

# Unix im Überblick

Ziel der Veranstaltung ist es, das Betriebssystem Unix aus der Sicht eines Systementwicklers kennen zu lernen. D.h. der Umgang mit Unix als Schnittstelle zu einer komplexen Hardware zur Entwicklung von Anwendungs- und Systemprogrammen steht im Vordergrund.

Dazu werden zunächst die wichtigsten Bestandteile von Unix aufgezeigt:

- die **Shell**,
- das **Dateisystem** und
- das **Prozesskonzept**

Dann wird auf die **Shell** näher eingegangen und **Shellprogrammierung** behandelt. Shellprogramme bilden in der Anwendungsentwicklung oft die Schnittstelle von kommerziellen Anwendungen, wie SAP/R3 zu anderen Anwendungen.

Am Schluss wird die systemnähere Programmierung in C verdeutlicht. Die **C Programmierung** hat sich im Unix Umfeld als Bindeglied zwischen Anwendungssystemen und Hardwarekomponenten etabliert. C Programmierung im Unix Umfeld stellt weiterhin den Standard bei der Schnittstellenprogrammierung dar.

# Inhalt

<a href="#">1. Einleitung.....</a>	<a href="#">3</a>
<a href="#">1.1. Geschichte von Unix.....</a>	<a href="#">5</a>
<a href="#">2. Überblick.....</a>	<a href="#">8</a>
<a href="#">3. Anmelden am Unix System.....</a>	<a href="#">11</a>
<a href="#">4. Die Shell.....</a>	<a href="#">12</a>
<a href="#">5. Dateien in Unix.....</a>	<a href="#">16</a>

## 1. Einleitung

Software kann grob in zwei Arten eingeteilt werden:

- Systemsoftware, die den Rechner selbst steuert und
- Anwenderprogramme, die die Aufgaben der Anwender lösen.

Das wichtigste Systemprogramm ist das **Betriebssystem**, es kontrolliert die Ressourcen des Rechners und ist Basis für die Entwicklung der Anwenderprogramme.

Ein moderner Rechner besteht u.a. aus den Komponenten:

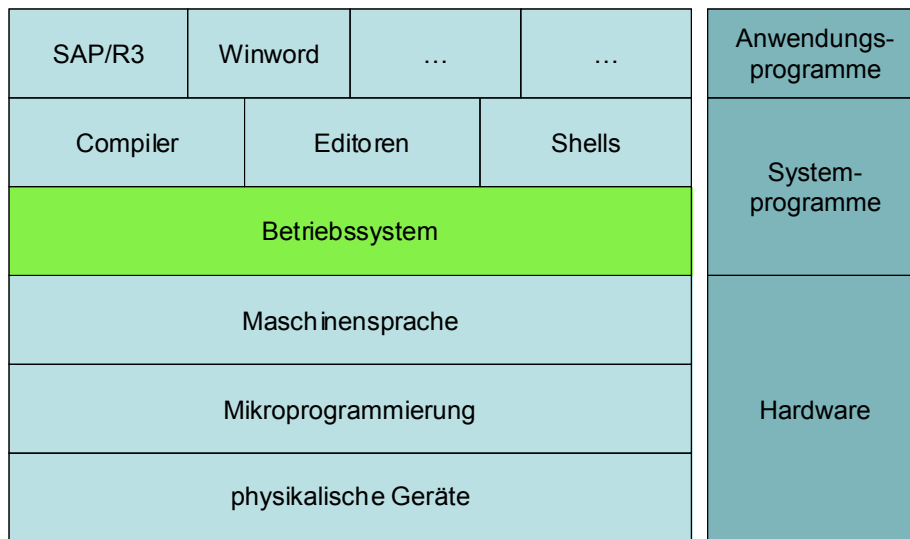
- ein oder mehrere Prozessoren,
- Hauptspeicher,
- Taktgeneratoren,
- Datenendgeräten,
- Plattenlaufwerken,
- Netzwerkkomponenten und
- DFÜ Einrichtungen.

