UNIX-System - Die Grundlage der heutigen Betriebssysteme

wissenschaftliche Arbeit

Studiengang: BSc Data Engineering
Author: Denis Andrei Szabo
Datum: 20. Mai 2025

1 Abstract

Inhaltsverzeichnis

2 Einleitung

Das Betriebssystem UNIX, welches in den 1970er Jahren von AT&T entwickelt wurde, wird als Fundament moderner Betriebssysteme angesehen. Linux und macOS werden bis heute von Prinzipien, wie Portabilität, Modularität und dem Konzept „Everything is a File“ geprägt [1], [2]. Windows, welches damals unabhängig von UNIX entwickelt wurde, beinhaltet heute mit dem „Windows Subsystem for Linux“ (WSL) Merkmale von UNIX [3]. Diese historischen Wurzeln sind nicht nur in der Informatik spürbar, sondern wirken bis in Unternehmen, Behörden und den Alltag vieler Nutzerinnen und Nutzer hinein, da zahlreiche digitale Geräte auf UNIX-Derivaten basieren [4], [5].

Der gegenwärtige Stand der Forschung macht deutlich, dass diese Gemeinsamkeiten und Unterschiede auf unterschiedlichen Ebenen existieren. Silberschatz [5] behandelt grundlegende Prozesse und Dateisysteme, während Raymond [1] den kulturellen Hintergrund der UNIX-Philosophie betont. McKusick [4] demonstriert die Weiterentwicklung dieser Konzepte in FreeBSD, während Tanenbaum [2] aufzeigt, dass auch andere Systeme wie Windows UNIX-ähnliche Funktionen integrieren.

Ausgehend von diesen Erkenntnissen lässt sich die zentrale Forschungsfrage folgendermassen formulieren:
In welchen Bereichen beeinflussen UNIX-Prinzipien moderne Betriebssysteme wie Linux, macOS und Windows und wo bestehen fundamentale Unterschiede?

Ziel dieser Arbeit ist es, einem sowohl technisch versierten als auch allgemein interessierten Publikum aufzuzeigen, welche Konzepte moderner Betriebssysteme direkt oder indirekt auf UNIX zurückgehen. Durch die Beantwortung der Forschungsfrage wird deutlich, in welchen Bereichen UNIX-Prinzipien bis heute relevant sind, wo sich im Lauf der Zeit abweichende Lösungen etabliert haben und welche praktischen Folgen sich daraus für Nutzerinnen, Nutzer und Systemarchitekturen ergeben. Die Untersuchung soll helfen, den Stellenwert dieser Prinzipien in aktuellen Systemen einzuordnen und ein besseres Verständnis für ihre Weiterentwicklung und heutige Bedeutung zu schaffen.

Die Methoden, die ich verwende, bestehen hauptsächlich aus einer Literaturrecherche zu UNIX und modernen Betriebssystemen. Ergänzend führe ich ein Interview mit einem Experten durch, um die theoretischen Erkenntnisse mit praktischen Erfahrungen zu verbinden.

3 UNIX und seine Relevanz für moderne Betriebssysteme

3.1 Kurzüberblick: Was ist UNIX?

Das Betriebssystem UNIX wurde Anfang der 1970er Jahren bei den Bell Laboratories von AT&T entwickelt. Ziel war es, ein portables, mehrbenutzerfreundliches und multitaskingfähiges Betriebssystem zu schaffen. Die daraus entstandene UNIX-Philosophie beinhaltet zentrale Prinzipien wie Modularität, Wiederverwendbarkeit, einfache Schnittstellen sowie das Paradigma „Everything is a File“. Diese Ideen prägten zahlreiche spätere Betriebssysteme. UNIX war eines der ersten Systeme, das vollständig in der Programmiersprache C geschrieben wurde, was die Portabilität auf verschiedene Hardwareplattformen stark vereinfachte [2].

UNIX wurde über die Jahre vielfach weiterentwickelt und in verschiedene Varianten verbreitet, darunter System V, BSD (Berkeley Software Distribution) sowie proprietäre und kommerzielle Derivate wie AIX, Solaris oder HP-UX. Aus dem BSD-Zweig entstanden auch moderne Open-Source-Betriebssysteme wie FreeBSD und NetBSD, dessen Aufbau und Weiterentwicklung im Detail von McKusick et al. beschrieben wird [4].

Zudem beeinflusste UNIX zahlreiche „UNIX-like“-Systeme [1] wie Linux, das heute eine zentrale Rolle in der IT-Infrastruktur weltweit einnimmt. Auch wenn Linux aus lizenzrechtlichen Gründen keinen originalen UNIX-Code enthält, basiert es klar auf den UNIX-Prinzipien und implementiert viele der ursprünglichen Konzepte [1].

Auch im Interview mit Herrn von Niederhäusern wurde die Relevanz von UNIX bestätigt: „UNIX ist mehr eine Philosophie als eine technische Implementierung [. . .] je länger, desto mehr natürlich relevant, weil die ganze Hyperscaler brauchen primär Linux [. . .] es hat sich bewährt“ [6].

3.2 Warum ist UNIX eine solide Grundlage?

Die Robustheit und Langlebigkeit der UNIX-Prinzipien ergeben sich aus ihrer strukturellen Klarheit und ihrer Fähigkeit, komplexe Systeme in einfache, verständliche Module zu zerlegen. UNIX setzt auf kleine, spezialisierte Programme, die über einfache Schnittstellen (meist über Standard-Eingaben und Standard-Ausgaben) miteinander kommunizieren. Diese Modularität fördert Wartbarkeit, Testbarkeit und Portabilität [1], [2].

Zudem ist UNIX skalierbar, von eingebetteten Systemen bis zu Grossrechnern, und bildet damit eine vielseitige Grundlage für unterschiedlichste Einsatzbereiche. Die POSIX-Norm (Portable Operating System Interface) standardisierte viele dieser Konzepte, was auch nicht-UNIX-Systemen wie Windows ermöglichte, kompatible Schnittstellen zu implementieren [5].

Silberschatz et al. heben besonders hervor, dass UNIX durch die klare Trennung von Kernel- und Benutzermodus sowie die konsistente Nutzung von Systemaufrufen eine stabile Plattform für viele Anwendungsbereiche bietet [5]. Raymond betont in seinem Werk zudem, dass die Kombination aus Einfachheit und Flexibilität der Tools eine nachhaltige Wirkung auf die Softwareentwicklung hatte [1].

Laut Herrn von Niederhäusern hat sich insbesondere die Modularität in modernen Infrastrukturen bewährt, etwa bei Docker, SeaGroups und Cloudsystemen: „Die Modularität ist ein sehr relevantes Prinzip, gerade im Hinblick auf Hyperscaler [. . .] das ist nur möglich, wenn es modular ist“ [6].

