

MATERIALSAMMLUNG  
BETRIEBSSYSTEME UND  
BETRIEBSSYSTEMNAHE  
PROGRAMMIERUNG

Prof. Dr. Hans-Jürgen Buhl



Sommersemester 2008

Bergische Universität Wuppertal  
Fachbereich C — Mathematik



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Dokumentation zu System Calls . . . . .	9
1.2	API-Dokumentation . . . . .	15
<b>2</b>	<b>Betriebssysteme - eine Übersicht</b>	<b>19</b>
2.1	Aufgaben eines Betriebssystems . . . . .	19
2.2	Modularer Aufbau vom Windows 2000 . . . . .	21
2.2.1	HAL = Hardware Abstraction Layer . . . . .	21
2.2.2	<b>Der Kern des Windows-Kernels</b> . . . . .	24
2.2.3	<b>Kernel-Module</b> . . . . .	24
2.2.4	<b>Systemdienste und Systemschnittstelle</b> . . . . .	26
2.2.5	<b>Subsysteme</b> . . . . .	26
2.2.6	<b>Booting</b> . . . . .	26
2.3	System-Architektur von UNIX/Linux . . . . .	27
2.3.1	Der Linux Bootprozess . . . . .	28
2.3.2	Der Solaris Bootprozess . . . . .	28
2.3.3	Kompilierung und Installation von Kernel-Modulen . . . . .	29
2.3.4	Ein professionelles (offenes) UNIX-Betriebssystem: Solaris . . . . .	29
2.4	Virtuelle Betriebssysteme . . . . .	29
2.5	OpenSolaris . . . . .	30
<b>3</b>	<b>Partitionen und Dateisysteme</b>	<b>31</b>
3.1	Partitionieren unter UNIX . . . . .	36
3.1.1	Partitionieren in Solaris . . . . .	36
3.1.2	Dateisystem-Erzeugung unter Windows . . . . .	38
3.1.3	Dateisystemerzeugung unter Linux . . . . .	39
3.1.4	Anlegen eines UNIX-Dateisystems unter Solaris . . . . .	39
3.1.5	Zusammenfügen von UNIX-Partitionen: die Mount-Tabelle /etc/vfstab . . . . .	40
3.1.6	MultiBoot-System: Windows-Partitionen im Boot-Chooser . . . . .	40
3.2	Dateisysteme: ein erster Überblick . . . . .	41
3.3	Dateisystem-Übersicht . . . . .	44
3.3.1	Ältere Betriebssysteme . . . . .	44
3.3.2	Neuere Betriebssysteme . . . . .	44
3.4	Access Control Lists — ACLs . . . . .	46
3.5	Namensräume im Netzwerk / Verteiltes Rechnen . . . . .	46
3.6	Verteilte Betriebssysteme . . . . .	47

3.7	Das FAT-Dateisystem	48
3.7.1	Aufbau eines FAT-Dateisystems	48
3.7.2	FAT-Verzeichnisse	49
3.8	Das UNIX Filesystem UFS/UFS2	50
3.8.1	Inodes	52
3.9	Das WINDOWS-Dateisystem NTFS	54
3.9.1	NTFS Schlüssel-Designkonzepte	55
3.9.2	Die NTFS Volumen- und Dateistruktur	56
3.9.3	Recoverability/Wiederherstellbarkeit	56
3.10	Weitere UNIX- und LINUX-Dateisysteme	58
3.10.1	S5FS	58
3.10.2	FFS oder UFS	58
3.10.3	LFS	58
3.10.4	ext2 und ext3	58
3.10.5	XFS	59
3.10.6	JFS	59
3.10.7	Reiserfs	59
3.10.8	HFS+	59
3.10.9	ZFS	60
<b>4</b>	<b>RAID</b>	<b>61</b>
4.1	RAID 0	61
4.2	RAID 1	62
4.3	RAID 5	62
4.4	RAID 6	63
4.5	Kombinations-RAIDs	64
4.6	Metadevices (Software-RAID) unter Solaris	65
4.7	Als Alternative: Self-Managing mit Hilfe von zfs	68
<b>5</b>	<b>Fallstudie: Using Solaris containers for building a secure webshop</b>	<b>71</b>
<b>6</b>	<b>Secure global desktop/Virtual network computing/Thin clients</b>	<b>73</b>
<b>7</b>	<b>Kooperativ arbeitende Applikationen</b>	<b>75</b>
7.1	Neue implizit parallelisierende Compiler/Programmiersprachen: Fortress	75
7.2	OpenMP	75
7.3	Das Klonen von Prozessen	75
7.4	Posix Threads: Starten von (detachten) Threads	76
7.5	Client/Server-Prozesse	76
7.6	strace	77

# Abbildungsverzeichnis

1.1	sun.doc.com . . . . .	9
1.2	Suche <b>dup2</b> -Dokumentation . . . . .	10
1.3	Suchresultat: <b>dup2</b> . . . . .	10
1.4	<b>dup2</b> im Manual-Abschnitt 2 . . . . .	10
1.5	<b>dup2</b> -Dokumentation . . . . .	11
1.6	Auszug aus dem INDEX für System Calls . . . . .	12
1.7	Das Manual in der Konsole: <b>man</b> . . . . .	13
1.8	Dokumentation für <b>date</b> . . . . .	13
1.9	annähernde Suche . . . . .	14
1.10	Manual-Seiten in kde . . . . .	14
1.11	man pages section 3 — die API-Dokumentation: . . . . .	15
1.12	Baise Library Funktions . . . . .	15
1.13	passwd, printf, ... . . . . .	16
1.14	printf-Dokumentation . . . . .	17
2.1	Windows 2000 Systemarchitektur . . . . .	21
2.2	HAL-Varianten . . . . .	23
2.3	Linux-Architektur . . . . .	27
2.4	Solaris2-Architektur . . . . .	27
2.5	Solaris Kernel Overview . . . . .	28
3.1	Vier primäre Partitionen . . . . .	32
3.2	Partitionieren einer Solaris-Festplatte mittels <b>format</b> . . . . .	36
3.3	Partitionieren einer Solaris-Festplatte mittels <b>format</b> (Fortsetzung) . . . . .	37
3.4	Partitionierung unter Windows . . . . .	38
3.5	Partitionierung unter Linux . . . . .	39
3.6	Anlegen eines Dateisystems mit newfs . . . . .	39
3.7	Mount-Tabelle . . . . .	40
3.8	Windows Partitionen . . . . .	40
3.9	Windows Shares . . . . .	46
3.10	Verteilte Betriebssysteme . . . . .	47
3.11	Struktur und Aufbau einer FAT-Partition . . . . .	48
3.12	Aufbau einer UFS-Partition . . . . .	51
3.13	Adressenschema für die Datenblöcke eines Inodes . . . . .	53
3.14	NTFS-Partition . . . . .	54
3.15	NTFS-Komponenten . . . . .	56

4.1	RAID 0	61
4.2	RAID 1	62
4.3	RAID 5	62
4.4	RAID 6	63
4.5	Kombinations-RAID 10	64

# Tabellenverzeichnis

- 2.1 Windows 2000 Kernel-Module . . . . . 25
- 3.1 Datenträgertypen . . . . . 34
- 3.2 Partitionstabellenfelder . . . . . 35
- 3.3 File und Datei Attribute . . . . . 49
- 3.4 UFS Datei- und Dateisystemgrößen . . . . . 50
- 3.5 Inode-Aufbau . . . . . 52
- 3.6 NTFS File- und Verzeichnis-Attribut-Typen . . . . . 55



# 1 Einleitung

*System Calls* anstatt reinen Hochsprachencodes braucht man wegen:

- Vermeidung von Overhead (schnellerer Code)
- Benutzung von Diensten, die eine Hochsprache nicht bietet (File- und Record-Locking, Interprozeßkommunikation, etc.)

**Ein Beispiel: test-read.c**

```
#include <stdio.h>

int main(void)
{
    char buf[1024];
    int len;

    if ((len = read(5, buf, sizeof(buf))) == -1)
        perror("read");

    return 0;
}
```

liefert beim Programmaufruf:

```
./test-read
read: Bad file descriptor
```

**Beachte:** Bei jedem SystemCall muß der Erfolgsstatus überprüft und im folgenden Programmablauf berücksichtigt werden!

## Einige weitere Beispiele:

### lock.c

```
/* lock.c -- simple example of record locking */

#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

/* displays the message, and waits for the user to press
   return */
void waitforuser(char * message) {
    char buf[10];

    printf("%s", message);
    fflush(stdout);

    fgets(buf, 9, stdin);
}

/* Gets a lock of the indicated type on the fd which is passed.
   The type should be either F_UNLCK, F_RDLCK, or F_WRLCK */
void getlock(int fd, int type) {
    struct flock lockinfo;
    char message[80];

    /* we'll lock the entire file */
    lockinfo.l_whence = SEEK_SET;
    lockinfo.l_start = 0;
    lockinfo.l_len = 0;

    /* keep trying until we succeed */
    while (1) {
        lockinfo.l_type = type;
        /* if we get the lock, return immediately */
        if (!fcntl(fd, F_SETLK, &lockinfo)) return;

        /* find out who holds the conflicting lock */
        fcntl(fd, F_GETLK, &lockinfo);

        /* there's a chance the lock was freed between the F_SETLK
           and F_GETLK; make sure there's still a conflict before
```

```

        complaining about it */
    if (lockinfo.l_type != F_UNLCK) {
        sprintf(message, "conflict with process %d... press "
            "<return> to retry:", lockinfo.l_pid);
        waitforuser(message);
    }
}

int main(void) {
    int fd;

    /* set up a file to lock */
    fd = open("testlockfile", O_RDWR | O_CREAT, 0666);
    if (fd < 0) {
        perror("open");
        return 1;
    }

    printf("getting read lock\n");
    getlock(fd, F_RDLCK);
    printf("got read lock\n");

    waitforuser("\npress <return> to continue:");

    printf("releasing lock\n");
    getlock(fd, F_UNLCK);

    printf("getting write lock\n");
    getlock(fd, F_WRLCK);
    printf("got write lock\n");

    waitforuser("\npress <return> to exit:");

    /* locks are released when the file is closed */

    return 0;
}

```

## daytime.c

```
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>

int main() {
    struct timeval tv;
    struct timezone tz;
    time_t now;
    /* beginning_of_time is smallest time_t-sized value */
    time_t beginning_of_time = 1L<<(sizeof(time_t)*8 - 1);
    /* end_of_time is largest time_t-sized value */
    time_t end_of_time = ~beginning_of_time;

    printf("time_t is %d bits long\n\n", sizeof(time_t)*8);

    gettimeofday(&tv, &tz);
    now = tv.tv_sec;
    printf("Current time of day represented as a struct timeval:\n"
           "tv.tv_sec = 0x%08x, tv.tv_usec = 0x%08x\n"
           "tz.tz_minuteswest = 0x%08x, tz.tz_dsttime = 0x%08x\n\n",
           tv.tv_sec, tv.tv_usec, tz.tz_minuteswest, tz.tz_dsttime);

    printf("Demonstrating ctime() %s:\n", sizeof(time_t)*8 <= 32 ? "" :
           " (may hang after printing first line; press control-C)");
    printf("time is now %s", ctime(&now));
    printf("time begins %s", ctime(&beginning_of_time));
    printf("time ends %s", ctime(&end_of_time));

    exit (0);
}
```

## cat1.c

```
/* cat1.c -- simple version of cat */
#include <stdio.h>
#include <unistd.h>

/* While there is data on standard in (fd 0), copy it to standard
   out (fd 1). Exit once no more data is available. */

int main(void) {
    char buf[1024];
    int len;

    /* len will be >= 0 while data is available, and read() is
       successful */
    while ((len = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
        if (write(STDOUT_FILENO, buf, len) != len) {
            perror("write");
            return 1;
        }
    }

    /* len was <= 0; If len = 0, no more data is available.
       Otherwise, an error occurred. */
    if (len < 0) {
        perror("read");
        return 1;
    }

    return 0;
}
```

## mysleep.c

```
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

int usecsleep(int secs, int usecs) {
    struct timeval tv;

    tv.tv_sec = secs;
    tv.tv_usec = usecs;

    return select(0, NULL, NULL, NULL, &tv);
}

int main(){
    usecsleep(0, 500000);
}
```

**Aufgabe 1.1** *Bringen Sie die Beispiele auf Ihrem Computer zum Ablauf und versuchen Sie die Quellcodes Zeile für Zeile zu verstehen, indem Sie die Dokumentation gemäß Abschnitt 1.1 zu Rate ziehen.*

# 1.1 Dokumentation zu System Calls und zur (C)-Laufzeitbibliothek

Auf den Web-Seiten der Firma SUN kann man unter <http://docs.sun.com/app/docs/prod/3253> nach einzelnen System Calls und deren Bedeutung suchen und sich die entsprechende Dokumentation anschauen:

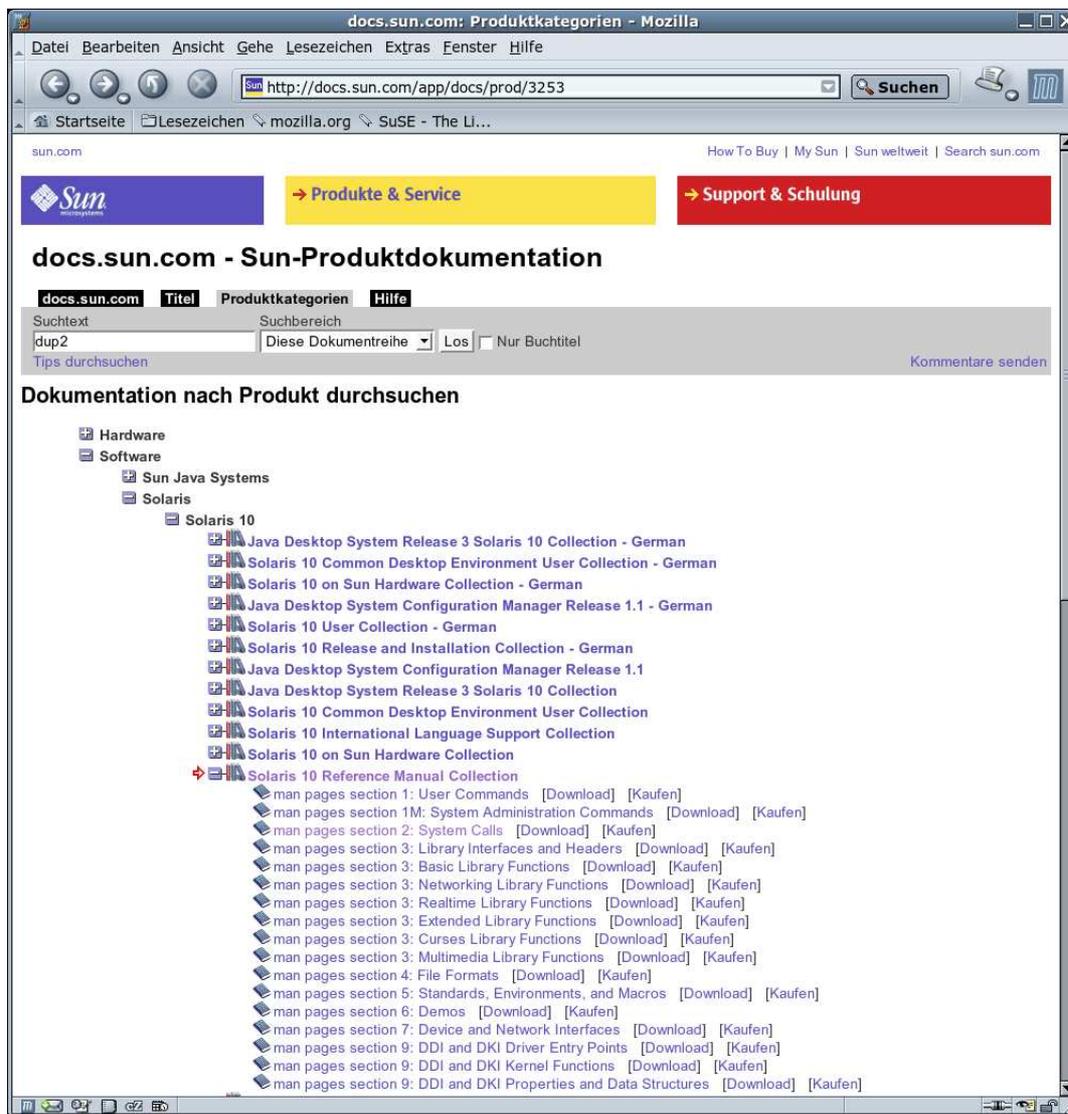


Abbildung 1.1: sun.doc.com

Als Beispiel suchen wir in den man-Pages (Manual-Seiten) nach dem System Call: **dup2**



Abbildung 1.2: Suche **dup2**-Dokumentation

Das Suchresultat liefert:

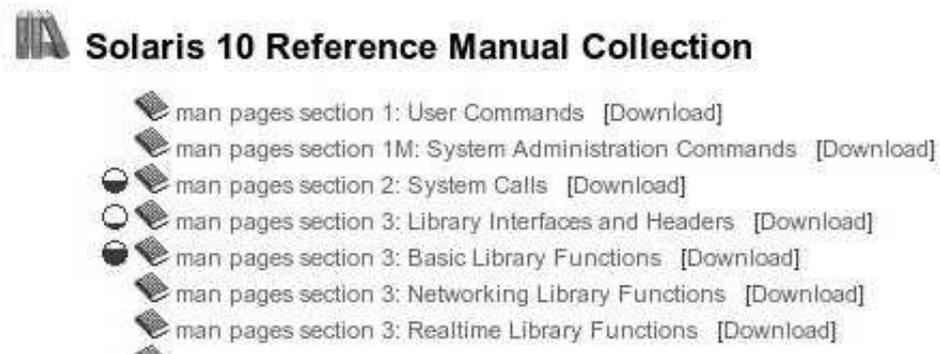


Abbildung 1.3: Suchresultat: **dup2**

In den **man pages section 2: System Calls** kann man sich die gewünschten Erläuterungen anzeigen lassen:

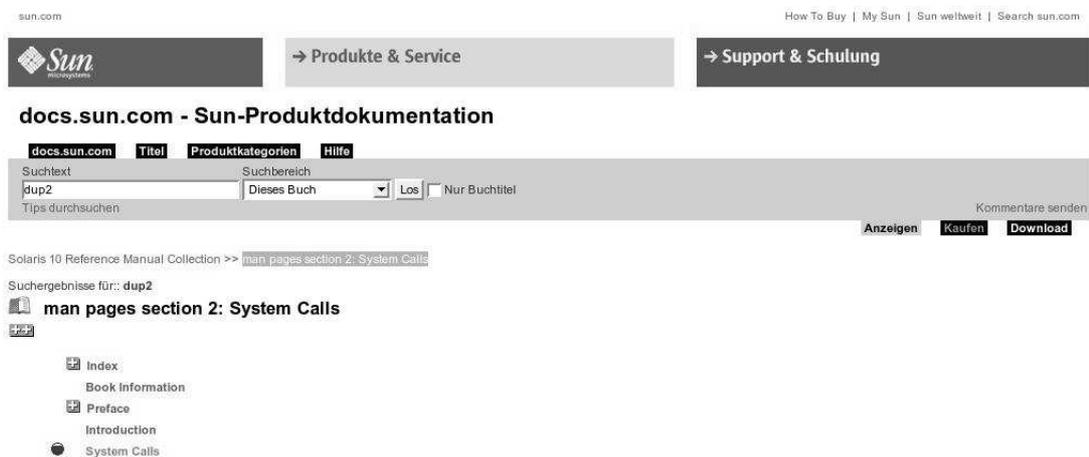


Abbildung 1.4: **dup2** im Manual-Abschnitt 2

**dup2** hat folgenden Eintrag:

# dup(2)

NAME | SYNOPSIS | DESCRIPTION | RETURN VALUES | ERRORS | ATTRIBUTES | SEE ALSO

## NAME

dup- duplicate an open file descriptor

## SYNOPSIS

```
#include <unistd.h>
int dup(int filides);
```

## DESCRIPTION

The `dup()` function returns a new file descriptor having the following in common with the original open file descriptor `filides`:

- same open file (or pipe)
- same file pointer (that is, both file descriptors share one file pointer)
- same access mode (read, write or read/write).

The new file descriptor is set to remain open across `exec` functions (see `fcntl(2)`).

The file descriptor returned is the lowest one available.

The `dup(filides)` function call is equivalent to:

---

```
fcntl(filides, F_DUPFD, 0)
```

---

## RETURN VALUES

Upon successful completion, a non-negative integer representing the file descriptor is returned. Otherwise, `-1` is returned and `errno` is set to indicate the error.

## ERRORS

The `dup()` function will fail if:

**EBADF**

The `filides` argument is not a valid open file descriptor.

**EINTR**

A signal was caught during the execution of the `dup()` function.

**EMFILE**

The process has too many open files (see `getrlimit(2)`).

**ENOLINK**

The `filides` argument is on a remote machine and the link to that machine is no longer active.

## ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Async-Signal-Safe

## SEE ALSO

`close(2)`, `creat(2)`, `cxxc(2)`, `fcntl(2)`, `getrlimit(2)`, `open(2)`, `pipe(2)`, `dup2(3C)`, `lockf(3C)`, `attributes(5)`

SunOS 5.9 Last Revised 28 Dec 1996

NAME | SYNOPSIS | DESCRIPTION | RETURN VALUES | ERRORS | ATTRIBUTES | SEE ALSO

Abbildung 1.5: dup2-Dokumentation

Der **INDEX** der Section 2 liefert eine sehr schöne Übersicht über die System Calls des UNIX-Betriebssystems.

Suchergebnisse für: **dup2**

 **System Calls**

A | B | C | D | E | F | G | H | I | J | K | L | M | N | C

**A**

- access(2) – determine accessibility of a file
- acct(2) – enable or disable process accounting
- acl(2) – get or set a file's Access Control List (ACL)
- adjtime(2) – correct the time to allow synchronization of the system clock
- alarm(2) – schedule an alarm signal
- audit(2) – write a record to the audit log
- auditon(2) – manipulate auditing
- auditsvc(2) – write audit log to specified file descriptor

**B**

- brk(2) – change the amount of space allocated for the calling process's data segment

**C**

- chdir(2) – change working directory
- chmod(2) – change access permission mode of file
- chown(2) – change owner and group of a file
- chroot(2) – change root directory
- close(2) – close a file descriptor
- creat(2) – create a new file or rewrite an existing one

**D**

- dup(2) – duplicate an open file descriptor

**E**

- exec(2) – execute a file
- exec(2) – execute a file
- execle(2) – execute a file

Suchergebnisse für: **dup2**

 **System Calls**

A | B | C | D | E | F | G | H | I | J | K | L | M | N | C

**A**

- access(2) – determine accessibility of a file
- acct(2) – enable or disable process accounting
- acl(2) – get or set a file's Access Control List (ACL)
- adjtime(2) – correct the time to allow synchronization of the system clock
- alarm(2) – schedule an alarm signal
- audit(2) – write a record to the audit log
- auditon(2) – manipulate auditing
- auditsvc(2) – write audit log to specified file descriptor

**B**

- brk(2) – change the amount of space allocated for the calling process's data segment

**C**

- chdir(2) – change working directory
- chmod(2) – change access permission mode of file
- chown(2) – change owner and group of a file
- chroot(2) – change root directory
- close(2) – close a file descriptor
- creat(2) – create a new file or rewrite an existing one

**D**

- dup(2) – duplicate an open file descriptor

**E**

- exec(2) – execute a file
- exec(2) – execute a file
- execle(2) – execute a file

Abbildung 1.6: Auszug aus dem INDEX für System Calls

**Aufgabe 1.2** Informieren Sie sich über den SystemCall `brk(0)`, der beim Start eines Programms vom Loader (für welchen Zweck?) benutzt wird.

Eine weitere Dokumentationsquelle für SystemCalls sind die lokalen man-pages. Der Aufruf erfolgt in einer Konsole mit dem Kommando man. Mittels man man erhält man zum Beispiel die Manualseite des Befehls man (für Manual).

```

nan(1)                                     Manual Hilfsprogramme                                     nan(1)

NAME
    man - Programm zum Einsehen der Online-Manuale

SYNTAX
    man [-acdhwtZU] [-m System[,...]] [-L locale] [-p Zeichenkette] [--M Pfad] [-P Pager] [-r Prompt] [-T Format] [-S Liste] [-c Erweiterung] [[Abschnitt] Seite ...] ...
    man -l [-tZ] [-p Zeichenkette] [-P Pager] [-r Prompt] [-T Format] Datei ...
    man -k Schlüsselwort ...
    man -f Seite ...

BESCHREIBUNG
    man ist der Manualbrowser des Systems. Jedes Argument Seite ist normalerweise der Name eines Programmes oder einer Funktion. Gefunden und angezeigt wird die Manualseite, die auf alle Argumente paßt. Wenn ein Abschnitt angegeben wird, sucht man nur in diesem Abschnitt der Manualseiten. Ohne Angabe eine explizite Angabe werden alle verfügbaren Abschnitte in einer vorher definierten Reihenfolge durchsucht. Wenn die Seite in mehreren Abschnitten vorkommt, wird nur die jeweils erste Seite angezeigt, die gefunden wird.

    Die folgende Tabelle zeigt die Nummern der Abschnitte der Manualseiten gefolgt vom Typ der dort zu findenden Seiten.

    1  Ausführbare Programme oder Shellbefehle
  
```

Abbildung 1.7: Das Manual in der Konsole: man

Nach Informationen des Kommandos date sucht man also mittels man date:

```

DATE(1)                                     User Commands                                     DATE(1)

NAME
    date - print or set the system date and time

SYNOPSIS
    date [OPTION]... [+FORMAT]
    date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
    Display the current time in the given FORMAT, or set the system date.

    -d, --date=STRING
        display time described by STRING, not 'now'

    -f, --file=DATEFILE
        like --date once for each line of DATEFILE

    --TIMESPEC, --iso-8601[=TIMESPEC]
        output date/time in ISO 8601 format. TIMESPEC='date' for date only, 'hours', 'minutes', or 'seconds' for date and time to the indicated precision. --iso-8601 without TIMESPEC defaults to 'date'.

    -r, --reference=FILE
        display the last modification time of FILE

    -R, --rfc-2822
        output RFC-2822 compliant date string

    -s, --set=STRING
        set time described by STRING

    -u, --utc, --universal
        print or set Coordinated Universal Time

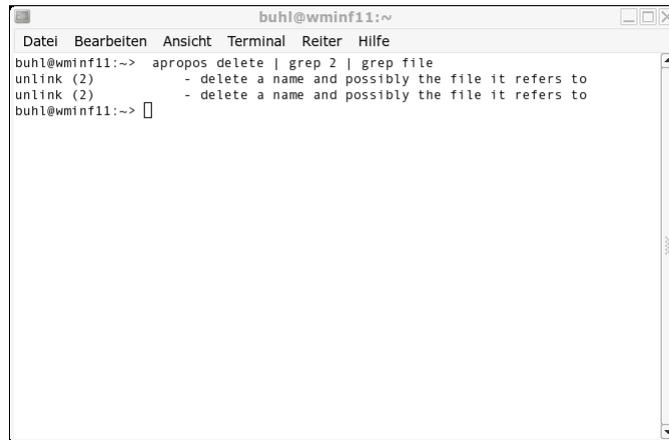
    --help
        display this help and exit

    --version
        output version information and exit

    FORMAT controls the output. The only valid option for the second form specifies Coordinated Universal Time. Interpreted sequences are:
  
```

Abbildung 1.8: Dokumentation für date

Bei Unkenntnis des genauen Kommandos für einen gesuchten Zweck, kann man mittels apropos einen Index durchsuchen:



```
buhl@wminf11:~  
Datei Bearbeiten Ansicht Terminal Reiter Hilfe  
buhl@wminf11:~> apropos delete | grep 2 | grep file  
unlink (2) - delete a name and possibly the file it refers to  
unlink (2) - delete a name and possibly the file it refers to  
buhl@wminf11:~> █
```

Abbildung 1.9: annähernde Suche

Schließlich kann man auch `khelppcenter man:read` (beziehungsweise `susehelp man:write`, `yelp man:link`, `konqueror man:unlink`) zur bequemen Suche nach Manual-Informationen benutzen:

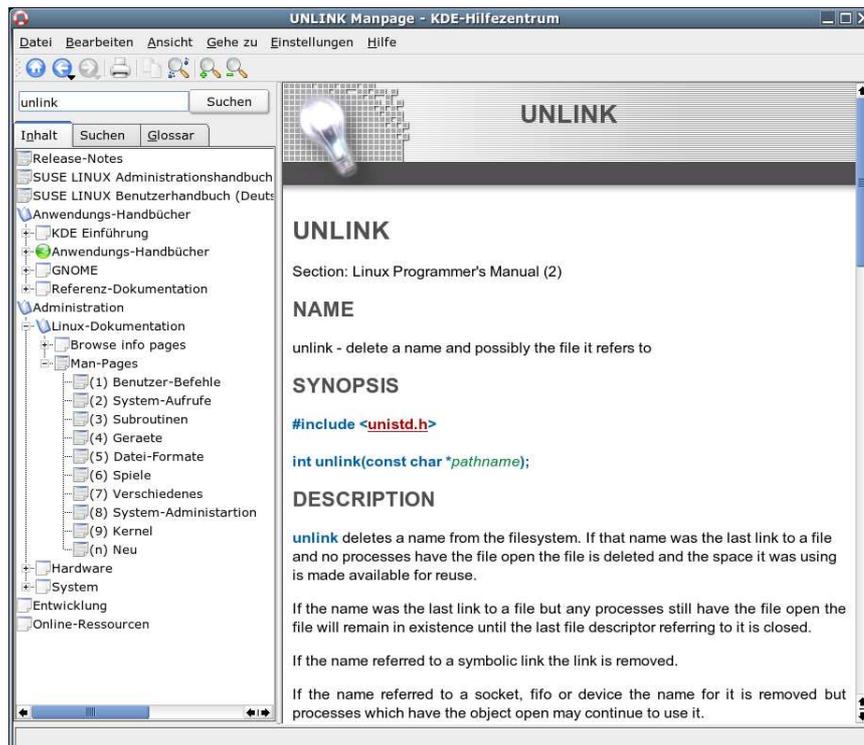


Abbildung 1.10: Manual-Seiten in kde

## 1.2 API-Dokumentation

Die **man pages section 3** enthalten die **API-Dokumentation** der Laufzeitbibliotheken:



Abbildung 1.11: man pages section 3 — die API-Dokumentation:

Beispiele:

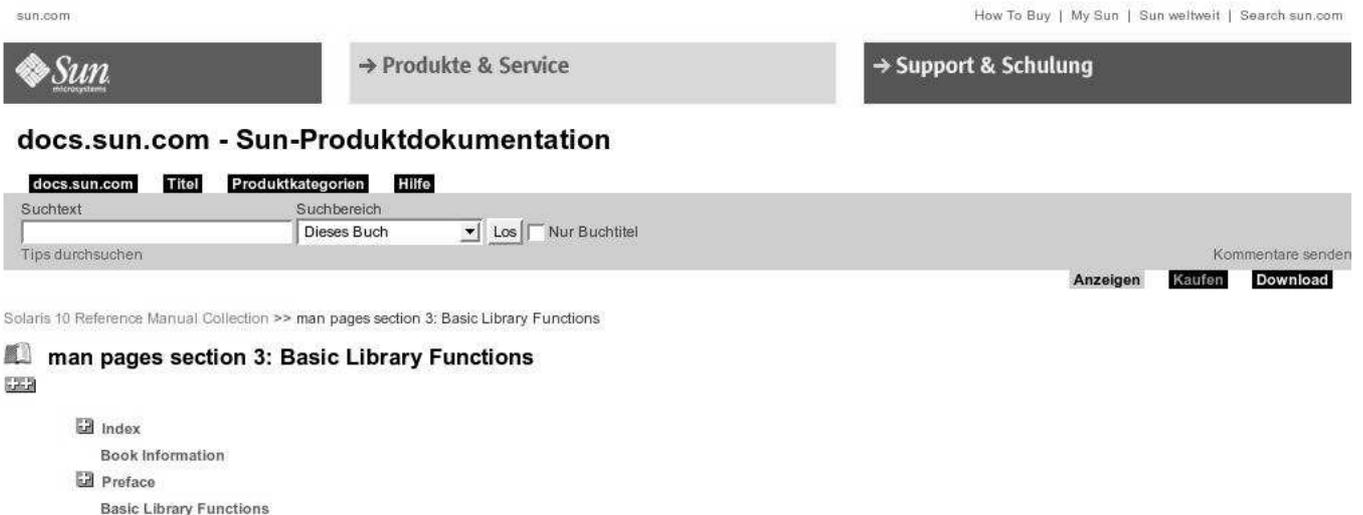


Abbildung 1.12: Basic Library Funktionen

Der INDEX unter P zeigt beispielsweise **psignal**, **printf**, **pthread** ... :

⋮

```
passwords
  get password entry from a file — fgetpwent ( 1 )
  get password entry from a file — fgetpwent_r ( 1 )
  get password entry in user database — endpwent ( 1 )
  get password entry in user database — getpwent ( 1 )
  get password entry in user database — getpwent_r ( 1 )
  get password entry in user database — getpwnam ( 1 )
  get password entry in user database — getpwnam_r ( 1 )
  get password entry in user database — getpwuid ( 1 )
  get password entry in user database — getpwuid_r ( 1 )
  get password entry in user database — setpwent ( 1 )
  get passwd entry from UID — getpw ( 1 )
  write password file entry — putpwent ( 1 )

⋮

print formatted output
  — fprintf ( 1 )
  — printf ( 1 )
  — snprintf ( 1 )
  — sprintf ( 1 )

print formatted output of a variable argument list
  — fprintf ( 1 )
  — printf ( 1 )
  — vsnprintf ( 1 )
  — vsprintf ( 1 )

print formatted wide-character output
  — fwprintf ( 1 )
  — swprintf ( 1 )
  — wprintf ( 1 )

printf — formatted output conversion ( 1 )
printf — print formatted output ( 1 )

⋮

psignal — system signal messages ( 1 ) ( 1 )
pthread_atfork — register fork handlers ( 1 )
pthread_attr_destroy — initialize and destroy threads attribute object ( 1 )
pthread_attr_getdetachstate — get or set detachstate attribute ( 1 )
pthread_attr_getguardsize — get or set the thread guardsize attribute ( 1 )
pthread_attr_getinheritsched — get or set inheritsched attribute ( 1 )
pthread_attr_getschedparam — get or set schedparam attribute ( 1 )
pthread_attr_getschedpolicy — get or set schedpolicy attribute ( 1 )

⋮
```

Abbildung 1.13: passwd, printf, ...

Die Dokumentation zu **printf** beginnt folgendermaßen:

## **printf(3C)**

NAME | SYNOPSIS | DESCRIPTION | RETURN VALUES | ERRORS | USAGE | EXAMPLES | ATTRIBUTES | SEE ALSO | NOTES

### **NAME**

printf, fprintf, sprintf, snprintf – print formatted output

### **SYNOPSIS**

```
#include <stdio.h>

int printf(const char *restrict format, /* args */ ...);
int fprintf(FILE *restrict stream, const char *restrict format, /* args */ ...);
int sprintf(char *restrict s, const char *restrict format, /* args */ ...);
int snprintf(char *restrict s, size_t n, const char *restrict format, /* args */ ...);
```

### **DESCRIPTION**

The printf() function places output on the standard output stream `stdout`.

The fprintf() function places output on the named output stream `stream`.

The sprintf() function places output, followed by the null byte (`\0`), in consecutive bytes starting at `s`; it is the user's responsibility to ensure that enough storage is available.

The snprintf() function is identical to sprintf() with the addition of the argument `n`, which specifies the size of the buffer referred to by `s`. If `n` is 0, nothing is written and `s` can be a null pointer. Otherwise, output bytes beyond the `n`-1st are discarded instead of being written to the array and a null byte is written at the end of the bytes actually written into the array.

⋮

Abbildung 1.14: printf-Dokumentation

**Aufgabe 1.3** Lesen Sie die Dokumentation zu `getchar()`, `getch()` sowie zu `cbreak()`. Stellen Sie kurz die Möglichkeiten dieser Funktionen dar.

Weitere Informationsquellen:

- <http://www.gnu.org/software/libc/manual>

**Aufgabe 1.4** Informieren Sie sich mittels der weiteren Informationsquellen über die POSIX-Funktionen und klassifizieren Sie diese nach dem Verwendungszweck.



# 2 Betriebssysteme - eine Übersicht

## 2.1 Aufgaben eines Betriebssystems

- Ablaufsteuerung

Starten und Beenden von Programmen/Prozessen, Abbrechen von Programmen (kill), ...

- Abstraktion

- techn. Einzelheiten vor Benutzern und Anwendungsprogrammen verstecken (z.B. HAL für Geräteschnittstellen, Chipsatz, usw.)

Vergleiche dazu: <http://support.microsoft.com/kb/q148659/#XSLTH3156121122120121120120>

- APIs (und ABIs) bereitstellen für

- Anwendungsprogramme (z.B. den System Calls, C-Laufzeitbibliotheken,...)
- Compilerhersteller (C-, C++-Calling-Konventionen, Word-Alignment in Structures, Speicherlayout in Unions, Register- und Stack-Nutzung zur Parameterübergabe bei Funktionsaufrufen, Name-Mangling bei überladenen Funktionen, ...)

Vergleiche: ABI LP64 in [http://www.unix.org/version2/whatsnew/lp64\\_wp.html](http://www.unix.org/version2/whatsnew/lp64_wp.html), Sun's C++ ABI in [http://developers.sun.com/tools/cc/articles/CC\\_abi/CC\\_abi\\_content.html](http://developers.sun.com/tools/cc/articles/CC_abi/CC_abi_content.html)

- Koordinierung und Zuteilung von Betriebsmitteln, (Multitasking)

- Prozessor(en)
- Hauptspeicher
- Hintergrundspeicher
- Zugriffsrechte auf E/A-Geräte, Dateisysteme, ...
- Initialisierung von Gerätetreibern
- Auftragsverwaltung (Jobs, Cronjobs, ...)
- Ausnahmebedinungsbehandlung, Interrupt-Tabellen initialisieren
- ...

Vergleiche Interrupt-Handler in:

<http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=interrupt+handler&action=Search>

- Schutz mehrerer nacheinander oder gleichzeitig arbeitenden Nutzer bzw. Prozesse voreinander (Multitasking/Multiuser)

- Schutz des Betriebssystems vor den Nutzern / Anwendungsprogrammen
- Bedienschnittstellen
  - Shells, awk, sed, Perl, Python, ...
  - GUIs
  - ...

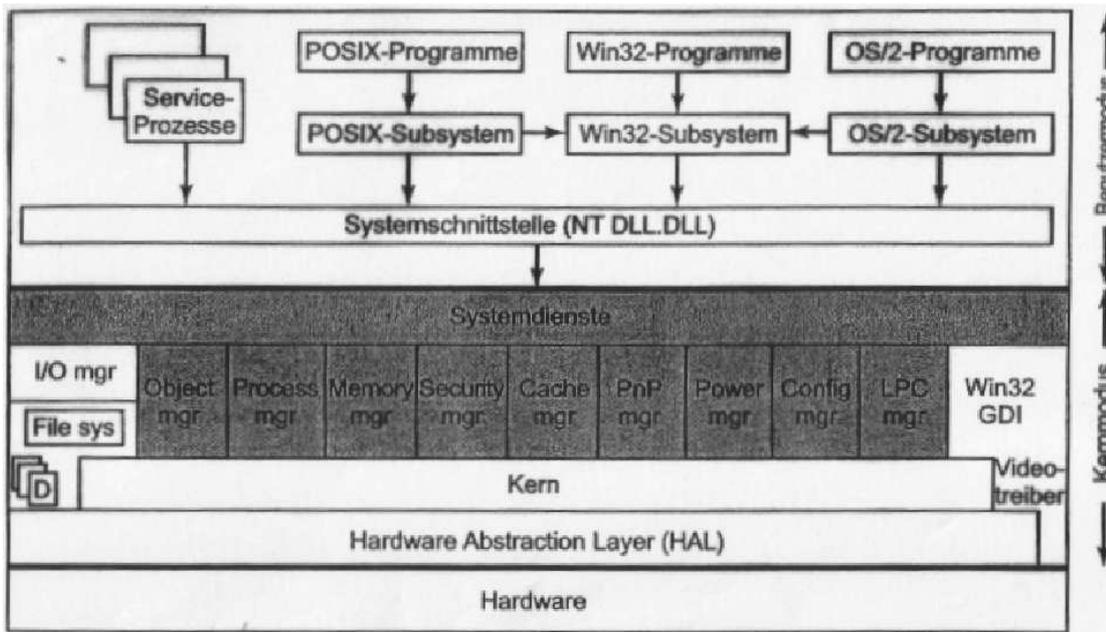


Abbildung 2.1: Windows 2000 Systemarchitektur

## 2.2 Modularer Aufbau vom Windows 2000 - Schichtweiser Aufbau (Layers)

Windows ist in Schichten aufgebaut, wobei die unteren (maschinennäheren) Schichten die Basis für die oberen (maschinenferneren) Schichten bilden. Die oberen Schichten können die Funktionen der unteren Schichten aufrufen, aber nicht umgekehrt.

### 2.2.1 HAL = Hardware Abstraction Layer (Hardwareabstraktions-Schicht)

- Chipsatz- und Motherbord-abhängige Unterschiede verstecken
- einheitlich Dienste bereitstellen:
  - Schnittstellen zu BIOS und BIOS-Setup
  - Abbildung **busbezogener Geräteadressen** auf **logische Adressen** (Beispiel: **con, aux, prn, com1, lpt1** in MSDOS)
  - nur HAL greift auf Gerätereister zu
  - Interruptroutinen setzen/Prioritäten festlegen
  - Vereinheitlichung dem Rest des OS gegenüber:
    - \* Systembus

- \* DMA-Controller
- \* Interrupt-Controller
- \* Memory Module
- \* System-Timer (Einheit: 0,1 ms)
- SMP (Symmetric Multiprocessing)

Benutzt werden standardisierte APIs wie z.B.: Win32, GDI, MDI, ...