Insgesamt also eine komplexe Hardware.

Ziel der Systemprogrammierung ist es, die Anwender von der Komplexität der Hardware zu bewahren. Um der Komplexität Herr zu werden, kann man Rechner in **Schichten** einteilen.

Die nachfolgende Abbildung zeigt das Schichtenmodell.

Schichtenmodell



Basis des Rechners ist die Hardwareebene. Die untere Schicht umfasst die physikalischen Geräte mit integrierten Bauelementen, Leitungen, Röhren usw. Darüber liegt eine einfache Schicht von Software, die unmittelbar auf die physikalischen Geräte einwirkt. Diese so genannten Mikroprogramme sind fest in ROM Speichern verdrahtet, es sind eigentlich Interpreter für Maschinenanweisungen, wie ADD, MOVE, JUMP. Die Menge aller solch elementaren Maschineninstruktionen nennt man Maschinsprache.

Oberhalb der Schicht Betriebssysteme befinden sich Werkzeuge und Dienstprogramme, die zur Entwicklung von Anwendungssoftware benötigt werden, wie z.B. C Compiler, Java Entwicklungsumgebungen, der vi oder die bash.

Man erkennt, dass das Betriebssystem die Verbindung zwischen Hardware und Programmen für Entwicklung und Betrieb von Anwendungssystemen darstellt.

## 1.1. Geschichte von Unix

UNIX wurde schon im Jahr **1969** bei den **Bell-Laboratorien** entwickelt, aber nicht als zu vermarktendes Produkt, sondern als Hilfsmittel für die Programmierung.

Ausgangspunkt war das Projekt "space travel" (Simulation der Bewegung der wichtigsten Himmelskörper im Sonnensystem). Die Programmentwicklung musste auf einer GE645 unter dem Betriebssystem **MULTICS** durchgeführt werden. Das lauffähige Programm wurde mittels Lochstreifen auf eine PDP7 übertragen, wo es in Produktion gehen sollte. Auf dem Zielsystem war keine Programmentwicklungsumgebung vorhanden. **Ken Thompson** entwickelte daraufhin UNIX, um die Programme direkt auf der PDP7 entwickeln zu können.

Die erste Vorstufe des Betriebssystems UNIX war sehr einfach gehalten. Zu dieser Zeit war es nicht nur ein Singleuser-System, sondern bot auch nur eine sehr einfache Benutzerschnittstelle. Der Name UNIX ist eine Abwandlung vom Namen des Multiuser-Systems MULTICS (**unique MULTICS**).

UNIX wurde im Gegensatz zu den meisten bis dahin entwickelten Betriebssystemen von einer kleinen Gruppe weiterentwickelt, deren erklärtes Ziel es war, eine für die Softwareentwicklung optimal angepasste Programmierumgebung zu schaffen. Bis dahin wurden die meisten Betriebs-

systeme an die Hardware eines Rechners angepasst. Im **Vordergrund** stand die optimale **Auslastung** der **Betriebsmittel**.

Diese für die damalige Zeit ungewöhnliche Konzeption eines Betriebssystems führte zur Entwicklung eines außerordentlich fortschrittlichen Systems. Dies ist auch daran zu erkennen, daß neuere, ebenfalls zum Standard avancierte Systeme, wie z.B. MS-DOS, einige Komponenten von UNIX übernommen haben.

UNIX hätte sich trotz aller Neuerungen nicht in dem Maße durchsetzen können, wenn sich nicht die Sprache, in der es geschrieben wurde, geändert hätte. Die erste Version von UNIX wurde vollständig in der Maschinensprache der PDP7 geschrieben. Ein solches System in Maschinensprache ist zwar sehr schnell, doch nicht portabel. Dies änderte sich mit der Entwicklung der Programmiersprache B.

UNIX wurde 1970 in die **Hochsprache B** umgeschrieben und somit portabel. B wurde von **Dennis Ritchie erweitert**. Es entstand **die Programmiersprache C**, die als höhere Programmiersprache annähernd über die Effizienz einer Assemblersprache verfügt. Der Beweis für die Portabilität des in C geschriebenen Betriebssystems wurde 1977 angetreten, als UNIX auf der Interdata 8/32 implementiert wurde.