[1] UNIX-like-Systeme sind von der UNIX-Philosophie inspiriert, folgen jedoch nicht zwingend einer vollständigen UNIX-Implementierung. Sie sind daher in der Regel weder UNIX- noch POSIX-zertifiziert.

4 Methodisches Vorgehen

4.1 Literaturauswahl und Vergleichsansatz

Zur Beantwortung der Forschungsfrage wurde eine umfassende Literaturrecherche durchgeführt. Die Auswahlkriterien umfassten Aktualität, Relevanz für das Thema „UNIX und moderne Betriebssysteme“, sowie wissenschaftliche Fundierung. Herangezogen wurden sowohl klassische Standardwerke zur Betriebssystemtheorie als auch praxisorientierte Quellen, darunter:

- *„The Art of Unix Programming“* von E.S. Raymond [1]
- *„Modern Operating Systems“* von A.S. Tanenbaum und H. Bos [2]
- *„Operating System Concepts“* von A. Silberschatz, P.B. Galvin und G. Gagne [5]
- *„The Design and Implementation of the FreeBSD Operating System“* von M.K. McKusick, G.V. Neville-Neil und R.N.M. Watson [4]

Zusätzlich wurde die offizielle Dokumentation des Windows Subsystem for Linux (WSL) verwendet [3].

Die Bewertung erfolgte durch einen systematischen Vergleich von Linux, macOS und Windows entlang zentraler UNIX-Prinzipien. Der Vergleich berücksichtigt technische Merkmale (z.B. Kernel Architektur, Dateisysteme), aber auch nutzerorientierte Aspekte (z.B. Kommandozeilenverfügbarkeit, Konfigurierbarkeit, Tool-Kompatibilität).

Die Literaturen wurde mithilfe von Plattformen wie Google, der Bibliothek der Berner Fachhochschule sowie Internet Archive ausgesucht. Die Suchbegriffe umfassten unter anderem „UNIX“, „Operating Systems“, „BSD“ und „Linux“. Die Auswahl der Quellen erfolgte auf Basis von Erscheinungsjahr, Inhalt und Thematik, Popularität und Relevanz.

Ein zusätzlicher Erkenntnisgewinn wurde durch ein Interview mit Herrn von Niederhäusern erzielt, einem erfahrenen Fachmann für Betriebssysteme. Herr von Niederhäusern arbeitet seit ca. 1999 professionell mit Linux, insbesondere im Umfeld von IBM-Mainframes, Embedded-Systemen und Datenservern. Zudem nutzt Herr von Niederhäusern Linux bereit seit ca. 1997 auch privat und hat schon da Erfahrungen sammeln können. Seine langjährige praktische Erfahrung trug wesentlich zur Einschätzung der Relevanz einzelner UNIX-Prinzipien for moderne Betriebssysteme bei.

4.2 Kriterien zur Bewertung von UNIX-Anteilen

Zur vergleichenden Analyse wurden folgende Bewertungskriterien verwendet:

- **Modularität**: Grad der Zerlegbarkeit in unabhängige Komponenten.
- **Portabilität**: Möglichkeit der Nutzung auf verschiedenen Hardwareplattformen.
- **Kommandozeileninteraktion**: Umfang und Tiefe der CLI-Unterstützung.
- **POSIX-Kompatibilität**: Einhaltung standardisierter Schnittstellen.
- **Toolphilosophie**: Vorhandensein spezialisierter, kombinierbarer Tools.
- **Architekturprinzipien**: z.B. Monolithisch vs. Hybridkernel

Diese Kriterien wurden auf die drei betrachteten Betriebssysteme angewendet. Die Ergebnisse wurden mithilfe der Literatur und eines Experteninterviews ergänzt und validiert.

Das Interview mit Herrn von Niederhäusern, der über Berufserfahrung mit diesen Systemen verfügt, lieferte wichtige praxisnahe Perspektiven. Beispielsweise wurde darauf hingewiesen, dass Windows heute über das Windows Subsystem for Linux (WSL) bewusste UNIX-nahe Funktionalitäten integriert, um Entwicklern eine vertraute Umgebung zu bieten. Ebenso wurde betont, dass die ursprüngliche UNIX-Philosophie in heutigen Systemen teils verwässert sei, jedoch modularer Aufbau und Automatisierung zunehmend relevanter würden, gerade im Kontext von Container, Cloud und DevOps [6].

Das Interview wurde transkribiert und zentrale Aussagen wurden den Bewertungskriterien zugeordnet und ergänzen die Literaturbasis durch erfahrungsbasierte Einschätzungen.

5 UNIX als Fundament moderner Betriebssysteme

UNIX ist mehr als nur ein historisches Betriebssystem, es bildet die Grundlage vieler moderner Systeme und beeinflusst bis heute deren Architektur, Konzepte und Philosophie. In diesem Kapitel werden drei bedeutende Betriebssysteme betrachtetet, nämlich Linux, macOS und Windows, mit Fokus auf ihren Bezug zu UNIX. Dabei werden sowohl Gemeinsamkeiten als auch Unterschiede analysiert. Neben wissenschaftlicher Literatur fliessen Aussagen aus dem Interview mit Herrn von Niederhäusern ein, der als Systemadministrator und Dozent für das Modul „Computing Infrastructure“ tätig ist.

5.1 Linux: Ein UNIX-like System

Linux ist kein direktes Derivat von UNIX, sondern ein sogenanntes UNIX-like System. Es wurde Anfang der 1990er Jahren von Linus Torvalds als freies Betriebssystem entwickelt und orientiert sich stark an den Prinzipien und der Struktur klassischer UNIX-Systeme wie BSD und System V. [2]

Wie Tanenbaum & Bos beschrieben, übernimmt Linux zentrale UNIX-Konzepte wie das Dateisystem Hierarchiemodell, die Rechteverwaltung, die Interprozesskommunikation (IPC) und die Nutzung von Textdateien zu Systemkonfiguration [2]. Auch das Prinzip „Everything is a File“ gilt nahezu uneingeschränkt [1].

Herr von Niederhäusern betonte: „Ja, also Linux hat es sehr konsequent umgesetzt. Ausser einfach eben auf der Ebene Kernel wahrscheinlich. Dort gibt es unterschiedliche Ansätze, eben ein monolithischer Kernel, das ist technisch versus einen dynamischen Kernel, dort gibt es sicher Unterschiede“ [6].

Ein grosser Unterschied liegt in der Lizenzierung. Während klassische UNIX-Systeme proprietär waren, ist Linux unter der GNU General Public License (GPL) frei verfügbar und wird gemeinschaftlich weiterentwickelt. Dennoch wurde Linux so konzipiert, dass es weitgehend POSIX-kompatibel ist, ein UNIX-Standard zur Probabilität von Anwendungen [5].

5.2 macOS: Ein UNIX-System

macOS ist ein zertifiziertes UNIX-System (seit Version 10.5 „Leopard“), das auf dem Darwin-Kernel basiert. Darwin wiederum vereint Komponenten von BSD, insbesondere FreeBDS, mit dem Mach-Mikrokernel [4]. Dadurch ist macOS technisch gesehen näher an UNIX als Linux.

