

Vorlesung

Wann/Wo:

- 13:30-16:45
- EBS DM3 DM5 MI3 I032c
- Klausur: 4.7.05 (90 Minuten)

Inhalte:

- Was ist ein Betriebssystem, Historie, Grundlagen
- Prozesse/Threads
- Deadlocks
- Scheduling
- Speicherverwaltung
- Dateisysteme
- Kommunikation
- Userinterface
- Linux-Praktikum: Shell, Prozessverwaltung, Utilities, Dienste

Literatur:

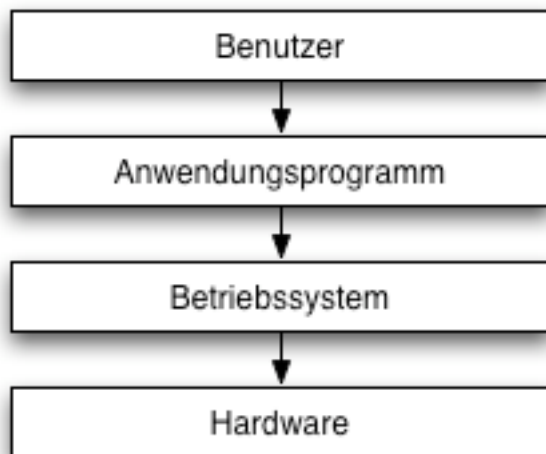
- Andrew S. Tanenbaum, „Moderne Betriebssysteme“
- Brause: „Betriebssysteme“
- Script / Mitschreiben
<http://www.herwig-henseler.de/lehre/betriebssysteme>

1 Einführung

Was ist ein Betriebssystem?

Allgemein verständlich:

Das BS stellt eine Brücke dar zwischen der Hardware und den Anwendungsprogrammen



Tanenbaum

2 Verschiedene Sichtweisen:

- 1.) *BS als erweiterte Maschine* (Architektursicht, Ebenendenken der Informatiker, Komplexitätsreduktion, Top-Down-Sicht, Hardwareabstraktion, API (Application Programmer Interface)). Beispiel: Programm verwendet Dateien. BS managen die Abbildung auf verschiedene physikalische Datenträger (Floppy, Platte, Memorystick, CD) sowie die Kommunikation mit den unterschiedlichen Geräten.
Idee: BS Als „Unterbau“
- 2.) *BS als Ressourcenmanager* (geordnete, kontrollierte und optimierende Zuteilung der Ressourcen für (ggfls. gleichzeitig ablaufende) Anwendungen und Anwender, Bottom-Up-Sicht). Räumt nach Beendigung eines Programmes alle Ressourcen auf.
Idee: Das BS als „Oberaufseher“ von Anwendern und Anwendungsprogramme

Definition: Betriebsmittel/Ressource:

Als Betriebsmittel bezeichnen wir die Prozessoren (Rechnerkerne), die Speicher (Haupt- und Hintergrundspeicher), Ein-Ausgabe-Geräte, Prozesse, Dateien, Übersetzer, Dienstprogramme, Benutzerprogramme usw.

Duden Informatik

Zusammenfassende Bezeichnung für alle Programme, die die Ausführung der Benutzerprogramme, die Verteilung der Betriebsmittel auf die einzelnen Benutzerprogramme und die Aufrechterhaltung der Betriebsart überwachen

DIN 44300

Die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die *Grundlage der möglichen Betriebsarten* des digitalen Rechensystems bilden und insbesondere die *Abwicklung von Programmen steuern* und überwachen.

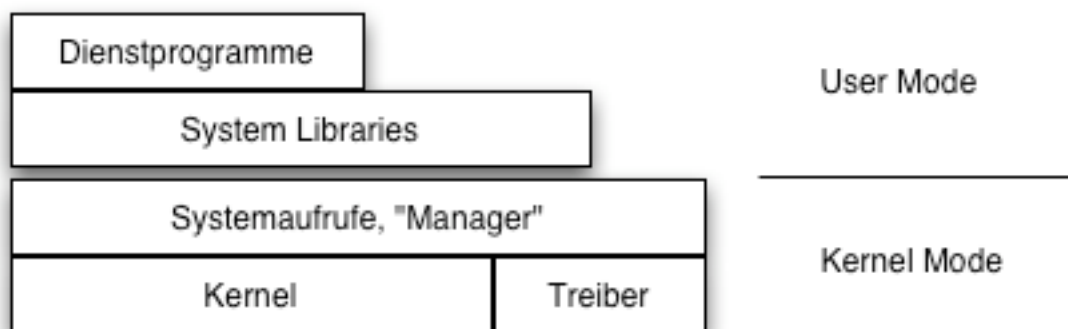


Abb.: Komponenten eines BS

Was umfasst ein BS?

