OS/2 – Linux – BSD

Moderne Betriebssysteme bestehen aus vier Basiskomponenten: Betriebssystemkern, C-Bibliothek/Userland-Anwendungen, Installationsroutine und Software-Verwaltung. Was unterscheidet hier BSD und Linux?

Der Kernel

Die ersten beiden Komponenten der Einleitung kennt man schon lang. Einen Kernel gibt es auch schon bei MS-DOS in Form von IO.SYS und MSDOS.SYS, bei OS/2 heißt er OS2KRNL).-Das grundlegende Userland sind die Systembefehle, zu denen der Kommandozeilenprozessor und je nach Betriebssystem sogar die grafische Oberfläche gehören kann. Der Kernel läuft immer mit erhöhten Privilegien im Ring 0 (bei OS/2 als einzigem Betriebssystem überhaupt außerdem im Ring 1), das Userland stützt sich auf den Kernel, auf den von einer gemeinsamen Bibliothek aller Anwendungen zugegriffen wird, der sogenannten C-Bibliothek. Das Userland läuft – daher der Name – im Userspace und ohne Kernelprivilegien.

Zum Userland

Wer unter OS/2, Windows oder auch DOS den Kommandozeilenprozessor *cmd* oder *command* aufruft, aber auch wer mit der Workplace Shell von OS/2 oder dem Explorer von Windows arbeitet, führt den wesentlichsten Bestandteil des Userlands aus. Das Userland ist alles, was ein Betriebssystem aus Anwendersicht ausmacht, denn mit dem Kernel, kommt der Anwender ja nicht in Berührung.

Die Installationsroutine

Ein Betriebssystem braucht in der Theorie keine Installationsroutine. Ein Betriebssystem kann auf die »harte Tour« dadurch auf eine Festplatte gebracht werden, indem diese mit externen Tools partitioniert und formatiert wird, dann werden der Kernel und die Userland-Anndungen auf die formatierte Festplatte kopiert und fertig. Nur wird heute niemand mehr ein Betriebssystem so »installieren« wollen. Zu diesem Punkt gehört aber auch noch etwas anderes: die Möglichkeit, Fixes einzuspielen.

Die Softwareverwaltung

Unter OS/2 gibt es keine Softwareverwaltung. Es sind zwar externe Tools vorhanden, die etwas derartiges rudimentär implementieren, aber zur Zeit des Entwurfs dieses Betriebssystems gab es noch gar keinen Anlaß, über Software-Installations- und -Wartungsroutinen nachzudenken. WarpIN ist der Versuch, eine Software-Verwaltung nachträglich aufzusetzen. In den Vorläufern DOS und OS/2 1.x installierte man auf kleinen Festplatten das, was man wirklich benötigte, die Auswahl an Anwendungen war nicht sehr groß und den Anwender selbst für die Aktualisierung verantwortlich. Auch das Problem der Einbrüche über das Netzwerk kannte man nicht.
Eine Softwareverwaltung bedeutet, daß

- alle installierte Anwendungen in einer Datenbank gehalten werden.

- die Möglichkeit besteht, Anwendungen so zu aktualisieren, daß sie nach dem Update noch funktionieren – und auch keine anderen Anwendungen durch Installationen, Updates und Löschaktionen negativ beeinflußt werden.
- nicht benötigte Anwendungen restlos wieder entfernt werden können.

Wie dies durchgeführt wird ist erst einmal egal. Der Microsoft Installer erfüllt die Hauptkriterien auch nur teilweise, denn die Aktualisierung der Anwendungen nimmt er nicht vor. Hier ist das Programm selbst verantwortlich.
Die bisherigen Schilderungen waren allgemeiner Art. Es gilt zu untersuchen, wo die Unterschiede und Gemeinsamkeiten zwischen Linux und BSD sind und wie beide den Anforderungen erfüllen.

Gemeinsamkeiten und Unterschiede der Basis

Für die Definition eines BSD- und eines Linux-Systems muß man sich die jeweiligen Herkünfte ansehen. BSD entstammt der ursprünglichen Unix-Kooperation der AT&T Labs mit der Universität in Berkeley. Aus der »Berkeley Software Distribution« stammen klassische Teile, die heute zum Standardumfang jedes unixartigen Betriebssystems gehören, wie beispielsweise der Editor vi, aber auch Kommandozeilentools und die C-Shell. BSD war von Anfang an ein vollständiges Betriebssystem bestehend aus einem Kernel, einer C-Bibliothek und dem Userland.
Durch die Kommerzialisierung von Unix kam es zu Rechtsstreitigkeiten. Daraufhin wurde das letzte BSD mit Namen 4.3BSD der Universität Berkeley um alles Problematische bereinigt. Das Ergebnis waren nicht lauffähige Betriebssystem-Rumpfquellen mit der Bezeichnung 4.4BSD Lite. Zu diesem Zeitpunkt hatten die Arbeiten an den freien BSD-Betriebssystemen Net/2 und 386BSD längst begonnen. Aus diesen entstanden NetBSD und FreeBSD, die dann die Quellen von 4.4BSD Lite übernahmen und sich seitdem stetig auseinanderentwickelten. Die Unterschiede zwischen den beiden Betriebssystemen sind heute gravierend. Beide sind aber keine Neu-, sondern Weiterentwicklungen und sie stammen von echten Unix-Quellen ab.
Linux ist ganz anders entstanden. »Linux« selbst ist kein Betriebssystem, sondern ein Betriebssystem-Kernel. Er wurde von Linus Thorvalds entworfen und dann um das GNU-Userland ergänzt. Die GNU.org versuchte sich nämlich schon länger am Entwurf eines freien Betriebssystems, kam aber mit seinem sehr anspruchsvollen Projekt GNU/Hurd nicht weiter. Allein das Userland und die C-Bibliothek, die den Namen GLibC trägt, waren voll funktionsfähig und sauber entworfen. Alle Bibliotheken und Programme waren und sind Neuschreibungen und Erweiterungen der Standard-Unix-Befehle unter GNU Public License, Diese Nachprogrammierungen waren ursprünglich für das Userland des GNU-Betriebssystem Hurd gedacht.
Linux und das GNU-Userland ergeben dann schon die beiden wichtigsten Teile eines modernen Betriebssystems. Die Installation dieses Rumpfs kann man jedoch niemandem wirklich zumuten. Aus diesem Problem entstanden die sogenannten Linux-Distributionen, wobei die erste Patrick Volkerdings »Slackware« war.
Eine Distribution ist kein neues eigenständiges Betriebssystem. Sie ergänzt ein vorhandenes Betriebssystem um dort mangelhafte oder fehlende Funktionen (oder um scheinbar Mangelhaftes oder Fehlendes). Die eComStation ist eine klassische Distribution, weil sie die Originalquellen von OS/2 um neue Funktionen wie eine verbesserte Installationsroutine erweitert und die Workplace Shell aufpoliert. Bei einer Distribution bleiben die ursprünglich »fremden« Teile unverändert und werden in das eigene Software-Repository nur als »Externals« eingepflegt. Die Linux-Distributionen ergänzen GNU/Linux um die Installationsroutine und die Softwareverwaltung.
Zur Verdeutlichung: FreeBSD, NetBSD und OpenBSD sind keine Distributionen von »BSD«, sondern eigenständige Betriebssysteme. Das bedeutet, daß sie in getrennten CVS-Repositories

