BETRIEBSSYSTEME
EINFÜHRUNG UND GESCHICHTE

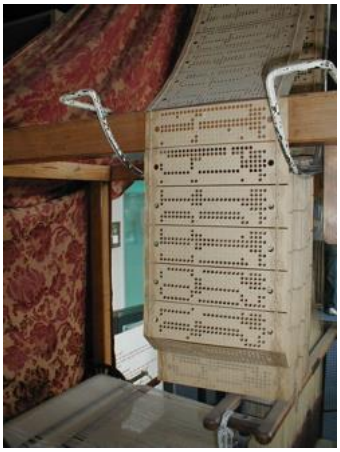
Prof. Dr. Jörg Mielebacher
mail@mielebacher.de

Die folgenden Folien führen in Eigenschaften und Geschichte von Betriebssystemen ein. Zu jeder Folie sind Notizenseiten erfasst.

Verbesserungsvorschläge und Fehlerhinweise können Sie gerne an die Adresse mail@mielebacher.de senden.

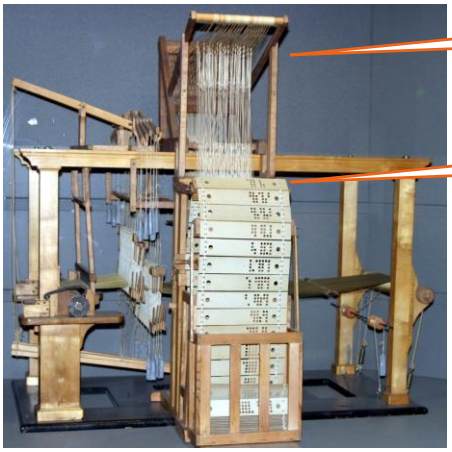
Ein Ausflug in die Geschichte

Was ist das?



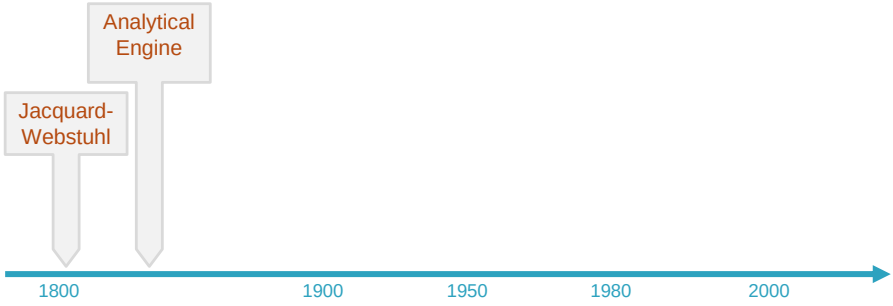
Erkennen Sie, was hier gezeigt ist? Auf den ersten Blick hat es wenig mit Computern oder Betriebssystemen zu tun.

Was ist das?



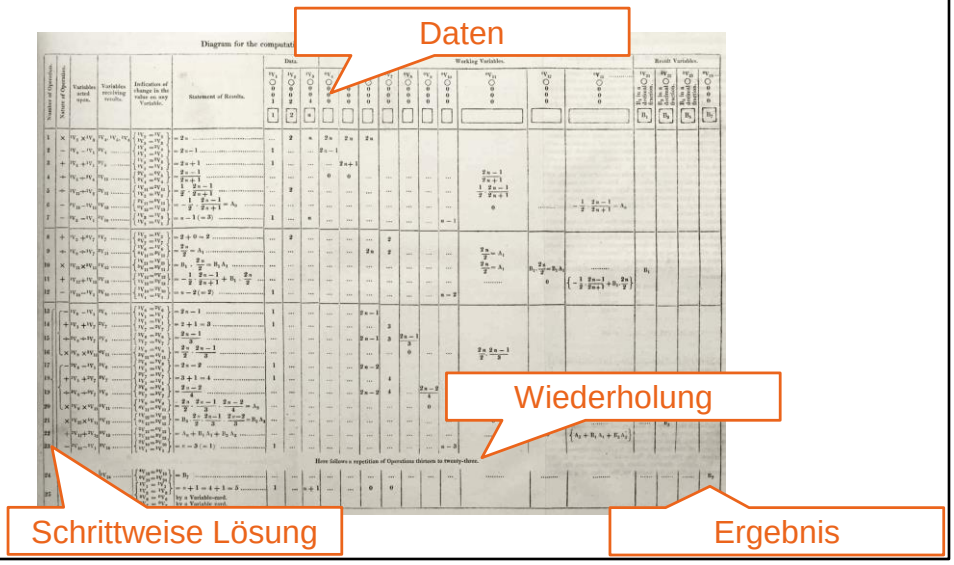
Es handelt sich um den sog. Jacquard-Webstuhl, der frühere Webstühle 1805 unter anderem um eine Lochkartensteuerung erweiterte. Die Idee, Daten und Programme auf Lochkarten abzulegen, blieb über die Textilproduktion hinaus erhalten. Musikinstrumente wurden damit gesteuert, vor allem aber Computer bis weit über die Mitte des 20. Jahrhunderts. Den Ursprung des Einsatzes in Computern geht auf die nie vollendete Analytical Engine von Charles Babbage zurück. Bleibt die Frage, was das mit Betriebssystemen zu tun hat – dazu später mehr.

Ein Blick zurück



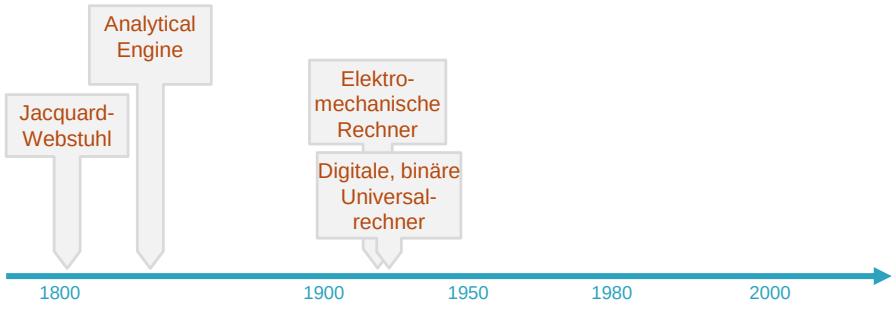
Die Meilensteine dieser Entwicklung lassen sich gut auf einem Zeitstrahl visualisieren. Allerdings wäre eine lineare Skala aufgrund der rasanten Entwicklungen, gerade seit der 2. Hälfte des 20. Jahrhunderts, nicht zielführend. Auch ist diese Übersicht nur ein Ausschnitt – es fehlen beispielsweise die Rechenmaschinen von Pascal und Leibniz oder auch von de Prony, der das Prinzip eines Fließbands für Berechnungen nutzte. Nicht zu vergessen die wichtigen formalen Grundlagen, unter anderem von de Morgan und Boole.
Für unsere Zwecke steht die Programmsteuerung im Vordergrund, verbunden mit der Idee des Universalrechners. Hier ist ein wichtiger Meilenstein mit dem Jacquard-Webstuhl gegeben, vor allem aber mit der Analytical Engine von Charles Babbage. Obwohl sie letztlich nur als Entwurf vorlag, bildete sie die Basis für elementare Weichenstellungen der Informatik; die wegbereitenden Arbeiten von Ada Lovelace beziehen sich auf diese Analytical Engine.

