# Übersicht: Aufbau von Betriebssystemen

- Bestandteile von Betriebssystemen
- Dienste von Betriebssystemen
- Systemaufrufe
- Systemprogramme
- Struktur von Betriebssystemen
- Virtuelle Maschinen
- Entwurf und Implementierung von Betriebssystemen
- Systemanpassung



# Typische Bestandteile von Betriebssystemen

- Prozessverwaltung
- Hauptspeicherverwaltung
- Dateiverwaltung
- E/A-Verwaltung
- Sekundärspeicherverwaltung
- Netzwerkdienste
- Zugriffsschutz
- Kommandozeilen-Interpreter

# Prozessverwaltung

- Ein *Prozess* ist ein Programm, das gerade ausgeführt wird.
- Ein Prozess benötigt bestimmte Ressourcen, um seine Aufgabe zu erfüllen: Prozessorzeit, Speicher, Dateien, E/A-Geräte usw.
- Im Zusammenhang mit der Prozessverwaltung ist das Betriebssystem für folgende Aktivitäten zuständig:
  - Erzeugung und Löschung von Prozessen
  - Prozessunterbrechung und Prozessfortsetzung
  - Bereitstellung von Mechanismen zur
    - Prozess-Synchronisation
    - Prozess-Kommunikation



# Hauptspeicherverwaltung

- Der Hauptspeicher ist ein großes Array von adressierbaren Speicherwörtern oder Bytes. Er ist ein Aufbewahrungsort für schnell zugängliche Daten und wird vom Prozessor und den E/A-Geräten gemeinsam genutzt.
- Der Hauptspeicher ist ein volatiles Medium. Sein Inhalt geht bei Ausfall der Stromversorgung verloren.
- Im Zusammenhang mit der Hauptspeicherverwaltung ist das Betriebssystem für die folgenden Aktivitäten zuständig:
  - Erfassen, welche Prozesse gegenwärtig welche Speicherbereiche verwenden
  - Entscheiden, welche Prozesse geladen werden sollen, wenn Speicherbereiche frei werden.
  - Nach Bedarf Speicher reservieren und freigeben

# Dateiverwaltung

- Eine Datei ist eine Sammlung von zusammengehörigen Informationen (definiert durch den Erzeuger der Datei). Üblicherweise repräsentieren Dateien Programme (Quellprogramme als auch ausführbare Programme) und Daten.
- Im Zusammenhang mit der Dateiverwaltung ist das Betriebssystem verantwortlich für die folgenden Aktivitäten:
  - Dateien erzeugen und löschen
  - Verzeichnisse erzeugen und löschen
  - Grundfunktionen zur Bearbeitung von Dateien und Verzeichnissen bereitstellen
  - Dateien auf den Sekundärspeicher abbilden
  - Dateien auf Backup-Medien sichern



# E/A-Verwaltung

- Die E/A-Verwaltung besteht aus:
  - Einem Puffer-Caching-System
  - Allgemeinen Gerätetreiber-Schnittstellen
  - Treibern für spezielle Geräte

# Sekundärspeicherverwaltung

- Da der Hauptspeicher volatil und relativ klein ist, muss ein Computersystem ausreichend Sekundärspeicher bereitstellen, in dem Programme und Daten dauerhaft gesichert werden können.
- Die meisten modernen Betriebssysteme nutzen Festplatten als zentrales Sekundärspeichermedium für Programme und Daten.
- Im Zusammenhang mit der Sekundärspeicherverwaltung ist das Betriebssystem verantwortlich für die folgenden Aktivitäten:
  - Freispeicherverwaltung
  - Speicherzuweisung
  - Festplatten-Scheduling