Laut Apple erfüllt macOS die Anforderungen der Single UNIX Specification (SUS), wodurch es sich offiziell als UNIX-System bezeichnen darf. Für Entwickler bedeutet das viele auf UNIX basierende Werkzeuge und Skripte nativ funktionieren, was macOS zu einer beliebten Plattform für Softwareentwicklung macht [5].

Herr von Niederhäusern erklärte dazu: „Also sagen wir man macOS hat so ein bisschen die Tendenz den User vor dem Computer zu schützen, also von dem her hat man eigentlich das recht gute Aqua GUI, also einfach die ganze Quarz Rendering Engine und ein User muss sich nie mit UNIX auseinandersetzten auf Mac, wenn er das nicht will. Von dem ist es dann sehr konsequent und sehr gut eigentlich gelöst, also am besten im Prinzip“ [6].

macOS integriert die UNIX-Basis in eine moderne, grafische Benutzeroberfläche. Dies zeigt, wie UNIX-Konzepte auch in einem kommerziellen Mainstream-System überleben und weiterentwickelt werden können [4].

5.3 Windows: UNIX-Elemente in einem nicht-UNIX-System

Microsoft Windows basiert historisch nicht auf UNIX, sondern entwickelte sich aus MS-DOS. Dennoch hat WWindows im Laufe der Zeit zahlreiche UNIX-ähnliche Elemente übernommen, primär im Server- und Entwicklerbereich [2].

Ein Beispiel ist das Windows Subsystem for Linux (WSL), das seit Windows 10 verfügbar ist. Es erlaubt die native Ausführung von Linux Distributionen unter Windows, inklusive Zugriff auf Bash, SSH, Git und andere UNIX-Tools [3]. Das macht Windows für Entwickler attraktiver, die bisher UNIX-Umgebungen bevorzugt haben.

Zudem implementiert Windows zunehmend POSIX-nahe Schnittstellen, allerdings nicht vollständig. Herr von Niederhäusern wies darauf hin: „Windows hat viele gute Tools aus der UNIX-Welt kopiert oder emuliert, aber es fühlt sich nie ganz gleich an“ [6].

Auch Systemdienste wie Prozess- und Speicherverwaltung folgen einem anderen Design. Dennoch zeigen Entwicklungen wie Windows Terminal oder native OpenSSH Unterstützung die zunehmende Annäherung an UNIX-Paradigmen, getrieben von den Anforderungen moderner Softwareentwicklung und Cloud-Infrastrukturen [1], [5].

6 Vergleich der UNIX-Anteile in modernen Betriebssysteme

Diese Kapitel untersucht die Gemeinsamkeiten und Unterschiede moderner Betriebssysteme in Bezug auf ihre UNIX-Bestandteile. Dabei liegt der Fokus auf der Systemarchitektur, der Entwicklungsmethodik, sowie der praktischen Relevanz für Entwicklerinnen und Entwickler. Grundlage für diese Analyse bilden Fachliteraturen und ein Interview mit Herrn von Niederhäusern [6].

6.1 Gemeinsamkeiten und Unterschiede

Obwohl Linux, macOS und Windows unterschiedliche Wurzen haben, lassen sich insbesondere zwischen Linux und macOS viele Gemeinsamkeiten feststellen, die auf ihr UNIX-Erbe zurückzuführen sind. Beide Systeme unterstützen POSIX-Standards, verfügen über eine Shell-Umgebung, bieten weitgehende Kompatibilität mit UNIX-Tools und setzen auf das Prinzip „Everything is a File“ [1], [2], [5].

macOS besitzt als zertifiziertes UNIX-System die engste technische Verbindung zur UNIX-Philosophie [4]. Es basiert auf BSD, einem klassischen UNIX-Derivate, und implementiert standardisierte UNIX-Schnittstellen vollständig. Linux hingegen ist nicht zertifiziert, erfüllt aber in der Praxis viele der gleichen Anforderungen und ist hochgradig kompatibel mit UNIX-Anwendungen [2], [5].

Windows stellt hingegen eine Sonderrolle dar, es ist weder UNIX-basiert noch vollständig POSIX-konform. Trotzdem hat Microsoft viele Elemente aus der UNIX-Welt übernommen, beispielsweise mit dem Windows Subsystem for Linux, das es ermöglicht, native Linux Binaries unter Windows auszuführen [3]. Auch Werkzeuge wie Git Bash, Windows Terminal und die PowerShell zeigen die Annäherung an UNIX-Konzepte. Dennoch bleibt das zugrunde liegende Systemdesign stark unterschiedlich, etwa in der Nutzung der Registry statt Konfigurationsdateien oder dem Fehlen eines nativen Paketmanagers [6].

Herr von Niederhäusern beschreibt die Situation folgendermassen: „Von dem her probiert jetzt Windows so ein bisschen die ganzen Philosophien miteinander zu verheiraten, aber dann sieht man, dass sich das Konzept so eigentlich bewährt hat oder, eben für moderne Betriebssysteme, mit einem Betriebssysteme auf dieser Ebene zu kommunizieren. [. . .] Das ist natürlich das Windows Subsystem schon sehr eine gute Alternative muss ich sagen. Vor allem jetzt die zweite version, bin ich auch recht ein Fan davon “ [6].

6.2 Einfluss auf Systemarchitekturen und Softwareentwicklung

Die UNIX-Philosophie hat die Art, wie Betriebssysteme konzipiert und Software entwickelt wird, massgeblich geprägt. Zentral sind Konzepte wie Modularität, textbasierte Schnittstellen, Skriptbarkeit und Wiederverwendbarkeit von Komponenten [1].

Bei Linux und macOS ist dieser Einfluss direkt sichtbar, beide Systeme setzten auf klare Trennung von Benutzer- und Kernel-Modus, auf einfache, kombinierbare Tools, sowie eine klare Prozesshierarchie [2], [4]. Die Verwendung von Shell-Skripten, cronjobs und systemd (bzw. launchd bei macOS) illustrieren diese Prinzipien.

Windows verfolgt traditionell einen stärker integrierten, GUI zentrierten Ansatz. Die Entwicklung der PowerShell und die Integration von WSL zeigen jedoch, dass auch hier zunehmend UNIX-artige Denkweisen einfliessen [3], [5]. Für die Softwareentwicklung bedeutet das eine grössere Vereinheitlichung über Plattformen hinweg. Entwickler nutzen heute plattformübergreifend Werkzeuge wie Git, Docker oder VSCode, alle stark beeinflusst von UNIX-Standards.

Auch Herr von Niederhäusern unterstreicht diesen Wandel: „Die Trennung der Welten ist vorbei. Ich sehe Entwickler, die unter Windows mit WSL und Docker arbeiten und andere die auf dem Mac native UNIX-Tools einsetzten. UNIX ist nicht tot, es ist überall, in verschiedenen Gewändern“ [6].