gepflegt und eigenständig weiterentwickelt werden. Eine Linux-Distribution nimmt im Gegensatz dazu aus den Repositories des Linux-Kernels und der GNU-Tools die aktuellen Daten und fügt sie den eigenen Routinen hinzu. Das kann so weit führen, daß PC-LinuxOS eine Subdistribution von Ubuntu ist, das nur eine Subdistribution von Debian GNU/Linux darstellt. Jede der Distributionen basiert auf dem Repository (mitsamt der externen Daten) des jeweiligen Vorgängers im Baum.
Es gibt mehr BSD-Betriebssysteme, aber nur eine ernstzunehmende Distribution von FreeBSD. Von FreeBSD sind abgeleitet DragonflyBSD und MidnightBSD, von NetBSD stammt OpenBSD ab, davon wieder MirOS. Die wichtigste FreeBSD-Distribution ist PC-BSD, das FreeBSD eine neue Installationsroutine und eine weitere Softwareverwaltung hinzufügt, aber auch tief ins Userland von FreeBSD eingreift. Es gab früher noch ein weiteres (kommerzielles) BSD, BSD-OS wurde eingestellt, relevante Teile davon sind in FreeBSD eingeflossen. Einen BSD-Entwurf mit einem Microkernel mit dem Namen xMach gab es auch einmal, Apples Darwin machte dieses Projekt, das nie in Produktionsstatus kam, dessen Quellen aber heute noch auf Sourceforge gehostet sind, obsolet. Sein Logo war ein blau eingefärbter Beastie.
Das Ableiten neuer Betriebssysteme ist bei BSD der übliche Weg, wenn man neue oder andere Funktionen benötigt. Erleichtert wird das durch die Lizenz, die es im Gegensatz zu GPL erlaubt, eigene Weiterentwicklungen unter Closed Source zu stellen, und damit, daß man im Gegensatz zu Linux immer ein komplettes Betriebssystem als Basis für die eigenen Weiterentwicklungen zur Verfügung hat. Das Bilden neuer Betriebssysteme und Einpflegen in ein eigenes Repository trägt den Namen Forking.

Aktualisieren des Basissystems

Wie die Geräteverwaltung von Linux aussieht, dürfte allgemein bekannt sein. Unterschieden wird einerseits zwischen hart in den Kernel einkompilierten Treibern und Modulen, die bei Bedarf geladen werden. Treiber, die nicht benötigt werden, werden gar nicht erst kompiliert. Die wenigsten Anwender kompilieren den Kernel aber selbst. Der Hauptgrund dafür ist, daß es unter Linux nicht üblich ist, die Betriebssystem-Quellen zu installieren. Das liegt daran, daß sie aus verschiedenen Repositories (kernel.org, GNU.org, Distribution) stammen. Manche kommerzielle Distributionen, speziell auch SuSE, hatten in der Vergangenheit sogar den Support abgelehnt, wenn ein eigener Kernel gebaut wurde.
Aktualisiert wird das Basissystem unter Linux deshalb in der Regel über eine Neuinstallation oder das Einspielen der von der Distribution zur Verfügung gestellten Binärdateien. Das ist durchaus mit dem Verhalten beispielsweise unter Windows vergleichbar und für Unix-Systeme sehr untypisch. Es spart dem Anwender Zeit und Festplattenplatz, außerdem braucht er sich mit der Materie auch nicht weiter zu befassen. Schwierig wird es, wenn man tatsächlich Spezialtreiber benötigt, die man sich selbst kompilieren muß.
Die BSD-Betriebssysteme unterscheiden sich in dieser Hinsicht kraß von Linux, in wichtigen Teilen aber wiederum auch voneinander. Hier soll nur ein kurzer Blick auf FreeBSD geworfen werden:
Es gibt mehrere gleichwertige Wege, das Betriebssystem zu aktualisieren. »Aktualisieren« bedeutet hier aber, daß nicht nur der Kernel, sondern immer auch das Basis-Userland neu aufgebaut wird.
Im Gegensatz zu Linux unterscheidet BSD nicht zwischen Anwender- und Entwicklerpaketen und der C-Compiler (übrigens der GNU-Compiler) ist standardmäßig installiert. Es ist möglich und auch sinnvoll, zum Betriebssystem selbst dessen komplette Quelltexte – Kernel und Userland – zu installieren. Das ganze Betriebssystem wird dann sehr einfach aus den Quellen kompiliert und über das installierte System überkopiert. Damit wird auch klar, wie man FreeBSD

aktualisiert: Man nimmt die neuesten Quellen, die man auf verschiedene Weise beziehen kann, kompiliert sie und installiert die neuen Binärdateien. Nach einem Neustart ist das System auf dem aktuellen Stand. Während das Betriebssystem kompiliert wird, kann man normal weiterarbeiten. Es ist auch möglich, ein System aus Binärdateien zu aktualisieren, was aber nicht lohnt. Ein System der regelmäßigen binären Patches wie bei Linux gibt es nicht.
So wie es üblich ist, das System aus den Quellen komplett zu kompilieren, ist es auch Usus, den Kernel anzupassen. Der FreeBSD-Kernel ist wie der von Linux monolithisch und gleichzeitig modular. Im Regelfall (man kann das aber auch anders einstellen) werden ein generischer Kernel mit den wichtigen Devices und zusätzlich alle Treibermodule kompiliert. Es werden auch die Module für solche Devices gebaut, die sich bereits im Kernel befinden. Manche Treiber können nicht fest in den Kernel eingebunden werden, weil deren Lizenz das nicht erlaubt, zum Beispiel die für das ZFS-Dateisystem.
Tatsächlich unbedingt fest eingebunden werden müssen aber nur die Devices, die das System zum Booten braucht. Es gibt auch keine initiale RAM-Disk wie bei Linux. Das Kernel- und Bootsystem ist sehr viel einfacher als bei Linux und auch besser zu durchschauen.

Die Softwareverwaltung

Bei Linux ist für die Softwareverwaltung ausschließlich die Distribution verantwortlich. Slackware nutzt modifizierte TGZ-Dateien, die RedHat-Abkömmlinge und openSUSE RPM-Dateien und Debian und seine Derivate wie Ubuntu nutzen DEB-Dateien. Alle diese Softwareverwaltungen gehen davon aus, daß an zentraler Stelle jemand sitzt und für alle Welt Binärpakete schnürt. Das ist für den Anwender schnell, bequem und leider extrem fehlerbehaftet, weil niemand gewährleisten kann, daß die Pakete miteinander wie vorgesehen harmonieren.
Ein weiterer Nachteil ist, daß´alle Programme mit Defaulteinstellungen kompiliert sind. Wer ein Programm mit besonderen Einstellungen benötigt, steht im Regen und muß es mit dem Selber-Kompilieren und dem GNU-Dreisatz versuchen, oder wechselt am besten zu BSD.
Bei FreeBSD und OpenBSD heißt das System für die Integration von externen Programmen »Ports«, bei NetBSD »Package Source«, es ist aber in allen drei Fällen sehr ähnlich. Entscheidender Unterschied ist, daß FreeBSD sehr viel mehr ausgelagert hat, zum Beispiel auch die gesamte grafische Oberfläche und neuerdings auch die Dokumentation.
Als Beispiel sollen hier die FreeBSD-Ports beschrieben werden, die Beschreibungen gelten weitgehend aber für alle BSD-Betriebssysteme:
Installiert man das Betriebssystem, umfaßt diese Installation den Kernel und das intern gepflegte Userland. Das Projekt selbst überwacht nur diese internen Teile, behebt Fehler und schließt bekanntgewordene Sicherheitslücken. Für alle anderen Programme wird die Verantwortung abgelehnt. BSD ist serverlastig. Zum Standardumfang gehören beispielsweise der FTP-, der HTTP und der NFS-Server aber weniger als bei Linux und bei FreeBSD nicht einmal die grafische Oberfläche.
Wo bekommt man diese Programme her? Einmal für die schnelle Installation ähnlich wie bei Linux in Form von Binärdateien, besser aber ist, man kompiliert sich die Programme selbst. Das ist einem Anfänger nicht zuzumuten, für den Administrator aber genial.
Um Programme unter FreeBSD kompilieren zu können, sollte das System entsprechend leistungsfähig sein. Auf einem Dual-Core-System benötigt OpenOffice.org 3 immer noch einen halben Tag, dann braucht man reichlich freien Festplattenplatz und ausreichend RAM. Moderne PCs haben mit den Voraussetzungen keine Probleme. Dann muß der Clou des Ganzen installiert werden: der Ports-Tree. Das ist ein Verzeichnisbaum mit Tausenden von Unterverzeichnissen. Strukturiert nach Rubriken und mit Meta-Ports zusammengefaßt werden – natürlich unter Berücksichtigung der Abhängigkeiten – spezielle Make-Dateien, mit denen ein