# Netzwerkdienste (Verteilte Systeme)

- Ein *verteiltes System* ist eine Menge von Prozessoren, mit jeweils eigenem Speicher und eigenem Systemtakt.
- Die Prozessoren sind über ein Kommunikationsnetzwerk verbunden.
- Die Kommunikation ist über *Protokolle* geregelt.
- Ein verteiltes System ermöglicht den Benutzern Zugriff auf (geographisch-verteilte) Systemressourcen.
- Verteilte Systeme werden aus verschiedenen Gründen realisiert:
  - Steigerung der Rechenleistung
  - Höhere Datenverfügbarkeit
  - Erhöhte Ausfallsicherheit

# Zugriffsschutz

- Unter *Zugriffsschutz* versteht man Mechanismen zur Regelung des Zugriffs von Programmen, Prozessen oder Anwendern auf das Computersystem und Anwender-Ressourcen.
- Ein Zugriffsschutzmechanismus muss:
  - zwischen autorisiertem und nicht-autorisiertem Zugriff unterscheiden
  - die Definition von Zugriffsrechten ermöglichen
  - Mittel zur Durchsetzung der Zugriffsregeln bereitstellen



# Kommandozeilen-Interpreter I

- Über verschiedene Steuerkommandos kann der Administrator/Anwender Einfluss auf das Computersystem zu nehmen.
- Häufig existieren Steuerkommandos:
  - zur Erzeugung und Verwaltung von Prozessen,
  - zur Behandlung von E/A-Geräten,
  - zur Verwaltung des Sekundärspeichers,
  - zur Regelung des Dateizugriffs,
  - zur Verwaltung von Benutzern und
  - zur Steuerung von Netzwerkfunktionen.

# Kommandozeilen-Interpreter II

- Das Programm, das die Steuerkommandos einliest und interpretiert, wird Kommandozeilen-Interpreter (command-line interpreter) genannt. Bei Betriebssystemen, die auf UNIX basieren, ist der Begriff „shell“ gebräuchlich.
- Die Aufgabe des Kommandozeilen-Interpreters ist es, das nächste Steuerkommando entgegenzunehmen und auszuführen.



# Dienste von Betriebssystemen

- **Programmausführung** – Routinen, um ein Programm in den Hauptspeicher zu laden und zu starten
- **Ein-/Ausgabe** – Da Anwendungsprogramme nicht selbst E/A-Operationen ausführen dürfen, muss das Betriebssystem Mittel zur Ein- und Ausgabe bereitstellen.
- **Dateibearbeitung** – Routinen zum Lesen, Schreiben, Erzeugen und Löschen von Dateien.
- **Prozesskommunikation** – Routinen zum Austausch von Informationen zwischen Prozessen auf dem gleichen Computer oder unterschiedlichen, über ein Netzwerk verbundenen Rechnern. (Realisiert über gemeinsam genutzten Speicher oder durch Nachrichtenaustausch)
- **Fehlererkennung** – Sicherstellung der korrekten Arbeitsweise des Systems durch Erkennung von Fehlern im Prozessor, in der Speicher-Hardware, ein E/A-Geräten und in Anwenderprogrammen.

# Verwaltungsfunktionen

Verwaltungsfunktionen dienen der Sicherstellung des reibungslosen und effizienten Betriebs des Systems.