Zusammenfassend lässt sich sagen, dass UNIX-Anteile heute die Basis für viele essenzielle Konzepte in moderner Betriebssystemen bilden, ob direkt integriert wie bei macOS, nachgebildet wie bei Linux oder teilweise emuliert wie bei Windows.

7 Zukunftsperspektiven für UNIX in Betriebssystemen

7.1 Technologische Entwicklungen und Herausforderungen

UNIX und seine Konzepte bilden weiterhin eine solide Grundlage für moderne Betriebssysteme. Trotz des starken Wandels in der IT-Landschaft, etwa durch Cloud-Computing, Containerisierung und Microservices, bleiben UNIX-Prinzipien wie Prozessisolation, Rechteverwaltung und Dateisystemhierarchien zentral [1], [2].

Zukunftstechnologien fordern jedoch Anpassungen. So gewinnen virtualisierte Umgebungen und Containerplattformen wie Docker und Kubernetes zunehmend and Bedeutung, die zwar auf UNIX-ähnliche Systeme aufsetzten, aber neue Anforderungen an Sicherheit, Ressourcenkontrolle und Skalierbarkeit stellen [5].

7.2 UNIX im Zeitalter von Cloud und Edge Computing

Die Verlagerung von Rechenressourcen in die Cloud verändert das Betriebssystemverständnis. UNIX-basierte System dominieren nach wie vor Server- und Cloud-Infrastrukturen, etwa Linux Distributionen in Rechenzentren grosser Anbieter [2], [5].

Auch im Edge-Computing Bereich, wo Rechenleistung näher am Nutzer verteilt wird, bleibt UNIX mit seiner Stabilität und Modularität eine wichtige Basis. Hier sind vor allem leite UNIX-Derivate und angepasste Kernel gefragt, die auf ressourcen beschränkten Geräten effizient laufen [6].

Das Interview mit Herrn von Niederhäusern bestätigt: „UNIX wird nicht verschieden, sondern sich anpassen müssen. Besonders im Cloud-Bereich sieht man, wie Linux mit Containern und Microservices immer wichtiger wird. Auch macOS und andere UNIX-Systeme profitieren von dieser Entwicklung“ [6].

7.3 Herausforderungen und Weiterentwicklung

Die grössten Herausforderungen für UNIX besteht in der Anpassung an neue Paradigmen, ohne dabei die bewährte Stabilität und Einfachheit zu verlieren. Projekte wie systemd bei Linux zeigen, dass Innovation notwendig ist, aber auch Debatten über Komplexität und Rückwärtskompatibilität auslösen [1], [2].

Die Integration von Sicherheitsmechanismen wie SELinux oder AppArmor ist ein weiterer wichtiger Punkt, um UNIX-basierte Systeme gegen moderne Bedrohungen zu wappnen. Auch die Interoperabilität mit nicht-UNIX-Systemen, wie Windows, wird durch Projekte wie WSL verbessert, wodurch hybride Arbeitsumgebungen entstehen [3], [6].

7.4 Ausblick

UNIX wir sich weiterhin als Fundament und Inspirationsquelle für Betriebssysteme behaupten, gerade weil es flexibel und modular ist. Die Zukunft gehört Systemen, die klassische UNIX-Prinzipien mit neuen Technologien wie Containerisierung, KI gestützter Systemüberwachung und Cloud Integration verbinden.

Wie Herr von Niederhäusern im Interview zusammenfasst: „UNIX lebt in den Herzen der Entwickler weiter. Es wird niemals ganz verschwinden, sondern sich stetig wandeln und neue Formen annehmen, vom Server bis zum Smartphone“ [6].

8 Literaturverzeichnis

[1] E. S. Raymond, *The Art of Unix Programming*, 1. Aufl. Upper Saddle River, NJ: Addison-Wesley, 2003, ISBN: 978-0-13-142901-7.

[2] A. S. Tanenbaum und H. Bos, *Modern Operating Systems*, 5. Aufl. Hoboken, New Jersey: Pearson, 2024, ISBN: 978-1-292-45966-0.

[3] Anonym, *Windows Subsystem for Linux Documentation*. besucht am 4. Mai 2025. Adresse: `https://learn.microsoft.com/en-us/windows/wsl/`.

[4] M. K. McKusick, G. V. Neville-Neil und R. N. M. Watson, *The Design and Implementation of the FreeBSD Operating System*, 2. Aufl. Upper Saddle River, NJ: Addison-Wesley, 2015, ISBN: 978-0-321-96897-5.

[5] A. Silberschatz, P. B. Galvin und G. Gagne, *Operating System Concepts*, 10. Aufl. Hoboken, New Jersey: John Wiley & Sons, 2018, ISBN: 978-1-118-06333-0.

[6] P. A. von Niederhäusern, *Experteninterview über UNIX-System und moderne Betriebssysteme*, März 2025.

9 Selbständigkeitserklärung

Ich bestätige, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe. Sämtliche Textstellen, die nicht von mir stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ort, Datum: 4710 Balsthal, 20. Mai 2025

Unterschrift:

A Transkript des Interviews

Ich habe die Fragen ein bisschen aufgeteilt. Das 1. Kapitel wäre jetzt über UNIX an sich. Die erste Frage wäre „Was macht aus Ihrer Sicht UNIX heute noch zu einem relevanten Konzept, beziehungsweise ist es noch relevant und sinnvoll?“Also grundsätzlich mal ist UNIX ja mehr eine Philosophie als eine technische Implementierung und man hat dann natürlich davon die verschiedenen Instanzen, so wie eben Linux als Klon eines UNIX Systems oder die, die sich halt wirklich quasi auf den Urahnen die sich auf den Ur-UNIX beziehen, wie irgendwelche AIX von IBM ist das glaube ich gewesen. SCO UNIX wo es nochmal gegeben hat. Free BSD vor allem also die ganzen BSD das sind eigentlich die richtigen UNIX Betriebssysteme, oder weil die haben alle noch den ursprünglich Source Code, als es ist nicht klonen oder. Zur Relevanz allgemein vielleicht, je länger, desto mehr natürlich, weil die ganzen Hyperscalers brauchen primär Linux, vielleicht weniger jetzt FreeBSD, halt über die ganzen Containerisierung kommt das ganze Konzept und es hat sich bewährt. Natürlich eine sehr hohe Relevanz, die es vielleicht in der Zwischenzeit mal nicht so gehabt hat in diesen 90er Jahren ist das vielleicht nicht so populär gewesen, es ist mehr richtung Desktop Operating Systems gegangen, wie eben das klassische windows oder ein klassisches macOS und was halt auch ein Hinweis ist wieso ein Konzept an und für sich ohne UI, das je länger, desto mehr relevant wird, oder nach wie vor einfach extrem ein Gewicht hat, ein erfolgsversprechndes Konzept darstellt ist dass ja jetzt Windows seit ja, da bin ich jetzt nicht ganz sicher, vielleicht seit 10 Jahren, eben auch in einer Core Version gibt die ihr also nicht mehr mit UI Tools müsste auseinander setzen oder auseinander schlagen oder sondern wo einfach alles gescriptet werden kann. Also es ist jetzt so eine Art eine Alternativansatz zu Linux. Dass es Windows Core Server Systems gibt, wo man sich mit anderen Befehlen, ähnlich im Prinzip eben bedienen und vor allem im Hinblick darauf auf die Automatisierung und eben, wenn sie irgendeinen Hyperscaler seid oder dann können sie natürlich nicht irgendwie mit einem UI herum hantieren, das ist ja nicht möglich und die irgendwie probieren zu automatisieren das ist das würde dem Konzept völlig widersprechen und darum hat das ganze Konzept eben ein extrem hohes Gewicht, dass es jetzt sicher ein höheres Gewicht bekommen hat, also vor 10-20 Jahren. Und vielleicht auch noch zu macOS, das ist ja im Prinzip auch ein Hybrid zwischen einem richtigen UNIX und so einem Klon, hat sich halt auch dort durchgesetzt. Und alle IOS Geräte und auch alle Android Geräte die basieren eigentlich auf diesem UNIX Konzept, also von dem her, ja „It’s here to stay“, es ist absolut eine dominante Philosophie und man will das so und kann das im Prinzip auch nur so über die Schnittstelle, also über Kernel, Tools und auch entsprechend nur so bewerkstelligen. Eben ein reines GUI OS, in dem Sinne, die haben keine Relevanz mehr. Das ist etwa das, was wir dazu spontan in den Sinn käme. Zweite Frage „Die klassischen UNIX Prinzipien wie das Everything is a File, Modularität und so, was denken Sie ist heute bei der Systementwicklung noch relevant und wieso?“Also Everything is a File ist es sowieso nie gewesen, das ist nur so ein Versuch gewesen, dass man das irgendwie so darstellen kann, aber es gibt x Ausnahmen oder sobald ihr irgendwelche Sockets oder so habt, dann ist es nicht mehr einfach ein File also, dann wird es nachher komplexer, was da sicher relevant, also man kann beides jetzt von diesen 3 Stichworten also kann man sicher Modularität und Portabilität mal in den Vordergrund rücken, wobei ich die Modularität noch fast als relevanter anschauen würde. Weil aus

der Modularität gibt es ihnen dann auch jeglichen Vorteil in der Portabilität, weil wenn ihr nicht einfach einen Monolith habt, sondern dann könnt ihr euch das modular eben anpassen erhört dir natürlich auch die Chance, dass es eben portabel ist. Aber von dem her eben ist das sehr eben die Modularität, vor allem ein sehr relevantes Prinzip eben gerade herum in Hinblick darauf auf die ganzen Hyperscaler welche nur gewisse Teile vom Betriebssystem zum Beispiel exponieren, in irgendeinem Dockerumfeld, wo über Sea Groups, also das Ganze Rechte Management auf Kernel Ebene, wo man abstrahieren kann, das ist nur möglich, wenn es modular ist. Obschon man sagen muss jetzt wenn man es noch weiter runter brechen würde, der Linux Kernel in dem Sinne ist nicht ein klassischer modularer Kernel, oder wie man zu der Zeit sagte Minikernel, wo man nachher zur Laufzeit erweitert, sondern ist ein monolithischer Konglomerat, das aber so ein bisschen Ansätze hat, dass man auch zur Laufzeit eben gewissen Treiber nachladen kann. Es ist etwa so ein bisschen vergleichbar, obschon Windows wäre an und für sich ein dynamischer Kernel, aber vom Verhalten her nicht. Also ist es etwa ähnlich vom Konzept her und es hat sich bewährt und eben gerade vor allem im Kontext eben wiederum von diesen hochskalierten Cluster Lösungen oder die es braucht. Und was auch vielleicht relevant ist, im ganzen Scientific Computing also die ganzen super Rechner, also Number crunches die profitieren natürlich eben von genau dieser Modularität oder dass man das sehr gut, einfach eben miteinander verknüpfen kann. Wenn man das eben kontrastiert mit den Main Frames von IBM, da hat man einfach wirklich einen Block, ein Rechner aber der ist so extremst performant, aber einfach gemacht für einen spezifischen Workload. Viele kleine Aufträge, aber jetzt nicht unbedingt eben High Performance und wenn man das UNIX Prinzip dort drauf verwendet, dann hat man vielleicht ein Problem, dann würde die Modularität dann nachher eher zu einem Hindernis werden. Aber sonst grundsätzlich hat sich das Konzept durchgesetzt. Zu diesen Philosophien, die man gemacht hat von UNIX, hat sich im Laufe der Zeit, also im Hinblick von früher zu heute etwas daran geändert oder weiterentwickelt? Geändert ja, leider muss ich fast sagen. Also von Anfang an hat sich halt, und das hat sich jetzt mit Linux natürlich fast verstärkt, die Aufsplittung gegeben. Die erste Aufsplittung hat es eigentlich gegeben, wo wir System Five und BSD getrennt haben aus lizentechnischen Gründen, aus politischen Gründen, zum Teil auch, was ich im Moment eben nicht so schätze, dass es einfach extremst viele Linux Distributionen gibt und jede funktioniert anders. Es hat andere Tools, weil da ist sich jetzt im Originalunixumfeld mit BSD, oder mit FreeBSD ist man sich der Philosophie eher noch treu geblieben. Also dort hat man die Ursprungsphilosophie eher noch und einen Trend, den ich auch nicht schätze ist, dass man jetzt alles natürlich was eigentlich ein bisschen ein Widerspruch ist zu dem, was ich einleitend gesagt haben, aber alles irgendwie hinter UI verstekcen, also irgendwie GNOME Desktop, KDE Desktop, X Desktop und gar nicht mehr wirklich einfach mit diesen Text Files arbeiten kann, mit diesen Tools arbeiten kann, sondern es ist jetzt fast eine Art ein gegenläufiger Trend eben. Wieder zurück zu den grafischen Tool, aber das ist dem Umstand geschuldet, dass man einfach dem User entgegen kommen will, am End User der vielleicht nicht gerade der UNIX Crack ist und eine Alternative bieten will zu Windows. Ein klassisches Extrembeispiel ist natürlich macOS. Und auch dort habt ihr einen Zoo an verschiedenen Tools, mal sind die auf der User Ebene als Kommando Befehl, nachher gibt es irgendwie ein grafisches Tool, die widersprechen sich oder sind nicht mehr aktuell. Das ist so ein wenig ein Trend, das hat sich

