

1 Allgemeines

In diesem Kapitel lernen Sie

- ▶ die Geschichte von UNIX, Linux, Freier Software und des GNU-Projektes kennen.
- ▶ den Aufbau des GNU/Linux-Systems und seine Architektur kennen.
- ▶ die Unterschiede von Linux zu DOS kennen.

1.1 Zur Geschichte von UNIX und LINUX

1.1.1 UNIX-Entwicklung

AT&T entwickelte Ende der 60-Jahre ein neuartiges Betriebssystem MULTICS (*Multiplexed Information and Computing Service*), das eine Reihe revolutionärer neuer Konzepte wie u.a. Timesharing (das Ausführen von mehreren Programmen auf einer CPU) implementieren sollte. Jedoch war das Projekt so ehrgeizig, dass MULTICS viel zu träge und zu groß für die damalige Hardware war.

Aus diesem Grunde setzte sich Ken Thompson, einer der MULTICS-Entwickler, an eine DEC PDP-7, die in einer Ecke stand, und fing an, ein kleines, schlankes Betriebssystem zu schreiben, das die wesentlichen wichtigen Konzepte von MULTICS verwirklichte. Ken Thompson hat deswegen dieses System zunächst UNICS (*Uniplexed Information and Computing Service*) genannt, als Anspielung auf MULTICS.

Gerüchten zu Folge war Ken Thompsons Motivation, UNIX zu schreiben, die, dass er auf der PDP-7 mit seinem Freund das Mehrbenutzerspiel „Space Travel“ spielen konnte.

Die erste UNIX-Version wurde in Assembler (Maschinencode) geschrieben. Um bei zukünftigen Projekten ein maschinenunabhängiges Betriebssystem zu erhalten, entwarf Thompson (AT&T) die Programmiersprache „B“, aus der dann die bekannte Programmiersprache „C“ weiterentwickelt wurde.

So war es kein Wunder, dass die nächste UNIX-Version in „C“ geschrieben wurde. Dies geschah zwischen 1972 und 1973.

Erst durch die Programmiersprache „C“ konnte ein neues Betriebssystem geschaffen werden, das relativ unabhängig von der bestehenden Hardware war. Durch die leichte Übertragbarkeit des neuen Betriebssystems UNIX konnte eine neue Rechnergeneration geschaffen werden, für die es zuvor kein entsprechendes Betriebssystem gegeben hatte.

1.1 Zur Geschichte von UNIX und LINUX

Auf einer RISC-, Power-PC- oder Alpha-Workstation, einem Server oder Großrechner, bis hin zu einem „normalen“ Rechner, auch PC genannt, kann das neue Betriebssystem eingesetzt werden.

Neben den kommerziellen, an bestimmte Hardwareplattformen gebundenen UNIX-Derivaten (Ableitungen), z.B. HP-UX von Hewlett Packard oder Tru64 UNIX von Compaq, bieten heute die Firmen Novell (Unixware), SUN (SUN/OS, Solaris), SCO (SCO-UNIX) auch UNIX-Versionen für PCs an, die Berkeley Software Distribution (BSD) ist als Open Source Software erhältlich, und als wesentliche neue Variante kommt seit etwa 1994 das Open Source-Betriebssystem Linux hinzu.

Die Komplexität des Betriebssystems wird erst sichtbar, wenn man sich vergegenwärtigt, dass z.B. die UNIX-Version System V4 aus ca. 1.200 Programmen bzw. Kommandos mit mindestens 200 MB Festplattenbedarf besteht. Das Geniale daran ist, dass sich viele kleine, spezialisierte Tools wie bei einem Baukasten je nach Bedarf zu größeren, leistungsfähigeren Werkzeugen verbinden lassen.