Lady Adas Programmentwurf

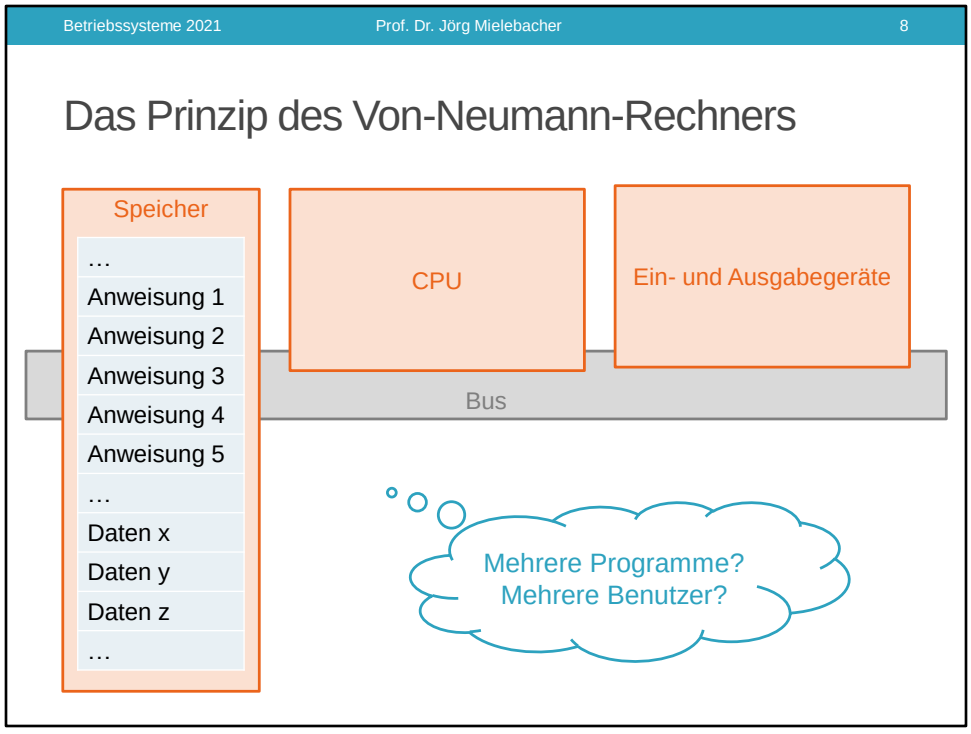


Lady Ada Lovelace war ihrer Zeit weit voraus, als sie für die unfertige Analytical Engine Konzepte einführte wie nummerierte Anweisungen, Register, Operationen, Kommentare, Wiederholungen usw. Sie legte damit wichtige Grundlagen für die Programmierung, wie wir sie heute noch kennen. Heute hoch relevant ist auch Ihr Einwand, dass Maschinen nicht denken können.

Ein Blick zurück

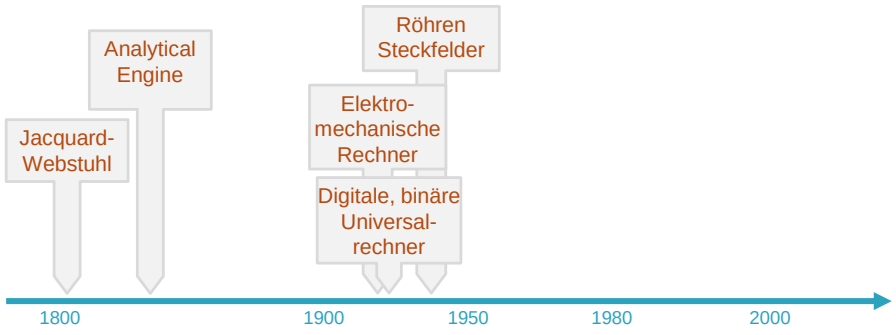


Die Jahre um 1937, von manchen auch als Annus Mirabilis der Computertechnik bezeichnet, bringen entscheidende Festlegungen mit sich. Verbunden ist dies mit den Arbeiten Turings zur Berechenbarkeit. Shannon zeigte die Umsetzung boolescher Algebra in elektrische Schaltungen. Stibitz setztes Relais ein, um Binärzahlen zu addieren. Aiken entwarf den Großrechner Mark I, und Konrad Zuse baute seinen ersten elektromechanischen Computer auf. Aus diesen elementaren Arbeiten ging die Idee des Universalrechners hervor, der digital (und nicht wie von anderen zu dieser Zeit propagiert analog) arbeitet und das Binärsystem für die Speicherung und Verarbeitung nutzt. Dies wird insbesondere während und nach dem 2. Weltkrieg weiterentwickelt, wobei die Architektur eng mit den Namen von Neumann, Eckert und Mauchly eng verbunden ist, während die Programmierung maßgeblich durch Grace Hopper beeinflusst wurde.



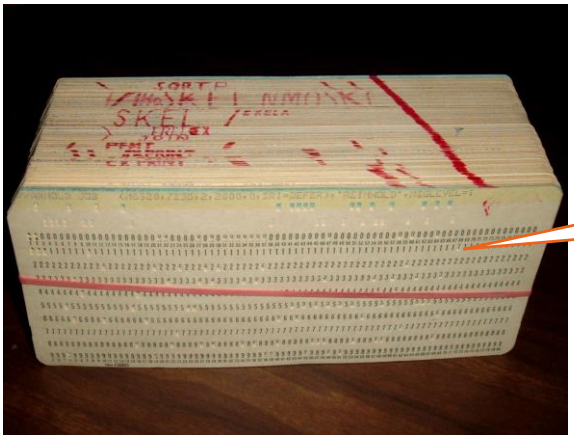
Kehrt man zurück zu den Computern wird häufig das Modell des Von-Neumann-Rechners bemüht, der als wichtigste Komponenten die CPU, den Speicher, die Ein- und Ausgabegeräte sowie einen Bus besitzt, der alle Komponenten miteinander verbindet. Eine Besonderheit ist die Tatsache, dass Daten und Anweisungen im selben Speicher liegen.
Wenn man sich dieses einfache Schema anschaut, fragt man sich aber schnell: Wie lassen sich mehrere Programme ausführen, womöglich sogar gleichzeitig? Oder auch: Wie können mehrere Benutzer diesen Computer nutzen?
In einer Vorlesung über Betriebssysteme mag auch die Frage angebracht sein: Wo ist das Betriebssystem?

Ein Blick zurück

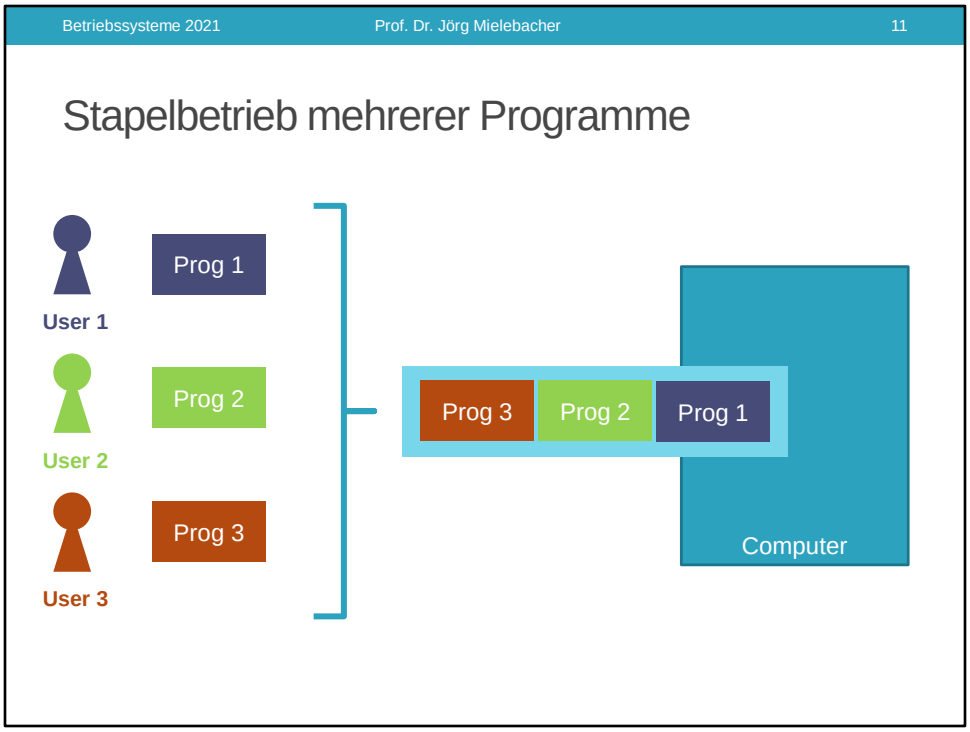


Die Jahre um 1937, von manchen auch als Annus Mirabilis der Computertechnik bezeichnet, bringen entscheidende Festlegungen mit sich. Verbunden ist dies mit den Arbeiten Turings zur Berechenbarkeit. Shannon zeigte die Umsetzung boolescher Algebra in elektrische Schaltungen. Stibitz setztes Relais ein, um Binärzahlen zu addieren. Aiken entwarf den Großrechner Mark I, und Konrad Zuse baute seinen ersten elektromechanischen Computer auf. Aus diesen elementaren Arbeiten ging die Idee des Universalrechners hervor, der digital (und nicht wie von anderen zu dieser Zeit propagiert analog) arbeitet und das Binärsystem für die Speicherung und Verarbeitung nutzt. Dies wird insbesondere während und nach dem 2. Weltkrieg weiterentwickelt, wobei die Architektur eng mit den Namen von Neumann, Eckert und Mauchly eng verbunden ist, während die Programmierung maßgeblich durch Grace Hopper beeinflusst wurde. Vorherrschende Technik anfangs fehleranfällige Relais und danach Elektronröhren, die Computer zu riesigen Anlagen mit enormem Energiebedarf machten.