Beispiele für Hardware-Abstraktionen sind:

- Druckertreiber
- Grafikschnittstellen (z.B. OpenGL oder DirectX)
- virtuelle Maschinen (VM)

DirectX umgeht HAL, um schnelle Multimedia-Prozeduren durch direkten Hardwarezugriff zu ermöglichen.

**HAL** ist für einen Großteil der Anwendungsprogramme und viele Systemprozesse notwendig. Wenige Bestandteile des Kernels und die Gerätetreiber kommunizieren direkt mit der Hardware, ohne HAL zu nutzen.

Auf einer Windows2000-System-CD liegen mehrere HAL-Versionen bereit, die je nach Hardware als

```
\WINNT\SYSTEM32\HAL.DLL
```

installiert werden.

In Windows NT gab es sogar Versionen für Digital Equipment Corporation (DEC) RISC-Workstations:

HAL NAME	COMPUTER MODEL NUMBER
halmikas.dll	= "Digital AlphaServer 1000 Family Uniprocessor"
hal0jens.dll	= "Digital DECpc AXP 150"
halsabmp.dll	= "Digital AlphaServer 2x00 4/xxx Family"
halsabmp.dll	= "Digital AlphaServer 2100A 4/xxx"
halavant.dll	= "Digital AlphaStation 200/400 Family"
halnonme.dll	= "Digital AXPPci 33"
halgs.dll	= "Digital Multia MultiClient Desktop"
halalcor.dll	= "Digital AlphaStation 500/600 Family"
halgampp.dll	= "Digital AlphaServer 2x00 5/xxx Family"
halgampp.dll	= "Digital AlphaServer 2100A 5/xxx"
haleb64p.dll	= "Digital AlphaPC64"
haleb164.dll	= "Digital Alpha EB164"
halavant.dll	= "Digital Alpha XL 233/266 Family"
halxl.dll	= "Digital Alpha XL 300/366 Family"
hallx3.dll	= "Digital AlphaStation 255"
halflex.dll	= "DeskStation Technology UniFlex and Raptor 3 Systems"
halrawmp.dll	= "Digital AlphaServer 4x00 5/xxx Family"
halrawmp.dll	= "Digital AlphaServer 4100-5 - Multiprocessor"
halalp.dll	= "Aspen Alpine/Telluride"
haltimbr.dll	= "Aspen Timberline/Summit"
halpinna.dll	= "Digital AlphaStation 600A 5/500"
halpinna.dll	= "Digital AlphaServer 1000 5/xxx Family"
halpinna.dll	= "Digital AlphaServer 1000a 5/xxx Family"
hallego.dll	= "Digital Alpha 21064A PICMG SBC"
*halmiata.dll	= "Digital Personal Workstation 433a, 500a"

Abbildung 2.2: HAL-Varianten

**Aufgabe 2.1** Informieren Sie sich in

[http://www.dewassoc.com/support/win2000/tshoot\\_hal.htm](http://www.dewassoc.com/support/win2000/tshoot_hal.htm)

über das Vorgehen, eine Windows-Installation für eine andere HW-Plattform zu modifizieren.

Vertiefung: [http://www.operating-system.org/betriebssystem/\\_german/index.htm](http://www.operating-system.org/betriebssystem/_german/index.htm)

## 2.2.2 Der Kern des Windows-Kernels

Der **KERN** ist der zentrale Bestandteil des Windows-Kernels. Er stellt die für alle weiteren Kernel-Module benötigte Funktionalität und Objekte bereit:

- Kontextwechsel von Prozessen
- Scheduling von Threads (Zeitscheibenwechsel, Interrupt-Behandlung, ...)
- Kontrollobjekte (z.B.: Prozessobjekte, deferred procedure calls — DCP —, asynchronous procedure calls — APC —, ...)
- DCP-Warteschlangen
- Dispatcher-Objekte wie Semaphore, Mutexe, Events, Stoppuhren, ...

## 2.2.3 Kernel-Module

Folgende **Dienste** werden von den Kernel-Modulen bereitgestellt:

- Prozess- und Auftragsverwaltung (Dispatcher; Ressourceneigentum)
- Speicherverwaltung (einschließlich virtuellem Hauptspeicher)
- Threadverwaltung (Scheduler; CPU-Zeitscheiben)
- Geräteverwaltung
- Dateisysteme
- Sicherheitsmanagement (Authentifizierung)

Die Module (Manager) im Einzelnen:

<b>Executive (=Kern-Oberteil)</b>	<b>Erleuterung</b>
- <b>E/A-Manager:</b>	geräteunabhängige Ein- und Ausgabe, hier befinden sich die Gerätetreiber, z.B. für FAT, NTFS, ...
- <b>Objektmanager:</b>	verwaltet Prozesse, Threads, Dateien, Verzeichnisse, E/A-Geräte, Semaphoren, ... (virtuelle Adressblöcke im Kernel-Adressraum: Anlegen / Freigabe)
- <b>Prozessmanager:</b>	erzeugt/vernichtet Prozesse und Threads
- <b>Speichermanager:</b>	realisiert virtuellen Speicher, bildet virtuelle Seiten auf physikalische Seiten ab, realisiert Schutzregeln
- <b>Sicherheitsmanager:</b>	stellt "orange book C2 standard security" bereit; Regeln für Authentifizierung (Anmeldung am System), Zugriffskontrolle (z.B. müssen virtuelle Seiten anfangs mit Nullen (0) gefüllt werden, ...) , ...
- <b>Cachemanager:</b>	enthält Funktionen für die Bereitstellung von Datei-Puffern, ...
- <b>PnP-Manager:</b>	"Plug and Play"-Funktionalität für USB-, PCMCIA-Geräte, ...
- <b>Powermanager:</b>	realisiert "Energy Star"-Anforderungen (Standbymodus des Monitors, Akku-Überwachung, Notabschaltung des Notebooks, ...
- <b>Konfigurationsmanager:</b>	realisiert die Registry-Funktionalitäten: (Schlüssel, Value)-Paare hinzufügen, löschen, abfragen
- <b>LPC-Manager: (Local Procedure Call)</b>	stellt hocheffiziente Prozesskommunikationsfunktionen zwischen Prozessen und OS-Subsystemen zur Verfügung
- <b>GDI: (graphical device interface)</b>	stellt geräteunabhängige Funktionen für Monitor- und Druckfunktionalitäten bereit, realisiert den Fenster-Manager, enthält Treiber für Grafikkarten

Tabelle 2.1: Windows 2000 Kernel-Module

## 2.2.4 Systemdienste und Systemschnittstelle

Darüber liegt die Schicht der **Systemdienste**, die die Aufrufschnittstellen auf die Executive-Funktionalitäten auch den Subsystemen (Win32, OS/2, POSIX) bzw. den Benutzerprozessen bereitstellt (wrapper-Funktionen, WIN-API, Native API NTDLL.DLL):

[http://en.wikipedia.org/wiki/Native\\_API](http://en.wikipedia.org/wiki/Native_API)

[http://en.wikipedia.org/wiki/Windows\\_API](http://en.wikipedia.org/wiki/Windows_API)

<http://www.microsoft.com/germany/technet/sysinternals/default.mspix>

## 2.2.5 Subsysteme

<http://www.microsoft.com/windowsserversystem/sfu/productinfo/overview/default.mspix>

<http://winntposix.sourceforge.net/>

**Aufgabe 2.2** Für welche Zwecke stellt Microsoft das POSIX-API zur Verfügung? Wo liegen im POSIX-API von Windows 2000 noch Defizite?

## 2.2.6 Booting

Informieren Sie sich unter

<http://dotnetjunkies.com/WebLog/unknownreference/articles/12284.aspx>  
über das Booten von Windows XP.

Welche Unterschiede treten in Vista/Server 2008 auf?

[http://en.wikipedia.org/wiki/Windows\\_Vista\\_startup\\_process](http://en.wikipedia.org/wiki/Windows_Vista_startup_process)

## 2.3 System-Architektur von UNIX/Linux

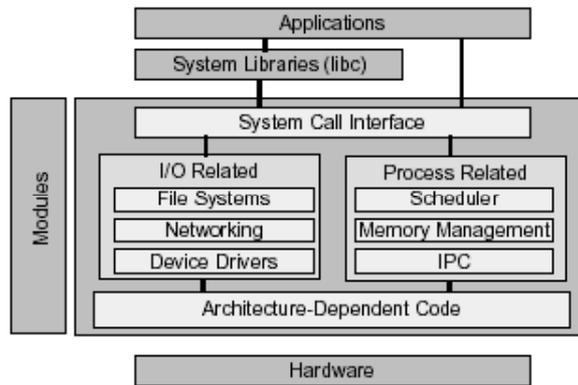


Abbildung 2.3: Linux-Architektur

Linux (und UNIX) besitzt einen **monolithischen Kernel**. Lediglich die Devicedriver werden als externe Module dynamisch nachgeladen.

Eine Einführung in den Linux-Kernel finden Sie unter:

- <http://docs.huihoo.com/linux/kernel/a1/index.html>
- [http://cs-pub.bu.edu/fac/richwest/cs591\\_w1/notes/wk1.pdf](http://cs-pub.bu.edu/fac/richwest/cs591_w1/notes/wk1.pdf)
- <http://bitterberg.de/tilmann/kernel-intro-acrobat.pdf>

Literatur: <http://lwn.net/Articles/161190/>

Die Darstellungen bei der Diskussion verschiedener Unix-Kernels weichen je nach Autoren-Intention etwas voneinander ab:

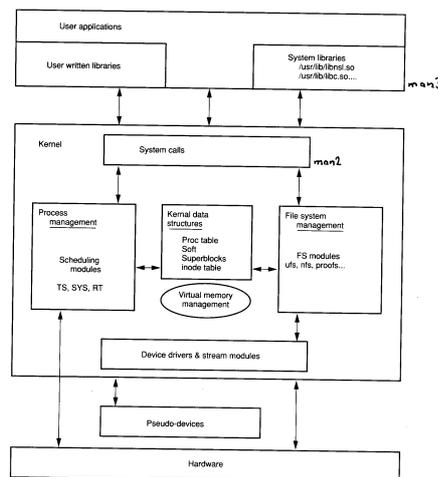


Abbildung 2.4: Solaris2-Architektur

<http://www.pearsonhighered.com/educator/academic/product/0,3110,0131482092,00.html>

(Vgl. auch: [http://www.nacs.uci.edu/indiv/dwatanab/usenix/01/solaris\\_internals.pdf](http://www.nacs.uci.edu/indiv/dwatanab/usenix/01/solaris_internals.pdf) sowie <http://www.solarisinternals.com/si/reading/t2-solaris-slides.pdf>)

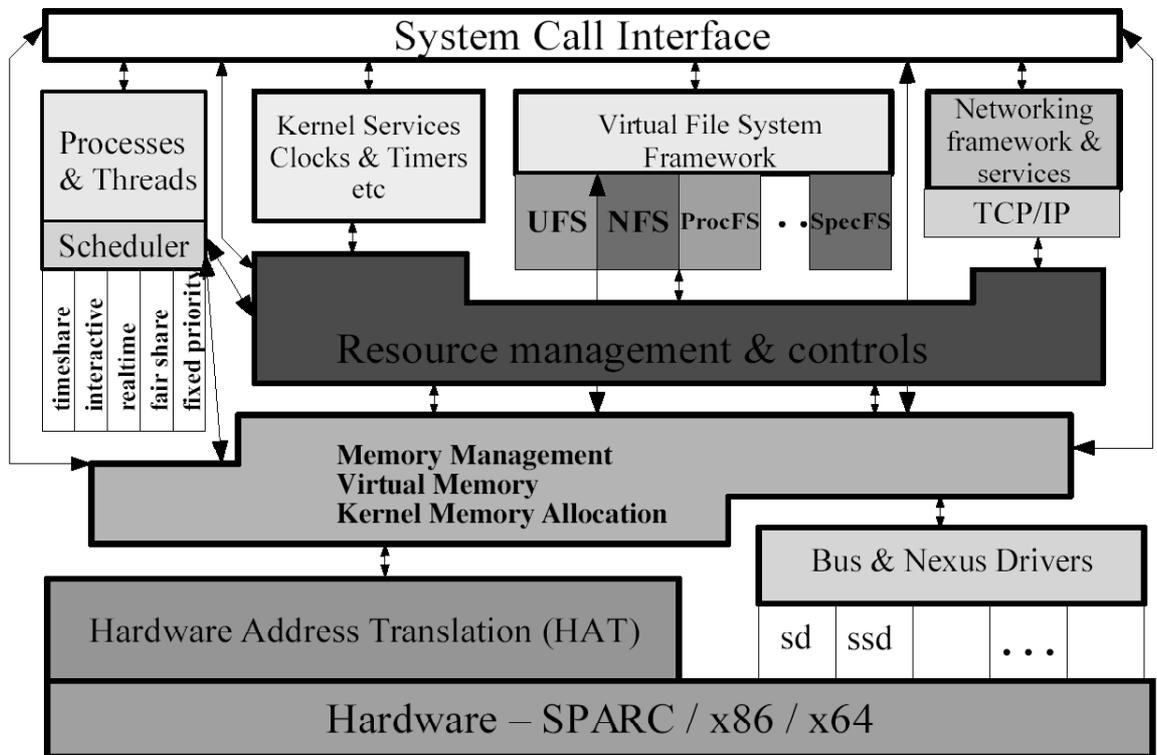


Abbildung 2.5: Solaris Kernel Overview

### 2.3.1 Der Linux Bootprozess

<http://www.debianhelp.co.uk/boot.htm>

<http://www.yolinux.com/TUTORIALS/LinuxTutorialInitProcess.html>

[http://en.wikipedia.org/wiki/Linux\\_startup\\_process](http://en.wikipedia.org/wiki/Linux_startup_process)

### 2.3.2 Der Solaris Bootprozess

[http://www.adminschoice.com/docs/booting\\_process\\_in\\_solaris.htm](http://www.adminschoice.com/docs/booting_process_in_solaris.htm)

## 2.3.3 Kompilierung und Installation von Kernel-Modulen

... am Beispiel des Cisco-Clients:

<http://www.rz.uni-augsburg.de/netz/vpn/linux/>

Fragen zu den Unbequemlichkeiten der Nutzung von Kernel-Modulen: Wann muß die Installation erneut durchgeführt werden? Welche Rechte braucht man bei der Installation? ...