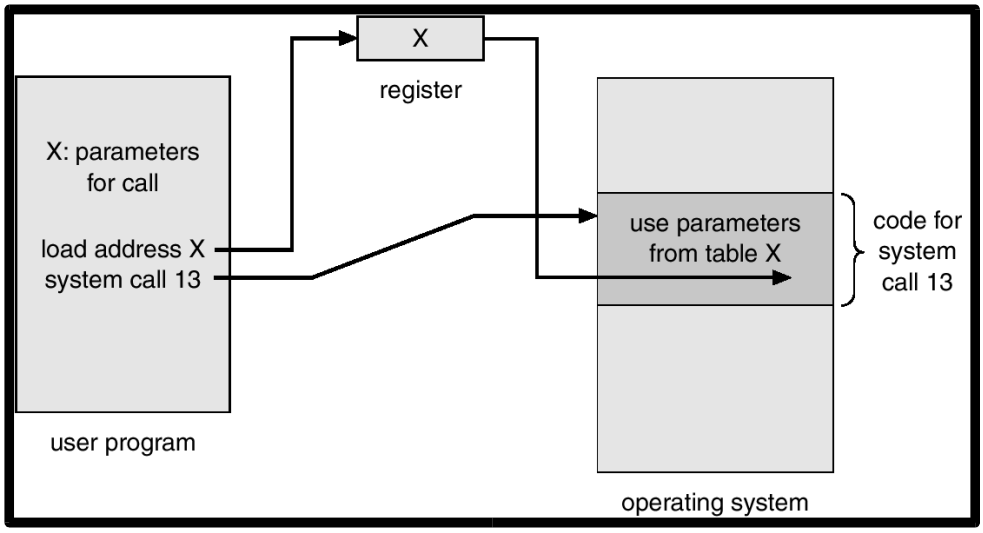
- **Zuweisung von Ressourcen** – Einflussnahme auf die Verteilung von Betriebsmitteln an verschiedene Benutzer oder verschiedene, gleichzeitig laufende Prozesse.
- **Buchhaltung** – Erfassung, welche Anwender/Prozesse welche Betriebsmittel wie lange verwenden. Diese Informationen können etwa zur Abrechnung oder Systemoptimierung dienen.
- **Sicherheitseinstellungen** – Steuerung der Zugriffsmöglichkeiten auf die Systemressourcen.



# Systemaufrufe

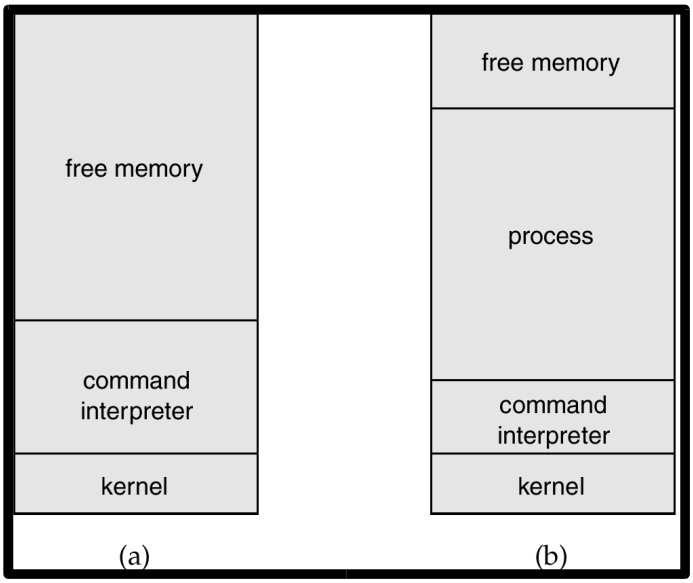
- Systemaufrufe bilden die Schnittstelle zwischen einem laufenden Programm und dem Betriebssystem.
  - Systemaufrufe stehen traditionell in Assembler zur Verfügung.
  - Systemaufrufe sind auch in einigen Hochsprachen direkt verfügbar, insbesondere, wenn die Sprachen zur Systemprogrammierung eingesetzt werden (z.B. C, C++)
- Zur Übergabe von Parametern zwischen dem laufenden Programm und dem Betriebssystem existieren drei unterschiedliche Vorgehensweisen:
  - Die Parameter werden in Prozessorregistern übergeben.
  - Die Parameter werden in einer Tabelle im Speicher abgelegt und die Adresse der Tabelle in einem Prozessorregister übergeben.
  - Die Parameter werden auf dem Programmstack abgelegt (push) und dort vom Betriebssystem gelesen (pop).