Thompson und Ritchie erhielten 1983 den ACM A.M.Turing Award (den sogenannten "Nobelpreis" der Informatik) für "die geschickte Auswahl von wenigen Schlüsselgedanken und deren eleganter Implementierung bei der Entwicklung von UNIX und C".

Heute gibt es das Betriebssystem UNIX auf Computern aller Größenordnungen vom PC bis zum Supercomputer. Der Verbreitung ist weiterhin zugute gekommen, daß AT&T, zu denen die Bell-

Laboratorien gehören, die Vermarktung von UNIX übernommen hatte. Mit der Version SYSTEM V sind alle nachfolgenden Versionen aufwärtskompatibel.

XENIX von SCO (Santa Cruz Operation) ist eine verbreitete UNIX-Implementierung für die Prozessoren der INTEL 80x86-Familie. Das Produkt ist durch SCO UNIX, das auf AT&T SYSTEM V 3.2 basiert, abgelöst worden. SCO UNIX ist das erste nicht von AT&T vertriebene Betriebssystem mit Namen UNIX.

386/IX von INTERACTIVE SYSTEMS und EURIX (deutschsprachig) von ComFood sind weitere wichtige UNIX-Derivate für PCs. Diese Produkte erhalten zunehmend Konkurrenz durch freiverfügbare (public domain) UNIX-Implementierungen (z.B. LINUX).

Das von AT&T lizenzierte SYSTEM V 4.0 führt alle bedeutenden UNIX-Entwicklungen (SYSTEM V, BSD: Berkeley Software Distribution - wichtige Weiterentwicklungen sind aus dieser Universitätsentwicklung hervorgegangen, z.B. die virtuelle Speicherverwaltung - ,SunOS: die UNIX-Implementierung von SUN, XENIX) zusammen. Besonderer Wert wird hier auf die Erfüllung von Sicherheitsanforderungen entsprechend dem Orange Book des DoD (Department of Defence: amerikanisches Verteidigungsministerium) gelegt. Dieses ist ein abgestufter Anforderungskatalog, der nach der Farbe des Einbandes benannt worden ist. Systemsoftwarekomponenten werden anhand der aufgeführten Kriterien zertifiziert.

Im Workstationbereich dominiert UNIX. Verbreitete Versionen sind hier AIX (IBM), HP-UX (HEWLETT-PACKARD), Sinix (SIEMENS) und SunOS (SUN).

Mittlerweile hat sich das freie Linux mit seinen Derivaten Debian, Suse, Redhat uvm etabliert.

## 2. Überblick

UNIX ist ein Mehrbenutzer-Betriebssystem, das vornehmlich zur **Programmentwicklung** konzipiert worden ist und im **Dialogbetrieb** eingesetzt wird. Es kann standardmäßig **nicht** für **Real-timeanwendungen** verwendet werden, da für Prozessunterbrechungen keine festen Abarbeitungszeiten eingehalten werden können.

UNIX besteht aus mehreren **Komponenten**, die **schichtenförmig angeordnet** sind. Der **Kernel** als unterste Schicht umfasst die Steuerprogramme. Er erledigt die Ein- und Ausgabe von Programmen, verwaltet den Haupt- und den Hintergrundspeicher, verwaltet die Prozesse, sorgt für den Datenschutz, legt die Logdaten ab und ist für vieles mehr zuständig. Der UNIX-Kernel ist kein Mikrokern, sondern immer noch eine monolithisches Gebilde. Der Kernel umfasst ca. 10.000 Programmzeilen C-Quellcode und weniger als 1.000 Programmzeilen Assembler-Code.