Weitere Beispiele:

Installiere „VirtualBox Guest Additions“ in Kapitel 4 des Manuals:

<http://www.virtualbox.org/wiki/Downloads>

Proprietäre 3D-Graphiktreiber in Linux: <http://www.suse.de/~ndirsch/nvidia-installer-HOWTO.html>

## 2.3.4 Ein professionelles (offenes) UNIX-Betriebssystem: Solaris

[http://de.wikipedia.org/wiki/Solaris\\_\(Betriebssystem\)](http://de.wikipedia.org/wiki/Solaris_(Betriebssystem))

Solaris 10 5/08 <http://www.sun.com/software/solaris/get.jsp>

Grub für Solaris 10 <http://docs.sun.com/app/docs/doc/820-4038/6nfhgs3ff?l=ru&a=view>

Solaris auf x86-Systemen: <http://www.linuxtag.org/2006/fileadmin/linuxtag/dvd/12443-solaris-x86-linuxtag-v4.pdf>

Die UNIX-Geschichte: <http://www.levenez.com/unix/>

Literatur: W. Richard Stevens: Advanced Programming in the UNIX Environment

## 2.4 Virtuelle Betriebssysteme

Ein **virtueller Computer** (häufig mit Hilfe eines virtuellen Host-Betriebssystems realisiert) erlaubt Nutzern, auf einer einzigen Hardware mit einer Vielzahl von „virtuellen“ Computern mit unterschiedlichen Betriebssystemumgebungen, die sich alle gegeneinander lediglich über (virtuelle) Netzwerkinterfaces sehen können, zu arbeiten. Der Einsatz Hilft unter gleichzeitiger Server-Konsolidierung, Strom einzusparen: <http://www.heise.de/newsticker/Strom-sparen-mit-Virtualisierung-/meldung/80863>.

<http://en.wikipedia.org/wiki/Hypervisor>

<http://www.sun.com/servers/coolthreads/ldoms/index.xml>

<http://en.wikipedia.org/wiki/LPAR>

[http://en.wikipedia.org/wiki/Workload\\_Partitions](http://en.wikipedia.org/wiki/Workload_Partitions)

Host/Guest-Integration

Repräsentative Beispiele sind:

- VMware: <http://en.wikipedia.org/wiki/VMware>
- VirtualPC: [http://en.wikipedia.org/wiki/Virtual\\_PC](http://en.wikipedia.org/wiki/Virtual_PC)
- Xen: [http://en.wikipedia.org/wiki/Xen\\_%28virtual\\_machine\\_monitor%29](http://en.wikipedia.org/wiki/Xen_%28virtual_machine_monitor%29)
- xVM VirtualBox: <http://www.sun.com/software/products/virtualbox/get.jsp>
- xVM Ops Center: [http://en.wikipedia.org/wiki/Sun\\_xVM](http://en.wikipedia.org/wiki/Sun_xVM)
- Hyper-V: <http://en.wikipedia.org/wiki/Hyper-V>,  
<http://www.heise.de/newsticker/Microsoft-hat-Hyper-V-fertig-/meldung/110072>

- Solaris 10 Zones:  
<http://www.softpanorama.org/Solaris/zones.shtml> und <http://docs.sun.com/app/docs/doc/817-1592>

Siehe auch: [http://www.deatech.com/deatech/articles/linux\\_pc\\_vc.html](http://www.deatech.com/deatech/articles/linux_pc_vc.html)  
und [http://en.wikipedia.org/wiki/Comparison\\_of\\_virtual\\_machines](http://en.wikipedia.org/wiki/Comparison_of_virtual_machines)

Hardware-Unterstützung für virtuelle Maschinen nicht nur in Großrechnern:  
<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/59885&words=Pacifica>

Hunderte von Cores: <http://de.wikipedia.org/wiki/Terascale-Prozessor>  
Terascale-Computing: <http://techresearch.intel.com/articles/Tera-Scale/1421.htm>

## 2.5 OpenSolaris

<http://www.opensolaris.com/get/index.html>  
<http://en.wikipedia.org/wiki/OpenSolaris>  
<http://en.wikipedia.org/wiki/Livecd>  
<http://dlc.sun.com/osol/docs/content/IPS/getst1.html>  
<http://opensolaris.org/os/community/documentation/docs-index/>

# 3 Partitionen und Dateisysteme

## Virtuelle Festplatten

### Zweck von Festplatten-Partitionierung

Für einige Zwecke sollte man eine Festplatte in mehrere logische Festplatten unterteilen; sie sieht dann dem Betriebssystem gegenüber wie mehrere (physikalisch vorhandene) Festplatten aus:

- Vorspiegelung mehrerer kleiner Festplatten, wenn die Gesamtkapazität sonst für das Betriebssystem/Dateisystem unbrauchbar groß ist (FAT16-Dateisysteme konnten z.B. nur max. 2 GB groß sein).
- Gleichzeitige Installation mehrerer Betriebssysteme auf einer Festplatte.
- Logische Trennung der Bereiche einer Festplatte nach ihrer Funktion: zum Beispiel Partitionen für das Betriebssystem, den Swapping-Bereich, die Anwenderprogramme, die Datenbereiche der einzelnen Benutzer, ...
- Weitergehende logische Trennung der Bereiche des Betriebssystems: Loggingdatenbereiche, Bereiche für optionale Software, Temporärbereiche (etwa 800 MB für das Image einer CD bzw. 5 GB bzw. 8.5 GB für ein DVD-Image),...
- Anlegen einer gemeinsamen Datenpartition (z.B. FAT) zum Datenaustausch zwischen verschiedenen installierten Betriebssystemen.
- Anlegen einer dedizierten nur zum Swapping benutzten Partition.
- Einrichten einer temporären Scratch-Partition.
- Einrichten einer Backup-Partition (Backup durch manuelles Spiegeln).
- Partitionen für Software-RAID-Volumes (Spiegeln, Stripes)
- ...

Man nennt solche logischen Festplatten **Partitionen**. Eine Festplatte wird für die genannten Zwecke *partitioniert*.

## Windows-Partitionen

IDE/ATA-Festplatten erlaubten zunächst maximal vier Partitionen. Später erweiterte man das zu einem Schema von drei primären und einer erweiterten Partition, wobei die erweiterte Partition viele Unterpartitionen besitzen darf.

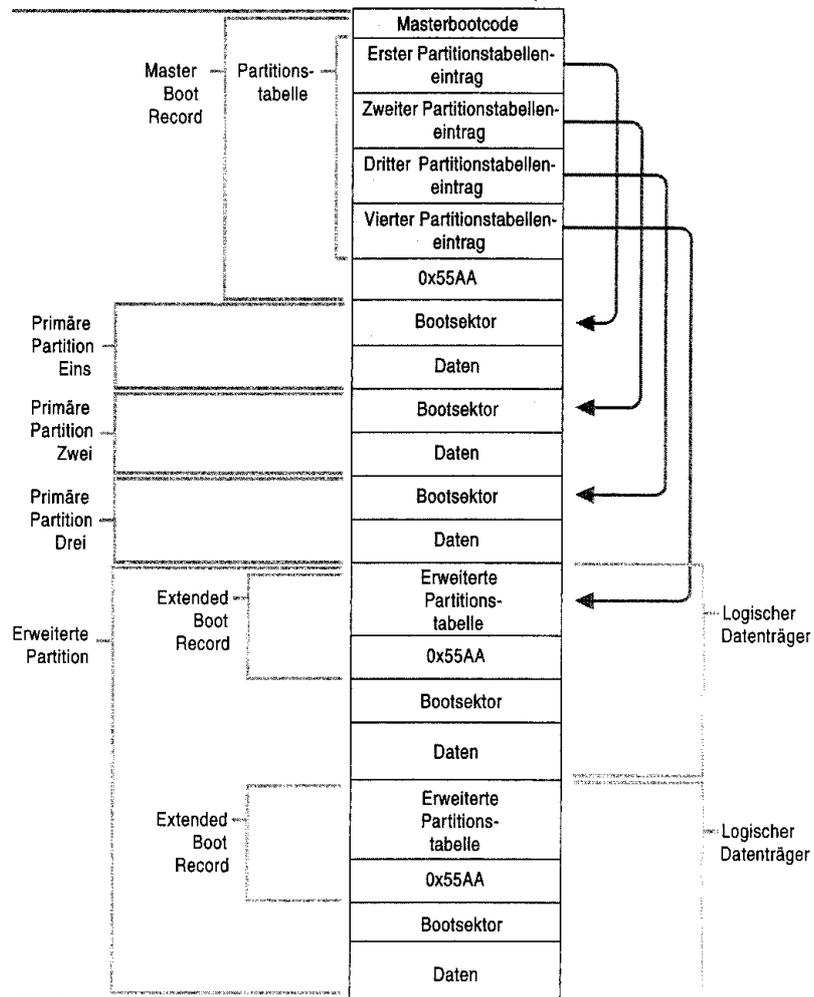
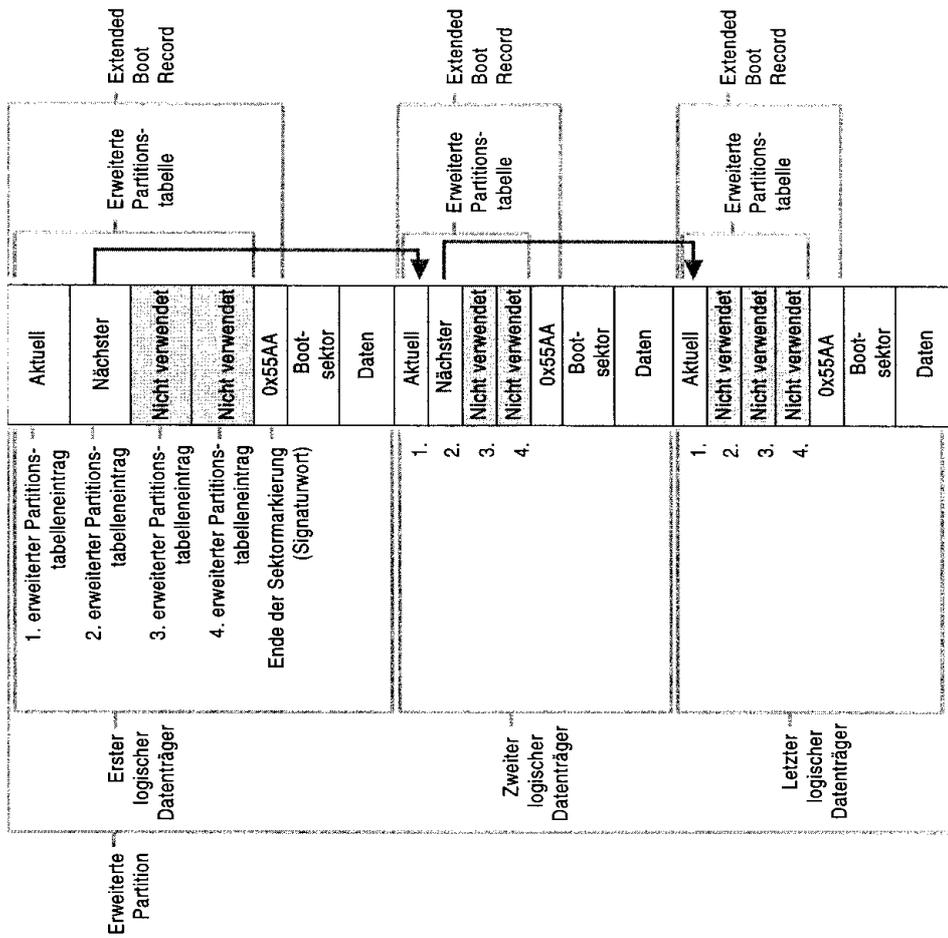


Abbildung 3.1: Vier primäre Partitionen



Partitionen werden durch eine *maggische Zahl* (die SYSTEM-ID) gekennzeichnet, die den Typ des auf der Partition installierten Dateisystems kennzeichnet:

EINTRAG	ERKLÄHRUNG
0x01	Primäre FAT12-Partition oder logisches Laufwerk (weniger als 32.680 Sektoren auf dem Datenträger)
0x04	FAT16-Partition oder logisches Laufwerk (32.680 - 65.535 Sektoren oder 16 MB - 33 MB)
0x05	Erweiterte Partition
0x06	BIGDOS FAT16-Partition oder logisches Laufwerk (33 MB - 4 GB)
0x07	Installierbares Dateisystem (NTFS-Partition oder logisches Laufwerk)
0x0B	FAT32-Partition oder logisches Laufwerk
0x0C	FAT32-Partition oder logisches Laufwerk mit Verwendung von BIOS INT 13h-Erweiterung
0x0E	BIGDOS FAT16-Partition oder logisches Laufwerk mit Verwendung von BIOS INT 13h-Erweiterungen
0x0F	Erweiterte Partition mit Verwendung von BIOS INT 13h-Erweiterungen
0x12	EISA-Partition
0x42	Datenträger einer dynamischer Festplatte
0x86	Fehlertolerante veraltete FAT16-Festplatte*
0x87	Fehlertolerante veraltete NTFS-Festplatte*
0x8B	Fehlertoleranter veralteter Datenträger mit FAT32-Formatierung*
0x8C	Fehlertoleranter veralteter Datenträger mit Verwendung von BIOS INT 13h-Erweiterung und FAT32-Formatierung*

Tabelle 3.1: Datenträgertypen

Mit einem Sternchen (\*) gekennzeichnete Partitionstypen können auch für nicht fehlertolerante Konfigurationen wie Stripset-Datenträger und übergreifende Datenträger verwendet werden.