Programme als Lochkartenstapel

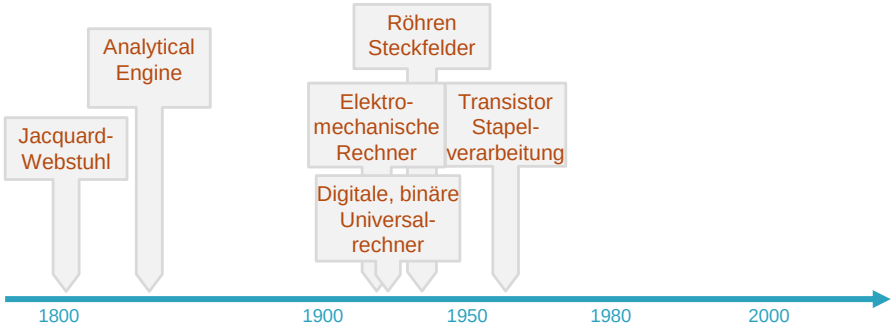


Während Computer anfangs oft durch Kabelverbindungen auf Steckfeldern programmiert wurden, setzten sich nach schnell Lochkarten (teilweise auch Filmstreifen) für die Programmierung durch. Programmieren bedeutete, ein Programm zu entwurfen, dieses auf Lochkarten zu codieren bzw. zu stanzen, die Lochkarten den Bedienern des Computers zu geben und auf das Ergebnis in Form von Ausdrucken oder dergleichen zu warten.
Diese Art, Computer zu verwenden, führt auf den Begriff der Stapelverarbeitung (Batch processing), der im Kontext von Betriebssystemen wichtig ist.



Sollen im Stapelbetrieb nun die Programme mehrerer Benutzer ausgeführt werden, so müssen die Programme sequenziell abgearbeitet werden, also in einer Art Warteschlage, ggf. mit Prioritäten. Programm 1 wird somit in den Speicher geladen, ausgeführt und das Ergebnis an User 1 gegeben, dann folgt Programm 2 in gleicher Weise und schließlich Programm 3. Insgesamt also ein sehr ineffizienter Prozess, den Möglichkeiten der damaligen Technik aber angemessen.

Ein Blick zurück



Die Erfindung des Transistors machte die fehleranfällige Röhren- und Relaistechnik überflüssig, erlaubte damit auch kleinere Computer, wobei die Transistoren kurz danach bereits in Form von Integrierten Schaltkreisen stark verkleinert wurden und zu einer rasanten Verkleinerung und Leistungssteigerung der Hardware führte. Dies ist zugleich die Zeit der klassischen Stapelverarbeitung.

Programmierung zu dieser Zeit

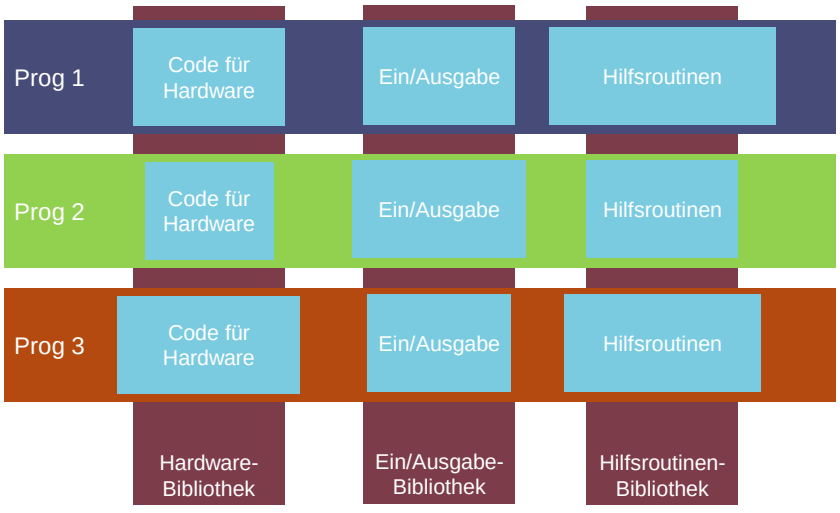
- Programmierung auf Papier
- Hardwarespezifischer Maschinencode
- Keine symbolische Programmierung
- Keine Compiler
- Fehleranfällig und ineffizient

➔ Entwicklung hardwareunabhängiger, standardisierter Programmiersprachen (z.B. COBOL)

➔ Entwicklung von Compilern, die Symbole in Maschinencode übersetzen

Die Programmierung war zu dieser Zeit ineffizient und fehleranfällig: Programme mussten entworfen und auf Lochkarten übertragen werden, wobei hardwarespezifischer Maschinencode notwendig war, d.h. ein Programm konnte nicht auf unterschiedlicher Hardware verwendet werden. Entsprechend gab es auch keine symbolische Programmierung, wie wir es heute von den Hochsprachen kennen und daher auch keine Compiler.
Diese führte zu dem Wunsch, hardwareunabhängige, standardisierte Programmiersprachen zu entwickeln, was vor allem durch Grace Hopper stark vorangetrieben wurde und letztlich zu der Programmiersprache COBOL führte, die man noch heute in manchen Großrechner-Anwendungen antrifft. In diesem Zuge wurden auch Compiler entwickelt, mit denen in der Programmiersprache verfasster Quellcode in die jeweiligen Maschinenanweisungen übersetzt wurde.

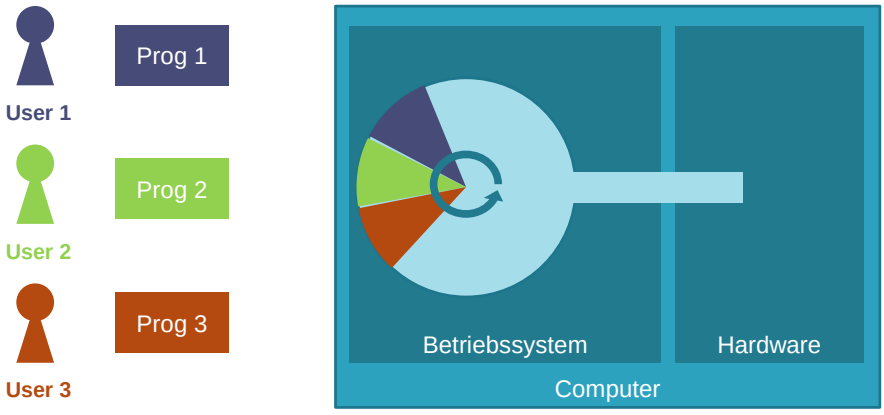
Ein Blick in die Programme



Bei einem genaueren Blick in die zuvor verwendeten Programme 1-3 würde man außerdem feststellen, dass es im Kern dieser Programme große Ähnlichkeiten gibt – alle drei Programme würden Code benötigen, der auf die jeweilige Rechnerhardware angepasst ist. Alle würden Code für die Ein/Ausgabe beinhalten. Und alle würden vermutlich auf vielerlei Hilfsroutinen zurückgreifen. Eine Idee wäre also, die Gemeinsamkeiten in Bibliotheken zusammenzufassen und sie somit wiederverwendbar zu machen.