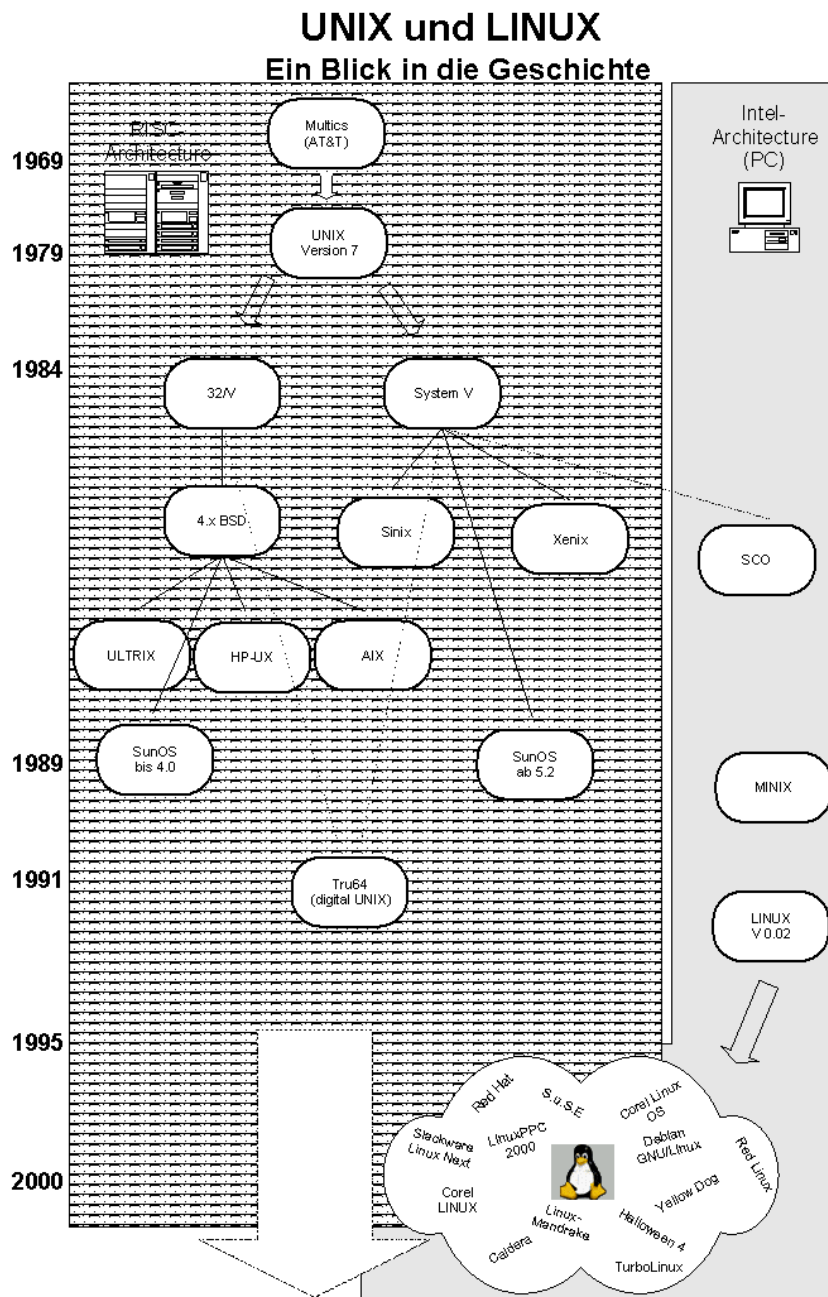
Allgemeines

Jahreszahl	Ereignis
1969	Ken Thompson, AT&T, beginnt mit der Entwicklung eines neuen Betriebssystems auf einer PDP7 in Anlehnung an MULTICS: UNIX Version 0 Ziel: einfache Datenstruktur, mehrbenutzerfähig, dialogorientiert, schnell, schlank.
1972–1973	UNIX wird umgeschrieben; nur noch ca. 10% in Assembler, alles andere in C. Durch Entwicklung und nachfolgender Verwendung der Hochsprache C wird einfache Portierbarkeit des Systems erreicht.
1977	AT&T lockert Lizenzpolitik; andere Hersteller dürfen UNIX auf ihren Plattformen anbieten.
1979	UNIX Version 7; Ausgangspunkt zahlreicher Portierungen. Die University of California Berkeley (UCB) bringt ihre UNIX-Version BSD 3.0 (Berkeley Software Distribution) heraus.
1983	UNIX System V.0 von AT&T; Dokumentation und Support werden nun mit angeboten.
1984–1989	Verschiedene Stufen der Weiterentwicklung bis zu UNIX System V Rel. 3 (SVR3) und BSD 4.3.
1988	Es bilden sich die zwei Pole OSF (Open Software Foundation) und UI (UNIX-International) mit dem Ziel, zukünftige UNIX-Versionen zu standardisieren: OSF: OSF/1 UI: UNIX System V Rel. 4 (SVR4)
1990	SVR4 wird ausgeliefert. Es integriert Funktionen aus SVR3, BSD 4.2, SunOS und Xenix.
1990	Grafische Oberfläche X-Windows mit dem Window-Manager OSF/Motif verbreitet sich auf UNIX-Systemen.
1991	OSF liefert das neue Betriebssystem OSF/1 aus. OSF/1 integriert Funktionen aus BSD 4.3, SVR3 und Mach. Die Derivate AIX, HP-UX und Digital UNIX integrieren OSF/1-Features. Die erste Linux-Version erscheint.
1995	Benutzeroberfläche CDE, eine Erweiterung von OSF/Motif, setzt sich als Standard-Benutzeroberfläche auf vielen Derivaten durch.