BS im engeren Sinne:

Der Kernel (plus Gerätetreiber): Hardwareabstraktionsschicht

- Ressourcenmanagement (Speicher, Prozessor, Ein-/Ausgabe)
- Hardware/Gerätmanagement (Treiber)
- Bootmanagement

BS im weiteren Sinne:

Kernel plus Werkzeuge und Benutzungsoberfläche:

- Dateibearbeitungswerkzeuge, Editoren
- Compiler, Skriptsprachen
- Shell, Kommandozeileninterface
- Graphische Oberfläche
- Benutzungsoberfläche
- Administrationswerkzeuge
- Reparatur-, Backup-, Netz-, Dienste-, Sicherheitsmanagement
- Filebrowser

- Webbrowser (behauptet Microsoft)
- Serverdienste (Webserver, Mailserver, Fileserver, ...)

Nicht mehr zum BS gehören:

- Anwendungsprogramme (Textverarbeitung, Spreadsheet, DB, Spiele)

Keine strenge Abgrenzung, „Grauzone“

Warum sollten sie sich mit BS beschäftigen?

- Auswahl und Installation eines BS für einen Anwendungsfall
- Kenntnis über Verhaltensstruktur von Speicher/Prozessen/Geräten
- Nutzung der API zur Programmierung
- Administration von Rechnern/Diensten
- Anpassung und Nutzung der Werkzeuge eines BS zur effektiveren Arbeit bzw. zur Anpassung an Anwendungsfälle
- Modifikation/Erweiterung eines BS (Treiber)
- Probleme im BS sind allgemeines Werkzeug für Informatiker

Historie

Historie: 1945-1955

Kein Betriebssystem. Raumgroße Rechner. Maschinensprache, Steckverbindungen, Schalter. Unzuverlässige Rechner

Historie: 1955-1965

Zuverlässigkeit stieg. Unterscheidung zwischen Entwicklern, Operateuren und Bedienern. Mainframes: Rechner in eigenen Räumen. Bediener mittels Lochkarte/streifen und Drucker. Stapelbetrieb/Batchbetrieb („Job“). Keine Interaktion.

Historie: 1965-1980

ICs, IBM 360, Multiprogramming: Mehrere Programme. Während eines wartete, konnte das andere rechnen. Timesharing-Systeme. Bildschirmarbeitsplätze. Kurze Antwortzeiten. Rechner mit dutzenden oder hunderten von Terminals. MULTICS, UNIX.

Historie: 1980 bis heute

Mikrocomputer, Minicomputer. CP/M, Apple DOS, MS-DOS. GUI. Windows, LINUX, BSD.

Exkurs: 1975-2001: Die Ära der Homecomputer. Von Altair 8800 bis Windows ME/MacOS9

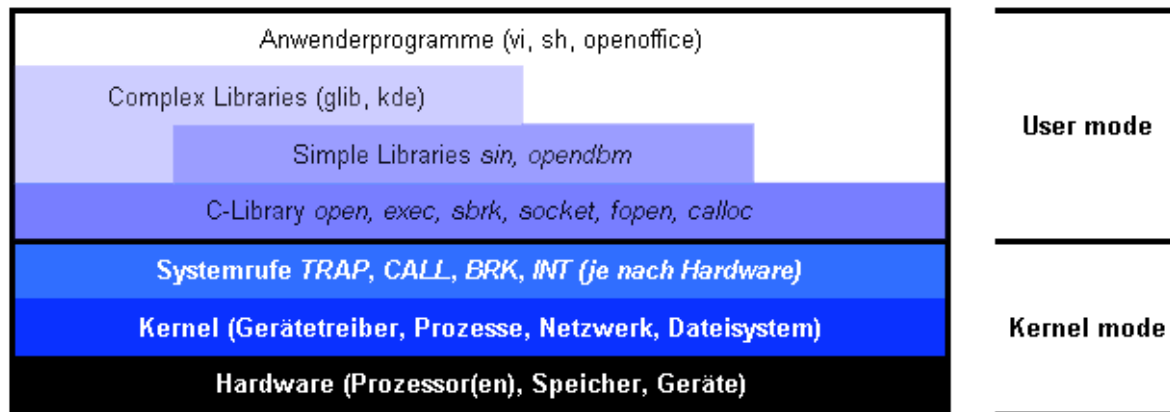
Übersicht und Einordnung verschiedener BS

