SelfLinux-0.13.1

Was ist Unix?

Autor: Florian Fredegar Haftmann (*florian.haftmann@stud.tu-muenchen.de*)
Formatierung: Alexander Fischer (*Selflinux@tbanus.org*)

Zwei Definitionsversuche:

> "Strictly, UNIX is a trademark administered by X/Open and refers to a computing operating system that conforms to the X/Open specification XPG4.2. This specification, also known as SPEC1170, defines the names of, interfaces to and behaviors of all UNIX operating system functions."

(aus *Beginning Linux Programming*, Wrox Press 1996).

> "UNIX ist das Betriebssystem der Zukunft und das schon seit über 30 Jahren!" - so ein ironischer Spruch eines *nicht-UNIXers*.

Inhaltsverzeichnis

1 Die Geburt von UNIX

Spaß beiseite. UNIX ist die Oberbezeichnung für eine Familie von Betriebssystemen, die alle sehr ähnlich aufgebaut sind und zum großen Teil sogar vom gleichen "Urahn" abstammen.

UNIX-Systeme laufen heutzutage auf jedem Rechner, vom Embedded System bis zum Mainframe, was insbesondere auch ein Verdienst von Linux ist.

Die Geschichte von UNIX liest sich stellenweise wie ein Heldenepos oder ein Kriminalroman, manchmal auch wie eine Groteske. Schon die Geburt von UNIX vollzog sich unter eigenartigen Umständen, denn es entsprang einem gescheiterten Projekt: Anfang 1969 gab es ein Gemeinschaftsprojekt des MIT, *General Electric* und den Bell Labs von AT&T, das Ideen für eine neue Generation von Betriebssystemen gesammelt hatte und daranging, diese Ideen unter dem Namen Multics umzusetzen. Da weder Zeitplan noch Budget eingehalten werden konnten, zog sich *Bell Labs* sehr schnell aus dem Projekt zurück. Ken Thompson und Dennis Ritchie, zwei Mitarbeiter von *Bell Labs*, die an *Multics* mitgearbeitet hatten, waren von den Einfällen und Erfahrungen, die sie mit *Multics* gesammelt hatten, so beeindruckt, dass sie kurzerhand eine abgespeckte Version des ursprünglichen *Multics* selbst schrieben und unter dem Namen ***Unics***, später ***UNIX***, in die Welt setzten.

Eine der Hauptmotivationen für UNIX war jedoch ein Spiel, das Thompson in seiner Freizeit geschrieben hatte. Um sein Spiel Space Travel auf eine PDP-7 zu portieren, benötigte Thompson u. a. ein Dateisystem, um das Spiel dauerhaft auf die PDP-7 zu speichern. (Zur Cross-Compilierung wurden noch Lochstreifen benutzt, die vom GECOS-System zur PDP getragen wurden.)

Übrigens:
Unics bzw. UNIX enthält ein zweifaches Wortspiel: Zum einen die Vorsilbe Uni- als Gegenstück zu Multi-, zum anderen wird UNIX genauso gesprochen wie *"Eunuchs"* - typischer Programmiererhumor dieser Zeit.

UNIX gewann sehr schnell eine große Verbreitung innerhalb der *Bell Labs*. *AT&T* war es aufgrund einer kartellgerichtlichen Entscheidung verwehrt, beliebig im kommerziellen Feld tätig zu werden, so auch im Falle von UNIX. Stattdessen lizenzierte *AT&T* UNIX gegen nominelle Gebühren an Universitäten, an denen UNIX seinen ersten Siegeszug antrat.

Eine besondere Rolle kam dabei der *University of California* in Berkeley zu, die einen eigenen Zweig des UNIX-Systems hervorgebracht hat, die *Berkeley Software Distribution*, kurz *BSD*. Sie war eng mit dem Quellcode von *AT&T* verwoben, zur Verwendung benötigte man also ebenfalls eine Lizenz von *AT&T*.

Auch das Interesse kommerzieller Anbieter an UNIX war geweckt worden: Sie erwarben Quellcode-Lizenzen von *AT&T* und brachten eigene, meistens auf besondere Hardware abgestimmte UNIX-Varianten heraus.