1995	Der Siegeszug von Linux beginnt. Der durchschlagende Erfolg des Open-Source-Modells schafft endlich ein dominierendes, auf allen Plattformen erhältliches UNIX-System.
1996	KDE als Benutzeroberfläche und nichtkommerzieller Ersatz für CDE beginnt, den Weg für Linux als Desktop-System zu bahnen.
2000	Linux hat einen Marktanteil von 18% auf dem Servermarkt erobert und ist nach Windows NT (Marktanteil 37%) das Server-Betriebssystem Nummer zwei. Der Einsatz von Linux als Desktop-System in den Unternehmen beginnt, es sind mittlerweile einige Office-Pakete in ausgezeichneter Qualität erhältlich, kaufmännische Software ist zu günstigen Preisen zu haben. Unternehmen wie IBM, SAP, HP und SGI portieren ihre Software nach Linux und/oder bieten die hauseigenen Rechner wahlweise mit Linux an. Linux breitet sich stark auf Embedded-Systemen wie Palmtops, Set-Top-Boxen oder Firewalls aus.

1.1 Zur Geschichte von UNIX und LINUX

Voraussetzung Betriebssystem UNIX:

- Leistungsfähiger Rechnertyp
- Rechner ist netzwerkfähig



Wesentliche Eigenschaften von UNIX und Linux:

- Mehrbenutzerbetriebssystem für Einzelplatz bis zum Großrechner (multiuser-fähig)
- Multitaskingfähigkeit durch Timesharing
- weitgehende Geräteunabhängigkeit
- Netzwerkfähigkeit
- relative Störungsunempfindlichkeit und Stabilität
- Portierbarkeit und Skalierbarkeit
- Sicherheit durch Zugriffsrechte
- Flexible Benutzerschnittstelle (mehrere Shells stehen zur Verfügung)

1.1.2 Linux-Entwicklung

Alles begann damit, dass ein finnischer Student namens Linus Benedict Torvalds sein Verständnis über die Funktionsweise des 80386-Prozessors vertiefen wollte. Dazu entwickelte er ein Steuerprogramm, welches zwischen zwei Programmen wechseln konnte.