	Open Source	Mainframe	DBServer	App, Webserv.	Workstation	Desktop/“PC“	Embedded	Echtzeit	Historisch	
z/OS [IBM] (VM, MVS, ...)		X	X	X						Klassisches Mainframe-BS
OS/400 [IBM]			X	X						Midrange-Server
BS2000 [Siemens]		X	X						X	
VMS [DEC] / OpenVMS	(x)	X	(x)	X					(x)	Exzellente Stabilität
UNIX (AIX, Solaris, HP-UX, IRIX, SCO, ...)	(x)	X	X	X	X					Familie der UNIX-Systeme
Linux	X	X	X	X		X	(x)			OSS, GPL
BSD (FreeBSD, NetBSD, OpenBSD, Darwin)	X			X		X	X			OSS, BSD-Lizenz. Stabilität, Sicherheit
Mac OS 1-9						X			X	Erstes BS mit GUI
Mac OS X, NeXTSTEP	(x)			X		X				Modernste GUI
MS-DOS, Netware				X		X			X	
Windows 3.X, 9x, ME						X			X	Kein BS
Windows NT, 2000/3, XP			X	X		X				Stabile Windows-Linie
VxWorks, QNX							X	X		Spezialanwendungen
Symbian, Windows CE, PalmOS, CiscoOS, EPOC							X			Für Handhelds
ITRON								X		In vielen Gebrauchsgegenständen
eComstation [IBM] (OS/2)						X	X		(x)	Ehemaliger DOS-Nachfolger
Zeta (BeOS)						X				Multimedia-OS

Weitere Eigenschaften:

- Portabilität
- Kosten
- Sicherheit
- Vorhandene Bibliotheken
- Mitgelieferte Software
- Verfügbare Software
- Mehrprozessorfähigkeit
- Kernelarchitektur
- Verbreitung (am weitesten Verbreitet: ITRON(!))
- Mehrbenutzerbetrieb (Multiuser), Mehrprogrammbetrieb (Multitasking)

UNIX



Was ist Unix? Ein eingetragenes Warenzeichen der Open Group.

- 1969: Bell Laboratories: Privates Projekt: „Simple and Elegant, Written in High-Level-Language“. Gegenstück zu MULTICS.
- 1974-1979: Universitäres Experimentiersystem (Berkeley University). Source code war verfügbar
- 1972: Kernighan, Ritchie. Entwicklung von „C“. Unabhängig von konkretem Prozessor (!)
- 1979: „System V“ (AT&T) und „BSD Unix“ (Berkeley).
- 1980er: Viele System V-Varianten, kommerzielle Nutzung.
- Notiz am Rande: Microsoft verkaufte XENIX
- 1980er: BSD: Wenige Varianten, technologischer Führer
- 1980er: Reifes System, Netzwerkfähig, Ethernet, TCP/IP. „Workstations“. GUI.

- 1990er: Zusammenführung von System V und BSD (POSIX) 1991: Linux als ein-Mann-Projekt
- 1992: Berkeley gibt BSD auf und damit frei. Rechtsstreit mit AT&T
- 1993: BSD-Varianten
- 1993: AT&T verkauft Unix. Aufspaltung in Coderechte (SCO) und Namensrechte (Open Group)

UNIX, Unixoid, Unix-artig. Alle irgendwie „ähnlich“, aber doch anders.

Heutige Varianten:

Kommerziell, auf System V beruhend („UNIX“): Solaris, AIX, HP UX, IRIX, SCO ...

Open Source, auf BSD beruhend: FreeBSD (Webserver), OpenBSD (Sicherheit), NetBSD (Portabilität), Darwin (MacOSX)

Open Source, „dazwischen“: Linux

BSDs sind komplette Betriebssysteme mit unterschiedlichen Entwicklungszielen.

Linux ist nur Kernel+Treiber. Distributionen schnüren ein Komplettpaket:

SuSE, Mandrake, Debian, RedHat (Novell), Knoppix, Gentoo, Ubuntu, etc.

