unbedingt die von Linux verwendete Programmiersprache C können, wie eine Detailverbesserung zeigt, die wir bei im Rahmen von Linux-Tests auf Windows-Notebooks entwickelt und zum Kernel beigesteuert haben. Ein Artikel dazu beschreibt, dass das nicht sonderlich kompliziert war [1].



index : kernel/git/torvalds/linux.git

Linux kernel source tree

Input: elantech - add Fujitsu Lifebook E547 to force crc_enabled

```
Temporary got a Lifebook E547 into my hands and noticed the touchpad
only works after running:

        echo "1" > /sys/devices/platform/i8042/serio2/crc_enabled

Add it to the list of machines that need this workaround.

Cc: stable@vger.kernel.org
Signed-off-by: Thorsten Leemhuis <linux@leemhuis.info>
Reviewed-by: Ulrik De Bie <ulrik.debie-os@e2big.org>
Signed-off-by: Dmitry Torokhov <dmitry.torokhov@gmail.com>

diff --git a/drivers/input/mouse/elantech.c b/drivers/input/mouse/elantech.c
index efc8ec342351..e73d968023f7 100644
--- a/drivers/input/mouse/elantech.c
+++ b/drivers/input/mouse/elantech.c
@@ -1118,6 +1118,7 @@ static int elantech_get_resolution_v4(struct psmouse *psmouse,
   * Asus UX32VD             0x361f02        00, 15, 0e      clickpad
   * Avatar AVIU-145A2       0x361f00        ?               clickpad
   * Fujitsu LIFEBOOK E544   0x470f00        d0, 12, 09      2 hw buttons
+ * Fujitsu LIFEBOOK E547   0x470f00        50, 12, 09      2 hw buttons
   * Fujitsu LIFEBOOK E554   0x570f01        40, 14, 0c      2 hw buttons
   * Fujitsu T725            0x470f01        05, 12, 09      2 hw buttons
   * Fujitsu H730            0x570f00        c0, 14, 0c      3 hw buttons (**)
@@ -1524,6 +1525,13 @@ static const struct dmi_system_id elantech_dmi_force_crc_enabled[] = {
                },
        },
        {
+               /* Fujitsu LIFEBOOK E547 does not work with crc_enabled == 0 */
+               .matches = {
+                       DMI_MATCH(DMI_SYS_VENDOR, "FUJITSU"),
+                       DMI_MATCH(DMI_PRODUCT_NAME, "LIFEBOOK E547"),
+               },
+       },
+       {
                /* Fujitsu LIFEBOOK E554  does not work with crc_enabled == 0 */
                .matches = {
                        DMI_MATCH(DMI_SYS_VENDOR, "FUJITSU"),
```

Jeder kann Änderungen zu Linux beitragen, manchmal sogar ohne C- oder Programmierkenntnisse; diese Anpassung zur korrekten Handhabung eines Notebook-Touchpads ist etwa im Rahmen eines c't-Tests entstanden.

Wie wird entschieden?

Nach welchen Kriterien entscheiden Torvalds und die Subsystem-Betreuer darüber, welche Änderungen sie integrieren, ablehnen oder gar ignorieren?

Vereinfacht gesagt: Die Entwickler integrieren alle Patches, solange die Änderung den Qualitätsansprüchen genügt. Hinter dieser Kurzaussage stecken allerdings zahlreiche, größtenteils ungeschriebene Richtlinien. Und selbst bei den niedergeschrieben gibt es Unterschiede zwischen den verschiedenen Subsystemen.

Letztlich muss ein Patch vor allem dem jeweiligen Subsystem-Maintainer gefallen; dazu sollte er normalerweise keinen Widerspruch von anderen Entwicklern ernten, die bei Linux generell oder im jeweiligen Bereich angesehen sind. Außerdem muss der Code zumindest gut strukturiert sein und dem Programmierstil des jeweiligen Subsystems entsprechen, sodass andere dort agierende Entwickler ihn leicht verstehen – das ist zur Wartung und Weiterentwicklung wichtig, denn manchmal steuern Programmierer kleine und größere Änderungen bei, um direkt danach wieder komplett von der Bildfläche zu verschwinden. Ein Patch darf zudem nichts kaputt machen, was schon mal funktioniert hat: Das wäre eine Regression, und solche sind bei Linux strikt untersagt. Ohnehin darf sich die Änderung nicht negativ auf die Weiterentwicklung von Linux oder Nutzer auswirken, die sich für die damit vorgenommene Verbesserung nicht interessieren.