Programm für FreeBSD vorbereitet wird und eventuell notwendige Patches zusammengefaßt. Wer X.org kompilieren will, wechselt beispielsweise nach */usr/ports/x11/xorg* und ruft *make install* auf. Entweder läßt man dann alles durchlaufen oder optimiert den Paketbau nach Wunsch. Mit dem unsicheren GNU-Dreisatz mit *configure/make/make install*, den der Linux-Anwender aufrufen muß, möchte er ein Programm anpassen hat das nur peripher etwas zu tun. Der große Vorteil der BSD-Ports gegenüber allen Paketsystemen bei Linux ist, daß jeder ohne Umstände aus den Originalquelltextpaketen die Programme und Bibliotheken kompilieren und sogar paketieren kann. Solch abenteuerliche speziell angepaßte Quellpakete wie die SRPMs bei SuSE und Red Hat gibt es nicht. Alle Patches gegenüber den Originalpaketen befinden sich im Ports-Tree selbst und werden damit unabhängig von den Originalpaketen gepflegt. Es gibt bei Linux Ähnliches in Form der Paketverwaltung von Gentoo und die Darwin-Ports waren – wenn auch umständlicher – den BSD-Ports nachempfunden.

Ich brauche aber ein Linux-Programm...

Mit der immer größeren Verbreitung von Linux auch in Kreisen, die sich nicht so gut mit Unix auskennen, und durch den Dunstkreis der Firmen, die vom Open Source profitieren wollen, findet man auch Firmen wie SAP mit R3, Adobe mit Flash und PDF, die die Anwendungen nicht als Quelltext, sondern gleich als Binärprogramme verteilen/verkaufen. Dies ist zwar eher unappetitlich, weil diese Firmen meist auch nicht in der Lage sind, qualitativ hochwertige und distributionsunabhängige Versionen zur Verfügung zu stellen, trotzdem gibt es Bedarf. Während ein SAP R/3 sicher nur von einem erfahrenen Administrator installiert werden wird, sieht es bei Endanwender-Software von Corel oder Adobe anders aus. Nehmen wir an, jemand möchte sich die neuesten Ergüsse der Community auf Youtube ansehen. Das geht nicht ohne den Flashplayer 10. Dieser wird für Linux, Windows (ab 2000), MacOS und Solaris bereitgestellt, nicht für BSD. Einen Adobe Reader gibt es auch nicht für BSD.
BSD und Linux sind aber tatsächlich enger miteinander verwandt, wie man als Außenstehender meinen könnte. So wie mit Odin eine Bibliothek für Windows-Anwendungen unter OS/2 bereitsteht und mit Wine eine Emulationsschicht für Windows-Anwendungen unter Linux und BSD und Mono .NET-Anwendungen wenigstens teilweise, gibt es mit dem Linuxulator eine Linux-Emulation bei BSD. Der Begriff »Emulation« ist dafür aber noch weniger gerechtfertigt als bei Wine.
Linux und BSD basieren auf den selben Grundkonzepten, die sich an Unix und POSIX anlehnen. Das bedeutet, daß der Kernel mit dem Userland bei beiden Betriebssystemen auf sehr ähnliche Weise kommuniziert. Was sich von System zu System unterscheidet, sind die Systemaufrufe selbst und die Funktionen und Parameter in der C-Bibliothek. Die LibC von BSD und die GLibC von Linux sind nicht kompatibel. Es ist aber sogar so, daß die eine Version von FreeBSD sich von der nächst älteren unterscheidet. Ein Programm, das gegen die Systembibliothek von FreeBSD 7 linkt, kann ohne Installation von Kompatibilitätsblibliotheken nicht in FreeBSD 8 gestartet werden.
Aus diesem Grund gibt es eine zweistufige Emulation in FreeBSD:

- Im Kernel können alle Linux-Systemaufrufe eingeschaltet werden, damit kann die GLibC installiert werden. Diese Funktion ist im Default-Kernel bis FreeBSD 7 tatsächlich schon freigeschaltet, ab Version 8 muß ein Modul (alias Treiber) geladen werden. Bei Bedarf werden die beiden wichtigsten Pseudodateisysteme sys und proc von Linux installiert.
- Es werden die GLibC und das grundlegende Userland bis inklusive der Shell einer definierten Linux-Distribution (bei FreeBSD Fedora 10, bei NetBSD OpenSUSE, bei OpenBSD Fedora 6) installiert.

Die Linux-Emulation selbst wird in einen eigenen Verzeichnisbaum getrennt von FreeBSD in-

stalliert und alle Linux-Anwendungen mappen auf diese Emulation. Damit funktionieren Programme wie Adobe Reader, Flash, Opera für Linux und selbst SAP R/3. Es laufen aber nicht alle Anwendungen. Wer die GLibC umgeht und statisch gelinkt ist (zum Beispiel Free Pascal für Linux), ist unter BSD nicht lauffähig. Auch Programme wie Kylix und Corel Photopaint, die gegen die Winelib gelinkt sind, stürzen ab. Solche Programme sind aber auch immer nur auf bestimmten Linux-Distributionen funktionsfähig.

Und die Schwächen?