# Parameter als Tabelle übergeben

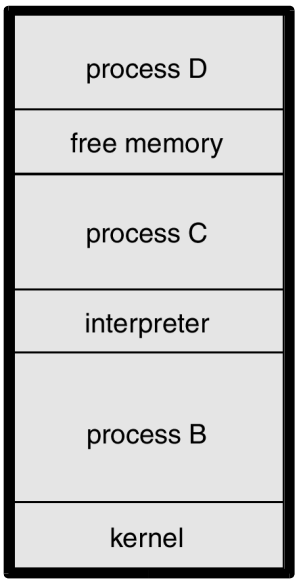





# Beispiel: Programmausführung unter MS-DOS



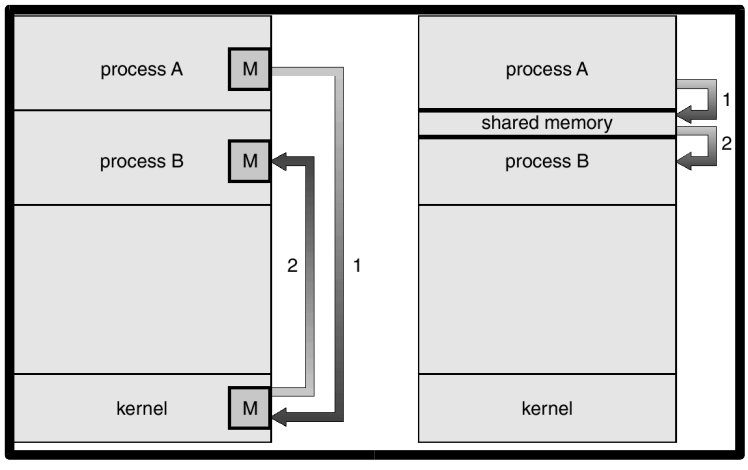
# Beispiel: Mehrere Programme ablaufen lassen (UNIX)






# Die zwei Modelle der Prozesskommunikation

- Die Kommunikation zwischen Prozessen geschieht entweder über Nachrichtenaustausch oder gemeinsam genutzten Speicher.




Nachrichtenaustausch gemeinsamer Speicher

# Systemprogramme

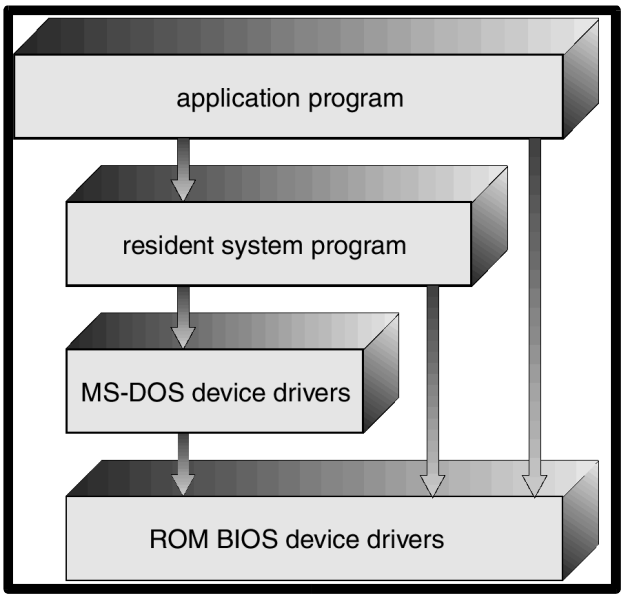
- Systemprogramme stellen eine benutzerfreundliche Umgebung zur Entwicklung und Ausführung von Programmen bereit. Systemprogramme dienen zur
  - Dateiverwaltung
  - Bereitstellung von Statusinformationen
  - Dateibearbeitung
  - Unterstützung der Programmierung
  - Programmausführung
  - Kommunikation
- Die Benutzersicht eines Betriebssystems ist im Wesentlichen bestimmt durch die Systemprogramme, nicht durch die Systemaufrufe.