Über dem Kern liegt die Schicht der **Benutzerprozesse**, die einerseits mit dem Anwender kommunizieren, andererseits Aufträge an den Kernel geben und von diesem überwacht werden. Die Anwenderprozesse rufen die Systemdienste mit Hilfe von "system calls" auf. Der Kernel bietet mehr als 70 verschiedene Systemaufrufe, die auf allen UNIX-Systemen im Wesentlichen gleich und insbesondere auch unabhängig von den angeschlossenen Geräten sind. Für die Systemaufrufe gibt es Standardisierungsbestrebungen, wie z.B. POSIX: portable operating system interface UNIX. Zu den Anwenderprozessen gehören nicht nur die Anwenderprogramme, sondern auch der UNIX-Kommandointerpreter (shell) und die Dienstprogramme.

Die **Shell** legt sich wie eine Schale um den Betriebssystemkern und ermöglicht es den Benutzern, über Kommandos die Systemleistungen abzurufen. Sie kann wie jedes andere Programm

gehandhabt werden. Für viele Anwendungen gibt es inzwischen eigene Kommandointerpreter, so z.B. Menü-Shells für die Nutzung der Systeme durch Nichtprogrammierer.

Die Shell ist zusätzlich ein einfaches und zugleich mächtiges Werkzeug für den Programmierer, denn mit wenigen Kommandos kann man durch die Nutzung von Jokerzeichen, Variablen (nur vom Typ Zeichenkette), Prozeduren und den von höheren Programmiersprachen her bekannten Anweisungen für Bedingungen (if-then-else, case) und Schleifen (while, for) komplexe Abläufe festlegen, ohne ein eigenes Programm schreiben zu müssen. Damit können auf einfache Art und Weise, auf die speziellen Anwendungen zugeschnittene neue Kommandos erstellt werden.

**Kommandosequenzen**, die man häufiger eingibt, können in Dateien abgelegt und mit einem einzigen Kommando ausgeführt werden (Kommandoprozeduren). Kommandos und Prozeduren können im Hintergrund gestartet werden.

Außer für die eingebauten Kommandos wird für jedes eingegebene Kommando ein **Dienstprogramm** gestartet. Es gibt mehr als 200, die in der Programmiersprache C geschrieben sind.

Unter UNIX kann die **Ein- und Ausgabe** von Programmen einfach umgelenkt werden. So kann die Ausgabe jedes Programms in eine Datei, auf den Bildschirm oder einen Drucker gelenkt werden, ohne dass das Programm sich darauf einstellen oder hiervon Kenntnis haben muss. Ebenso kann die Eingabe nicht nur vom Bildschirm, sondern auch aus einer Datei heraus erfolgen. Außerdem kann die Ausgabe eines Programms die Eingabe eines anderen sein, wobei beide Programme zur gleichen Zeit ablaufen. Die dafür notwendige Synchronisation wird vom Kernel vorgenommen (Synchronisationsmechanismus Pipe).

Eine weitere wesentliche Eigenschaft von UNIX ist das **hierarchische Dateikonzept** mit einem differenzierten Dateischutz (für den Benutzer, eine Gruppe und alle Systembenutzer) und mit

montierbaren und demontierbaren Geräten, die selber als Dateien angesprochen werden. Dateien und Prozesse sind die Objekte, mit denen das UNIX-Betriebssystem arbeitet. Ein/Ausgaben für Dateien und Geräte sowie selbst zwischen Prozessen (über Pipes) sind einfach und effizient implementiert und für alle Bereiche einheitlich.

Viele der UNIX-Eigenschaften finden sich in den Betriebssystemen MS-DOS und OS/2, bezogen auf einen Singleuser-Betrieb, wieder.

### 3. Anmelden am Unix System

Unix ist ein Multiuser- und Multitasking-System. Das bedeutet, dass in einem bestimmten Augenblick sowohl mehrere Benutzer auf einer Unix-Maschine gleichzeitig arbeiten können, als auch, dass jeder einzelne dieser Benutzer mehrere Programme aufrufen kann, die alle zur gleichen Zeit ausgeführt werden.