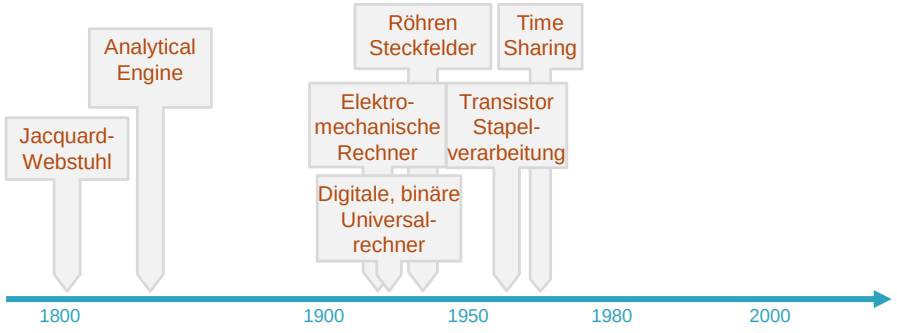
Dies wurde auch so gehandhabt, insbesondere vor den ersten Betriebssystemen: Zu den Rechner gab es Bibliotheken, die für eigene Programme genutzt werden konnten. Diese Bibliotheken wurden in Form

Time Sharing mehrerer Programme



1957 wurde durch Bemer und McCarthy das sog. Time Sharing eingeführt. Dieses löst das Problem des ineffizienten, sequenziellen Stapelbetriebs, indem mehrere Benutzer parallel einen Computer nutzen können. Die Ausführung des Programms wird hierfür in kleine Abschnitte unterteilt, wobei jeder Abschnitt nacheinander für eine kurze Zeitscheibe die Ressourcen des Computers (vor allem den Prozessor) nutzen kann. Dies brachte außerdem mit sich, dass die einzelnen Nutzer nun durch Terminals mit dem Computer verbunden waren und ihn im Dialogbetrieb, also interaktiv nutzen konnten.
Hiermit war es möglich mehrere Programme zur selben Zeit auf einem Computer auszuführen. Vor allem aber kommen wir nun zum Kern dieser Veranstaltung, dem Betriebssystem. Denn nun ist es notwendig, dass eine zusätzliche Software die Nutzung der Hardware steuert und auch den Zugriff der einzelnen Nutzer überwacht.

Ein Blick zurück



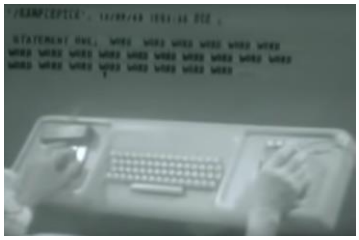
Die fortschreitende Integration von Transistoren in Form integrierter Schaltkreise (IC) trieb die Leistungsfähigkeit der System weiter voran und führte zu Konzepten wie dem Time Sharing, das es erlaubt, mehrere Programme zeitgleich ablaufen zu lassen und damit mehrere Nutzer zu bedienen. Die Entwicklung nahm jedoch nicht alleine diesen zentralisierten Weg, sondern führte über Bastler, Hippies und innovative Unternehmer zu einem dezentralen Ansatz in Form von Personal Computern.

Vannevar Bush: As we may think

- Memex
  - Informationszugriff
  - Hypertext
  - Assistenzfunktion
- Verknüpfung von Militär, Wissenschaft und Industrie

Die Geschichte des Personal Computers umfasst neben der eigentlichen Geschichte von Hard- und Software vor allem auch die besonderen Konzepte, die den PC ausmachen – nämlich von einem einzigen Menschen genutzt zu werden und unmittelbar nutzbar zu sein. Das steht im klassischen Widerspruch zu der Philosophie des Großrechners ausmachen. Dies führte 1974 sogar Ken Olsen von DEC zu der Aussage „I can‘t see any reason that anyone would want a computer of his own“.
Tatsächlich reichen die Wurzeln bis in das Jahr 1945 zurück, als Vannevar Bush sein Essay As we may think veröffentlichte. Darin beschreibt er die Idee des Personal Computers in Form des von ihm erdachten Memex, das Menschen bei der effizienten Nutzung von Informationen unterstützen soll. Er beschreibt darin z.B. Konzept wie Hypertext oder auch Assistenzfunktionen und inspirierte damit zahlreiche Informatiker wie Licklider und Engelbart, die wiederum zur Keimzelle der sich an der Ostküste schnell entwickelnden Szene von Bastlern und visionären Unternehmern wurde.
Bush spielte noch aus einem anderen Grund eine wichtige Rolle bei der Entwicklung der Informatik in den USA: Er vereinte drei Rollen als MIT-Forscher, als Gründer des Unternehmens Raytheon und als wichtigster Militärberater im 2. Weltkrieg. Diese drei Rollen führten dazu, dass er Militär, Wissenschaft und Industrie in den USA eng miteinander verknüpfte und die rasante Entwicklung der Informatik in den USA massiv förderte – diese Verknüpfung ist aber nicht unumstritten.

Douglas Engelbart: Mother of all demos

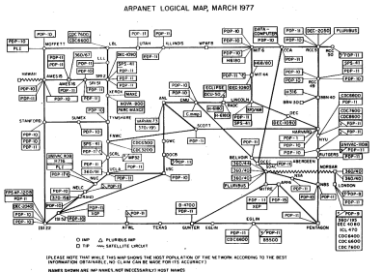


https://www.youtube.com/watch?v=yJDv-zdhzMY

Douglas Engelbart hatte mit seinem Team am Augmented Human Intellect Research Center in den 1960er Jahren visionäre Entwicklungen auf den Weg gebracht, die für den Erfolg des Personal Computers unerlässlich waren. Prominentestes Beispiel ist die Computermaus. Die Vorstellung bei der sog. Mother of all Demos (1968) vermittelt einen Eindruck davon, wie weit voraus das Team seinerzeit war; sie ist verfügbar unter https://www.youtube.com/watch?v=yJDv-zdhzMY Die Übergang zu grafischen Benutzeroberflächen (GUI) wurde hier vorweggenommen, setzte sich aber letztlich erst in den 1980er Jahren zunächst in Apples Computern, später auch in Microsoft Windows durch und wurde Teil des Betriebssystems.

JCR. Licklider: Man-Computer Symbiosis

© Joi Ito



Mensch-Computer-Interaktion in Echtzeit

Verteilte Computernetze

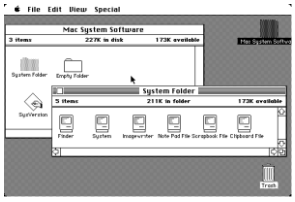
JCR Licklider hat, wie auch Engelbar, früh den Nutzen grafischer Bedienschnittstellen erkannt und sie bereits in den 1950er Jahren für das am MIT entwickelte sog. SAGE System eingesetzt. Er befasste sich aber auch stark mit der Symbiose von Mensch und Computer bzw. wie der Computer die menschliche Intelligenz verstärken kann. Er suchte die Mensch-Computer-Interaktion in Echtzeit, anstatt der damals vorherrschenden Stapelverarbeitung. Licklider steht aber zusammen mit Bob Taylor hinter einer weiteren Idee, die unser leben grundlegend verändert hat: Seine Vision waren verteilte Computernetze. Auf dieser Grundlage entstand das ARPANET, der Vorläufer dessen, was wir heute als Internet bezeichnen.
Netzwerkfähigkeit ist neben der GUI zu einem wesentlichen Teil des Personal Computers geworden und ist ebenfalls ein integraler Bestandteil des Betriebssystems.