2 Eigenschaften von UNIX

Was ist nun das Besondere, das UNIX auszeichnet? UNIX hatte Eigenschaften, die heute zwar selbstverständlich geworden sind, aber damals noch recht neu waren:

* ein hierarchisches Dateisystem, d. h. die Möglichkeit, Dateien in Ordnern zu strukturieren - die ersten DOS-Versionen ab 1980 beherrschten dies noch nicht.
* Multitasking, d. h. mehrere Programme (oder Prozesse) können gleichzeitig laufen, ohne sich zu stören. Von Anfang an war dieses Multitasking preemptiv, d. h. die Steuerung und Zuteilung des Prozessors wird direkt vom Betriebssystemkern übernommen, ohne dass einzelne Programme fehlerhaft oder böswillig das ganze System in den Abgrund ziehen können - bei *Windows* und *Mac* war man erst in der zweiten Hälfte der 90er Jahre auf dem Weg dorthin.
* Multiuser-System, d. h. mehrere Benutzer können am gleichen System arbeiten und ihre Daten und Ressourcen beliebig für den Zugriff durch andere Benutzer sperren oder freigeben. Es können auch mehrere Benutzer am System gleichzeitig arbeiten, wenn die Hardware es zulässt (z. B. zusätzliche Konsolen oder Arbeiten über Netzwerkverbindung).
* Netzwerkfähigkeit, schon sehr früh wurden die UNIX-Kernel mit einem TCP/IP-Stack ausgestattet und bildeten so sehr schnell das Rückgrat des damals noch jungen Internets.

Darüber hinaus haben UNIX-Systeme charakteristische Eigenschaften:

* UNIX war unabhängig von irgendeiner bestimmten Hardwareplattform. Die meisten Betriebssysteme damals (und auch noch heute) waren auf einen bestimmten Prozessortyp zugeschnitten, und diese Abhängigkeit setzte sich fort in den Programmen, die für diese Systeme geschrieben wurden. UNIX abstrahiert so weit von der Hardware, dass es möglich wird, das System auf andere Plattformen zu portieren und dann UNIX-Programme ohne große Änderungen dort laufen zu lassen. Zwar gab es schon damals standardisierte Programmiersprachen, die auf mehreren Plattformen verfügbar waren, aber immer in leicht abgewandelter Form, so dass es praktisch unmöglich war, einfach portierbare Programme zu schreiben.
  Ein Grund für die Portabilität lag darin, dass UNIX nicht in *Assembler* sondern in *C* programmiert war (eine Hochsprache, die 1972 von *Thompson* und *Ritchie* entwickelt wurde). Zunächst war der UNIX-Code jedoch, obwohl er zum großen Teil in *C* geschrieben war, sehr stark an die Architektur der PDP gebunden. Erst mit UNIX Release 7 (1979) wurde der Code wirklich portabel.
* UNIX kommt von Hause aus mit einer Fülle an Entwicklungswerkzeugen und Bibliotheken: man muss nicht erst lange viel Software nachinstallieren, sondern findet das Wichtigste schon vor und kann davon ausgehen, dass auf jedem entsprechenden UNIX-System das gleiche vorhanden sein wird.
  Ein legendäres Programm, das schnell zum Grundbestandteil eines UNIXes wurde, war z. B. die Shell `sh`, die *Steve Bourne* schrieb. Der Linuxgemeinde ist sein Name in der allseits bekannten bash (*Bourne Again Shell*) überliefert.
* Zahlreiche Aufgaben lassen sich unter UNIX sehr einfach automatisieren.
* UNIX verfügt über elegante Programmierkonzepte, die dem ästhetischen Empfinden vieler Programmierer entgegenkommen.

Zur Schreibweise: Sowohl "UNIX" als auch "Unix" sind gültige Schreibweisen. UNIX (in dieser Schreibweise) ist darüber hinaus ein eingetragenes Warenzeichen der Open Group, was bei der Auswahl der Schreibweise ggf. eine Rolle spielen kann. Innerhalb von SelfLinux hat eine mögliche unterschiedliche Verwendung durch die Autoren keine Bedeutung.