verändert eher, jetzt in dem Kontext, zum negativen. Da sehe ich eine gewisse Gefahr. Aber es führt mich gleich zur nächste Frage, es ist nach wie vor eine solide Grundlage, also rein das Konzept. Und das zeigt sich auch sonst hätte isch ja Linux nie so durchgesetzt, sonst hätte man das irgendwann mal als Hobbyprojekt abgeschlossen, ist gut oder der Linus Torvalds hat da 2-3 Jahre Spass daran gehabt und dann hätte man jetzt irgendwie andere Konzepte, aber sonst als Betriebssystem als moderne Basis bietet das nach wie vor eine gute Grundlage. Also denken Sie wenn an jetzt selber ein Betriebssystem machen würde, dass man gut noch UNIX als Grundlage gebrauchen kann? Sicher also entweder man macht eben einen Klon, oder vom Konzept her so wie das Linux ist in Bezug auf UNIX oder man übernimmt gewissen Konzepte, Kernkonzepte ins Betriebssystem, welches von UNIX kommen. Also zum Beispiel Haiku OS oder BI-OS, wo es früher einmal war. Haiku OS das ist auch ein Klon, das sit ein Single User Grafikbasiertes Betriebssystem, hat aber darunter natürlich eine Schnittstelle, welche Posix Compliant ist oder welche von den Befehlen her eben UNIX-OID ist ohne, dass es selber wirklich UNIX ist, auch vom Kernel her nicht. Also eben lassen sich andere natürlich auch sehr gerne inspirieren und was man jetzt auch bisschen gemerkt hat, dass man im Prinzip die Befehle, wo halt mitkommen in einem BSD oder einem UNIX, also Command Line Tools, dass man die auf jedem Betriebssystem, das nicht unbedingt UNIX ist, anbietet. Also viele Befehle, die ähnlich sind unter UNIX gibt es in dieser Form auch unter Windows oder vice versa. Und eben auf die Spitze getrieben worden ist es mit dem ganzen Windows Subsystem for Linux, wo man ja wirklich das emuliert, was auch relative performant ist, ohne dass man eine virtuelle Maschine braucht. Von dem her probiert jetzt Windows so ein bisschen die ganzen Philosophien miteinander zu verheiraten, aber dann sieht man, dass sich das Konzept so eigentlich bewährt hat oder, eben für moderne Betriebssysteme, mit einem Betriebssysteme auf dieser Ebene zu kommunizieren. Jetzt zu Linux selber, wo UNIX-like ist. Wo gibt es dort klare Gemeinsamkeiten und Unterschiede. Und wie konsequent hat es die UNIX Prinzipien umgesetzt? Ja, also Linux hat es sehr konsequent umgesetzt. Ausser einfach eben auf der Ebene Kernel wahrscheinlich. Dort gibt es unterschiedliche Ansätze, eben ein monolithischer Kernel, das ist technisch versus einen dynamischen Kernel, dort gibt es sicher Unterschiede. Vieles ist natürlich nachimplementiert, also dort unterscheidet sich Linux auf der technischen Ebene von diesen Tools, das ist nichts vom ursprünglichen Sourcecode drin, aus lizentechnsichen Gründen nicht. Das ist vielleicht auch eine Eigenart oder eine fundamentale Unterscheidung von Linux zu UNIX, sie dürfen also nicht die klassischen Source brauchen und wenn man das klassisch lizenziertes UNIX nimmt, dann sind sie in der Regel lizenziert. Aber man findet die klassischen UNIX Betriebssysteme fast gar nicht mehr, also entweder ist es ein FreeBSD, da kann man damit machen, was man will, also man darf das auch kommerzialisieren und bei Linux, dann muss man sich unter Umständen einfach die verwendeten Lizenz im Klaren sein. Darf man das jetzt abschotten was man da gemacht hat als Linux Klon, als Betriebssystem Also da gibt es Lizenzrechtliche Unterschiede. Entweder das UNIX FreeBSD Paradime, wo alles offen ist, „mach damit was du willst“oder komplett „nein du musst es lizenzieren auf der Hardware“für so und so viele Ressourcen, Speicher, RAM, Storage wo wahrscheinlich jetzt bei einem IBM UNIX, also AIX der Fall wäre mehr gibt es fast gar nicht mehr. Und bei Linux, wo man aufpassen muss, wie man es angibt, weil wenn man jetzt irgendwie für eine Root oder für ein embedded Device Linux verwendet und man

massgebende Änderungen macht, müsste man die Änderung im Prinzip nachher auch public machen. Dort sind so die verschiedensten Finessen und Unterschiede, wo es zu einem reinen UNIX eben jetzt hat wenn man Linux anschaut. Aber sonst so von den Oberflächen Prinzipien her setzt Linux die Prinzipien eigentlich sehr gut um. macOS ist ja ein UNIX basierendes System. Genau. Inwiefern sin die Systemarchitektur und der Alltag von den Nutzern von den UNIX Prinzipien beeinflusst? Also sagen wir man macOS hat so ein bisschen die Tendenz den User vor dem Computer zu schützen, also von dem her hat man eigentlich das recht gute Aqua GUI, also einfach die ganze Quarz Rendering Engine und ein User muss sich nie mit UNIX auseinandersetzten auf Mac, wenn er das nicht will. Von dem ist es dann sehr konsequent und sehr gut eigentlich gelöst, also am besten im Prinzip. Aber wen man ein Power User ist hat man natürlich eben diese Tools zur Verfügung, allerdings muss man aufpassen, es gibt Unterschiede in der Bedienung zum Teil von den BSD Tools, also von dem richtigen UNIX Tools, wo es unter Mac gibt oder unter den FreeBSD's dieser Welt oder eben dann den Linux Entsprechungen. Die tun sich nicht immer genau gleich verhalten, also zu Teil gibt es andere Optionen, zum Teil sind die Befehle anders, da muss man natürlich immer ein bisschen aufpassen, wenn man häufig hin und her switcht, dass man da sich nicht irgendwie täuschen lässt. Aber sonst für einen normalen Benutzer, für eine normale Benutzerin die nicht mit dem Terminal arbeiten will, dann beeinflussen die UNIX Prinzipien im Prinzip da nicht gross. Wenn man sich technisch damit beschäftigt dann hilft es und das ist ein grosser Pluspunkt aus meiner Sicht eben mit einem Mac Betriebssystem zu arbeiten. Gibt es auch gewisse Sachen, wo Apple von UNIX Prinzipien herausgenommen hat? Ja leider. Viele Einstellungen kann man nicht konsequent über die Kommandozeile lösen, da muss man also Tools haben. Und zum Teil gibt es auch irgendwelche Kommandos, die man so nur unter macOS kennt, die gibt es auf keine Art und Weise irgendwo in einem BSD, also überhaupt nicht standardisiert und zum Teil kann man mit verschiedensten Kommandos, von High Level bis Low Level, wo sich aber komplett anders benennen, anders angewendet werden, das Gleiche machen. Also Orthogonalität, die saubere Trennung ist nicht da, man hat immer irgendwie so gewisse Vektoren, wie man es sich vorstellen kann, die etwa in die ähnliche Richtung zeigen, aber nachher trotzdem nicht das Gleiche genau erreichen und das ist manchmal ein bisschen in einem Konflikt und das finde ich schade. Aber eben diese Tendenz, die merkt man auch bei Linux Distributionen, also zum Beispiel bei irgendwie Red Hat oder Ubuntu, da findet man zu Teil Kommandos, die gibt es genau nur auf diesen Plattformen und da gibt es irgendwelche UI Tools die man brauchen muss, weil man es über die Kommandoziele machen würde, würde man alles kaputt machen und vieles ist schon nicht mehr nur ein reinen Textdateien gespeichert, sondern zum Teil irgendwelche Settings Files, also so Portety Lists jetzt unter macOS oder irgendwelche Binärdaten und das ist dort wo jetzt spezifisch macOS der Pfad schon auch verlässt, respektiv die Philosophie ein bisschen ritzt oder verletzt. Da gehe ich gleich weiter zu Windows, also dort ist es ähnlich, dort kommt man einfach von der anderen Seite, mann probiert ein bisschen Linux reinzuholen, aber auch dort dann haben sie ewig den Konflikt zwischen den UI Tools und zum Teil halt noch dort ist jetzt noch extremer, Windows 11 auf Windows 10 Dialoge und über die Kommandozeile kann man gewisse Sachen machen, gewisse dann nicht, dann gibt es 4 Lösungsansätze, je nachdem zum Beispiel die ganze User Management, Access Management das ist ein rissiger hassle unter