Diese Beschreibungen sollten nur BSD und Linux grob gegeneinander abgrenzen. Es gibt aber vieles, was man als OS/2-Anwender bei beiden Systemen eher als störend empfinden wird. Einige Punkte sollen hier noch diskutiert werden:
Wer unter OS/2 oder Windows ein Programm installiert, erhält entweder ein Installationsarchiv oder eine Zip-Datei. Mit wenigen unrühmlichen Ausnahmen wie beispielsweise der Forderung, daß EMX installiert sein muß, wird man ein ordentlich zusammengestelltes und packetiertes Programm nach der Installation ausführen können. Auch wenn OS/2 keine eingebaute Softwareverwaltung hat und man als Anwender selbst für Updates, den Bezug und so weiter sorgen muß, ist das doch recht komfortabel.
Linux und BSD kommen aus dem Serverbereich. Hier wird mit anderen Prämissen gearbeitet. Eine Software besteht hier aus voneinander abhängigen Paketen, möchte ich OpenOffice.org einspielen, kann das durchaus schiefgehen, weil irgend eine benötigte Bibliothek mit einer bereits installierten kollidiert. Ein krasses Negativbeispiel sind die X-Bildschirmschoner unter FreeBSD, von denen es eine generische, eine KDE- und eine Gnome-Version gibt. Sie sind zueinander inkompatibel und die Installationen für die jeweilige Oberfläche zwingend. Solche Beispiele findet man zuhauf. Die einzige Lösung unter allen freien Betriebssystemen bietet die FreeBSD-Distribution PC-BSD mit dem PBI-System, einer Abkürzung von *PC BSD Installer* oder *Push-Button Installer*, bei dem alle Anwendungen immer alle benötigten Bibliotheken enthalten. Leider wird dieses innovative System bei PC-BSD selbst nicht konsequent verfolgt. Im Gegensatz zu Ubuntu/Xubuntu/Kubuntu wird bei PC-BSD die KDE-4-Oberfläche immer installiert, selbst wenn man sie nicht will.
Einer der sicher größten Problemfälle aller unixartigen Betriebssysteme ist die grafische Oberfläche. Im Gegensatz zu den Desktop-Betriebssystemen OS/2 und Windows wird der Bildschirm nicht direkt angesteuert. Ein ausgeklügeltes Client/Server-System trennt die Anwendungen und die Ausgabe streng voneinander. Der Vorteil des Systems ist, daß Thin Clients problemlos zu installieren sind und die Anzeige im Netzwerk sehr leicht exportiert werden kann. Erkauft wird das aber mit einem zusätzlichen Netzwerkverkehr und Geschwindigkeitseinbrüchen gegenüber beispielsweise Windows. Apple hat das beim Entwurf von OS X erkannt und X über Bord geworfen. Es gibt auch experimentelle Ansätze, ein X aufzubauen, das ohne die Netzwerkkommunikation auskommt. Sie sind aber nicht weit gediehen.
Ein weiteres Problem bei X ist die Trennung von X-Server, Window Manager und Anwendungen, außerdem, daß X immer extern aufgesetzt ist und auch so wirkt, bei BSD noch viel stärker als bei Linux. Man kann den Textmodus zwar ausschalten, alle Reparaturarbeiten finden aber im Textmodus statt.
Weiterhin wurde mit X.org 7 das Konfigurationsprogramm abgeschafft, Endanwender haben bei FreeBSD – abgesehen von PC-BSD – fast keine Chance mehr, X auf Anhieb zum Laufen zu bringen. Nur mit NVidia-Karten sieht das ein bißchen besser aus, weil es hier außer den proprietären Bildschirmtreibern auch ein proprietäres Konfigurationstool gibt.
Dies sind nur einige der Probleme, auf die man stößt, wenn man sich mit Linux und BSD auf dem Desktop beschäftigt. Wer nur mit dem PC E-Mails bearbeitet, ein klein bißchen Textverar-

beitung macht und Flash-Videos ansieht, kommt mit beiden Betriebssystemen sehr weit. Nur stellt sich die Frage, warum man solche leistungsfähigen Systembetriebssysteme einsetzen will, wenn man ohnehin nicht ernsthaft mit dem Computer arbeiten will.
Interessant an BSD ist aber – abgesehen von den geschilderten eher technischen Details – die Vielzahl der möglichen Desktops. Hier findet sich alles, was man sich nur vorstellen kann. Außer den typischen modernistischen Desktop-Monstern KDE 4 und Gnome gibt es auch einfacheres und schlankeres wie XFCE 4 und KDE 3, aber auch eher spleeniges wie Enlightenment, minimalistisches wie Blackbox und objektorientiertes wie das in Objective X geschriebene GNUStep/Étoilé.

Nachsatz

Es gibt natürlich noch viel mehr technische und auch philosophische Unterschiede zwischen Linux und BSD. Für den Desktop-Anwender sind sie nicht so wichtig. Daß man GNU und BSD sogar kombinieren kann, zeigt außerdem Debian GNU/kFreeBSD, ein voll funktionsfähiges Debian, das nicht auf dem Linux-, sondern dem FreeBSD-Kernel beruht und dabei das GNU Userland nutzt. Debian GNU/kFreeBSD zeigt aber die gleichen Probleme wie Linux: Das Update ist schwierig, weil die Quellen hier wieder aus verschiedenen Quellen stammen.
Würde ich ein eigenes Desktop-Betriebssystem entwerfen müssen, würde ich als Basis FreeBSD wählen und mir dann überlegen, wie ich den grafischen Desktop aufbaue – sicherlich nicht auf dem veralteten und zu unixlastigen Xorg 7. Apple hat in dieser Hinsicht erfolgreich vorgemacht, wie man vorgehen muß. Ich würde aber von FreeBSD keinen Fork ziehen, sondern eine Distribution bilden, wie es iXsystems mit PC-BSD tut. Damit spart man sich die Sicherheits-Fixes, die Updates und damit ungeheuer viel Ärger.
Ganz auf X würde ich – wie Apple – aber nicht verzichten, damit X-Anwendungen wenigstens genutzt werden können – vielleicht ist der Ansatz von X für OS/2 nicht verkehrt. Einen weiteren »YAXD« (Yet Another X Desktop) braucht die Welt nicht.

Kernelwirrungen

Es betrifft den Vergleich zwischen BSD und Linux nicht, aber sollte nicht unterschlagen werden: Der Typ des Kernels.
Es gibt zwei Basisdefinitionen von Betriebssystem-Kernen: den sogenannten monolithischen und den Microkernel. Für den Anwender hat es keine Bedeutung, ob ein Betriebssystem einen monolithischen oder einen Microkernel hat. Typische aktuelle Betriebssysteme mit Microkernel sind

- Windows NT bis zur aktuellen Version 6.1
- Hurd
- Mac OS X
- Minix

Die Kernel von MacOS X und Windows NT basieren auf den Quellen des Mach-Kernels, der ursprünglich als Forschungsprojekt an der Carnegie Mellon University in Pittsburgh entwickelt wurde. Speziell der Hurd-Kernel ist technisch interessant, weil das Basissystem sehr klein ist. Dieser abgeleitete GNU-Mach-Kernel befindet bis heute aber nicht in einem produktionsfähigen Status und das Betriebssystem befindet sich seit 1997 in der Versionsstufe 0.2. Seitdem wurden keine neuen Versionen veröffentlicht, obwohl an Hurd gearbeitet wird.
Linux, alle BSD- und sonstigen Unix-Derivate arbeiten mit monolithischen Kerneln. Linus Thorvalds wurde wegen des monolithischen Aufbaus seines Kernelentwurfs von Andrew S. Tanenbaum, dem Entwickler von Minix und eifrigen Microkernel-Verfechter, heftig angegriffen.