# MS-DOS-Systemstruktur (I)

- MS-DOS – wurde entwickelt, um bei möglichst geringem Speicherbedarf eine möglichst große Funktionalität bereitzustellen.
  - Keine Unterteilung in Module
  - Obwohl MS-DOS eine gewisse Struktur aufweist, sind seine Schnittstellen und Funktionsebenen nicht deutlich abgegrenzt.

# MS-DOS-Systemstruktur (II)

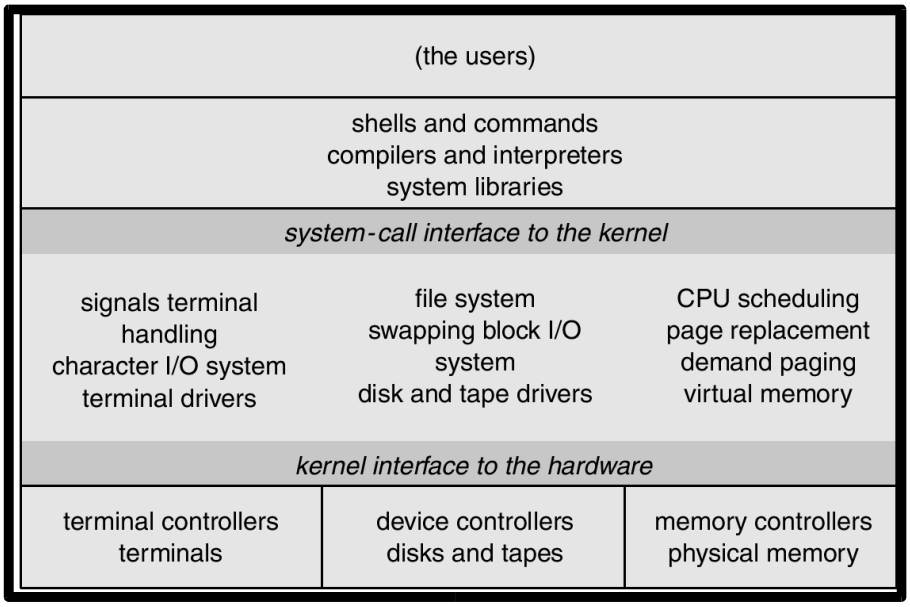





# UNIX-Systemstruktur (I)

- Aufgrund von Hardwarebeschränkungen besitzt das ursprüngliche UNIX ebenfalls nur eine wenig ausgeprägte Gliederung. Bei UNIX lassen sich zwei grundlegende Bestandteile unterscheiden:
  - Der Kernel
    - Umfasst alles unterhalb der Systemaufruf-Schnittstelle und oberhalb der physikalischen Hardware.
    - Ist zuständig für das Prozessor-Scheduling, die Speicherverwaltung, das Dateisystem und einer Vielzahl weiterer Betriebssystemfunktionen. (Sehr viele Funktionen für nur eine Ebene)
  - Systemprogramme