(z.B. www.linuxiso.org)

Exkurs: Open Source, GPL, BSD

Designphilosophien:

- Schreibe kleine Programme, die eine Aufgabe tun und diese besonders gut
- Schreibe Programme, die miteinander arbeiten können
- Speichere Daten in Textdateien
- „Alle Ressourcen sind Dateien“

Henry Spencer:

"Those who do not understand Unix are condemned to reinvent it, poorly."

Windows

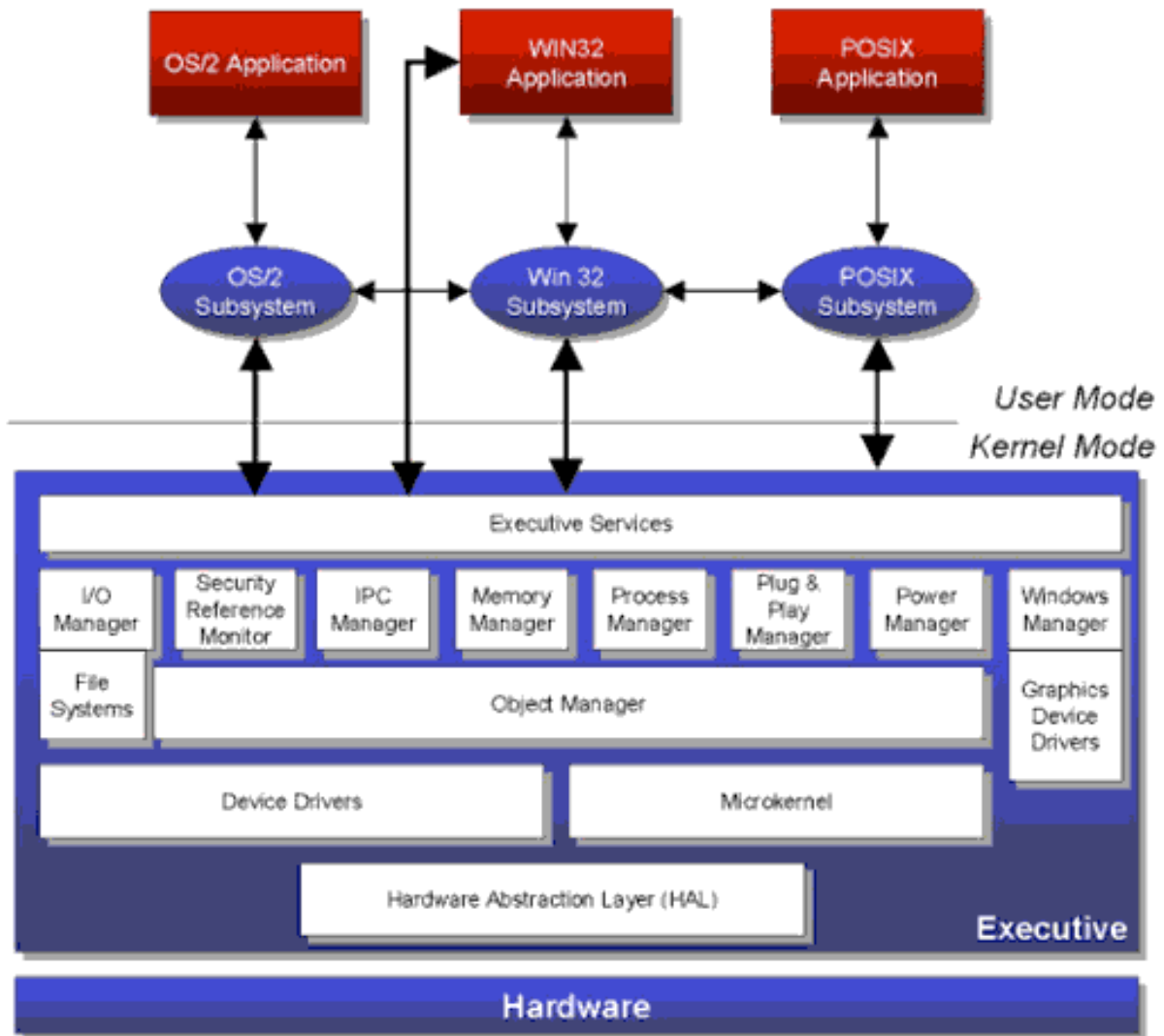


Figure 2.– System architecture

- 1981: MS-DOS als Nachäffung von CP/M. Original als QDOS (Quick & Dirty.OS)
- Windows 1/2: Gescheiterter Versuch eines Nachbaus von Apples MacOS
- 1990er: Durchbruch mit Windows 3.1, später 95/98/SE/ME. Allesamt auf MS-DOS basierend, keine echten BS.
- 1990er: Parallel Entwicklung von Windows NT/2000 als OS/2 3.0 mit echtem Kernel
- 2001: Vereinigung der Versionen zu XP
- Immer noch zu 100% ausgerichtet auf x86-Prozessoren