Der Unterschied zwischen den beiden Kernel-Paradigmen ist schnell erklärt:
In einem monolithischen Kernel laufen alle Kernelprozesse und alle Treiber in einem gemeinsamen privilegierten Adreßraum. Die Kommunikation zwischen den einzelnen Bereichen kann direkt erfolgen, weil im gemeinsamen Adreßraum der Datenaustausch problemlos und schnell vonstatten geht. Der Nachteil: stürzt ein schlecht programmierter Treiber ab, reißt er alles andere im Adreßraum und damit auch den ganzen Kernel mit in den Abgrund. Weiter kommt hinzu, daß amoklaufende Treiber auch auf Speicherbereiche schreibend zugreifen können, die ihnen nicht gehören.
Bei einem Microkernel ist das in der Theorie alles anders: Der Kernel selbst ist sehr klein (daher der Name). Er selbst ist für gar nichts zuständig außer dem Einrichten und der anschließenden Kommunikation der Treiber untereinander, er ist also nur für das Speicher- und Prozeßmanagement verantwortlich. Die Treiber selbst laufen in einem oder mehreren sogenannten Servern. Die Treiber erhalten nur Zugriff auf das, was sie selbst angeht. Stürzt ein Treiber ab, wird er (in der Theorie) einfach neu gestartet und nichts geschieht. Der Nachteil dieses Konzepts ist die mangelhafte Performance, da alle Kommunikation über festgelegte und streng gegeneinander abgeschottete Pipes läuft. Ob es wirklich besser ist, alles gegeneinander abzukapseln und eine hochkomplizierte Interprozeßkommunikation zu programmieren, darüber kann man streiten. Benchmark mit MkLinux und L4Linux, Linux-Betriebssystementwürfen mit Mach und L4-Microkernel anstelle des eigentlichen Linux-Kernels konnten jedenfalls nicht überzeugen. Sie waren schlicht zu langsam.
Aus diesem Grund beruhen die kommerziellen Betriebssystemen mit Microkernel auf einem Mischkonzept: Alles, was an wesentlichen und performancerelevanten Operationen durchgeführt werden muß, kommt in den Kernel selbst, alle anderen Treiber bleiben außen vor. Wie gut dieses Konzept ist, sieht man am BSOF von Windows, der die Notbremse bei Fehlern im Kernel ist und der Schadensvermeidung dient.
Welchem der Typen der OS/2-Kernel angehört, war nicht herauszufinden. Der eigentliche Kernel lädt bei OS/2 die Basistreiber für Tastatur, Bildschirm, Uhr und Drucker und danach die Basis-DLLs. Das spricht für ein Mischsystem. Entscheidend ist aber nicht, daß der Kernel Treiber lädt, sondern wohin er sie lädt und das konnte nicht ermittelt werden.

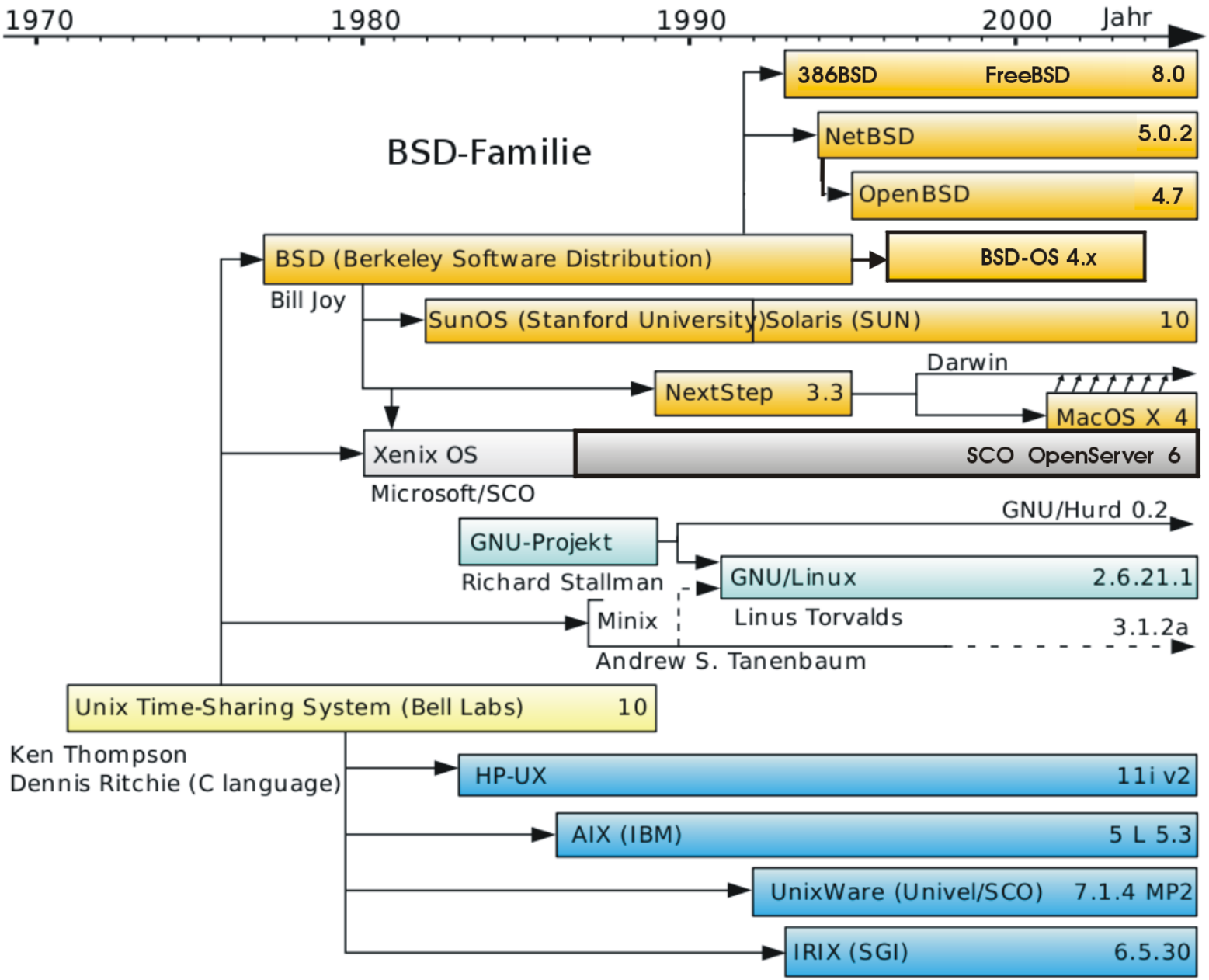
(Jörg Braun)

1	Ubuntu	2410
2	Fedora	1704
3	Mint	1588
4	openSUSE	1375
5	Mandriva	1172
6	Debian	1083
7	PCLinuxOS	1017
8	Sabayon	872
9	Arch	822
10	MEPIS	761
11	Puppy	712
12	FreeBSD	648
13	Slackware	585
14	Ultimate	572
15	CentOS	543
16	Tiny Core	540
17	Gentoo	489
18	PC-BSD	439
19	Kubuntu	427
20	KNOPPIX	405
21	CrunchBang	397
22	PC/OS	387
23	Zenwalk	345
24	sidux	339
25	TinyMe	331
26	Vector	329
27	Elive	328
28	Dreamlinux	317
29	BackTrack	314
30	Red Hat	295
31	gOS	294
32	ClearOS	290
33	Xubuntu	288
34	Chakra	280
35	Lubuntu	273
36	OpenSolaris	268
37	Ubuntu Stud.	267
38	Macpup	263