Personal Computer

Altair 8800

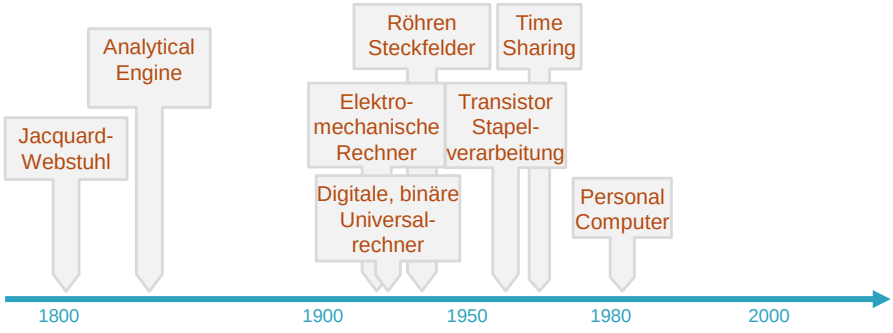
Xerox Alto

Apple Macintosh



Die Entwicklung des PC wurde maßgeblich durch die Verfügbarkeit von Mikrocontrollern vorangebracht. Bastler konnten Rechnerbausätze wie die Altair 8800 kaufen und zusammenbauen, während in den Laboren von Xerox – allen voran das legendäre Palo Alto Research Center PARC - Geräte wie der Alto entstanden, der mit GUI und Maus ausgestattet war. Von der Konferenz bei Xerox ist überliefert, dass die männlichen Führungskräfte an den ausgestellten Altos keinerlei Interesse zeigten, ihre Ehefrauen jedoch begeistert waren von den Möglichkeiten und der intuitiven Bedienung – ein deutlicher Hinweis, dass der Personal Computer gerade auch für die private Nutzung interessant sein könnte. Der Alto wiederum diente letztlich als Vorlage für den Apple Lisa (1983) und den weitaus erfolgreicheren Apple Macintosh (1984), die beide mit einer GUI ausgestattet waren.
Einen Personal Computer zu verwenden, bedeutete aber auch, gegenüber einem Großrechner deutlich weniger Leistung zur Verfügung zu haben, was natürlich auch Auswirkungen auf die dort eingesetzten Betriebssysteme hatte, z.B. das System 1 des Macintosh.

Ein Blick zurück



Der Personal Computer prägt seit den 1980er Jahren unseren Alltag – beruflich, wie privat.

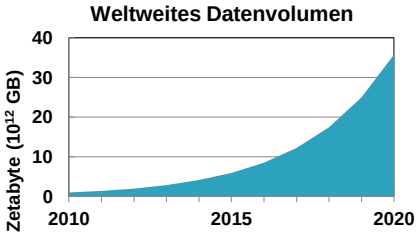
Und dann?

Internet/WWW

Mobile Computing

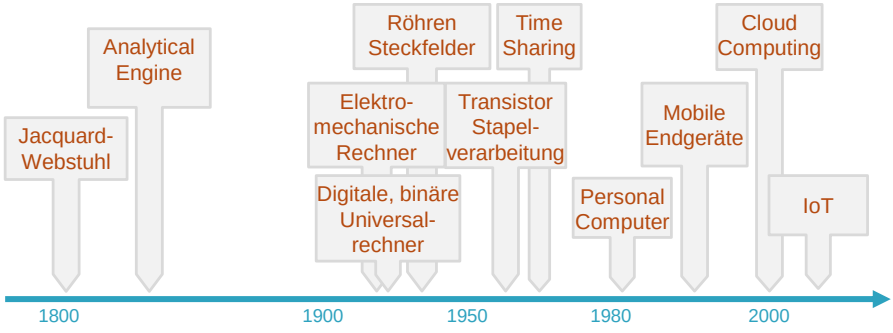
Cloud Computing

Internet der Dinge



Seit den 1990er Jahren hat die Entwicklung weiter rasant an Geschwindigkeit zugenommen, einen großen Anteil daran hatte die Verbreitung des Internet bzw. des World Wide Web, aus dem in den 2000er Jahren das Cloud Computing entstand, durch das sich Ressourcen über das Internet flexibel nutzen lassen. Auch das Internet der Dinge (Internet of Things, IoT) steht in direktem Zusammenhang damit. Während sich durch das Cloud Computing die Ressourcen virtuell beliebig zentralisieren lassen, werden im IoT viele leichtgewichtige Systeme in der Fläche eingesetzt, also auf dezentrale Erfassung und Verarbeitung (z.B. Edge Computing) gesetzt - in beiden Fällen mit wichtigen Auswirkungen auf die Ausgestaltung der Betriebssysteme.
Das Mobile Computing wiederum baut zwar auch auf dem Internet und anderen Kommunikationsnetzen auf, ist aber besonders deshalb interessant, weil es die Idee des PC noch weiter in unseren Alltag transportiert, z.B. durch Mobiltelefone oder Smartwatches. Die massiv vorangeschrittene Miniaturisierung der Hardware erlaubt es, selbst auf kleinem Raum hohe Rechenleistung abzurufen und damit GUIs, Spracheingabe oder Gestensteuerung zu ermöglichen.
Einen Trend sollte man sich dabei stets vor Augen führen: Nach Expertenschätzung verdoppelt sich das weltweite Datenvolumen etwa alle zwei Jahre. Die Weiterentwicklung von Hard- und Software wird also vor allem auch von der Frage getrieben werden, wie man diese riesigen Datenmengen sinnvoll nutzen kann.

Ein Blick zurück



Mobile Endgeräte werden besonders seit Mitte der 1990er Jahre immer leistungsfähiger und haben Einzug in den Privatkundenmarkt gehalten, hervorzuheben sind die Smartphones und in den letzten Jahren die Smartwatches. Das Cloud Computing folgte wenige Jahre später, ebenso das Internet der Dinge, das z.B. im Kontext von Industrie 4.0 und Smarthome eine große Rolle spielt.

Wenn Sie sich für Hintergründe der auf den letzten Folien beschriebenen Entwicklung interessieren, lohnt ein Blick die Bücher Isaacson: The Innovators und Markoff: What the Dormouse said. Einen interessanten Überblick über die Geschichte des Computer bieten außerdem das Heinz-Nixdorf-Museum in Paderborn und das Deutsche Museum in München.

Betriebssysteme und ihre Merkmale

Grundbegriffe

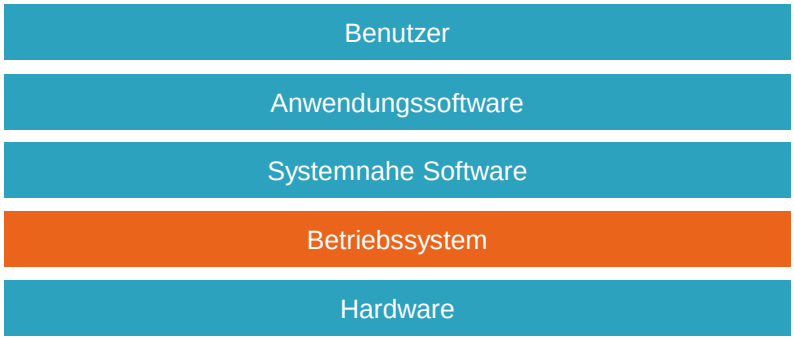
- Betriebssystem / Operating System (OS)
- Betriebssystemkern / Kernel
- Prozess / Task

Tanenbaum liefert in seinem Standardwerk zu Betriebssystemen (engl. Operating System, OS) eine sehr griffige Definition von Betriebssystemen: Demnach ist das Betriebssystem der Teil der Software, der die Ressourcen des Computers verwaltet und Anwendungsprogrammen zur Verfügung stellt.

Der Betriebssystemkern/Kernel ist der grundlegende, unverzichtbare Teil des Betriebssystems, auf dem alle übrigen Teile des Betriebssystems aufbauen; er hat direkten Zugriff auf die Hardware. Welche Teile dem Kernel zuzurechnen sind, hängt von der Architektur des Betriebssystems ab, dazu später mehr. Typische Beispiele für Aufgaben des Kernels sind die Prozessverwaltung und die Speicherverwaltung.

Unter einem Prozess bzw. Task versteht man ein Programm in Ausführung, d.h. eine Instanz eines Programms, wobei es zu einem Programm auch mehrere Instanzen geben kann. Das Programm selbst ist nur eine Folge von Anweisungen und wird es durch die Ausführung zu einem Prozess. Das Betriebssystem erfasst weitere Daten zu einem Prozess, z.B. die eindeutige Prozess-ID. Innerhalb eines Prozesses gibt es mindestens einen Thread, bei nebenläufiger Programmierung mehrere.