Das sind aber nur einige von vielen Kriterien, die Maintainer bei der Entscheidung anlegen.

Nicht sonderlich entscheidend ist indes, wie viel Nutzer die Verbesserung betrifft, die jemand zur Integration einreicht. Die Linux-Entwickler haben daher auch schon Treiber für Spezial-Hardware aufgenommen, die nur eine einzelne Firma hausintern an weniger als zehn Arbeitsplätzen einsetzt. Durch solche Treiber wächst zwar der Umfang des Quellcodes, aber da man Treiber beim Kompilieren eines Kernel-Images individuell ausknipsen kann, hat das keine negativen Auswirkungen auf andere Anwender. Das verkompliziert zwar Wartung und Pflege leicht, aber darin sehen die Linux-Entwickler bei Treibern meist kein Hindernis.

Treiber im Kernel?

Warum sind Treiber ein fester Bestandteil des Linux-Kernels? Das bei Windows verwendete Modell ist doch viel besser!

Das mag für Sie so wirken, bei näherem Hinsehen ist das aber längst nicht so eindeutig. Klar, bei Notebooks und Desktop-PCs ist Windows aus verschiedenen, teils historischen Gründen führend – in nahezu allen anderen Bereichen dominiert aber das jüngere Linux. Dass es Unmengen an Treibern gleich mitbringt, ist ein Faktor, der zu diesem Erfolg maßgeblich beigetragen hat. Es hat nämlich viele Vorteile, die Treiber unter eigener Kontrolle an einer Stelle zu haben. Der Ansatz erleichtert etwa Wartung und Weiterentwicklung von Treibern enorm. Ein Entwickler, der einen Fehler in einem Treiber entdeckt, kann ihn dadurch in Sekundenschnelle gleich bei anderen Treibern korrigieren. Die Herangehensweise erleichtert auch größere Umbauten enorm, wenn sich Anforderungen stark ändern, wie es etwa bei der Einführung neuer USB-Generationen mehrfach passiert ist. Die Linux-Entwickler können dann die Basisinfrastruktur für USB-Treiber und alle darauf aufbauenden Treiber in einem Handstreich anpassen, statt sich jahrelang mit einem festen Programmierinterface zwischen Kernel und USB-Treibern quälen zu müssen, auf dem sich neue Anforderungen nicht recht abbilden lassen.

Die Bündelung zwingt konkurrierende Firmen außerdem zu besserer Zusammenarbeit, durch die sie mehr an einem Strang ziehen. Die Begutachtung durch erfahrene Kernel-Entwickler ist zudem eine gute Qualitätssicherung. Durch sie eignen sich viele Treiber etwa von vornherein für unterschiedliche Prozessorplattformen; dadurch lief viel auf x86-PCs ausgelegte Hardware auch sofort auf dem Raspberry Pi, weil die meisten Kernel-Treiber keine Anpassungen für dessen ARM-basierten Prozessor brauchten. Das Linux-Modell hat zudem Knoppix, Desinfec't und dessen Vorläufer Knoppicillin möglich gemacht: Dank der Kernel-Treiber konnten diese und andere Live-Linuxe enorm viel Hardware von Haus aus unterstützen und liefen so auf nahezu allen PCs.

Neben diesen und weiteren Vorteilen hat der Linux-Ansatz auch einige Nach-

teile. Damit etwa ein neuer Grafikprozessor gleich bei der Markteinführung unterstützt wird, muss der jeweilige Hersteller den Support bereits zwei bis drei Monate vorher in den Linux-Kernel einbringen. Das ist kein Hexenwerk, wie AMD und Intel zeigen, denn deren Entwickler schaffen das in der Regel oder hängen nur ein bisschen hinterher. Außerdem hält nichts die beiden davon ab, ihre Treiber auch unabhängig anzubieten; das ist zwar sehr aufwendig, aber das liegt weniger am Kernel, sondern vor allem an den vielen, im Detail unterschiedlich arbeitenden Linux-Distributionen.