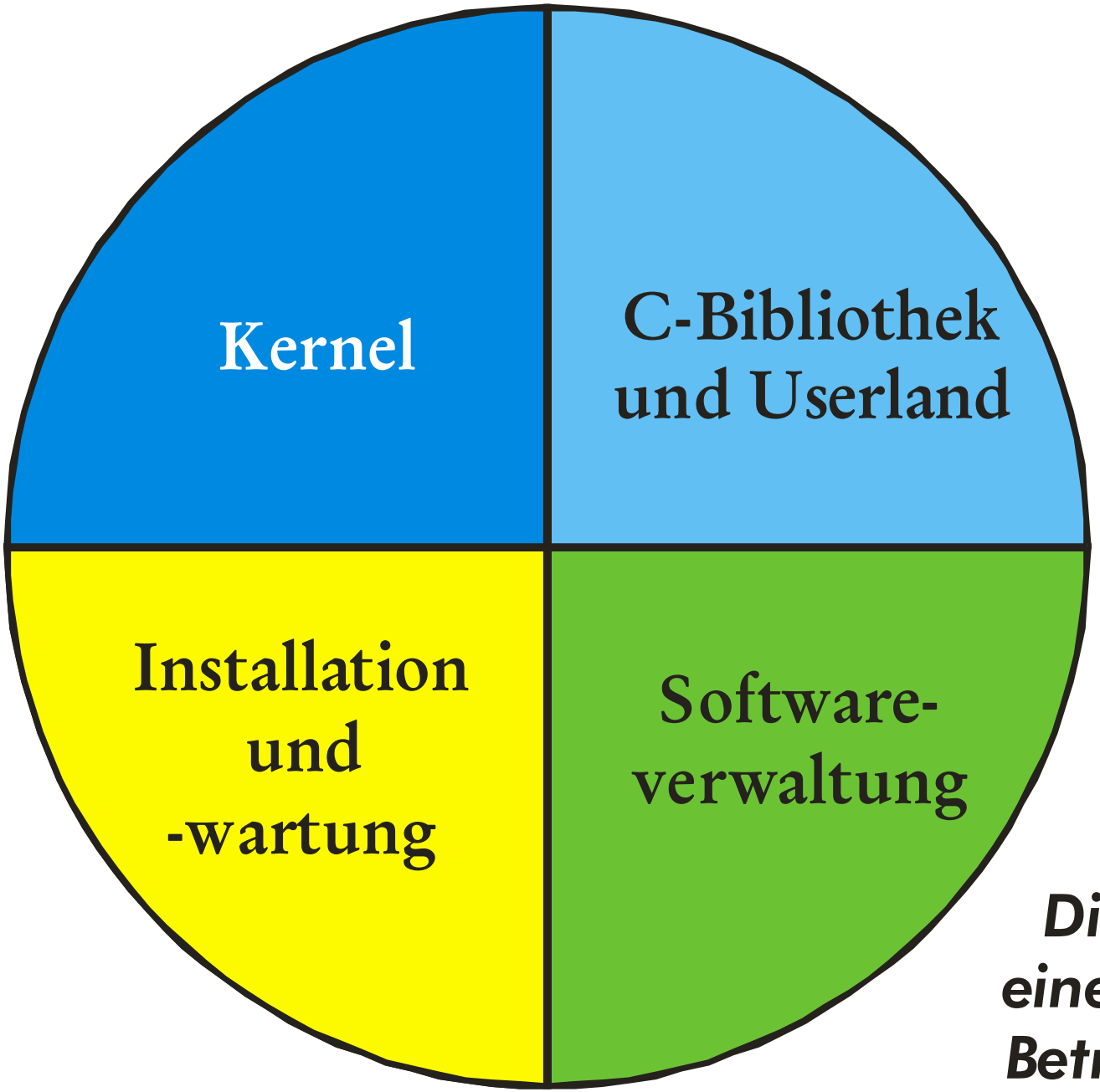
39	SliTaz	249
40	EasyPeasy	241
41	Pardus	240
42	Frugalware	236
43	Unity	231
44	SystemResc.	223
45	Super OS	220
46	Slax	213
47	Parted Magic	209
48	Absolute	204
49	Mythbuntu	199
50	Moblin	197
51	blackPanther	182
52	Nexenta OS	181
53	moonOS	178
54	Clonezilla	174
55	PureOS	171
56	Scientific	169
57	Yoper	164
58	MOPSLinux	159
59	GeeXboX	156
60	DragonFly	153
61	64 Studio	152
62	Linux XP	151
63	Zorin	149
64	Salix	149
65	Xandros	146
66	wattOS	145
67	OpenBSD	143
68	OpenGEU	138
69	Calculate	138
70	ZevenOS	137
71	U-lite	136
72	Ubuntu Chr.	135
73	CDlinux	135
74	LFS	134
75	FreeNAS	133
76	Yellow Dog	130

77	Element	130
78	Epidemic	128
79	Wolvix	127
80	linuX-gamers	124
81	Debris	122
82	eBox	121
83	Solaris	118
84	GoblinX	118
85	VortexBox	117
86	GParted	116
87	Igelle	115
88	ALT	115
89	Hymera	114
90	Novell SLE	113
91	Parsix	112
92	Musix	112
93	KahelOS	111
94	Turbolinux	109
95	PUD	108
96	LinuxConsole	108
97	Linpus	108
98	NuTyX	107
99	NetBSD	107
100	DEFT	107

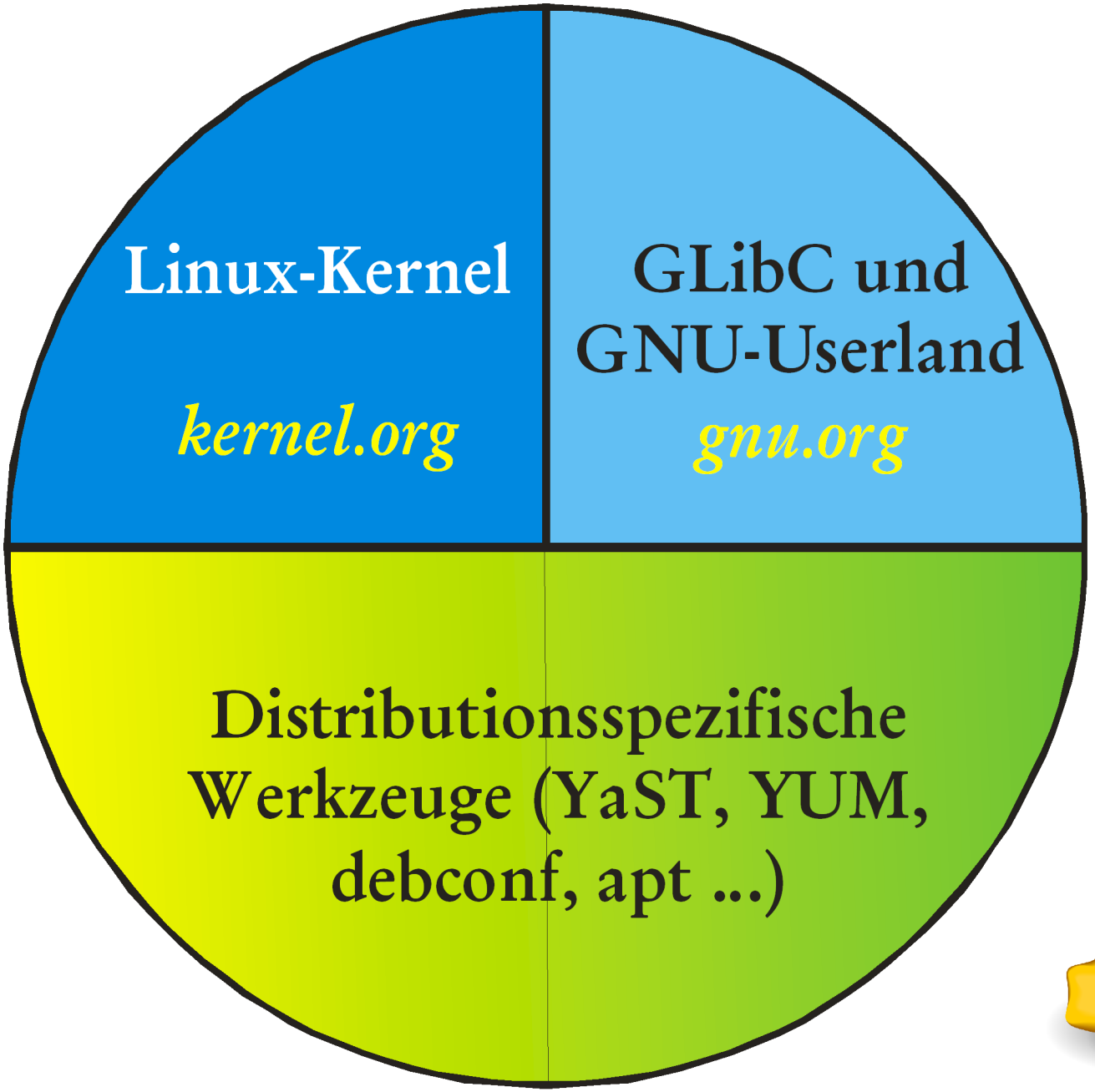
Ränge der Distributionen und Betriebssysteme auf distrowatch.org Stand 30.04.2010