Daraus entstand dann ein einfaches Terminalprogramm: Ein Teil des Programms liest Zeichen von der seriellen Schnittstelle und stellt sie auf dem Bildschirm dar, ein anderer überträgt die eingegebenen Zeichen von der Tastatur zur seriellen Schnittstelle. Nach einigen Weiterentwicklungen hatte Linus Torvalds den Urvater des heutigen Linux-Betriebssystems geschaffen: Linux Version 0.1.

Nach zahlreichen Erweiterungen und Verbesserungen erschien im September 1991 die erste „öffentliche“ Version von Linux. Seit dieser Zeit beteiligen sich viele Studenten, Angestellte von spezialisierten Firmen und andere Interessierte an der Weiterentwicklung und ermöglichten im Frühjahr 1994 die Version 1.0. Im Laufe der Zeit hat die Funktionalität und die Zahl der Programme erheblich zugenommen und übersteigt momentan viele kommerziell erhältlich UNIX-Derivate bei weitem.

1.1.3 Freie Software

GNU steht für „GNU is Not UNIX“ und ist ein Projekt der *Free Software Foundation* (FSF), deren Ziel die Schaffung eines völlig frei verfügbaren, mit UNIX kompatiblen Betriebssystems ist. Im Zuge der Entwicklung wurden alle UNIX-Hilfsprogramme neu entwickelt und teilweise mit mehr oder verbesserter Funktionalität versehen. Die

rechtliche Grundlage liefert die GNU General Public License (GPL). (Sie finden die GPL bei jeder Linux-Distribution und auch in vielen Linux-Büchern.)

Das GNU-Projekt hatte sein Ziel, ein freies Betriebssystem, bereits vor einiger Zeit fast erreicht, es fehlte nur der Betriebssystemkern. Mit *HURD* war zwar ein solcher geplant und in Arbeit, aber man hatte nie wirklich einen einsatztauglichen Kernel fertiggestellt. Deshalb kombinierte man die Programme und Bibliotheken von GNU mit dem Linux-Kernel, und es entstand das, was man heute gerne als „Linux“ bezeichnet, obwohl die Bezeichnung „GNU/Linux“ eigentlich gerechter und sinnvoller ist.

Unzählige Programmierer aus aller Welt arbeiten an der Entwicklung von GNU/Linux, unentgeltlich, in ihrer Freizeit, mit. Sie treibt kein kommerzielles Interesse, sondern das Vergnügen, alleine oder im Team, Probleme und Schwierigkeiten zu lösen, neue Funktionen hinzuzufügen und so das Betriebssystem zu perfektionieren.

1.2 LINUX-Übersicht

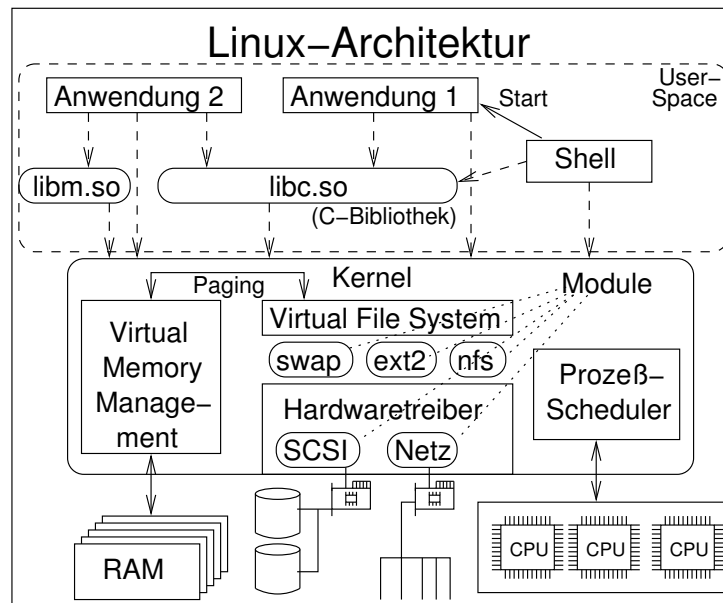
In diesem Kapitel werden einige der grundlegenden Eigenschaften von Linux vorgestellt.