# UNIX-Systemstruktur (II)

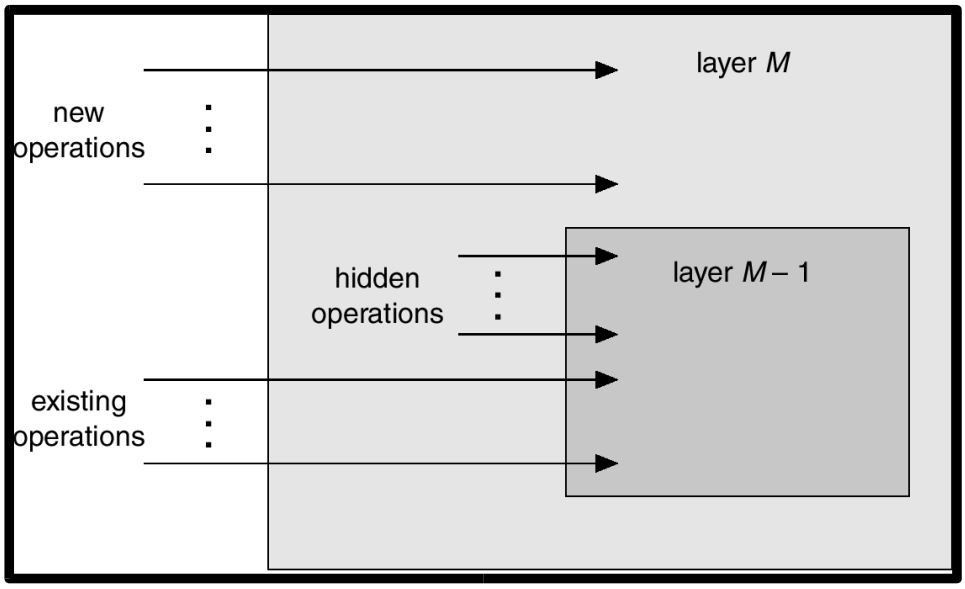





# Schichtenansatz

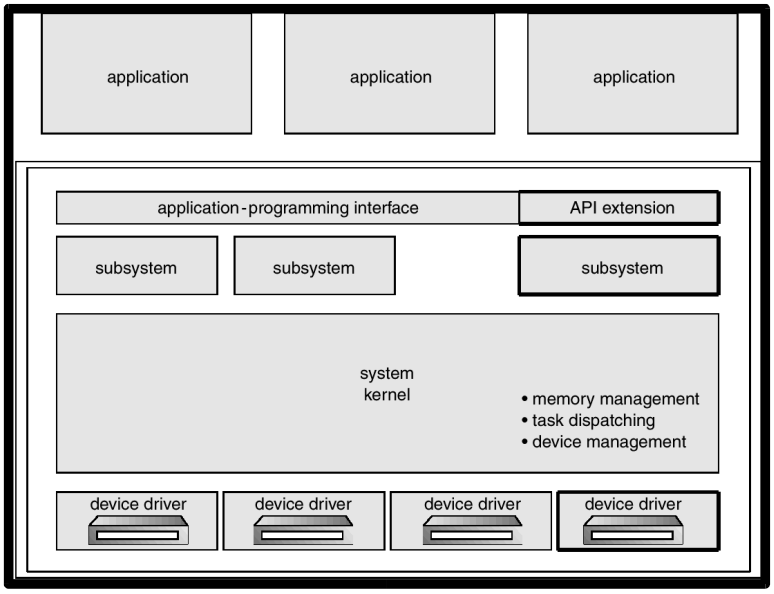
- Das Betriebssystem ist in eine bestimmte Anzahl von Schichten (Ebenen) unterteilt. Die unterste Schicht (Schicht 0) ist die Hardware, die oberste (Ebene N) die Benutzerschnittstelle.
- Die Schichten werden so definiert, dass eine Schicht nur Funktionen und Dienste der darunter liegenden Schicht(en) verwendet. Dadurch wird eine gewisse Modularität erreicht.