Eine aktuelle neuere Übersicht der genutzten SYSTEM-IDs finden Sie unter:

[http://www.win.tue.nl/~aeb/partitions/partition\\_types-1.html](http://www.win.tue.nl/~aeb/partitions/partition_types-1.html)

Bytewerte für Offset	Feldlänge	Beispielwert	Feldname und Definition
0x01BE	BYTE	0x80	<b>Boot Indikator:</b> Kennzeichnet ob es sich bei dem Datenträger um eine aktive Partition handelt. Gültige Werte sind: <ul style="list-style-type: none"> <li>• 00. Keine Verwendung für den Startvorgang</li> <li>• 80. Aktive Partition</li> </ul>
0x01BF	BYTE	0x01	<b>Starting Head</b>
0x01C0	6 Bits	0x01*	<b>Starting Sector:</b> Es werden nur die Bits 0...5 verwendet. Die oberen zwei Bits, 6 und 7, werden durch das Feld Startzylinder mitverwendet.
0x01C1	10 Bits	0x00*	<b>Starting Cylinder:</b> Verwendet zusätzlich ein Byte zu den oberen 2 Bits des Startsektorfeldes zur Bildung des Zylinderwertes. Der Startzylinder ist eine 10-Bit-Nummer mit einem maximalen Wert von 1023.
0x01C2	BYTE	0x07	<b>System ID:</b> Definiert den Datenträgertyp. Siehe hierzu Tabelle <b>Datenträgertypen</b> .
0x01C3	BYTE	0xFE	<b>Ending Head</b>
0x01C4	6 Bits	0xBF*	<b>Ending sector:</b> Es werden nur die Bits 0...5 verwendet. Die oberen zwei Bits, 6 und 7, werden durch das Feld Endzylinder mitverwendet.
0x01C5	10 Bits	0x09*	<b>Ending Cylinder:</b> Verwendet zusätzlich ein Byte zu den oberen 2 Bits des Endsektorfeldes zur Bildung des Zylinderwertes. Der Endzylinder ist eine 10-Bit-Nummer mit einem maximalen Wert von 1023.
0x01C6	DWORD	0x3F000000	<b>Relative Sectors:</b> Das Offset vom Beginn der Festplatte bis zum Beginn der Partition gezählt in Sektoren.
0x01CA	DWORD	0x4BF57F00	<b>Total Sectors:</b> Die Gesamtzahl der Sektoren der Partition.

Tabelle 3.2: Partitionstabellenfelder

## 3.1 Partitionieren unter UNIX

### 3.1.1 Partitionieren in Solaris

SOLARIS-Festplatten können in sieben Partitionen unterteilt werden, von denen eine (Nummer 2) in der Regel als Pseudopartition die ganze Festplatte erfaßt.

Unter SOLARIS partitioniert man eine Platte mit dem Befehl `format`:

```
% format>
Serching for disks...done

AVAILABLE DISK SELECTIONS:
  0. c0t0d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
    /pci@1f,4000/scsi@3/sd@0,0
  1. c0t8d0 <SEAGATE-ST336607LC-0003 cyl 49780 alt 2 hd 2 sec 720>
    /pci@1f,4000/scsi@3/sd@8,0
  2. c0t9d0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
    /pci@1f,4000/scsi@3/sd@9,0
  3. c0t10d0 <SEAGATE-ST318404LC-HP02 cyl 14078 alt 2 hd 6 sec 421>
    /pci@1f,4000/scsi@3/sd@a,0
  4. c0t11d0 <SEAGATE-ST3146807LC-0004 cyl 49780 alt 2 hd 8 sec 720>
    /pci@1f,4000/scsi@3/sd@b,0
  5. c0t12d0 <SEAGATE-ST3146807LC-0004 cyl 49780 alt 2 hd 8 sec 720>
    /pci@1f,4000/scsi@3/sd@c,0
Specify disk (enter its number): 5
selecting c0t12d0
[disk formatted]

FORMAT MENU:
FORMAT MENU:
disk          - select a disk
type          - select (define) a disk type
partition    - select (define) a partition table
current      - describe the current disk
format       - format and analyze the disk
repair       - repair a defective sector
label        - write label to the disk
analyze      - surface analysis
defect       - defect list management
backup       - search for backup labels
verify       - read and display labels
save         - save new disk/partition definitions
inquiry      - show vendor, product and revision
volname      - set 8-character volume name
!<cmd>       - execute <cmd>, then return
quit
```

Abbildung 3.2: Partitionieren einer Solaris-Festplatte mittels `format`

```

PARTITION MENU:
  0 - change `0' partition
  1 - change `1' partition
  2 - change `2' partition
  3 - change `3' partition
  4 - change `4' partition
  5 - change `5' partition
  6 - change `6' partition
  7 - change `7' partition
select - select a predefined table
modify - modify a predefined partition table
name - name the current table
print - display the current table
label - write partition map and label to the disk
!<cmd> - execute <cmd>, then return
quit
partition> pr
Current partition table (original):
Total disk cylinders available: 49780 + 2 (reserved cylinders)

Part      Tag      Flag      Cylinders      Size      Blocks

  0 unassigned  wm        0              0          (0/0/0)      0
  1 unassigned  wu        0              0          (0/0/0)      0
  2 backup      wu        0 - 49779     136.72GB   (49780/0/0) 286732800
  3 unassigned  wm        0              0          (0/0/0)      0
  4 unassigned  wm        0              0          (0/0/0)      0
  5 unassigned  wm        0              0          (0/0/0)      0
  6 unassigned  wm        0              0          (0/0/0)      0
  7 unassigned  wm        0 - 49779     136.72GB   (49780/0/0) 286732800

partition> q
format> inq
Vendor: SEAGATE
Product: ST3146807LC
Revision: 0004

```

Abbildung 3.3: Partitionieren einer Solaris-Festplatte mittels **format** (Fortsetzung)

### 3.1.2 Dateisystem-Erzeugung unter Windows

- Low-level-format/soft sectors: herstellerspezifische Tools, siehe etwa <http://de.wikipedia.org/wiki/Formatierung>
- Partitionieren unter Windows
- Filesystem-Erzeugung unter Windows

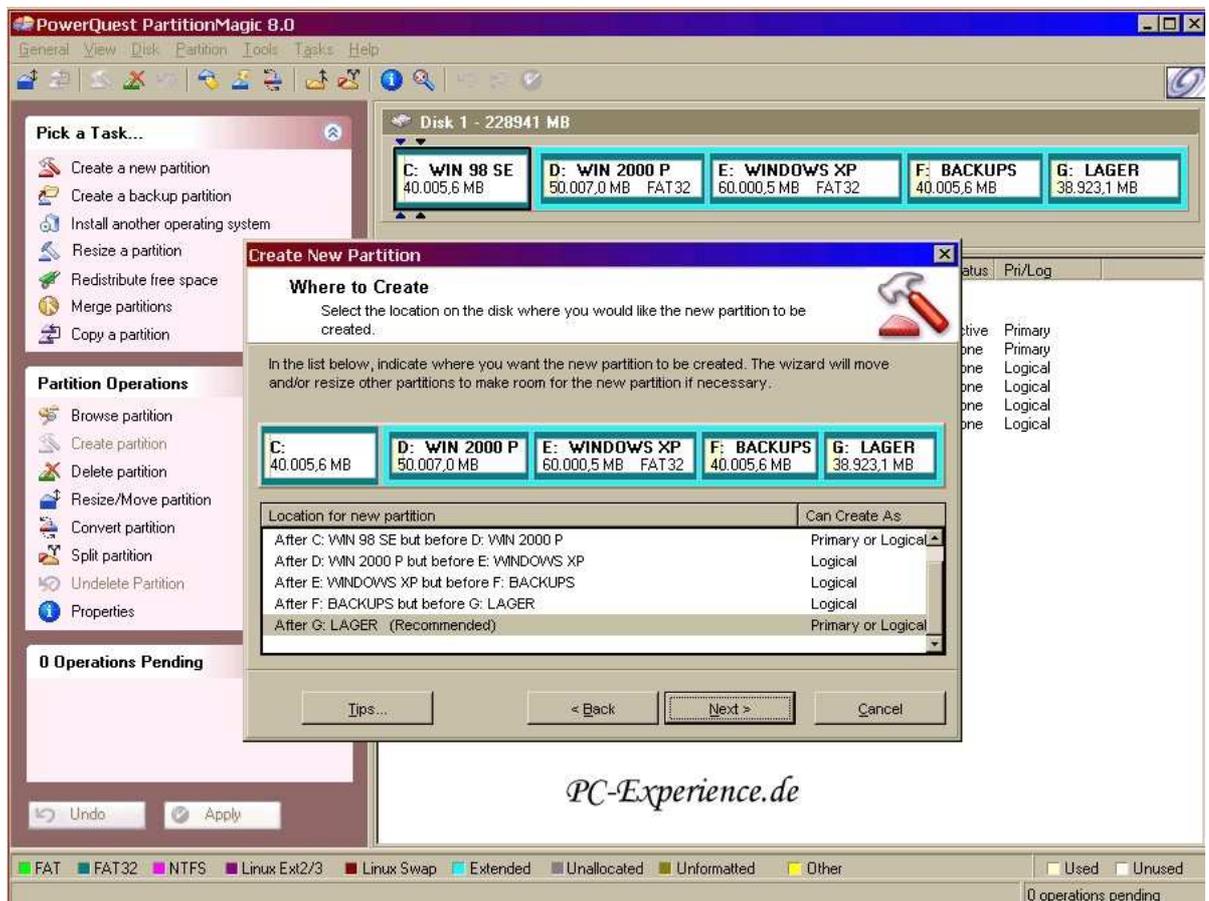


Abbildung 3.4: Partitionierung unter Windows

Die Partitionierung und Filesystemerzeugung geschieht unter Windows entweder bei der **Installation des Betriebssystems**, mittels des Tools Computerverwaltung (Item Datenträgerverwaltung) beziehungsweise mit speziellen Zusatz-Applikationen wie zum Beispiel **PartitionMagic**, ...

### 3.1.3 Dateisystemerzeugung unter Linux

Auch unter Linux werden die Partitionen sowie die Dateisysteme entweder bei der **Installation** oder mittels des Betriebssystemtools Partitionieren



Abbildung 3.5: Partitionierung unter Linux

erzeugt.

### 3.1.4 Anlegen eines UNIX-Dateisystems unter Solaris

```
newfs -Nv /dev/rdisk/c0t11d0s7
s -F ufs -o N /dev/rdisk/c0t11d0s7 275212800 720 8 8192 1024 170 1 167 8192
v/rdisk/c0t11d0s7: 275212800 sectors in 47780 cylinders of 8 tracks, 720
134381.2MB in 2655 cyl groups (18 c/g, 50.62MB/g, 6144 i/g)
er-block backups (for fsck -F ufs -o b=#) at:
, 104432, 208832, 313232, 417632, 522032, 626432, 730832, 829472, 933872,
38272, 1142672, 1247072, 1351472, 1455872, 1560272, 1658912, 1763312,
67712, 1972112, 2076512, 2180912, 2285312, 2389712, 2488352, 2592752,
97152, 2801552, 2905952, 3010352, 3114752, 3219152, 3317792, 3422192,

2885792, 272990192, 273094592, 273198992, 273303392, 273407792, 273512192,
3616592, 273715232, 273819632, 273924032, 274028432, 274132832, 274237232,
4341632, 274446032, 274544672, 274649072, 274753472, 274857872, 274962272,
5066672, 275171072,
```

Abbildung 3.6: Anlegen eines Dateisystems mit newfs

Anschließend der Produktionslauf ohne die Option „-N“.

### 3.1.5 Zusammenfügen von UNIX-Partitionen: die Mount-Tabelle /etc/vfstab

Manuell:

```
# mount /dev/dsk/c0t11d0s7 /mnt
```

oder mit Hilfe einer Mount-Tabelle:

#device #to mount #	device to fsck	mount point	FS type	fsck pass	mount at boot	mount options
fd	/dev/fd	fd	-	no	-	
/proc	/proc	proc	-	no	-	
/dev/dsk/clt1d0s1	-	-	swap	no	-	
/dev/dsk/clt1d0s0	/dev/rdisk/clt1d0s0	/	ufs	1	no	-
/dev/dsk/clt2d0s7	/dev/rdisk/clt2d0s7	/export/home	ufs	2	yes	-
swap	/tmp	tmpfs	yes	-		

Abbildung 3.7: Mount-Tabelle

### 3.1.6 MultiBoot-System: Windows-Partitionen im Boot-Chooser

```
bootini
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(5)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(5)\WINDOWS="Microsoft Windows XP Professional"
/fastdetect
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Microsoft Windows 2000 Professional"
/fastdetect
```

Abbildung 3.8: Windows Partitionen

## 3.2 Dateisysteme: ein erster Überblick

Vergleiche die freie Enzyklopädie Wikipedia: <http://de.wikipedia.org/wiki/Dateisystem>.

Ein Dateisystem ist ein System, das Daten in Form von Dateien auf einem Computersystem speichert und verwaltet. Die meisten Betriebssysteme verwenden Dateisysteme. Historisch wurden Dateisysteme zur Organisation des Zugriffs auf Massenspeicher wie Festplattenlaufwerke entwickelt. Jede Datei belegt einen Teil des Massenspeichers. Ein Dateisystem bietet die Möglichkeit per Namen auf eine Datei zu zugreifen. Das Konzept der Dateisysteme wurde dann soweit abstrahiert, dass auch Zugriffe auf Dateien im Netz und in Geräten, die virtuell als Datei verwaltet werden, über Dateisysteme geregelt werden können.

Dateien haben in einem Dateisystem fast immer mindestens einen Dateinamen sowie Attribute, die nähere Informationen über die Datei geben. Die Dateinamen sind in speziellen Dateien, den Verzeichnissen, abgelegt. Über diese Verzeichnisse kann ein Dateiname und damit eine Datei vom System gefunden werden. Ein Dateisystem bildet somit einen Namensraum. Alle Dateien (oder dateiähnliche Objekte) sind so über einen eindeutigen Namen/Adresse (Dateiname inkl. Pfad oder URL) – innerhalb des Dateisystems – aufrufbar.

```
user@host> man mount
```

```
...
```

```
-t vfstype
```

```
The argument following the -t is used to indicate the file system type. The file system types which are currently supported are: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, ramfs, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, usbfs, vfat, xenix, xfs, xiafs. Note that coherent, sysv and xenix are equivalent and that xenix and coherent will be removed at some point in the future -- use sysv instead. Since kernel version 2.1.21 the types ext and xiafs do not exist anymore. Earlier, usbfs was known as usbdevfs.
```

```
...
```

```
user@host> man filesystems
```

```
...
```

```
Below a short description of a few of the available filesystems.
```

```
minix is the filesystem used in the Minix operating system, the first to run under Linux. It has a number of shortcomings: a 64MB partition size limit, short filenames, a single time stamp, etc. It remains useful for floppies and RAM disks.
```

```
ext is an elaborate extension of the minix filesystem. It has been completely superseded by the second version of the extended filesystem (ext2) and has been removed from the kernel (in 2.1.21).
```

```
ext2 is the high performance disk filesystem used by Linux for fixed disks as well as removable media. The second extended filesystem was designed as an extension of the extended file system (ext). ext2 offers the best performance (in terms of speed and CPU usage) of the filesystems supported under Linux.
```

```
ext3 is a journaling version of the ext2 filesystem. It is easy to
```

switch back and forth between ext2 and ext3. ext3 offers the most complete set of journaling options available among journaling filesystems.

xiafs was designed and implemented to be a stable, safe filesystem by extending the Minix filesystem code. It provides the basic most requested features without undue complexity. The xia filesystem is no longer actively developed or maintained. It was removed from the kernel in 2.1.21.

msdos is the filesystem used by DOS, Windows, and some OS/2 computers. msdos filenames can be no longer than 8 characters, followed by an optional period and 3 character extension.

umsdos is an extended DOS filesystem used by Linux. It adds capability for long filenames, UID/GID, POSIX permissions, and special files (devices, named pipes, etc.) under the DOS filesystem, without sacrificing compatibility with DOS.

vfat is an extended DOS filesystem used by Microsoft Windows95 and Windows NT. VFAT adds the capability to use long filenames under the MSDOS filesystem.

proc is a pseudo-filesystem which is used as an interface to kernel data structures rather than reading and interpreting /dev/kmem. In particular, its files do not take disk space. See proc(5).

iso9660 is a CD-ROM filesystem type conforming to the ISO 9660 standard.