1.2.1 Architektur

Routinen von zentraler Bedeutung für Verwaltung von Daten und Prozessen werden in einem Systemkern (Kernel) integriert. Diese Routinen laufen im sogenannten *Kernel-Space* (auch *Kernel-Mode* genannt) ab und erhalten spezielle Privilegien, wie z.B. direkten Hardware-Zugriff.

Alles andere wird strikt vom Systemkern ferngehalten und auf die Ebenen der Dienstprogramme verlagert. Diese laufen dann im *User-Space* (auch *User-Mode* genannt) ab und haben keine Möglichkeit, die Privilegien eines Kernel-Mode-Prozesses zu erlangen. Dies verleiht dem Betriebssystem eine hohe Stabilität und Robustheit.

In der folgenden Abbildung sind die Beziehungen der Systemkomponenten zum Systemkern schematisch dargestellt. Die verschiedenen Benutzer, die bei Linux gleichzeitig auf dem Rechner arbeiten können, werden hierbei nicht berücksichtigt.



In der obigen Abbildung existieren drei wesentliche Bereiche:

- Die *Hardware*, im Schaubild das RAM, ein SCSI-Hostadapter mit zwei Festplatten und drei CPUs.
- Der *Kernel*, der die notwendigen Systemroutinen enthält, um auf die Hardware zugreifen zu können. Er gaukelt den Anwendungen vor, sie hätten einen eigenen Rechner für sich alleine. Der Kern ist durch *Kernel-Module*, während der Laufzeit, beliebig durch neue Funktionen oder Treiber erweiterbar. Die Hauptteile des Kerns sind:

Der *Prozessscheduler*, der die Prozessorzeit der Prozessoren auf die einzelnen Anwendungen (= Prozesse) verteilt. Er schaltet so schnell zwischen den Prozessen um, dass der Eindruck entsteht, sie liefen gleichzeitig. Damit können auch mehrere Benutzer gleichzeitig und unabhängig voneinander arbeiten.

Das *Speichermanagement* oder *Virtual Memory Management*. Jedem Prozess auf dem System steht scheinbar der ganze Speicheradressraum (4 GB auf 32-bit-Systemen) zur Verfügung. In diesen werden sowohl Daten geschrieben als auch Programmdateien geladen, und erst die der Memory Manager sorgt dafür, dass diese virtuellen Speicheradressen einem physikalischen Ort im Arbeitsspeicher oder auf der Festplatte (evtl. gemeinsam mit anderen Prozessen bei nur lesbaren Bereichen wie Programmcode) zugeordnet werden.

Der ganze virtuelle Speicher ist in Speicherseiten unterteilt, die meist eine Größe von 4 kB haben. Wird nun eine Speicherseite (Page) einer Anwendung längere Zeit nicht benötigt, so wird sie auf die Festplatte ausgelagert. Den freiwerdenden Platz im RAM kann dann eine andere Anwendung verwenden. Diesen Vorgang nennt man *Paging*. Früher hat man stets ganze Prozesse auf die Platte ausgelagert (*Swapping*), was jedoch zu ineffizient ist. Daher kommt der Name *Swap-Speicher*, mit dem der virtuelle Festplattenspeicher bezeichnet wird. Die Menge an Swap-Speicher, die Linux verwalten kann, ist von der CPU und der Kernelversion abhängig. Enterprise-Kernel können, entsprechende CPU und genügend Speicherplatz vorausgesetzt, Swap-Speicher bis in den Terabyte-Bereich verwalten.

Unter Linux wird generell immer *der komplette verfügbare Arbeitsspeicher* voll ausgenutzt. Falls die Anwendungen weniger RAM brauchen, als verfügbar ist, wird das restliche RAM verwendet, um Dateien, auf die häufig zugegriffen wird, zwischenzupuffern. Damit beschleunigt sich der Dateizugriff beträchtlich. Diesen Puffer nennt man *Festplatten-Cache*.