Abstraktion und Hardwareunabhängigkeit



Da jeder Prozessor einen eigenen Befehlssatz besaß, bestand ein großes Problem in der Anfangszeit der Programmierung darin, dass Programm speziell für die eingesetzte Hardware geschrieben wurden, d.h. die Maschinenanweisungen konnten nur von der jeweiligen Hardware korrekt umgesetzt werden.
Eine wichtige Aufgabe des Betriebssystems ist es deshalb, Hardwareunabhängigkeit zu schaffen, d.h. es stellt eine virtuelle Maschine bereit, die von der tatsächlichen Hardware abstrahiert. Programme greifen somit auf die virtuelle Maschine zu, während das Betriebssystem diese Zugriff in hardwarespezifische Zugriff übersetzt. Das Betriebssystem stellt somit eine eigene Schnittstelle für Systemaufrufe bereit, die von darüber liegenden Schichten genutzt werden kann.
Auf dem Betriebssystem baut die systemnahe Software auf, dazu zählen z.B. Datenbanken und Interpreter.
Hierüber wiederum folgt die Anwendungssoftware, die entweder direkt auf das Betriebssystem zugreift oder auf Dienste der systemnahen Software nutzt. Der Benutzer wiederum interagiert vorrangig mit der Anwendungssoftware.

Aufgaben des Betriebssystems

Wie bereits gesehen besteht eine wichtige Aufgabe des Betriebssystems darin, eine virtuelle Maschine bereitzustellen, die von der vorhandenen Hardware abstrahiert, darüber hinaus verwaltet das Betriebssystem die vorhandenen Hardwarekomponenten (Prozessen, Speicher, Laufwerke, Ein-/Ausgabegeräte usw.).

Der zweite Aufgabenblock betrifft die sog. Prozesse, also die Programme in Ausführung. Sämtliche Prozesse werden verwaltet (meist unter einer sog. Prozess-ID samt einiger Metadaten). Da üblicherweise mehrere Prozesse zeitgleich auf einem Prozessor ausgeführt werden, spielt das Scheduling eine entscheidende Rolle, d.h. wann welcher Prozess den Prozessor nutzen kann. Ein weiteres großes Aufgabengebiet ist die Kommunikation der Prozesse untereinander, die sog. Interprozesskommunikation, z.B. durch Sockets oder Pipes.

Betriebssysteme sind heutzutage i.d.R. mehrbenutzerfähig. Das Betriebssystem verwaltet also die Benutzer des Systems, meist anhand einer eindeutigen User-ID. Auch stellt es sicher, dass die Benutzer nur diejenigen Operationen durchführen können, zu denen sie berechtigt sind.

Hinsichtlich des Speichers verwaltet das Betriebssystem den Hauptspeicher (RAM) und die damit verbundenen Caches und Auslagerungsspeicher. Die Verwaltung ist dabei vergleichsweise aufwendig, weil heutzutage der verfügbare Speicher so aufgeteilt wird, dass jedem Prozess ein eigener, virtueller Speicherbereich zur Verfügung steht, um die Prozesse voneinander zu isolieren. Neben dem Hauptspeicher muss das Betriebssystem aber auch den Zugriff auf die nichtflüchtigen Speicher, z.B. die Festplatte sicherstellen. Das schließt die Nutzung von Dateisystemen ein, aber auch den tatsächlichen Zugriff auf das

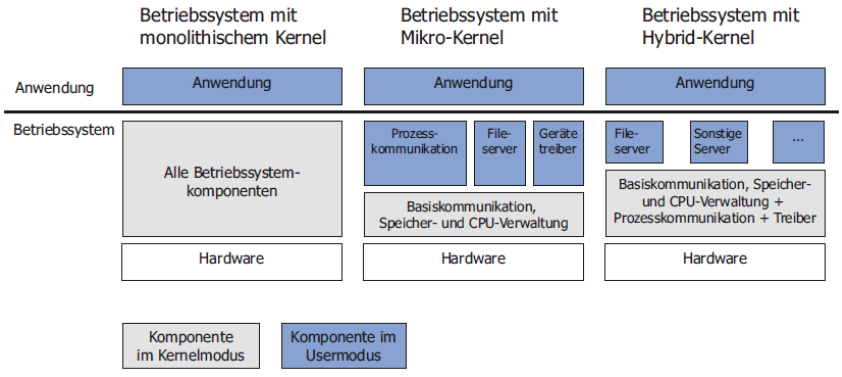
Speichermedium.

Zugriffsschutz

- Kernelmodus (privilegiert):
  - Aktivierung durch Prozessorregister
  - Nutzung nur durch das Betriebssystem selbst
  - Zugriff auf Kernel-Code und –Daten erlaubt
- Usermodus (nicht privilegiert):
  - Anwendungsprogramme
  - Kein direkter Zugriff auf Kernel-Code und –Daten
  - Unerlaubte Zugriffsversuche führen zu Ausnahme (Trap)
  - Nutzung der Schnittstelle für Systemaufrufe (Systemcall)

Der Zugriffsschutz des Betriebssystems basiert i.W. auf der Unterscheidung in den privilegierten Kernelmodus und den nichtprivilegierten Usermodus. Der gerade aktive Modus wird in einem Prozessorregister hinterlegt. Der Kernelmodus ist dem Betriebssystem selbst vorbehalten; er erlaubt Zugriff auf den Kern des Betriebssystems (Kernel) – seinen Code und seine Daten.
Anwendungsprogramme werden im sog. Usermodus ausgeführt. Aus dieserm Usermodus sind Zugriffe auf Kernel-Code und –Daten nicht erlaubt, der Versuch wurde zu einer Ausnahme (Trap) führen. Da jedoch Funktionen des Betriebssystem aufgerufen werden müssen (z.B. das Reservieren von Speicher, das Öffnen von Dateien usw.), können Anwendungsprogramme über die vom Betriebssystem bereitgestellte Schnittstelle Systemaufrufe starten.

Architektur von Betriebssystemen



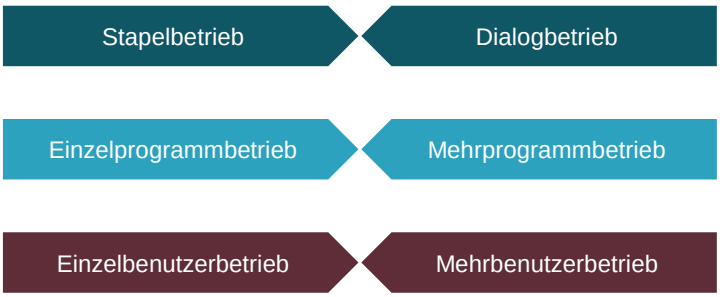
Quelle: Mandl: Grundkurs Betriebssysteme

Es gibt unterschiedliche Ansätze für den Aufbau des Betriebssystemkerns (Kernel): Früher setzte man vor allem auf sog. Monolithische Kerne, d.h. der Kern enthielt alle Bestandteile (Prozessverwaltung, Speicherverwaltung, Gerätetreiber usw.). Er wurde vollständig im Kernelmodus ausgeführt, nur die Anwendung selbst im Usermodus.

Die Idee sog. Microkernel-Architekturen basiert darauf, den priviligiert ausgeführten Teil des Kernels so schlank wie möglich zu halten, sodass darin nur Basiskommunikation sowie Speicher- und CPU-Verwaltung enthalten sind. Prozesskommunikation, Gerätetreiber usw. laufen dann im Usermodus.

Einen Mittelweg stellen sog. Hybridkernel dar. Prozesskommunikation, Speicherverwaltung, CPU-Verwaltung und Treiber laufen im Kernelmodus ab, lediglich die Serverdienste im Usermodus.