Oftmals liegt es nicht am Kernel und seinen Entwicklern, dass viele Mainstream-Distributionen neue Grafikchips erst Monate oder manchmal Jahre nach der Markteinführung unterstützen. Daran ist vielmehr die Pflegestrategie von Distributoren wie Debian, Ubuntu oder Linux Mint schuld: Sie liefern neuere Linux-Kernel meist nicht als Update nach, sondern integrieren sie erst viele Monate später in die nächste Version ihrer Distribution.

Ignorierte Treiber

In meinem Notebook sitzt ein WLAN-Chip von Realtek, für den das Unternehmen selbst einen Linux-Treiber anbietet, der super läuft. Warum nehmen die Kernel-Entwickler diese Treiber nicht einfach auf? Der Code steht unter der GPLv2, daher ist die Lizenz kein Hindernis. Eine ähnliche Situation gibt es beim Notebooks eines Freundes, wobei es dort ein Mediatek-Chip ist.

Die Kernel-Macher haben eine Reihe solcher Treiber aufgrund von Qualitätsmängeln außen vor gelassen. Oft lag das am Ansatz: Die Treiber bauten nicht auf der im Kernel enthaltenen Basis-Infrastruktur für WLAN-Treiber auf, sondern brachten stattdessen eigene WLAN-Stacks mit. Diese Treiber aufzunehmen hätte nicht nur den Kernel unnötig aufgebläht, sondern auch die Entwicklung und den Einsatz von Managementprogrammen wie dem NetworkManager verkompliziert, die sich auf Eigenarten und Fehler verschiedener WLAN-Stacks einstellen müssten. Darüber hinaus hatten die Treiber oft Schwachstellen und größere Macken – Sie hatten nur Glück, dass die auf Ihrem System nicht hervortraten.

Betriebssysteme wie Android wechseln immer mal auf neue Linux-Versionen von Kernel.org, um die vielen Verbesserungen aufzugreifen, die dort einfließen.

Das Ablehnen dieser Treiber hat deren Entwicklern und vor allem Anwendern das Leben kurzfristig schwer gemacht. Programmierer und Unternehmen haben dadurch aber mittelfristig kapiert, Treiber auf Basis der Basisinfrastruktur des Kernel schreiben zu müssen. Dadurch ziehen mittlerweile viel mehr an einem Strang und erfinden das Rad nicht immer wieder neu. Vielfach haben Firmen dadurch mit der Zeit sogar festgestellt, dass solch eine Zusammenarbeit auch für sie selbst viel einfacher ist und viele Vorteile bietet.

Ursprung aller Linux-Kernel

Sind die von Linus Torvalds und seinen Mitstreitern entwickelten Linux-Kernel überhaupt wichtig für mich? Android, FritzOS!, Ubuntu & Co. haben doch schon einen Kernel auf Linux-Basis – teilweise sogar welche, die der jeweilige Anbieter stark angepasst hat, um seine Bedürfnisse besser zu erfüllen.

Was die Entwickler um Torvalds treiben, ist auch für Sie enorm wichtig, denn früher oder später ist es im Eigeninteresse praktisch jedes Herstellers, wenigstens bei der Entwicklung neuer Produkte auf frischere Kernel zu setzen. Zwar ist der Linux-Kernel nur eine von vielen Betriebssystemkomponenten – aber halt die zentrale. Und die wandelt sich ständig, denn aus der Hauptentwicklungslinie gehen alle neun bis zehn Wochen neue Mainline-Kernel hervor, die einen Riesenschwung größerer Verbesserungen bringen.

Diese Neuerungen steigern oft Sicherheit und Performance; andere rüsten neue Mechanismen oder Protokolle nach, deren Unterstützung zum Bestehen in der sich ständig ändernden Welt manchmal schnell essenziell werden. Weil die meisten Treiber bei Linux im Kernel stecken, müssen Hersteller allein schon zur Unterstützung neuer Hardware auf eine neue Linux-Version der Hauptentwicklungslinie wechseln, denn das Zurückportieren von Treibern wird schnell immens aufwendig.

Das motiviert selbst Unternehmen zum Wechsel auf frische Kernel-Versionen, die lediglich ein Linux-basiertes Betriebssystem für Internet-fähige Waschmaschinen entwickeln.

Triebkraft der Entwicklung

Torvalds und die Subsystem-Maintainer sitzen doch an Schlüsselpositionen, da alle Änderungen durch ihre Hände gehen. Können sie diese Macht nicht irgendwie nutzen, damit jemand endlich mal Problembereiche angeht?