MacOS X



- 1984: Ur-Mac, erstes erfolgreiches BS mit GUI, kein echtes BS
- Prinzip „plug&play“. Benutzer bekommt Schreibtisch-Metapher
- 1990er: Gescheiterte Versuche, das Single-Tasking OS modern zu gestalten
- 1997: Übernahme von NextStep
- 2001: MacOSX basiert auf Unix
- MacOSX ist gebunden an Hardware von Apple. □

Merke: In den 70er Jahren waren sämtliche Konzepte, die ein (heute) modernes BS umfasst, bereits bekannt.

Merke: Es gibt kein „bestes“ Betriebssystem. Unterschiedliche Anforderungen und Lösungskonzepte.

Merke: Computer in 10 Jahren werden ganz anders aussehen als heutige

Ressourcen

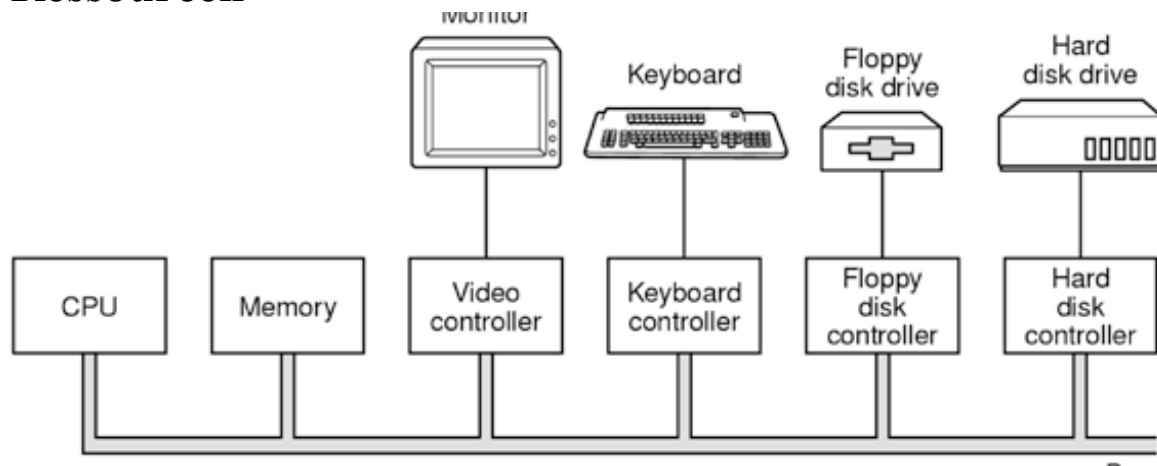


Abb.: Der (*sehr stark*) vereinfachte Aufbau eines Computers:

Prozessor

- Holt Anweisungen aus dem Speicher, Führt sie aus.
- Nur eine Anweisungsfolge zur Zeit (-> Prozess, -> Thread)
- Programmzähler, Keller, Statusregister
- BS: Muss beim Taskwechsel den „Zustand“ retten und wiederherstellen
- BS: Wann ist welcher Zugriff erlaubt?
- Kernel-Modus (alles erlaubt) vs. User-Modus (Zugriff auf Hardwareregister und Befehlssatz eingeschränkt)
- Wechsel von User- in Kernelmodus durch sog. TRAP
- Interrupts durch I/O

Hauptspeicher

- Linear Adressierbar
- BS: Schützen der Programme gegeneinander (und vom Kernel/BS!)
- BS: Laden von Programmen an unterschiedlichen Adressen
- BS: Trennen von Programm und Code
- BS: Gemeinsames Nutzen von Codebibliotheken
- BS: Mehrprozessorbetrieb: Synchronisation von Caches

Eingabe-/Ausgabegeräte

- BS: Vielfältige Probleme: Ansteuerung der Hardware, Abbilden von Informationen auf Sektoren, fehlerhafte Sektoren, Verteilen von Informationen auf mehrere Festplatten, Rechteverwaltung
- In der Regel werden Aufgaben von einem Controller übernommen