Betriebsarten



Weitere Merkmale

- Anwendungsgebiet (Großrechner, PC, Embedded usw.)
- Sicherheit
- Echtzeitfähigkeit
- Lizenzierung / Open Source
- Länge der Speicheradressen (z.B. 32bit oder 64bit)

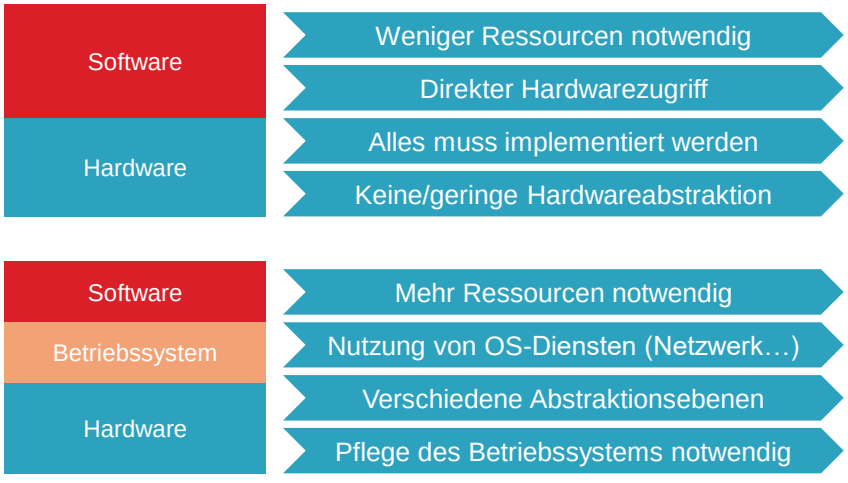
Betriebssysteme lassen sich darüber hinaus beschreiben hinsichtlich ihres Anwendungsgebiets (z.B. Großrechner, PC, Embedded, Mobilgeräte, Smartcards usw.), hinsichtlich seiner Sicherheit, hinsichtlich ihrer Echtzeitfähigkeit, hinsichtlich des Lizenzmodells, insbesondere ob ihr Quellcode offengelegt ist (Open Source), und hinsichtlich der Länge der Speicheradressen, wobei 8bit- und 16bit-System heute nur noch in Nischen anzutreffen sind. Überwiegend verwendet man heute 64bit-Betriebssysteme, teilweise auch 32bit-Systeme.

Exkurs: Arduino vs. Raspberry Pi

Kein Betriebssystem vs. Betriebssystem

Nach diesen Einordnungen ein kleiner Exkurs: Viele von Ihnen kennen sicher die beliebten Plattformen Arduino und Raspberry Pi. Beide sind kostengünstig und vielseitig. Sie unterscheiden sich jedoch in einem wichtigen Merkmal: Der Arduino verwendet kein Betriebssystem, der Raspberry Pi verwendet ein Betriebssystem. Das hat zur Folge, dass ein Programm, das man auf den Arduino lädt, unablässig ausgeführt wird – und es gibt genau ein Programm (einem etwaigen Bootloader kommt hier eine besondere Rolle zu). Der Raspberry Pi wiederum erlaubt es durch das Betriebssystem mehrere Prozesse parallel auszuführen, auch systemnahe Software (Datenbanken usw.) und bietet einen Mehrbenutzerbetrieb.

Exkurs: Arduino vs. Raspberry Pi

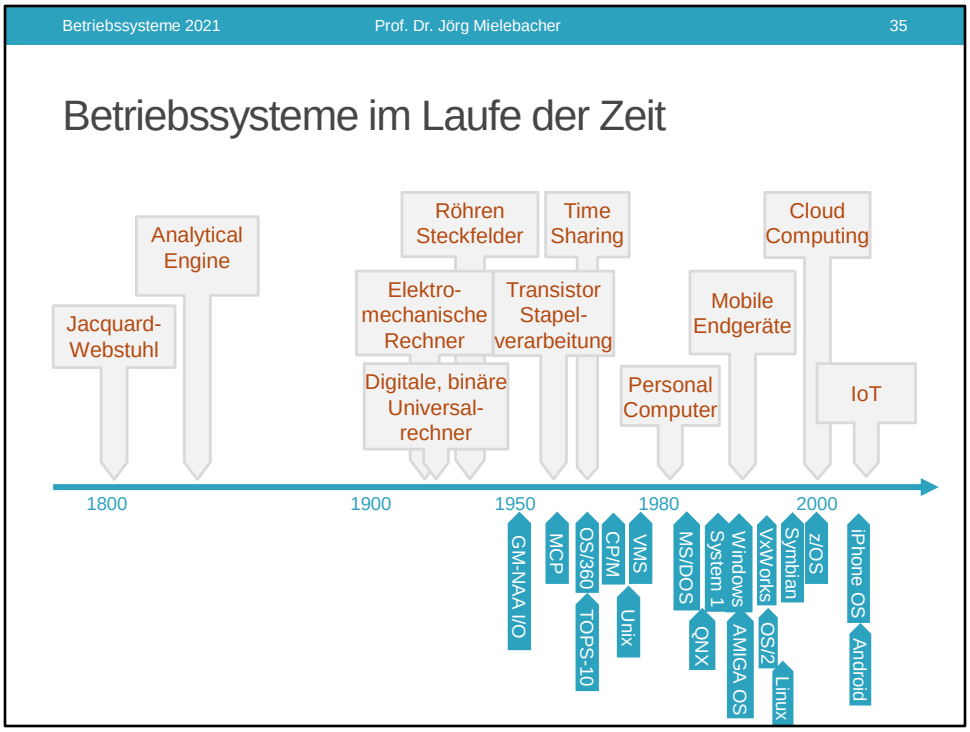


Diesen Vergleich kann man etwas verfeinern:
Ohne Betriebssystem benötigt man weniger Ressourcen, kann die Hardware unmittelbar nutzen, muss jedoch alles implementieren bzw. durch Bibliotheken hinzufügen (z.B. Zugriff auf Dateisysteme). Auch gibt es keine bzw. nur geringe Hardwareabstraktion.
Mit Betriebssystem benötigt man mehr Ressourcen, kann jedoch auf die Dienste des Betriebssystems zurückgreifen (Netzwerkzugriff, Zugriff auf Dateisysteme, Berechtigungskonzepte usw.). Das Betriebssystem bietet dabei unterschiedliche Abstraktionsebenen und erlaubt es i.d.R. mehrere Prozesse zur selben Zeit auszuführen. Allerdings ist ein Betriebssystem ein weiteres Stück Software, das gepflegt werden muss (Sicherheitsupdates usw.).
Es kommt also auf den Anwendungszweck an, ob ein Betriebssystem nützlich ist oder nicht.

Zwischenfrage

- Welche Betriebssysteme kennen Sie?

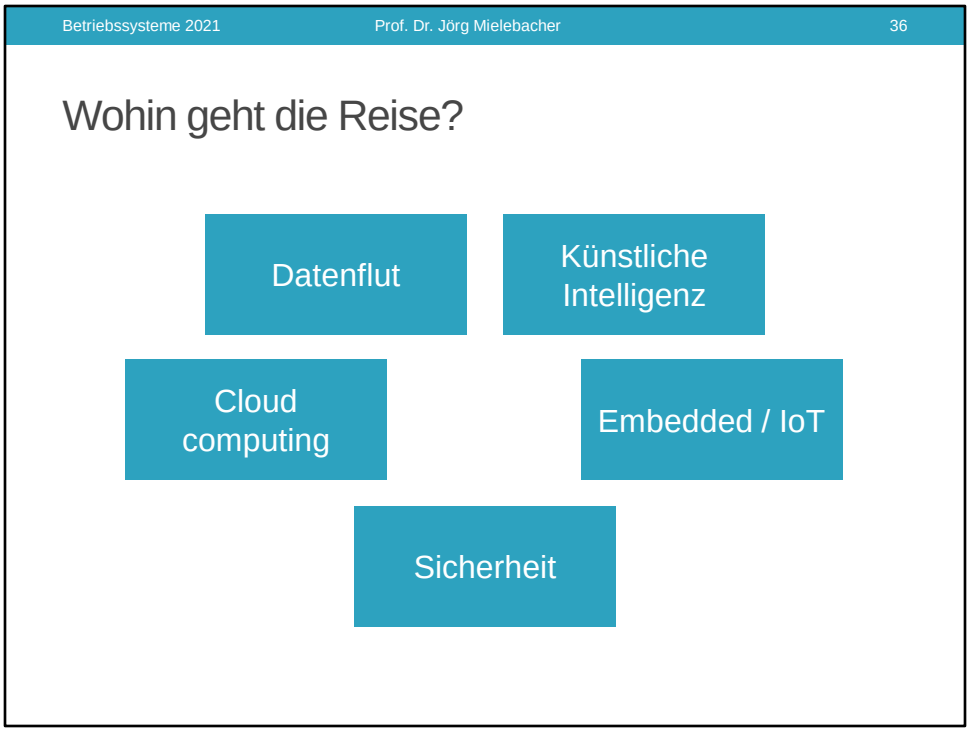
Welche Betriebssysteme kennen Sie? In welchem Umfeld hatten Sie mit ihnen zu tun? Gibt es Besonderheiten?