# Schicht eines Betriebssystems






# Schichtenstruktur von OS/2

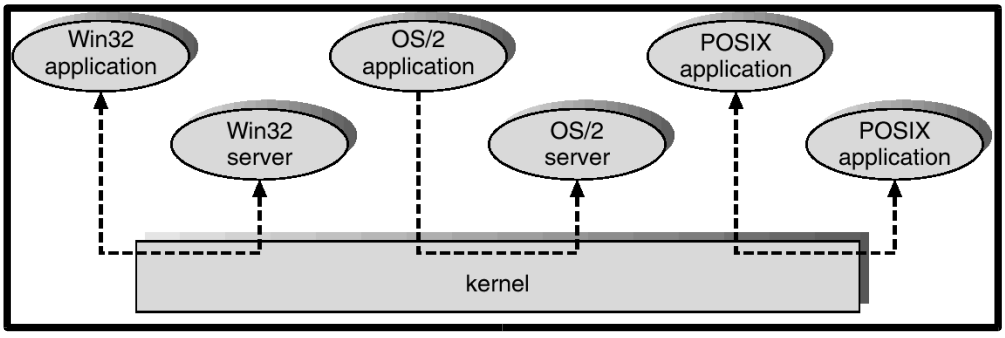


# Mikrokernel-Systemstrukture

- Auslagerung von möglichst viel Funktionen aus dem Kernel in Module, die im Anwendermodus ablaufen.
- Die Kommunikation zwischen den Modulen erfolgt über Nachrichtenaustausch.
- Vorteile:
  - Leichtere Erweiterbarkeit
  - Leichtere Portierbarkeit des Betriebssystems auf andere Hardware-Plattformen
  - Höhere Zuverlässigkeit und Sicherheit, da weniger Systembestandteile im Systemmodus ablaufen



# Client-Server-Struktur von Windows NT



# Virtuelle Machinen (I)

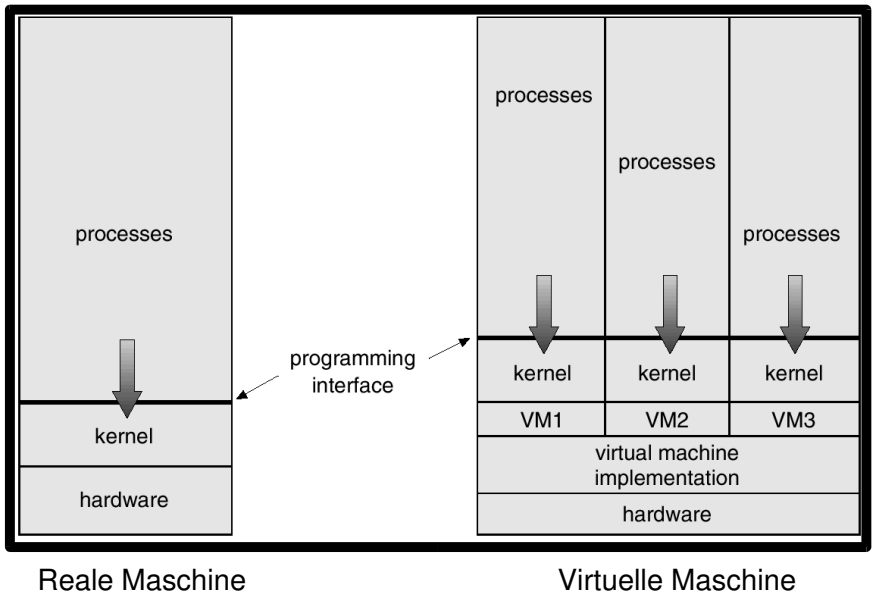
- Eine *virtuelle Maschine* ist die konsequente Fortführung des Schichtenansatzes. Zwischen Aufrufen des Betriebssystemkernels und Aufrufen der Hardware wird nicht unterschieden.
- Eine virtuelle Maschine stellt eine Schnittstelle bereit, die *identisch* ist mit der zugrundeliegenden Hardware.
- Das Betriebssystem erweckt so den Eindruck, als ob jeder Prozess auf seiner eigenen Hardware (eigener Prozessor, eigener (virtueller) Speicher) abläuft.



# Virtuelle Maschinen (II)

- Die Betriebsmittel eines (physikalisch realen) Computers werden gemeinsam genutzt, um virtuelle Maschinen zu realisieren:
  - Durch Prozessor-Scheduling entsteht der Eindruck, als habe jeder Prozess seinen eigenen Prozessor.
  - Virtuelle Ein- und Ausgabegeräte können durch Spooling und Umlenkung vom/zum Dateisystem simuliert werden.
  - Eine normales Benutzerterminal dient als Konsole für den Operator der virtuellen Maschinen.