- Software zur Kommunikation mit Controller heißt Treiber.
- Treiber arbeitet im Kernel-Modus. Liefert selber Hardwareabstraktion für das BS.
- Datenübernahme in Puffern zum Ausgleich des Geschwindigkeitsunterschiedes
- Kommunikation der CPU mit dem Gerät:
 - o Polling („geschäftiges Warten“): CPU fordert Daten an und wartet auf Ergebnis (oder fragt regelmäßig ab, ob Ergebnis inzwischen fertig). Einfach, aber großer Overhead.
 - o Interrupt („Unterbrechung“): CPU fordert Daten an und fährt mit anderer Aufgabe fort. Controller unterbricht CPU, wenn Ergebnis bereitliegt. Kompromisier aber Effizient
 - o DMA (Direct Memory Access): Chip steuert Kommunikation zwischen Speicher und Controller. Interrupt, an CPU, wenn fertig.

Interrupts

- BS verwaltet eine ganze Reihe von verschiedenen Interrupts (Timer, I/O, Ausnahmen)
- Unterbrechung des aktuell laufenden Programms
- Interruptroutinen i.Allg. sehr kurz. Kein Taskwechsel.
- Interrupts sind Sperrbar (z.B. während der Behandlung eines Interrupts). Geräte wiederholen Interrupts.
- Interruptcontroller, Prioritäten

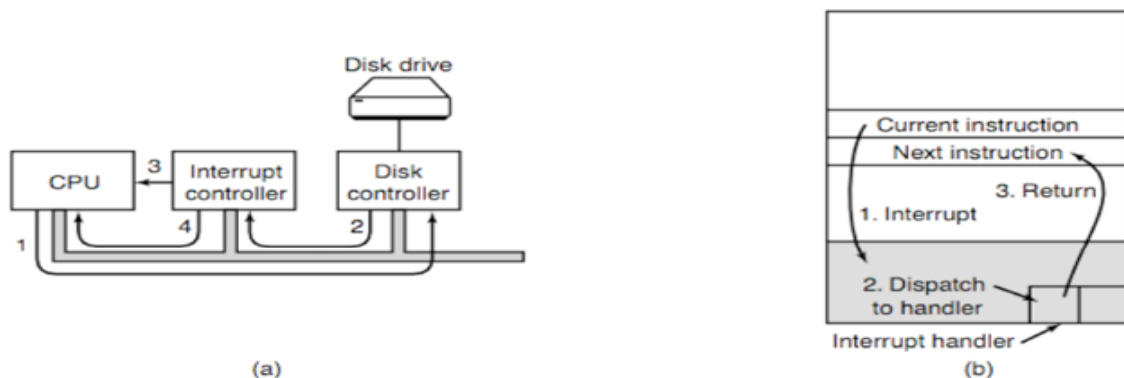


Abb.: Basisaufbau einer Interruptbehandlung

Bussysteme

- PCI, SCSI, USB, Frontside, ...
- Mini-Betriebssystem im ROM: BIOS (Basic Input Output System), OpenFirmware, IEEE Boot Standard. Bus- und Geräteerkennung beim Systemstart

Beispiele für Systemaufrufe (API des BS)

Prozessmanagement

- Erzeugen eines neuen Kindprozesses (fork)
- Warten auf die Beendigung eines Kindes (wait)
- Speicherabbild eines Prozesses ersetzen (exec)
- Prozess beenden (exit)

Dateimanagement

- Datei öffnen (open), schliessen (close), lesen (read), schreiben (write)
- Lesezeiger bewegen (seek), Status ermitteln (stat)

Verzeichnismanagement

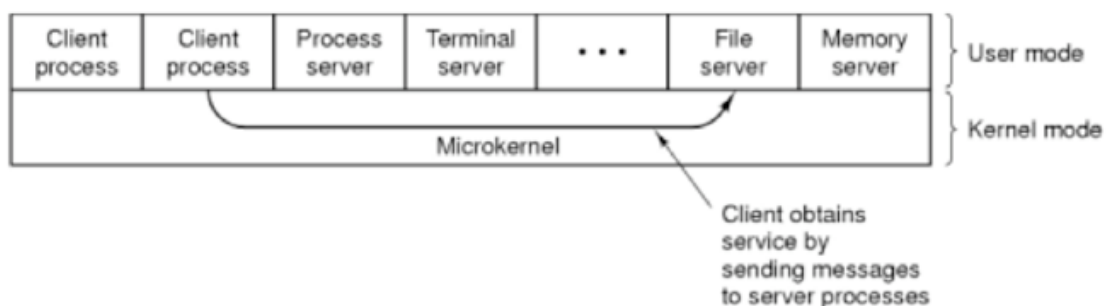
- Verzeichnis erzeugen (mkdir), löschen (rmdir)
- Dateisystem einhängen (mount), aushängen (umount)