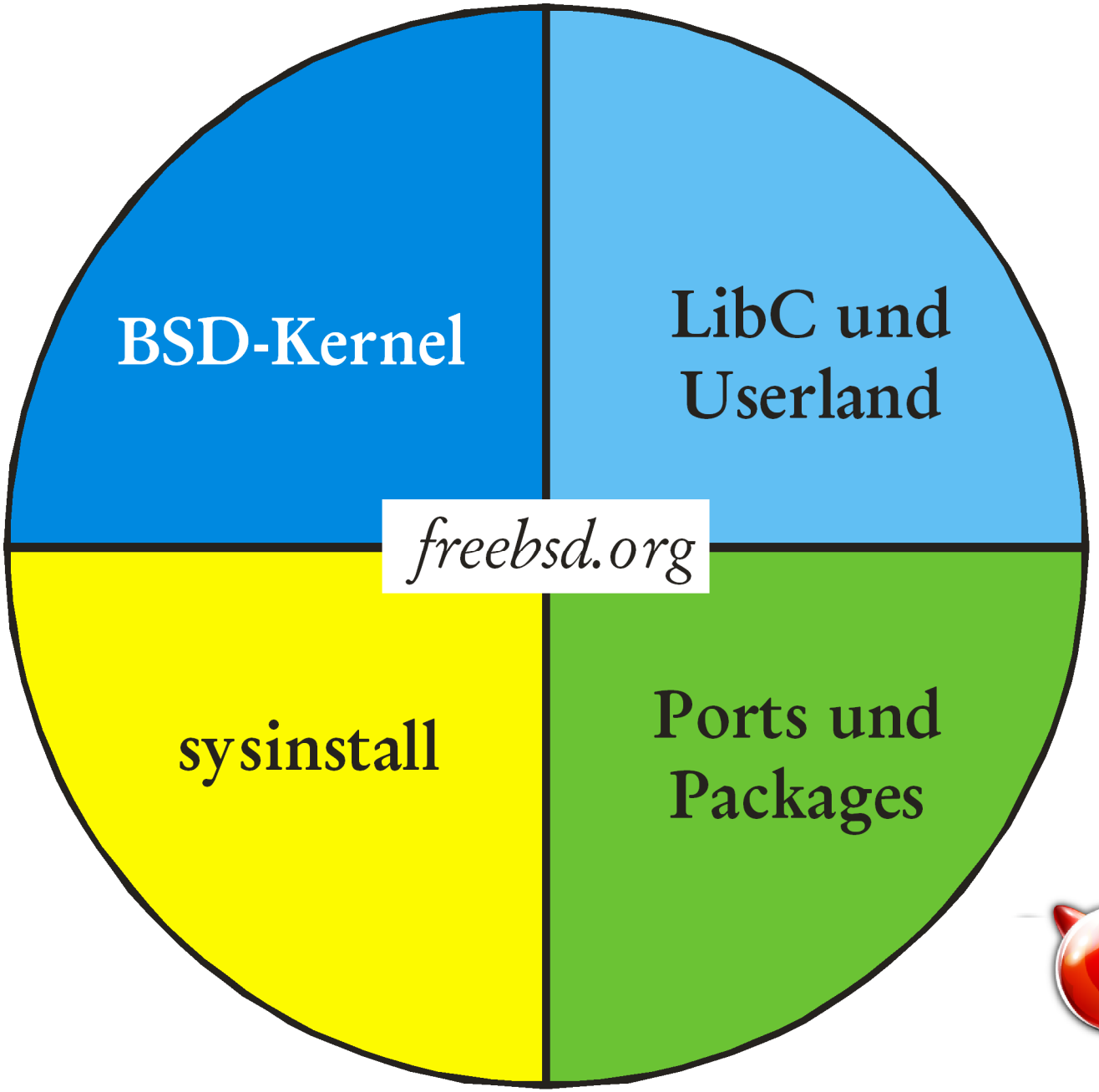


Leicht angepaßt/korrigiert nach: http://de.wikipedia.org/w/index.php?title=Datei:Unix_History_DE.svg