Damit jeder dieser Benutzer seine Daten vor dem Zugriff der anderen Benutzer schützen kann, muss man sich, bevor man mit einem Unix-System arbeiten kann, erst einmal **anmelden**, das heißt, einen speziellen Benutzernamen und ein Passwort eingeben. Dadurch erfährt das System, welcher Benutzer da gerade die Arbeit aufnehmen möchte, und kann diesem Benutzer seine persönliche Arbeitsumgebung (inclusive aller privater Daten) zur Verfügung stellen.

```
login: benutzername  
password: passwort
```

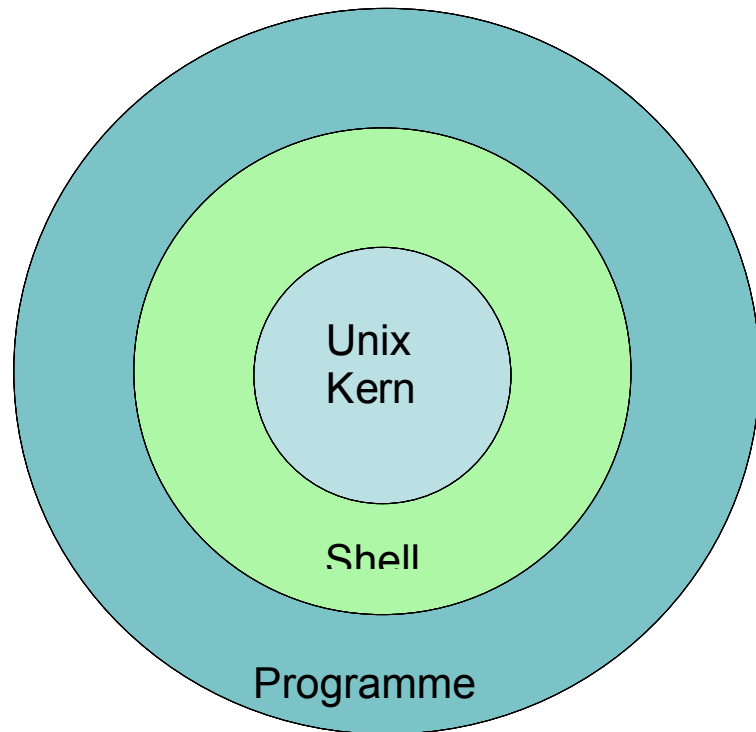
Man erkennt die erfolgreiche Anmeldung daran, dass die Eingabeaufforderung des Systems (genauer der Shell), der sog. *Prompt* erscheint:

```
$
```

## 4. Die Shell

Wenn man nach dem Unix Prompt ein Kommando eingibt, so antwortet nicht das Betriebssystem, sondern der Kommandointerpreter, die Shell. Sie ist die Schnittstelle zum eigentlichen Betriebssystem.

Die Shell bildet eine Schale, die den Unix Kern umgibt und so den Zugriff auf den Kern vereinfacht.



Die Shell ist selbst ein normales Programm, d.h. man kann beliebige Kommandointerpreter verwenden.

Übliche Shells sind

- sh, die Bourne Shell (Standard),
- csh, die C Shell,
- ksh, die Korn Shell,
- bash.

Wir werden hier die Korn Shell verwenden. Alle gezeigten Programme werden aber auch auf der bash ablauffähig sein.

Eine neue Shell (hier die csh) wird gestartet durch:

```
$ csh
%
```

und verlassen durch:

```
% exit
$
```

oder

```
% ^D
$
```

Die Kommandozeilen für die Shell sind allgemein wie folgt aufgebaut:

Kommandoname Parameter1 Parameter2 , , ,

Kommandos sind ausführbare Programme; die Parameter sind Größen, die die Ausführung der Kommandos beeinflussen; dabei werden Optionen meistens mit einem vorangestellten Minuszeichen gekennzeichnet.