#### High Sierra

Linux supports High Sierra, the precursor to the ISO 9660 standard for CD-ROM filesystems. It is automatically recognized within the iso9660 filesystem support under Linux.

#### Rock Ridge

Linux also supports the System Use Sharing Protocol records specified by the Rock Ridge Interchange Protocol. They are used to further describe the files in the iso9660 filesystem to a UNIX host, and provide information such as long filenames, UID/GID, POSIX permissions, and devices. It is automatically recognized within the iso9660 filesystem support under Linux.

hpfs is the High Performance Filesystem, used in OS/2. This filesystem is read-only under Linux due to the lack of available documentation.

sysv is an implementation of the SystemV/Coherent filesystem for

Linux. It implements all of Xenix FS, SystemV/386 FS, and Coherent FS.

nfs is the network filesystem used to access disks located on remote computers.

smb is a network filesystem that supports the SMB protocol, used by Windows for Workgroups, Windows NT, and Lan Manager.

To use smb fs, you need a special mount program, which can be found in the ksmbfs package, found at <ftp://sunsite.unc.edu/pub/Linux/system/Filesystems/smbfs>.

ncpfs is a network filesystem that supports the NCP protocol, used by Novell NetWare.

To use ncpfs, you need special programs, which can be found at <ftp://linux01.gwdg.de/pub/ncpfs>.

Siehe auch: [http://en.wikipedia.org/wiki/List\\_of\\_file\\_systems](http://en.wikipedia.org/wiki/List_of_file_systems) und [http://en.wikipedia.org/wiki/Comparison\\_of\\_file\\_systems](http://en.wikipedia.org/wiki/Comparison_of_file_systems).

## 3.3 Dateisystem-Übersicht

### 3.3.1 Ältere Betriebssysteme

Die älteren Betriebssysteme (CP/M, Apple DOS, Commodore DOS) hatten nur ein Dateisystem, welches keinen eigenen Namen trug. Diese kann man einfach als CP/M-Dateisystem, Apple-Dateisystem, ... bezeichnen.

### 3.3.2 Neuere Betriebssysteme

- Linux/UNIX:
  - minix (vom gleichnamigen Betriebssystem)
  - ext2 (second extended file system, lange Zeit das Linux-Standard-Dateisystem)
  - ext3 (weiterentwickelte Variante von ext2, mit journaling)
  - FFS (Vorgänger von UFS unter BSD)
  - ReiserFS (Linux Journaling File System von Hans Reiser)
  - JFS (Journaled File System von IBM)
  - UFS (UNIX File System, verwendet unter Solaris und BSD)
  - XFS (ein journaling Dateisystem von SGI)
  - SYSV (das klassische Dateisystem des System V-Unix von AT&T)
  - ADFS (Acorn StrongARM)
  - GNOME Storage (Datenbank-basierendes Dateisystem)
- (MS-)DOS:
  - FAT bzw. FAT12 (File Allocation Table, für Disketten noch heute gebräuchlich)
  - FAT16 (Erweitertes FAT-System für Festplatten)
  - FAT32 (Erweitertes FAT für große Festplatten)
  - **exFAT** (extended FAT FAT64)
- MS-Windows unterstützt sämtliche DOS-Dateisysteme, zusätzlich:
  - VFAT (Virtual FAT: längere Dateinamen für alle FAT-Systeme)
  - NTFS (Journaling Dateisystem von Windows NT und Nachfolgern)
  - WinFS (für die Zukunft angekündigtes Datenbank-basierendes Dateisystem)
- AmigaOS:
  - OFS (Amiga Old File System)
  - FFS (Amiga Fast File System)
- Apple Macintosh
  - ProDOS (Dateisystem der späten Apple II-Modelle)
  - MFS (Macintosh File System)
  - HFS (Hierarchical File System)
  - HFS+ (Erweiterung von HFS u. a. auf Dateinamen mit mehr als 32 Zeichen)
  - HFSX Case sensitive Variante von HFS+
- OS/2:
  - HPFS (High Performance File System)
  - JFS (Journaled File System)
- BeOS/Haiku:

- BFS
- OpenBFS (64 Bit, multithreaded, journaliertes, Datenbank-ähnliches Dateisystem)
- CD-ROM/DVD:
  - ISO9660 (Dateisystem für CD-ROMs)
    - \* Joliet (Erweiterung des ISO9660 von der Firma Microsoft)
    - \* Rockridge (Erweiterung des ISO9660 für UNIX)
  - UDF (Universal Disk Format, u. a. auf DVDs aller Typen gebräuchlich)
  - HFS/HFS+ Auf Macintosh-CDs wird häufig das Festplatten-Filesystem genutzt
- Netzwerk:
  - **NFS** (Network File System; ein über Netzwerke erreichbares Dateisystem für Unix-artige Systeme)
  - Coda (ein fortgeschrittenes Netzwerk-Dateisystem ähnlich zu NFS)
  - SMB (ein über Netzwerke erreichbares Dateisystem vor allem für Windows-Systeme)
  - xFS (ein verteiltes und dezentrales Netzwerk-Dateisystem)
  - AFS (Andrew File System)
  - NCP (NetWare Core Protocol)
  - DFS (distributed file system der Open Group, eine Weiterentwicklung des Andrew File System / Filesystem von Microsoft)

Alle Dateisysteme haben gemeinsam, dass auf sie auch von Fremdsystemen zugegriffen werden kann, sofern das Betriebssystem dies direkt unterstützt oder dies dem Betriebssystem über entsprechende Treibersoftware ermöglicht wird.

Ausnahmen bilden Dateisysteme, die eine erweiterte Berechtigung unterstützen, die Möglichkeit der Verschlüsselung bieten, oder deren genaue Funktionsweise ein Betriebsgeheimnis (proprietär) ist (zum Beispiel NTFS).

## 3.4 Access Control Lists — ACLs

Zugriffsrechte:

- discretionary
- mandatory

Access Control Lists erlauben die Spezifikation der Zugriffsrechte auf die Objekte des Dateisystems:

- POSIX-ACLs, POSIX-ACLs in Linux
- Windows-ACLs, Windows-AC
- ACLs in heterogenen Windows-/UNIX-Clustern

## 3.5 Namensräume im Netzwerk / Verteiltes Rechnen

- Newcastle-Connection ...
- Federated Naming Service — fns/xfn
- Samba: Windows-Freigaben

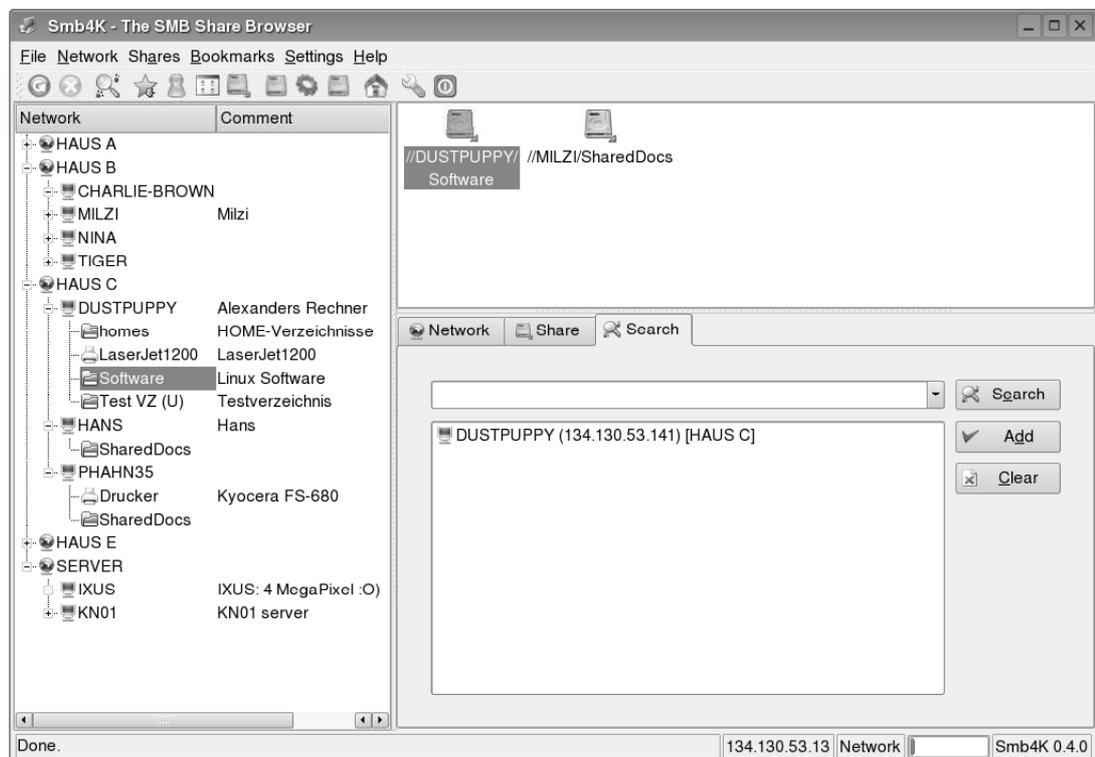


Abbildung 3.9: Windows Shares

- Windows DFS, DFS-Administration
- Service Location Protocol — SLP
- DCE

## 3.6 Verteilte Betriebssysteme

Vergleiche: [Distributed Computing](#)

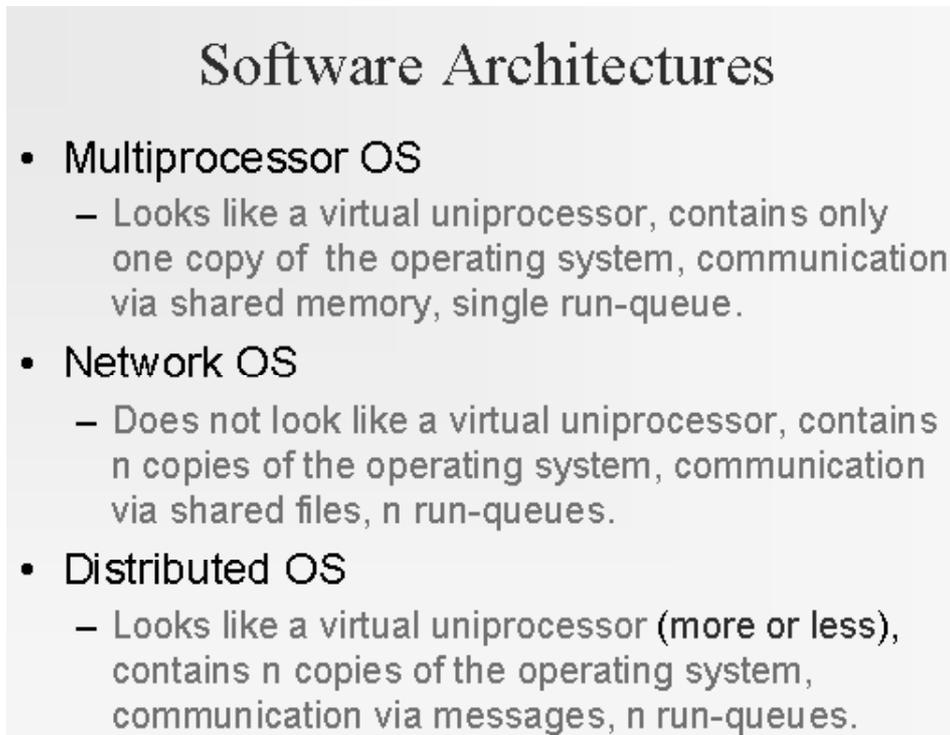


Abbildung 3.10: Verteilte Betriebssysteme

Yair Amir: [Distributed Operating Systems](#)

Exemplarisch:

- [Amoeba Introduction](#)
- [Amoeba User Guide](#)
- [Amoeba Administration Guide](#)
- [Amoeba Home Page](#)

## 3.7 Das FAT-Dateisystem

Die **File Allocation Table** (zu Deutsch: Dateibelegungstabelle) wurde von **Seattle Computer Products** in einem Dateisystem für dessen **Betriebssystem QDOS**, dem direkten Vorgänger von **MS-DOS**, entwickelt. Es ist die einzige bedeutende Neuerung von QDOS gegenüber CP/M. Zu der Familie der FAT-Dateisysteme gehören:

- FAT12 (wird heute immer noch für jede DOS- oder Windows-Diskette gebraucht)
- FAT16 (wird heute für alle Arten von mobilen Datenträgern verwendet, die kleiner als 2 GB sind.)
- FAT32 (wird für alle Arten von mobilen Speichern von mehr als 2 GB Kapazität genutzt)

Die FAT verkettet die Speicher-Cluster der Dateien. Jeder Datei ist im Directory-Eintrag lediglich die Nummer des ersten Clusters des Dateiinhalts ( $\geq 2$ ) zugeordnet, etwa N. In der FAT steht dann an der Position N die Cluster-Nummer des folgenden Clusters, usw. bis eine spezielle Cluster-Nummer (0xFF8...0xFFF bei FAT12, 0xFFF8...0xFFFF bei FAT16, ...) das Dateiende markiert. Die Cluster-Nummer 0xFF7 (bei FAT12) markiert einen Cluster als unbrauchbar, 0x0 kennzeichnet ihn als frei und 0x1 ist reserviert.