Der VFS oder *Virtual Filesystem Switch* ist eine einheitliche Schnittstelle zu allen benutzbaren physikalischen Dateisystemen, sei es ein Standard-Linux-Dateisystem (ext2), ein Windows-Dateisystem oder gar ein Netzwerk-Dateisystem wie NFS. Neue Dateisysteme können über Kernel-Module jederzeit eingebunden werden. So beherrscht Linux heute fast jedes gängige Dateisystem, und es ist zum Beispiel möglich, dass Sie den SCSI-Festplattenturm eines MAC-OS-Rechners an eine Linux-Workstation anschließen, und alle Daten des MAC-OS lesen können.

Die Hardwaretreiber existieren wiederum als Kernel-Module und können zur Laufzeit, ohne Rechnerneustart, geladen oder entfernt werden. Sie regeln den Zugriff auf Festplatten, Netzwerkkarten, Schnittstellen und Motherboard.

- Der *User-Space*, oder auf Deutsch, der Anwendungsbereich. Dieser ist hermetisch vom Kern getrennt und auch die Anwendungen sind voreinander geschützt. Da der Kern sehr stabil ist, stürzt Linux praktisch nicht ab, was für eine stabile Mehrbenutzerumgebung unabkömmlich ist. Die wichtigsten Akteure im Benutzerbereich sind:

Die Bibliotheken oder auf englisch *Shared Libraries*. Damit Programmierer, z.B. beim Ausgeben von Text auf einem Terminal, nicht bei jeder Anwendung das Rad neu erfinden müssen, sammelt man diese Funktionen in dynamisch (also zur Laufzeit eines Programms) ladbare Funktionsbibliotheken. Diese teilen sich dann mehrere Anwendungen, die Funktionen dieser Bibliothek verwenden. Das prominenteste Beispiel ist mit Sicherheit die C-Bibliothek `libc.so`, die die Standardfunktionen der Sprache C enthält. Da die meisten Anwendungsprogramme unter Linux in dieser Sprache oder in

C++, das externe C-Bibliotheken mitbenutzen kann, geschrieben sind, ist sie ein sehr wesentlicher Bestandteil des Systems.

Die Shell bietet klassischerweise die Benutzerschnittstelle in Form einer Kommandozeile. Den Namen verdankt sie der Tatsache, dass sie sich aus Benutzersicht wie eine „Schale“ um den Kern schmiegt, um für den Benutzer das System bedienbar zu machen.

Aus der Shell heraus können nun beliebig viele Anwendungen, oder auch neue Shells, so genannte *Subshells*, gestartet werden.

Die Anwendungen oder *Prozesse* sind Instanzen von Programmen, die auf der Festplatte gespeichert sind. Ein Programm kann also mehrmals gestartet sein. Jede dieser Instanzen läuft unabhängig von den anderen Instanzen in ihrem eigenen Speicherbereich, wie bereits oben erklärt. Auch die Shell ist nichts anderes als ein ganz normaler Prozess. Jeder Benutzer, der sich auf dem System anmeldet, bekommt einen eigenen Shell-Prozess gestartet, der die Instanz einer Shell ist. Diese erste Shell nach der Anmeldung nennt man *Login-Shell*.

1.2.2 Hardware

Beim Einsatz als Einzelrechner: Damit Sie sich an konkreten Werten orientieren können, finden Sie in der nachfolgenden Tabelle einige Beispielkonfigurationen zusammengestellt.

Installation	Benötigter Plattenplatz
Minimum	bis 200 MB
Klein	200 MB bis 500 MB
Mittel	500 MB bis 1,2 GB
Groß	1,2 GB bis 6 GB

Beim Einsatz als Fileserver: Hier kommt es auf die Festplattenperformance an. SCSI-Geräten sollte unbedingt Vorzug gegeben werden. Achten Sie auch auf Leistungsfähigkeit der Platten und des verwendeten Controllers.