# Reale und virtuelle Maschinen






# Vor- und Nachteile virtueller Maschinen

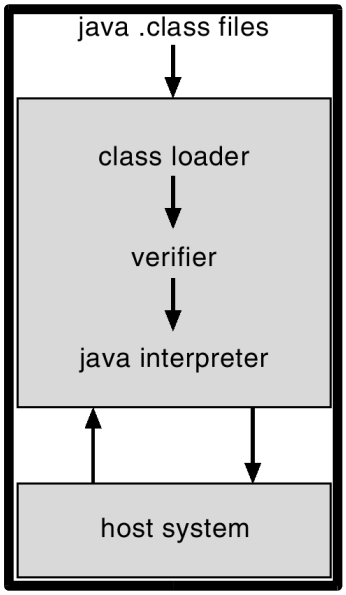
- Durch das Konzept der virtuellen Maschine wird der vollständige Schutz der Systemressourcen gewährleistet, da jede virtuelle Maschine vollständig von allen anderen isoliert ist. Eine direkte, gemeinsame Nutzung von Ressourcen ist somit allerdings nicht möglich.
- Virtuelle Maschinen eignen sich gut für die Forschung an und die Entwicklung von Betriebssystemen. Die Entwicklung kann auf virtuellen Maschinen erfolgen ohne den normalen Betrieb des physikalischen Systems zu stören.
- Da ein exaktes Duplikat der zugrundeliegenden Maschine benötigt wird, ist der Aufwand zur Realisierung einer virtueller Maschinen hoch.

# Java-Virtual-Machine (I)

- Compilierte Java-Programme bestehen aus plattformunabhängiem Bytecode, die von einer Java-Virtual-Machine (JVM) ausgeführt wird.
- Die JVM besteht aus:
  - dem Class-Loader
  - dem Class-Verifier
  - dem Laufzeitinterpreter
- Durch Just-In-Time-Compiler (JIT-Compiler) wird die Performanz erhöht.



# Java-Virtual-Machine (II)



# Ziele des Systementwurfs

- Benutzerziele – Das Betriebssystem soll einfach zu bedienen, leicht zu lernen, zuverlässig, sicher und schnell sein.
- Systemziele – Das Betriebssystem soll einfach zu entwerfen, zu implementieren und zu warten sein. Ferner soll es flexibel und zuverlässig sein sowie fehlerfrei und effizient arbeiten.



# Trennung von Mechanismen und Regeln

- Mechanismen bestimmen, *wie* etwas getan wird, Regeln bestimmen, *was* getan wird.
- Die Trennung von Mechanismen und Regeln ist ein wichtiges Prinzip beim Betriebssystementwurf. Da die Regeln auch nach der Entwicklung an die speziellen Bedürfnisse eines Systems angepasst werden können, wird ein Höchstmaß an Flexibilität erreicht.

# Implementierungsaspekte

- Ursprünglich wurden Betriebssysteme in Assembler geschrieben. Heutzutage werden sie (größtenteils) in Hochsprachen entwickelt.
- Programmcode in einer Hochsprache:
  - kann schneller entwickelt werden,
  - ist kompakter,
  - leichter zu verstehen und einfacher zu debuggen.
- Ein Betriebssystem ist wesentlich leichter auf andere Hardware-Plattformen zu portieren, wenn es in einer Hochsprache geschrieben wurde.



# System Generation (SYSGEN)

- Betriebssysteme werden für eine ganze Klasse von Rechnersystemen entworfen. Um auf einem speziellen System laufen zu können, muss das Betriebssystem bei der Installation bzw. beim Systemstart (booten) entsprechend konfiguriert werden.
- Beim Konfigurationsvorgang (SYSGEN) werden alle benötigten Daten über das Computersystem ermittelt. Dazu gehören: Informationen über Prozessoren, den Hauptspeicher, Festplatten, Geräte, Geräteadressen, Puffergrößen usw.
- *Booten* – Starten des Computers durch das Laden des Betriebssystemkernels. Das *Bootstrap-Programm* ist im ROM gespeichert, ermittelt die Position des Kernels, lädt ihn in den Hauptspeicher und startet ihn.