Das tun sie, aber mit Augenmaß. Gerade die Betreuer von Teilbereichen des Linux-Quellcodes motivieren Entwickler beim Einbringen eines Features immer wieder dazu, allgemeine Probleme im direkten Umfeld des jeweiligen Schauplatzes anzugehen. Die leitenden Entwickler des Grafiktreiber-Subsystems von Linux bewegen damit etwa AMD, Intel und andere Firmen dazu, für viele oder alle Grafiktreiber relevante Dinge einmal in der Basisinfrastruktur zu lösen, die der Kernel für Grafiktreiber mitbringt. Das schafft nicht nur einheitliche Konfigurations- und Programmierschnittstellen, sondern vermeidet auch, dass Entwickler die gleichen Probleme immer wieder auf Neue und als Inselösungen in ihren Treibern angehen.

Dieser Hebel der Subsystem-Betreuer wird ab einem gewissen Punkt aber zur Erpressung. Und die funktioniert bei Open-Source-Software nur begrenzt, schließlich könnten AMD, Intel, Red Hat, Suse und jeder andere jederzeit sagen: Das wird uns jetzt aber zu bunt mit Linus Torvalds und seinen Leuten, wir tun uns

zusammen und entwickeln den Linux-Kernel unter anderem Namen und unserer Führung weiter. Eine solche Zersplitterung hätte allerlei Nachteile, daher versucht Torvalds sie von jeher zu vermeiden – bislang äußert erfolgreich.

Beim Einreichen einer von Intel beigesteuerten Infrastruktur zur Prozessor-Cache-Partitionierung wird Torvalds daher nicht verlangen, dass das Unternehmen zuerst einen Treiber für irgendeinen Grafik- oder WLAN-Chip schreiben muss, den Linux bislang nicht oder nur schlecht unterstützt. Und erst recht würde er Entwickler nicht nötigen, Probleme in Bereichen zu korrigieren, die den Programmierer oder seinen Arbeitgeber überhaupt nicht jucken.

Welche Firmen tragen bei?

Wie viele verschiedenen Firmen tragen zum Linux-Kernel bei? Und wie hoch ist der Anteil von Entwicklern, die Kernel-Entwicklung als Hobby betreiben?

Die Linux Foundation veröffentlicht alle ein bis zwei Jahre eine „State of Linux Kernel Development“ genannte Analyse, die grobe Antworten zu dieser und ähnlich gelagerten Fragen liefert. Bei der letzten, 2017 veröffentlichten Analyse (siehe ct.de/yzpg) trugen zu jeder neuen Linux-Version meist Entwickler von zwei- bis dreihundert Firmen bei, viele davon regelmäßig. Weit über 80 Prozent der Änderungen stammen im analysierten Zeitraum von Entwicklern, die Unternehmen für die Arbeit im Linux-Bereich bezahlen.

Änderungen Linux 4.8 bis 4.13 – Top 12

Firmenzugehörigkeit	Commits	Anteil
Intel	10.833	13,1 %
Keine	6819	8,2 %
Red Hat	5965	7,2 %
Linaro	4636	5,6 %
Unbekannt	3408	4,1 %
IBM	3359	4,1 %
Consultants	2743	3,3 %
Samsung	2633	3,2 %
SUSE	2481	3,0 %
Google	2477	3,0 %
AMD	2215	2,7 %
Renesas Electronics	1680	2,0 %

Quelle: 2017 Linux Kernel Development Report, Linux Foundation

Von Firmen gesteuert?

Sie sagten, dass Unternehmen die meisten Änderungen zum Linux-Kernel beisteuern und Linus Torvalds keine Marschrichtung vorgibt. Bedeutet das also, das letztlich Firmen entscheiden, wo es mit Linux hingeht?

Jein: Die Sache ist komplizierter. Erst mal ist egal, wer eine Änderung entwickelt hat. Wer gute Arbeit macht und hilft, kann Einfluss durch seine Aktivitäten nehmen und sich so einen Ruf erarbeiten. Für ihre Arbeit am Kernel bezahlte Entwickler haben dazu allerdings meist mehr Zeit, daher sind sie in einer deutlich besseren Position mit mehr Einfluss. Durch richtig gute Arbeit können Entwickler auch zum Betreuer eines Subsystems aufsteigen und damit die erwähnten Schlüsselpositionen besetzen. Diese Aufgabe kostet aber einiges an Zeit und geht üblicherweise an Leute, die sich bereits bewiesen haben. Daher ist es auch kein Wunder, dass die meisten Subsystem-Betreuer von irgendwem für ihre Linux-Arbeit bezahlt werden.