Windows. Je nach dem auf welcher Ebene der User erstellt worden ist, muss man ein grafisches Tool brauchen oder dann muss man es oder kann man es nur über die Kommandozeile lösen, das ist super mühsam. Eben es ist gerade für viele attraktiv geworden, die ein bisschen frustriert sind von Linux, wo ein Linux Betriebssystem möchten, aber auf den Komfort von Windows nicht verzichten möchten. Vielleicht jetzt Leute, die sich mehr in einem Web Developer Umfeld bewegen, die halt darauf angewiesen sind, weil Windows die weitverbreitete Plattform ist, dass sie es direkt darauf test können, aber trotzdem irgendwelche Treiber brauchen, irgendwelche Hardware brauchen, aber eine Flexibilität von einem Linux möchten. Das ist natürlich das Windows Subsystem schon sehr eine gute Alternative muss ich sagen. Vor allem jetzt die zweite version, bin ich auch recht ein Fan davon. Und dort sehe ich schon, dass sich Windows auch ein bisschen dieser Gruppe von Special Need Users annähert, durch das Subsystem. Von dem her geht das sicher in die richtige Richtung, auch wenn es dann Richtung Docker geht, obwohl das ist eine Krücke natürlich unter Mac und Windows das sind nicht wirklich Dockerfähige Systeme, weil das baut ja auf einem native Linux Kernel auf, aber trotzdem Docker for Desktop als Hilfsapplikation wären dort auch noch so eine Alternative, allerdings nicht so performant wie eben jetzt ein Windows Subsystem. Aber diese Zielgruppe sehe ich dort schon von Profi User, die auf den Komfort von Windows nicht verzichten möchten, weil es gibt einfach gute Tools unter Windows und es ist einfach die Plattform, die dominant ist mit ja, vielleicht sind es mittlerweile nur noch 95% anstatt 98%. Aber es ist halt nach wie vor extremst relevant und nachher als zweites dominantes Betriebssystem ist Mac vielleicht irgendwie 15%, wobei das würde dann nicht gehen wenn man auf 100 aufrundet, und Linux die dann ganz einen kleinen Teil ausmacht. Jetzt noch eine Zwischenfrage, finden Sie es gut, dass Windows auf Platz 1 ist, von den Nutzer Betriebssystem oder würden Sie persönlich sagen, dass jetzt macOS oder Linux besser wäre? Nein ich habe nicht das Gefühl, dass das ein Problem ist. Es ist halt eine historisches, politisches Problem, dass Windows so dominant ist, wäre es nicht Windows wäre es ein anderes Betriebssystem und dann würde man das halt einfach als Target ins Auge fassen sowohl politisch und technisch, darum gibt es die meisten Angriffe auf Windows. Deshalb kritisieren die meisten auch Windows. Ja, es ist nicht Open Source ok, macOS ist auch nicht Open Source letzte Konsequenz. Linux übrigens auch nicht, viele Tools die gibt es nicht in Source Form, also von dem her ist es jetzt halt einfach so wie es ist, wird sich mittelfristig auch nicht ändern, bis man dann natürlich unter Umständen einfach mal das Konzept vom klassischen Desktop Computer nicht mehr braucht. Wenn dann irgendwelche andere Paradigmen kommen, eben jetzt irgendwie über Smartphone oder über irgendwelche ja Brain Implants oder so, wobei da kann ich mich nicht zu einer Prognose konkret äussern jetzt ist es einfach mal so wie es ist und das schadet sicher Windows nicht, eben dass es so bisschen die Embrace Philosophie hat, die Linux Komponenten aufgreift, jetzt einfach rein auf der Ebene vom Benutzer. In the cloud dort sieht es ein bisschen anders aus. Dort gibt es eigentlich nur Linux, dort gibt es sehr wenige Windows Systeme. Von dem her balanciert sich das ein bisschen aus. Frage 11 haben Sie eigentlich schon beantwortet. Die Frage 12 wo sind Ihrer Meinung nach grundlegende Unterschiede von Architektur, Design und Nutzerphilosophie von diesen 3 Betriebssysteme untereinander von Linux, macOS und Windows? Ja eben Architektur, da habe ich mich schon dazu geäussert, also da ist man dann irgendwo auf der Eben vom Kernel, Micro versus Monlithic Kernels oder Hybrid Kernels, so