Ein Fileserver bietet die Möglichkeit, Daten zentral zu halten. Hier kann es sich um Benutzerverzeichnisse, eine Datenbank oder sonstige Archive handeln. Der Vorteil ist eine wesentlich einfachere Administration.

Falls der Fileserver ein größeres Netz bedienen soll (ab 20 Usern), wird die Optimierung des Plattenzugriffs essentiell.

Angenommen, Sie möchten einen Linux-Fileserver aufbauen, der 20 Benutzern ihre Heimverzeichnisse (Home) zur Verfügung stellen soll. Sie wissen, jeder Benutzer wird maximal 1 GB für seine persönlichen Daten in Anspruch nehmen. Damit reicht eine 20 GB-Platte, welche einfach unter `/home` gemountet wird.

Haben Sie 40 Benutzer, so wäre eine 40 GB-Platte notwendig. Besser wäre es in diesem Fall jedoch, /home auf zwei 20 GB-Platten aufzuteilen, da sich diese dann die Last (Zugriffszeit) teilen.

Beim Einsatz als Applikations-Server: Ein Applikations-Server ist in der Regel ein leistungsstarker Rechner, der berechnungsintensive Aufgaben im Netz erfüllt. Solch ein Rechner verfügt normalerweise über einen etwas größeren Hauptspeicher (ab 512 MB RAM). Um schnelleren Plattendurchsatz zu erreichen, können mehrere Swap-Bereiche eingerichtet werden, wobei jede Swap-Partition maximal 2 GB groß sein darf, und es dürfen nur maximal 8 Swap-Partitionen eingerichtet werden.

1.2.3 Systemkern (Kernel)

Der Kernel ist der Teil des Betriebssystems, der direkt mit der Hardware des Computers zusammenarbeitet. Er stellt die Dienstleistungen zur Verfügung, die die verschiedenen Programme benutzen können, und isoliert damit diese Programme von der zugrundeliegenden Hardware.

Die Hauptfunktionen des Kernels sind die Verwaltung des Speichers, die Steuerung des Zugriffs auf den Computer, die Verwaltung des Filesystems, die Behandlung von Unterbrechungen („Interrupts“), die Fehlerbehandlung, die Durchführung von Ein- und Ausgabeanweisungen, die es dem Computer erlauben, mit Terminals, Speichermedien und Druckern zusammenzuarbeiten, und die Aufteilung der Computerressourcen, wie zum Beispiel der CPU oder der Ein- und Ausgabegeräte, auf die einzelnen Benutzer.

Die Programme kommunizieren mit dem Kernel über ungefähr 100 Systemaufrufe. Die Systemaufrufe lösen im Kernel die Ausführung verschiedener Funktionen für das Programm aus, so, zum Beispiel, das Öffnen einer Datei, das Schreiben in eine Datei, das Lesen von Informationen über eine Datei, die Ausführung eines Programms, den Abbruch eines Prozesses, eine Prioritätsänderung eines Prozesses oder das Lesen der Uhrzeit.

Die verschiedenen Implementierungen von UNIX System V haben alle kompatible Systemaufrufe mit jeweils gleichen Funktionalitäten. Der interne Aufbau, der die Funktion des Systemaufrufes bewerkstelligt (normalerweise in „C“ geschrieben) und die Systemarchitektur der Implementierungen müssen dazu allerdings kaum eine Ähnlichkeit aufweisen.

1.2.4 Die Shell und ihre Bedeutung

Unter UNIX wird der Kommandointerpreter als Shell bezeichnet (entspricht in etwa der COMMAND.COM unter DOS). Die Shell ist also jenes Programm, in dem die Kommandozeilen eingegeben und ausgeführt werden. Die Shell kann auch zur Aus-