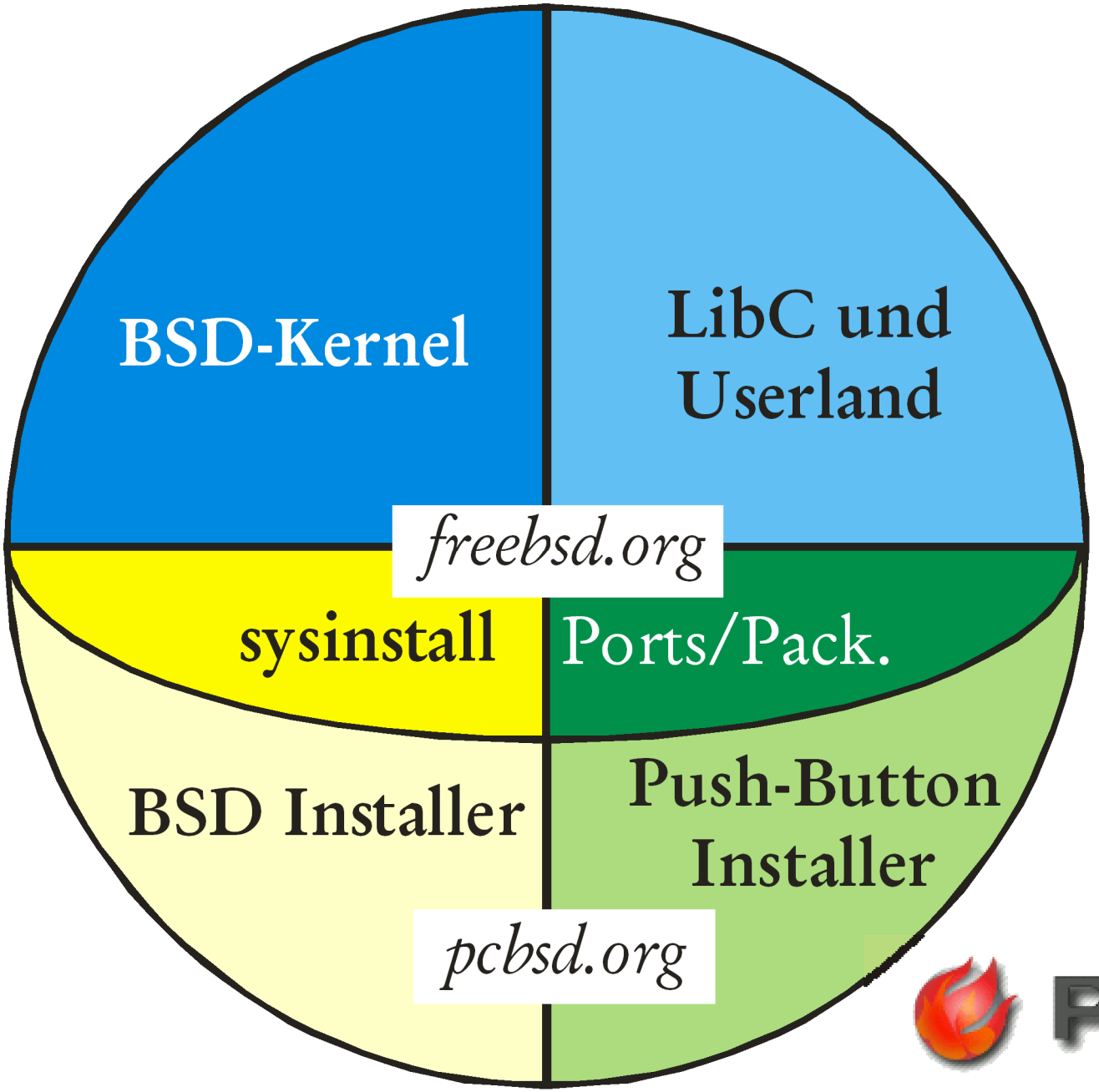


Die vier Teile eines modernen Betriebssystems


Linux-Kernel
kernel.org
GLibC und
GNU-Userland
gnu.org
Distributionsspezifische
Werkzeuge (YaST, YUM,
debconf, apt ...)


BSD-Kernel
LibC und Userland
freebsd.org
sysinstall
Ports und Packages

FreeBSD®


BSD-Kernel
LibC und Userland
freebsd.org
sysinstall
Ports/Pack.
BSD Installer
Push-Button Installer
pcbsd.org

PC-BSD

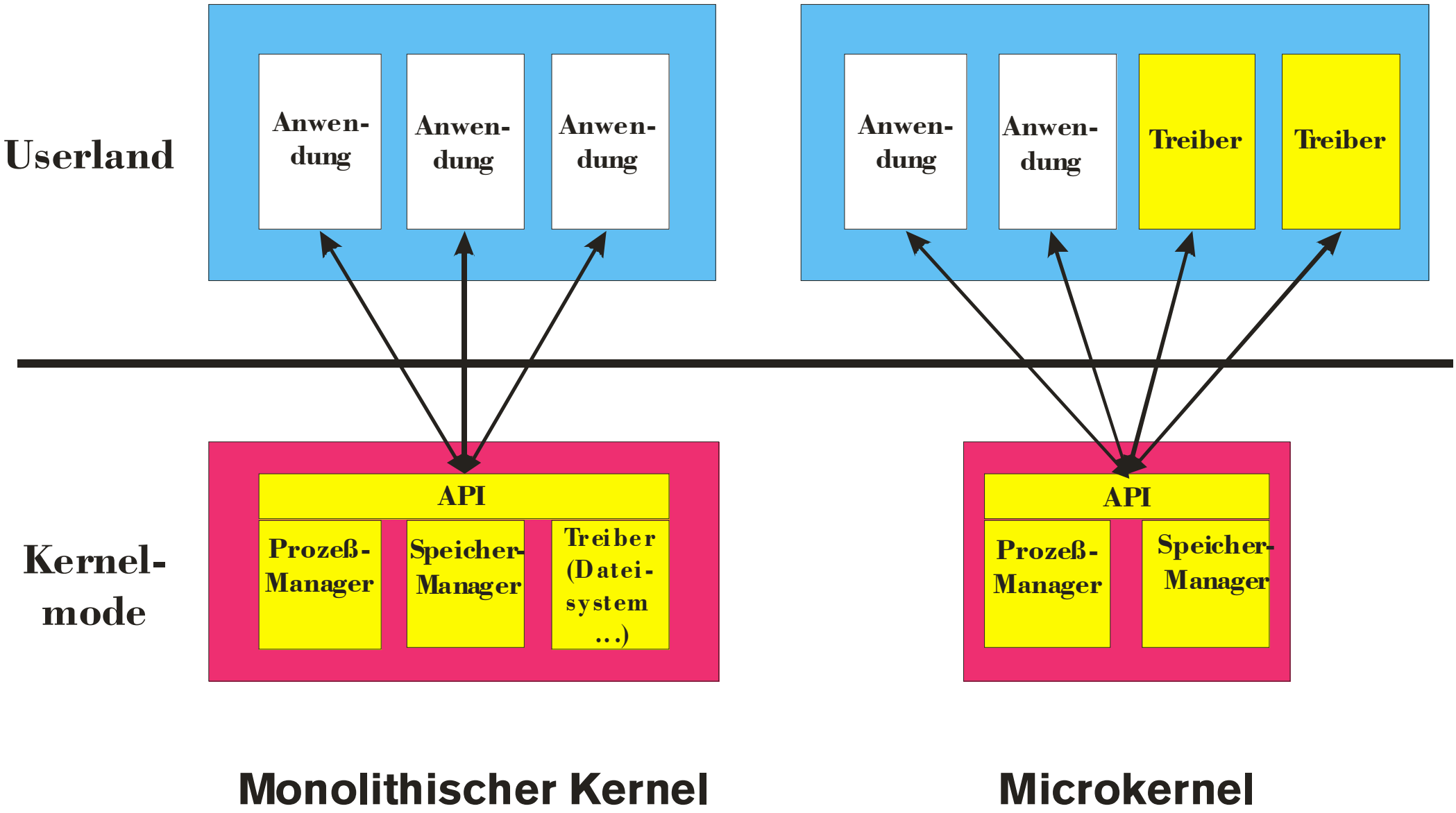
Programmausführung in FreeBSD

- FreeBSD
  X.org 7, OpenOffice.org, Lazarus, Gimp, Inkscape, alle Mozilla-Anwendungen, JDK 1.6; insgesamt über 10.000 Pakete)
- Linux (Fedora-10-Emulation)
  Adobe Reader, Flash 10, Opera, Firefox, Seamonkey etc.
- System V Rel. 4 (Solaris-2.6-CD wird benötigt)
  nur rudimentär, aktuell ist SunOS 5.10, weiter ausgebaut bei OpenBSD (z.B. für Java).
- Win32 (mit Wine)
  Nur in FreeBSD/i386 verfügbar, Wintricks und ähnliches muß manuell installiert werden.
- .NET mit Mono
  identischer Status wie bei Linux.
- Virtualisierung mit VirtualBox/Qemu, (XEN)
  Aktuelle VirtualBox OSE ohne USB, Qemu mit Kernelmodul und Usermode-Emulation.

Vom Server zum Desktop?

- Einfaches, gut wartbares Basisbetriebssystem
- Sehr gut skalierbarer und konfigurierbarer Kernel
- Hohe Stabilität/gute Performance des IP-Stacks
- Dateisysteme UFS/UFS2 und ZFS, verschlüsselt und mit Journaling auch in Kombination
- Virtualisierungsfähig als Server und Client

- Bekannte ACPI-Probleme bei Notebooks
- X: Nur proprietäre Treiber für Nvidia-Grafikk.
- Kein Wine für den x86_64
- Laufwerksaustausch mit Linux schwierig.


Userland
Anwen-
dung
Anwen-
dung
Anwen-
dung
Anwen-
dung
Anwen-
dung
Treiber
Treiber
Kernel-
mode
API
Prozeß-
Manager
Speicher-
Manager
Treiber
(Datei-
system
...)
API
Prozeß-
Manager
Speicher-
Manager
Monolithischer Kernel
Microkernel