Bis in die 1950er Jahre war es üblich, dass Rechner ohne Betriebssystem verwendet wurden, d.h. Programme wurden – wie eingangs beschrieben – sequenziell und exklusiv ausgeführt.

Die Literatur nennt als erstes Betriebssystem das von General Motors für eine IBM 704 entwickelte GM-NAA I/O. Die Grundidee bestand darin, das nächste Programm automatisch zu starten, wenn das vorherige endete. Auch enthielt es häufig benötigte Routinen und erlaubte die Überwachung des Systems. Wie die meisten anderen Betriebssysteme der Anfangszeit wurde es nicht vom Hardwarehersteller entwickelt, sondern vom Kunden selbst.
1961 erschien mit dem Master Control Program MCP von Burroughs das erste Betriebssystem, das vollständig in einer Hochsprache geschrieben wurde.
1966 veröffentliche IBM das OS/360 für seine System/360 Mainframes.
1967 erschien TOPS-10 für den PDP-10 von DEC, das besonders an Universitäten und insbesondere des ARPANET sehr verbreitet war. Auch war es bekannt für die ersten darauf entwickelten Spiele.
Für den Intel 8080 Microcomputer entstand 1974 das Betriebssystem CP/M von Digital Research, im einfachsten Fall handelte es sich um ein 8bit-System. Neben Zugriff auf Disketten und dergleichen, enthielt es auch ein Basic Input Output System (BIOS). Bill Gates und Paul Allen setzten in den Anfangstagen von Microsoft ebenfalls stark auf CP/M.
Das Betriebssystem Unix wurde in den 1970er Jahren in den Bell Labs entwickelt und seit dem Ende der 1970er auch außerhalb der Bell Labs eingesetzt.
1977 erschien von DEC das bis heute weiterentwickelte VMS (OpenVMS), das unter anderem auf den verbreiteten VAX-Workstations zum Einsatz kam.
1981 erschien mit MS/DOS das für die Marktstellung von Microsoft wichtige Betriebssystem, das von IBM lizenziert unter dem Namen IBM PC DOS zum Einsatz kam. Es war günstig

eingekauft und clever lizenziert, sodass Microsoft an jedem verkauften IBM-Rechner mitverdiente.
1982 erschien das Echtzeitbetriebssystem QUNIX, später QNX, das Verbreitung im Embedded-Umfeld fand, verwendet wurde es außerdem im Automobilbau. QNX verwendet einen Microkernel.
1984 erschien Apples System 1, das auf Lisa OS basierte und viele Ideen des Xerox Alto aufgriff. Aus dem System 1 wurde MacOS.
1985 erschien Microsofts Windows 1.0, das zunächst nur eine Grafische Benutzeroberfläche für MS-DOS war. Kritiker wenden ein, dass Microsoft viele der Ideen von Apple übernommen hatte, wobei Apple selbst intensiv auf Ideen des Xerox PARC zurückgegriffen hatte. Windows wandelte sich mit Windows 95 von einer GUI zu einem integrierten Betriebssystem.
1985 erschien Commodores Amiga OS, das auf den seinerzeit enorm vielseitigen Amiga-Computern eingesetzt wurde. So erlaubte der Amiga 500 Mitte der 80er Jahre bereits Sprachausgaben, leistungsfähige Grafik- und Soundverarbeitung, für die damalige beeindruckende Spiele und vieles mehr.
Das 1987 erschiene VxWorks zählt zu den Echtzeitbetriebssystemen und wurde im Embedded-Umfeld, vor allem in Luft- und Raumfahrt eingesetzt.
Ebenfalls 1987 erscheint das von IBM und anfangs auch Microsoft entwickelte OS/2, das als Nachfolger von PC DOS angesiedelt war. OS/2 erzielte auch einige Aufmerksamkeit, wurde jedoch 2001 eingestellt.
Linux tritt erstmals 1991 in Erscheinung, basierend auf dem von Linus Torvalds entwickelten Open-Source-Kernel, der an Unix und das von Tanenbaum entwickelte MINIX angelehnt war. Daraus gingen eine Vielzahl unterschiedlichster Distributionen hervor, die heute von Embedded-Anwendungen bis hin zu leistungsfähigen Cloud-Servern eingesetzt werden – man schätzt, dass etwa 90% der Cloud-Infrastruktur Linux verwenden.
1997 erschien das für Mobiltelefone entwickelte Symbian, das auf Psions EPOC basierte und vor allem durch die damals verbreiteten Nokia-Geräte bekannt wurde.
2000 erschien IBMs 64bit-System z/OS.
2007 erschien mit Apples iPhone das iPhone OS 1, später iOS.
2008 erschien das quelloffene und auf einem Linuxkernel basierende Android, hinter dessen Entwicklung vor allem Google steht.
Eine genauere Darstellung der verbreiteten Betriebssystem folgt zu einem späteren Zeitpunkt.



Die gegenwärtigen Trends der IT bestimmen natürlich auch die weitere Entwicklung der Betriebssysteme. Die Flut an Daten wird neue Konzepte im Umgang mit Daten erfordern, möglicherweise werden bestehende Dateisysteme sich in Richtung dokumentenorienterter Datenbanken weiterentwickeln. Auch ist zu erwarten, dass KI-Methoden tiefer in das Betriebssystem eingebettet werden, da sie teilweise auch bereits durch spezielle Prozessoren unterstützt werden. Dies kann z.B. auch für neue Bedienkonzepte genutzt werden. Das Cloud Computing wiederum legt nahe, dass verteilte Betriebssysteme, die man bereits seit Jahrzehnten diskutiert, an Bedeutung gewinnen. Besonders spannend wird die Frage, wie sich Betriebssysteme durch das IoT verbreiten – oft werden Komponenten der Automatisierung und Sensorik noch ohne Betriebssystem genutzt, was sich angesichts der wachsenden Leistungsfähigkeit und Anforderungen rasch ändert. Über allem stehen die stark gestiegenen Anforderungen an die Sicherheit der Systeme – letztlich bedingt durch die Vernetzung der Systeme, durch die wachsende Komplexität und durch den Einsatz in sicherheitskritischen Anwendungen (Medizin, Versorgung usw.).

Zusammenfassung

- Das Betriebssystem verwaltet die Ressourcen des Computers und stellt sie Anwendungsprogrammen bereit.
- Das Betriebssystem abstrahiert von der tatsächlichen Hardware.
- Das Betriebssystem verwaltet Hardware, Prozesse, Benutzer und Daten.
- Der Übergang von Stapelverarbeitung zum Dialogbetrieb hat die Betriebssysteme stark beeinflusst, außerdem die stark gestiegene Leistungsfähigkeit der Hardware.
- Betriebssysteme lassen sich nach unterschiedlichen Merkmalen einordnen, z.B. Architektur, Anwendungs-gebiet, Echtzeitfähigkeit, Betriebsarten usw.
- Es gibt zahlreiche Betriebssysteme für unterschiedlichste Anwendungen.

Aufgaben

1. Worin liegen die Unterschiede von Stapelverarbeitung und Time Sharing? Was wurde durch Time Sharing möglich?
2. Grenzen Sie die Begriffe Programm, Prozess und Thread voneinander ab.
3. Was versteht man unter Abstraktion durch das Betriebssystem?
4. Was macht die Pflege des Betriebssystem im Embedded-Umfeld so herausfordernd?
5. Versuchen Sie, Ihnen bekannte Betriebssysteme nach den vorgestellten Kriterien einzuordnen,