Ein Kommando zur Ausgabe des Datums und der aktuellen Uhrzeit ist „date“.

```
$ date  
Son Dez 23 16:36:11 CET 2001  
$
```

Ein Kommando zum zählen von Worten, Zeilen und Zeichen einer Datei ist wc.

```
$ wc /etc/passwd  
  46      104    2430 /etc/passwd  
$  
$ wc -l /etc/passwd  
  460 /etc/passwd  
$
```

## Hörsaalübung

Melden Sie sich mit Ihrem „User“ (u4e01 – u4e32) am Linux-Rechner mit der IP-Adresse „142.100.42.200“ an.

Sie benötigen einen ssh-Client:

- Windows: <http://www.chiark.greenend.org.uk/~sgtatham/putty>
- Unix: ssh ist als Kommando verfügbar

## 5. Dateien in Unix

Informationen werden in Dateien gespeichert. Die einfachste Art, eine Datei zu erzeugen, ist die Verwendung von Programmen, die eine Ausgabe auf dem Bildschirm (Standardausgabe) erzeugen und dann diese **Ausgabe umlenken**.

Das Kommando **echo** reflektiert seine Argumente auf der Standardausgabe.

```
$ echo dieser text wird auf den Bildschirm geschrieben  
dieser text wird auf den Bildschirm geschrieben  
$
```

Durch Umlenkung (mittels „>“) wird eine Datei beschrieben.

```
$ echo dieser text wird in die Datei dat1 geschrieben > dat1  
$
```

Den Inhalt einer Datei kann man sich mit dem Kommando **cat** (**concatenate**) ansehen.

```
$ cat dat1  
dieser text wird in die Datei dat1 geschrieben  
$
```

**cat** kann als einfacher **Editor** verwendet werden, wenn man die Standardausgabe umlenkt und keine Datei als Parameter angibt.

```
$ cat > dat2
Dies ist ein Text mit 3 Zeilen
2. Zeile
Letzte Zeile
$ cat dat2
Dies ist ein Text mit 3 Zeilen
2. Zeile
Letzte Zeile
$
```

In Verzeichnissen kann man mehrere Dateien ablegen. Ein **Verzeichnis** ist selbst eine Datei; somit ist ein hierarchisches Dateisystem verfügbar.

Das Kommando zum auflisten des Inhalts eines Verzeichnisses ist **ls**.

```
$ ls -l
-rw-rw-r--  1 as          users          53 Dec 24 11:28 dat2
$
```

ls mit der Option l (long) listet den Verzeichnisinhalt im Long Format mit folgenden Informationen:

- Zugriffsrechte
- Anzahl Verweise auf die Datei
- Name des Besitzers der Datei

- Gruppe
- Größe in Bytes
- Datum und Uhrzeit der letzten Änderung
- Name der Datei

Zum Manipulieren von Verzeichnisinhalten existieren mehrere Kommandos, wie Kopieren, Umbenennen oder Löschen.

Kopieren:

```
$ cp date1 dat4  
$
```

Löschen:

```
$ rm dat1  
$
```

Umbenennen:

```
$ mv dat4 dat1  
$
```

Mit diesen elementaren Kommandos ist man in der Lage, sich an einem Unix System anzumelden und einfache Dateioperationen durchzuführen.

## Hörsaalübung

Das Kommando „mkdir“ soll verwendet werden, um ein Verzeichnis „d1“ anzulegen. Durch das Kommando „cd“ kann in ein Verzeichnis gewechselt werden.

Die Verwendung von Kommandos kann man aus den Manual-Seiten erfahren:

```
$ man mkdir
NAME
    mkdir -- make directories

SYNOPSIS
    mkdir [-pv] [-m mode] directory_name ...

DESCRIPTION
    The mkdir utility creates the directories named as operands, in the order
    specified, using mode rwxrwxrwx (0777) as modified by the current
    umask(2).

    The options are as follows:

    -m mode
        Set the file permission bits of the final created directory to
        the specified mode.  The mode argument can be in any of the
```

- Wechseln Sie in das gerade angelegte Verzeichnis d1
- Verwenden Sie das Kommando „cat“, um eine Datei „dat1“ anzulegen, die aus mehreren Zeilen besteht.
- Benennen Sie die Datei „dat1“ in „dat2“ um.