3 Die Zersplitterung von UNIX und Standardisierungsbestrebungen

1984 trennte sich AT&T von etlichen Tochterfirmen, womit ihr auch gestattet wurde, sich als gewöhnlicher Wettbewerber auf dem Computermarkt zu betätigen. Damit wurden auch die Lizenzgebühren für UNIX drastisch angehoben und der Zugang zum Quellcode mehr und mehr eingeschränkt. Die Folge war, dass die Kooperation zwischen den Firmen, die UNIX kommerziell vermarkteten, immer mehr zurückging und jeder in "seiner" UNIX-Version seine eigenen Erweiterungen und Verbesserungen einbaute, bis UNIX heillos in unterschiedliche Versionen aufgesplittert war: *SunOS* von *SUN*, *HP-UX* von *Hewlett-Packard*, *AIX* von *IBM*, *Ultrix* von *Digital*, *SINIX* von *Siemens*, auch *Microsoft* versuchte sich auf dem UNIX-Markt mit *Xenix*. Ein großer Vorteil, die leichte Portierbarkeit der UNIX-Programme, drohte mit dieser Zersplitterung zu verschwinden und viele Stimmen prophezeiten auch ein mittelfristiges Ende von UNIX.

Als *AT&T* nach der Zerschlagung als Wettbewerber auftreten durfte, versuchte es auch, einen Standard zu schaffen: System V (wobei das V für die Zahl 5 steht und nicht für den Buchstaben, also "System Five" gesprochen).

1985 brachte *AT&T* das *"System V Interface Definition"* oder auch "lila Buch" heraus. Dieses Dokument stellte ein Standard für die UNIX-Schnittstellen dar. Zusätzlich enthielt es auch eine Menge Werkzeuge, die ein System auf die Konformität mit dem Standard V überprüfte. Diese von *AT&T* 1983 freigegebene Version *"UNIX System V"* war zu dieser Zeit die domierende Version, dies stellte den Versuch dar, die Hersteller auf einen Standard zu einen. Wegen des Widerstandes, der unter anderem dadurch entstand, dass man sich nicht von einer einzigen Firma abhängig machen wollte, entstanden im Laufe der Zeit andere Standards, so z. B. POSIX (Portable Operating System based on UNIX). Weil *AT&T* alle Rechte an dem Namen UNIX hatte, wurde vom IEEE (Institute of Electrical and Electronic Engineers) dieser Name für diesen Standard gewählt. Ein anderes Beispiel hierfür ist *X/Open*: Das X/Open Konsortium ist ein Zusammenschluss verschiedener Computerhersteller, die einen De-Facto-Standard schaffen wollten. 1988 wurde der X/Open Portability Guide veröffentlicht.

4 UNIX, GNU & Linux

All diesen Standardisierungsversuchen blieb der Durchbruch verwehrt. Erfolg hatte dagegen ein Projekt, gestartet Anfang der 80er-Jahre am *MIT* von Richard Matthew Stallman, dem "letzten Hacker der Altvorderenzeit", das GNU-Projekt: GNU's Not UNIX. Sein Ziel war es, von Grund auf ein neues, UNIX-ähnliches Betriebssystem zu schreiben, das frei verfügbar sein sollte. Durch seinen intensiven Einsatz und Beiträge anderer Programmierer entstand bis Ende der 90er eine beachtliche und leistungsstarke Sammlung an UNIX-Werkzeugen. Auch wenn das System bislang nicht vollständig ist, konnten sich die GNU-Werkzeuge dennoch auf vielen UNIX-Varianten etablieren, unter anderem auch deshalb, da einzelne UNIX-Anbieter ihre Einnahmequellen noch etwas auszubauen gedachten. Mit dem Grundpaket wurde z. B. kein C-Compiler mitgeliefert, worauf viele Systembetreuer, um Geld zu sparen, auf *Stallmans* GNU C-Compiler zurückgriffen, der ohnehin qualitativ besser war. So wurden die GNU-Werkzeuge ein systemübergreifender Quasi-Standard. Die freie Entwicklungsmethode hatte erreicht, woran die proprietären Standardisierungsversuche bislang gescheitert waren.