wie es unter Mac ist, aber sonst eben Design und Nutzer Philosophie, eben es verschmilz alles. Als auch wenn der Trend auf Linux sitzt, möglich vieles grafisch anzubieten, ich meine das spricht Bände und von dem her das nimmt sich nicht mehr viel, man fährt mit der Maus über irgendwelche Masken, klickt irgendwelche Buttons oder Touchscreen ob es jetzt ein Windows Tablet ist oder ein IOS Phone oder ein Android Tablet oder wie auch immer, grundsätzlich das ist alles so ähnlich, man kann die Paradigmen nicht jedes mal neu erfinden. Es ist schon seit 30 Jahren so, es hat sich bewährt und das bleibt jetzt einmal so, bis man dann eventuell mal so ein Brain Computer hat. Aber sonst, im Moment denke ich bleibt das mal sehr sehr ähnlich. Jetzt im Bereich Softwareentwicklung, bevor man jetzt diese UNIX Philosophie festgelegt hat, hat sich nachher in der Softwareentwicklung etwas geändert? Ja ich würden doch schon sagen, also vorher hat man sehr stark Main Frame fokussiert gearbeitet, mit irgendwelchen, wie soll ich das sagen? Nicht sehr flexiblen Programmiersprachen, also irgendwelche Cobalts oder irgendwelche Fortrans, Algols dieser Welt, in den 50er 60er Jahren. Vorher war es noch schlimmer, dann hat man einfach genau für diese Maschinen den Programmcode mehr oder weniger in Hardware gegossen und was halt die UNIX Philosophie über die Pipes eingeführt hat ist so das funktionale, also eben auch dank der Modularität, dass man nicht mehr einfach sehr statisch programmiert, obschon C ja eher dem ein bisschen widerspricht, also das C als die Programmiersprache, die eigentlich der Siegeszug von UNIX oder ist es umgekehrt gewesen? Es ist immer so schwierig oder eine Hähne Ei Problematik. Erst ermöglicht hat weil man das UNIX dank C auf verschiedene Plattformen Portieren konnte, weil es so eine einfache Programmiersprache ist. Aber denken wir jetzt ein Level weiter oben, also über diese Kommandotools, wo wir Shell brauchen, Sie kennen es ja vom Kurs oder wo man mit Pipes miteinander eben arbeiten kann. Das ist schon ein sehr modernes Konzept, das man jetzt eigentlich eher so erfährt, wenn man jetzt so modernere Programmiersprachen anschaut, die fangen jetzt alle an, mit der eben sequence base Programmierung an, also mit den Ranges in C++, mit Link in C# oder .NET, mit den Streams in Java, das ist alles oder PowerShell, ganz extrem, das ist alles irgendwie auf dieser Philosophie basierend, dass man so funktional arbeitet, dass man alles in einer Sequenz hat, anstatt dass man so geschachtelte Funktionsaufrufe hat, wo man im Prinzip von innen nach aussen denken muss, das macht es extrem unergonomisch. Jetzt kann man einfach wie man Test liest, von links nach rechts kann man sich irgendwelche Befehle aneinander Reihen und man sieht sofort was da passiert. Das denke ich ist schon so eine Lasting Legacy oder ebene ein Tool Design, das sich absolut durchgesetzt hat. Das muss man ganz klar sagen. Aber eben ich muss noch dazusagen, das sind einfach meine Ansichten oder vielleicht jemand, der jetzt bei Microsoft arbeitet hat ja da irgendwie einen guten Einwand, wo man sagen kann „OK ja doch das Bärenpunkt“, aber jetzt von mir aus gesehen eben denke ich das es wirklich ein relevanter Faktor wo da UNIX in die ganze Betriebssystemlandschaft eingeführt hat. Denken Sie, dass die UNIX Prinzipien so in den nächsten Jahren oder Jahrzehnten in den Betriebssystemen immer noch vorkommen oder dass man bis dahin irgend etwas anderes wieder festlegt? Ja, irgendwann ist man ausgeschossen mit den Konzepten. Also von dem her man hat fast keine andere Wahl als das so zu machen, ja vielleicht irgendwelche Real Time Operating Systems, die ganz extreme Constraints, also Einschränkungen haben. Der braucht vielleicht jetzt nicht ein Linux Kernel oder einen UNIX Kernel, da hat man

mehr so ein RTOS Kernel, das sind aber, aber schlussendlich probiert man das Konzept zu wiederholen, weil das hat sich bewährt. Also darum sehe ich jetzt nicht, zumindest auf der technischen Ebene, da jetzt noch extreme Revolution passiert. Und ein Quantencomputer, der braucht kein Betriebssystem, also das sind Optimierungsmaschinen, das ist völlig etwas anderes. Also ich kann jetzt nicht mit dem Argument kommen „Ja wie sieht jetzt aus, wenn man dann mal eben die Quantencomputer hat?“Auch der wird Schnittstellen anbieten, die sich genau so verhalten werden, weil es sich einfach bewährt hat über die Zeit. Aber vielleicht mehr so, dass das Halt weg abstrahiert wird, dass man als User, als technischer Benutzer sich gar nicht mehr ganz im klaren ist, und es ist auch nicht mehr relevant was darunter passiert, also under the hood, es kann im Prinzip dann nachher wie egal sein. Aber jetzt einfach, wen ich ein Ingenieur wäre und ein Betriebssystem entwickeln würde, würde ich wohl ähnliche Ansätze verwenden halt auch, weil es sich bewehrt hat. Und auch verschiedene Hardware, welche es jetzt gibt, wie ARM, Risk W, also Risk 5, die brauchen alle irgendwelche Linuxoiden Betriebssysteme, weil die sich auch schnell rapportieren lassen. Da warten Sie nicht bis Microsoft sich dazu durch gerungen hat, eben dort irgendwie Windows drauf laufen lassen zu können. Passen zu der Antwort gehe ich schnell zur letzte Frage, denken Sie in dem Fall dass UNIX Philosophie, in den nächsten Jahren vielleicht stärker umgesetzt werden als sie jetzt sind? Ja ich denke der Trend aktuell ist schon in diese Richtung, definitiv. Also eben, wenn man das ganze Cloudcomputing, Cloud Infrastructure, Hyperscaling anschaut, sieht man diesen Trend ganz klar und auch im ganzen Machine Learning Umfeld, weil es sind schlussendlich auch Hyperscaler, da braucht man einfach ein Linux und mit Windows ist das sehr sehr schwer umzusetzen, weil die Iterationszeiten, für etwas neues zu entwickeln bei Windows zu hoch ist, weil es ein komplexes grafische System ist, mit den ganzen Überbau, da probiert man das trotzdem zu Shoehorne, also rein zu drücken, dass man das dann nachher sauber machen kann, bringt es aber nie wirklich ganz hin. Also darum, die UNIX Prinzipien, ob konkret oder abstrakt, über die Modularität eben von irgendwelchen Containern, Kubernetes oder so, das wird sich so sicher stärken. Und man hat auch verschiedene Abstraktionslevel und das ermöglicht eben UNIX sehr gut, dass man sich in diesen verschiedenen oder auf diesen verschiedenen Abstraktionslevels ohne viel Reibung bewegen kann. Also im Prinzip ganze Tech Stack hoch und runter gehen kann. Ja, die Frage 15 habt ihr ja auch schon beantwortet mit diesen Containersierung. Genau. Das wär’s eigentlich auch gewesen. Ja, und eben sonst, falls Ihnen jetzt dann beim Zusammenfassen oder beim normalen Auswerten eine Frage kommt oder wenn es eine Frage gibt oder noch eine Klärung braucht oder so, dann schreiben Sie mir einfach und nachher probieren wir diese noch mal besser zu beantworten oder vielleicht im Nachgang kommt mir dann etwas noch in den Sinn, da kann ich Ihnen das schon noch bekannt geben, oder würde sie Ihnen das auch noch kommunizieren. Ja merci.