Arbeitgeber können davon profitieren, Subsystem-Maintainer zu beschäftigen. Die Vorteile sind aber nicht so groß, wie es auf den ersten Blick scheint, schließlich findet die Entwicklung in der Öffentlichkeit statt. Ein beispielsweise bei Intel angestellter Subsystem-Maintainer kann nicht einfach alle von Intel-Mitarbeitern eingesandten Patches durchwinken und alle Änderungen von AMD-Leuten ablehnen: Er würde schnell kritisiert werden, worauf er Torvalds' Vertrauen schnell verlieren dürfte und den Posten bald los ist.

Entwickler wissen um diese und ähnlich gelagerte Aspekte, durch solche würden sie ihren Ruf auf Spiel setzen, der ist in Linux-Kreisen viel Wert ist. Wer bei welcher Firma arbeitet, ist bei der Kernel-Entwicklung dadurch viel weniger wichtig als etwa Code-Qualität, persönliche Sympathien und einiges andere. Ein Maintainer-Job hängt zudem an Personen, nicht an Firmen, daher wandert die Verantwortung mit, wenn ein Entwickler zum ärgsten Konkurrenten wechselt.

Motivation für Firmen

Was veranlasst Unternehmen dazu, Änderungen für Linux zu entwickeln und auch in den offiziellen Kernel einzubringen?

Die Firmen tragen aus Eigeninteresse zur Entwicklung bei. So hat Intel vor einer Weile eine Infrastruktur zur Partitionierung des Prozessor-Caches in seine Server-CPUs eingebaut. Da Linux im Server-Bereich dominierend ist, hat Intel auch gleich die zugehörige Linux-Unterstützung entwickelt. Dadurch greifen Linux-Distributionen sie automatisch oder mit Intels Hilfe auf, sodass Kunden das Feature leicht nutzen können. Im Idealfall schneiden Intel-CPUs dadurch bei Tests besser ab; sie verkaufen sich dann besser als Prozessoren, wo eine solche Funktion von Linux nicht unterstützt wird oder ganz fehlt.

Diese Vorteile hat nicht nur Intel, sondern auch AMD längst erkannt. Beide sorgen daher dafür, dass ihre Chips gut mit Linux laufen, vor allem die Bausteine für Bereiche, in denen Linux dominiert, darunter Server, Smartphones oder Tablets. Beide Unternehmen kümmern sich aber auch um Linux-Support für Komponenten, die auf Desktop-Systeme und Notebooks zielen. In solchen PCs stecken allerdings noch Bausteine anderer Unternehmen, die vielfach keine vergleichbare Motivation an den Tag legen. Kein Wunder, da die wenigsten solcher Systeme mit Linux ausgeliefert werden, ist der Druck durch Kunden und Systemhersteller nicht sonderlich groß – außer bei Chromebooks, denn da nötigt Google die Zulieferer zu gutem Support im Mainline-Kernel.

Bei Desktop-PCs und Notebooks hängt es daher stärker vom Engagement Freiwilliger oder der Distributions-Entwicklern ab, wie gut Linux auf solchen Systemen läuft. Extrem schwierig wird es beim Linux-Support zudem, wenn Unternehmen sich dem Linux-Modell verwehren und keine oder nur unzureichende Informationen zur Programmierung ihrer Hardware rausrücken. Das ist etwa bei der PC-Grafikchip-Abteilung von Nvidia der Fall. Sie baut sogar Sicherheitsfunktionen ein, die es quelloffenen Treibern schwer machen. Das ist etwa der Grund, warum Linux-Distributionen derzeit das volle Funktions- und Leitungspotenzial aktueller GeForce-Chips nicht mal ansatzweise auszuschöpfen vermögen. *(thl@ct.de)*

Literatur

[1] Thorsten Leemhuis, Kratz dein Jucken, Kompatibilitätsprobleme von Linux nachhaltig beseitigen, c't 9/2017, S. 102 oder https://heise.de/4466282

Basics zum Linux-Kernel, Teil 1, 2 & 3: ***ct.de/yzpg***