Bemerkenswert: Als *Stallman* den Entschluss fasste, GNU zu starten, hatte er noch nie mit UNIX gearbeitet, geschweige denn eine Zeile C-Code programmiert. Alles, was er wusste, waren ein paar grundlegende Konzepte und die Tatsache, dass UNIX seine Plattformunabhängigkeit bereits unter Beweis gestellt hatte.

Auch von akademischer Seite wurde der immer zugeknöpfteren Haltung der UNIX-Vertreiber begegnet: Zu Anfang wurde der Quellcode von *AT&T* den Universitäten offen zur Verfügung gestellt und so vielerorts als Tutorial für die Arbeitsweise eines Betriebssystems verwendet. Als *AT&T* den Quellcode unter Verschluss brachte, fiel diese Möglichkeit weg. Andrew S. Tanenbaum, Informatik-Professor an der Freien Universität Amsterdam, entschloss sich daher, für seine Studenten eine eigene Version von UNIX zu schreiben, die nichts mit dem urheberrechtlich geschützten Code von *AT&T* zu tun hatte. Nach zwei Jahren harter Arbeit brachte er sein System unter dem Namen Minix heraus. Es war weniger für die praktische Arbeit, sondern in erster Linie als Lehrobjekt gedacht. Dennoch wurde es von sehr vielen Studenten auch praktisch auf dem heimischen PC eingesetzt, da es im Gegensatz zu den kommerziellen UNIX-Varianten für einen moderaten Preis zu haben war. Allerdings stieß *Minix* in diesem Einsatzgebiet sehr schnell an seine Grenzen. Viele seiner Anwender machten *Tanenbaum* Vorschläge und schickten Patches für Erweiterungen und Verbesserungen. *Tanenbaum* allerdings war damit sehr zurückhaltend. Da er *Minix* in erster Linie als Tutorial sah, kam es ihm mehr auf eine knappe und klare Struktur als auf eine möglichst umfassende Funktionalität an.

Ein *Minix*-Anwender mit Namen *Linus Torvalds* gab sich damit nicht zufrieden. Das GNU-System war bis auf den Kernel vollständig, aber das Release des GNU-Kernels mit Namen *HURD* schien noch auf sich warten zu lassen. Um die zeitliche Lücke bis dahin zu füllen, begann er selbst einen Kernel zu schreiben, der sehr rasch unter dem Namen Linux Verbreitung fand und eine große Entwickler- und Benutzergemeinde zusammenbrachte. Da die meisten Entwickler auf UNIX-Varianten arbeiteten, auf denen die GNU-Werkzeuge liefen, lag es nahe, den Linux-Kernel so einzurichten, dass er zusammen mit den GNU-Werkzeugen verwendet werden konnte: GNU/Linux. Der Kernel HURD ist über "akademische" Anfänge bislang nicht hinausgekommen, so dass das anfänglich als "Provisorium" gedachte *Linux* sich an seiner Stelle etabliert hat.

Zur gleichen Zeit löste sich *BSD* aus seiner ursprünglichen Abhängigkeit von *AT&T*: Eine Gruppe von *BSD*-Entwicklern ersetzte alle Anweisungen im Quellcode, die noch von *AT&T* beigesteuert waren, durch neue und erstritt in einem langwierigen Gerichtsverfahren für *BSD* die Freiheit. Daraus gingen die Projekte FreeBSD, NetBSD und OpenBSD hervor, die auch eine beachtliche Verbreitung gefunden haben und manchmal als *Linux*-Vettern bezeichnet werden (und so manche Linux-Distribution enthält das ein oder andere "Schmankerl" aus einem der drei Projekte).

Seither hat sich Linux zu einem bedeutenden UNIX entwickelt: Kommerzielle UNIX-Anbieter haben Marktanteile an Linux verloren und mussten neue Strategien entwickeln. Nicht selten mündeten diese

Überlegungen in offener Unterstützung für Linux, dessen weitere Verbreitung ohnehin nicht mehr zu verhindern war.

UNIX (insbesondere seine freien Versionen) ist heute auf dem Servermarkt eine feste Größe. Ob ihm auch auf dem Desktop ein Durchbruch beschieden sein wird, ist eine der spannendsten Fragen der Gegenwart.