Verschiedenes

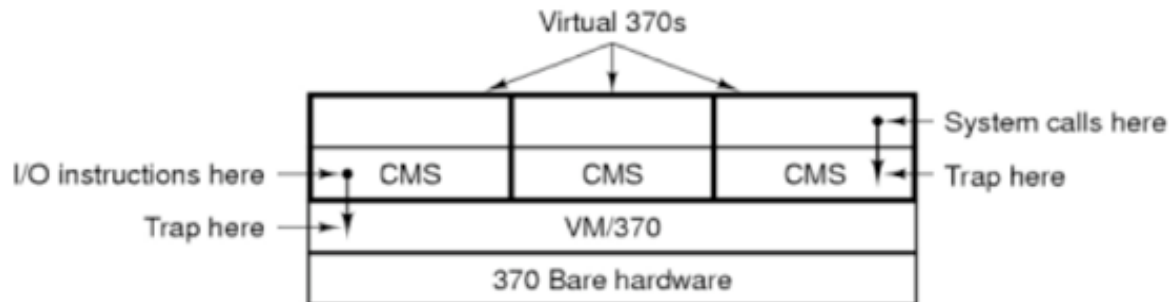
- Dateirechte (chmod)
- Verzeichniswechsel (chdir)
- Signal an Prozess schicken (kill)
- Zeit erfragen (time)

Strukturen von Betriebssystemen

- Monolithische Systeme (z.B. Linux, Windows)
 - o BS ist ein „Block“ mit geringer interner Struktur. Keine strenge Kapselung der Module.
- Client-Server-Modell, Mikrokern (MacOSX)
 - o BS ist unterteilt in Module, die sich nur über eine wohldefinierte Schnittstelle benutzen dürfen (message passing)
 - o Vorteile: Leichter wartbar, einfacher verstehbar, Verteilbar, Fehlerresistenter, Sicherer, portierbarer.
 - o Nachteil: Overhead durch komplexere Aufrufe der Module



- Virtuelle Maschinen (BS hat identische Schnittstelle wie zugrunde liegende Hardware: VM, DOS, VMWare, Java. Kann von der Hardware unterstützt sein).



Heute BS meist mit monolithischem Kern.

Virtuelle BS erfahren zunehmende Bedeutung:

- Emulation einer CPU X auf CPU Y, um artfremdes BS zu emulieren. Beispiel: VirtualPC auf MacOSX, um einen Windows-Rechner zu emulieren.
- Emulation alter Hardware, um ein altes BS auf moderner Hardware zu fahren: Migration existierender Systeme ohne Änderung der Konfiguration („never touch a running system“)
- Parallelbetrieb von verschiedenen BS. Beispiel: Testen von Webseiten/Anwendungen unter verschiedenen BS auf der gleichen hardware ohne Reboot
- Parallelbetrieb von gleichartigen BS auf einer hardware. Beispiel: Webhoster, der mehreren Kunden gleichzeitig Administratorzugang zu einem Rechner gibt. Jeder Kunde glaubt, er wäre „alleine“ auf dem Rechner (kann sogar seine virtuelle maschine rebooten, ohne die anderen virtuellen Maschinen zu stören).
- Parallelbetrieb gleicher BS, Um Development, Staging und Live-Server auf gleicher hardware zu trennen.
- Mainframe: Dynamische Anpassung von Ressourcenzuteilungen an verschiedene BS je nach Auslastung
- Trennung von Diensten vorbereiten, um bei steigenden Ressourcenanforderungen nahtlos auf schnellere Hardware umzusteigen oder Dienste zu migrieren („one service = one operating system“)

Ziel: Maximale Abschottung des Gast-BS vom Wirts-BS und zwischen den Gast-BS.

Realisierung auf PC-Hardware z.B. durch VMWare oder VirtualPC.

Meta-Betriebssysteme (VM, Xen)

Betriebssystemeigene Konzepte (Jails, Domains)