Die **FAT** hat bei den FAT-Dateisystemen in der Regel eine **Kopie**, um bei Datenverlust noch immer eine funktionsfähige alternative FAT zu besitzen. Mit diversen Programmen ist wegen dieser Verkettung der Dateicluster eine Datenwiederherstellung in vielen Fällen möglich, wenn sie früh genug nach dem Datei-Löschbefehl versucht wird.

Vergleiche [FAT — How it seems to work](#).

### 3.7.1 Aufbau eines FAT-Dateisystems

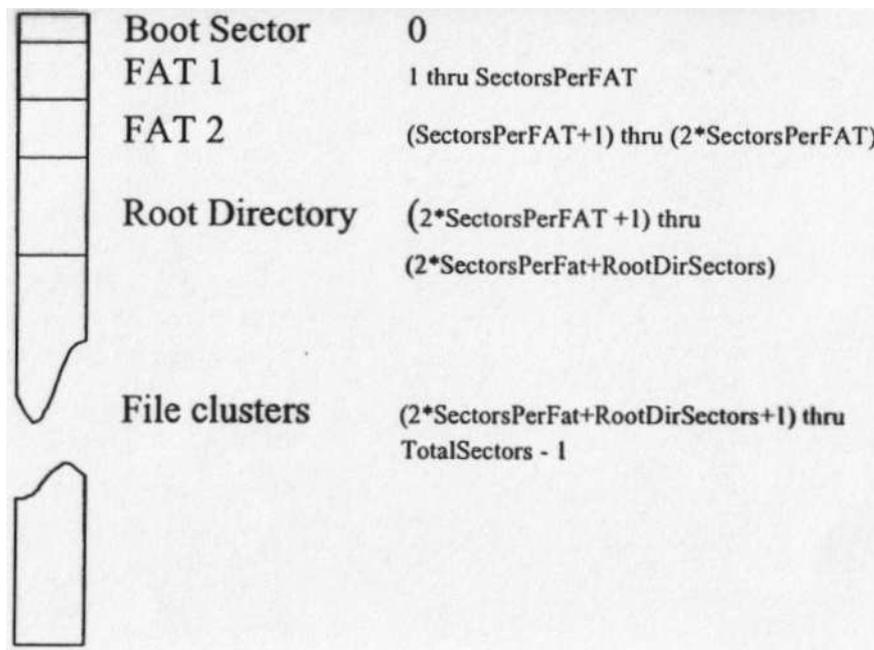


Abbildung 3.11: Struktur und Aufbau einer FAT-Partition

### 3.7.2 FAT-Verzeichnisse

Inhaltsverzeichnisse/Directories:

- Aneinanderreihung von Verzeichniseinträgen
- Das Wurzelverzeichnis hat eine feste Anzahl von Einträgen
  - dafür sind 14 Sektoren reserviert
- Alle anderen Verzeichnisse sind normale Dateien
  - keine Längen-Beschränkung
- Verzeichniseinträge:
  - 32 Bytes lang
  - das erste Zeichen des Dateinamens ist ein Indikator für:
    - \* 0x00 Eintrag wurde noch nie benutzt
    - \* 0xE5 Eintrag wurde benutzt, aber wieder freigegeben (d.h. die Datei wurde gelöscht)
  - Aufbau des Eintrags:

BYTE	Filename[8]	evtl. mit Leerzeichen aufgefüllt
BYTE	Extension[3]	
BYTE	Attribute	
BYTE	Reserviert[10]	
WORD	Zeit	Stunde*2048+Min*32+Sec/2
WORD	Datum	(Jahr-1980)*512 + Monat*32 + Tag
WORD	StartCluster	
DWORD	FileSize	

Die Datei-Attribute:

BIT	MASKE	ATTRIBUTE
0	0x01	Read Only / Leserecht
1	0x02	Hidden / Versteckt Datei
2	0x04	System / Systemdatei
3	0x08	Volume Label / Datenträger-Name
4	0x10	Subdirectory / Unterinhaltsverzeichnis
5	0x20	Archive / Datei ist seit letztem Backup geändert worden
6	0x40	Unused / unbenutzt
7	0x80	Unused / unbenutzt

Tabelle 3.3: File und Datei Attribute

## 3.8 Das UNIX Filesystem UFS/UFS2

Der UNIX Kern sieht alle Dateien als Byte-Ströme an. Jede innere logische Struktur ist anwendungsspezifisch. Dennoch ist Unix mit physischen Strukturen von verschiedenen Dateiabstraktionen vertraut. Sechs Dateitypen werden unterschieden:

- **Gewöhnlich Dateien:** Dateien, die Informationen von einem Benutzer, einem Anwendungsprogramm oder einem Systemdienstprogramm enthalten.
- **Verzeichnisse:** Dateien, die eine Liste von Verzeichniseinträgen enthalten. Diese enthalten wiederum lediglich den Dateinamen, sowie einen Zeiger zum zugehörigen Inode, der die Datei repräsentiert. Verzeichnisse sind hierarchisch organisiert. Verzeichnis-Dateien sind gewöhnliche Dateien mit speziellem Schreibschutz, so dass nur das Dateisystem in sie schreiben kann, während lesender Zugang für Benutzerprogramme vorhanden ist. Das Root-Verzeichnis des Dateisystems wird durch den ersten Inode realisiert.
- **Spezielle (Geräte-)Dateien:** Sie werden verwendet, um auf Peripheriegeräte wie Terminals und Drucker zuzugreifen. Jeder Eingabe-/Ausgabe-Treiber (I/O-Device) wird mit einer speziellen (Geräte-)Datei angesprochen. Es gibt Block- und Byte-Geräte-dateien, die gepufferte bzw. ungepufferte byteweise Ein- und Ausgabe anbieten.
- **Benannte FIFOs:** unidirektionale FIFOs, die durch einen Datei-Pfad im Filesystem erreicht werden können.
- **Symbolische Links:** Durch (symbolische) Indirektionen vergebene weitere Namen für vorhandene Dateien.
- **POSIX IPC-Sockets:** Dateien, die je einen lokalen bidirektionalen Kommunikationsmechanismus für den Datenaustausch zwischen Prozessen anbieten.

Vergleiche auch die man-Pages zu fs(5) und newfs(1M).

**Maximale UFS Datei- und Dateisystemgrößen:**

BETRIENSSYSTEM	DATEIGRÖSSE	DATEISYSTEMGRÖSSE
SunOS 4.1.X	2 GB	2 GB
SunOS 5.X	2 GB	1 TB
IRIX 5.X	2 GB	8 GB
IRIX 6.2	9,000,000 TB (64-bit Kernel) 1 TB (32-bit Kernel)	9,000,000 TB (64-bit Kernel) 1 TB (32-bit Kernel)

Tabelle 3.4: UFS Datei- und Dateisystemgrößen

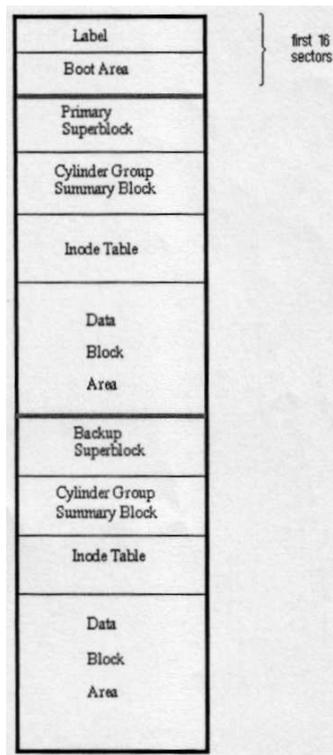


Abbildung 3.12: Aufbau einer UFS-Partition

A **UFS filesystem** is composed of the following parts:

- a few blocks at the beginning of the partition reserved for boot blocks (which must be initialized separately from the filesystem)
- a superblock, containing a magic number identifying this as a UFS filesystem, and some other vital numbers describing this filesystem's geometry and statistics and behavioral tuning parameters
- a collection of cylinder groups; Each cylinder group has the following components:
  - a backup copy of the superblock
  - a cylinder group header, with statistics, free lists, etc, about this cylinder group, similar to those in the superblock
  - a number of inodes, each containing file attributes
  - a number of data blocks

Inodes are numbered sequentially. The first several inodes are reserved for historical reasons, followed by the inode for the root directory.

Directory files contain only the list of filenames in the directory and the inode associated with each file. All file metadata is kept in the inode.

### 3.8.1 Inodes

FELD	BEDEUTUNG	
Datei-Zugriffsrechte	16-Bit-Flag für die die Zugriffs- und Ausführungsrechte	
	12-14	Dateityp (Datei, Verzeichnis,...)
	9-11	Ausführungs-Flags
	8	Eigentümer-Leseerlaubnis
	7	Eigentümer-Schreiberlaubnis
	6	Eigentümer-Ausführerlaubnis
	5	Gruppen-Leseerlaubnis
	4	Gruppen-Schreiberlaubnis
	3	Gruppen-Ausführerlaubnis
	2	Andere-Leseerlaubnis
	1	Andere-Schreiberlaubnis
0	Andere-Ausführerlaubnis	
Link-Zähler	Anzahl der Verzeichnissverweise auf diesen Inode	
Benutzer-ID	Eigentümer der Datei	
Gruppen-ID	Eigentümergegruppe der Datei	
Dateigröße	Anzahl der Bytes in der Datei	
Dateiadressen	15 zum Teil mehrfach indirekt verwertete Adressen der zur Datei gehörenden Datenblöcke	
Letzter Zugriff	Zeit des letzten Zugriffes	
Letzte Modifikation	Zeit der letzten Modifikation	
Letzte Inode-Modifikation	Zeit der letzten Inode-Modifikation	

Tabelle 3.5: Inode-Aufbau

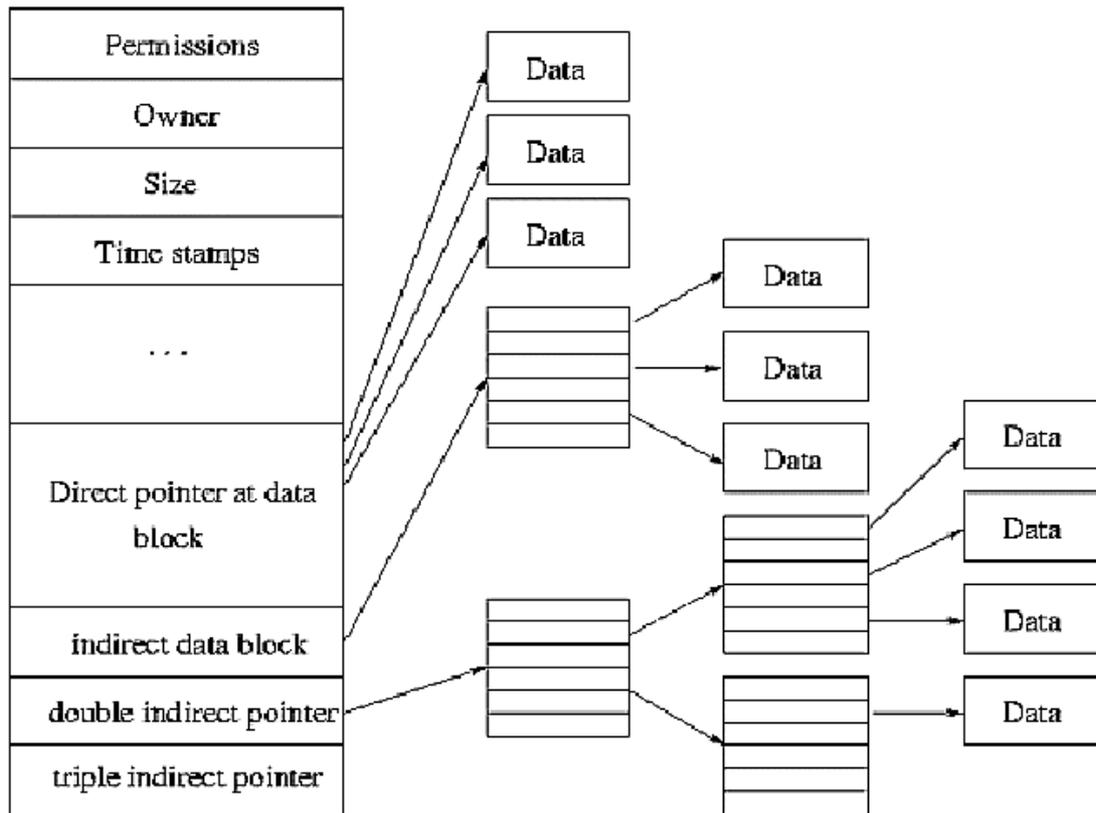


Abbildung 3.13: Adressenschema für die Datenblöcke eines Inodes

Vergleiche: <http://neptune.netcomp.monash.edu.au/cpe5013/Lecture/Week03/UFS.html> und [http://www.sun.com/software/whitepapers/solaris10/fs\\_performance.pdf](http://www.sun.com/software/whitepapers/solaris10/fs_performance.pdf).

### 3.9 Das WINDOWS-Dateisystem NTFS

WINDOWS 2000 (W2K oder WIN 2K) unterstützt mehrere Dateisysteme, einschließlich FAT, das schon auf Windows 95, MS-DOS, und OS/2 lief. Die Entwickler von W2K entwarfen jedoch ein neues Dateisystem, das W2K Dateisystem (NTFS), mit dem beabsichtigt wurde, den hohen Anforderungen eines sicheren Arbeitsplatzes beziehungsweise Servers im kommerziellen Umfeld zu genügen.

#### Windows NTFS Partition und Cluster Größen:

VOLUMEN GRÖSSE	SEKTOREN PRO CLUSTER	CLUSTERGRÖSSE
≤ 512 Mbyte	1	512 bytes
512 Mbyte - 1 Gbyte	2	1K
1 Gbyte - 2 Gbyte	4	2K
2 Gbyte - 4 Gbyte	8	4K
4 Gbyte - 8 Gbyte	16	8K
8 Gbyte - 16 Gbyte	32	16K
16 Gbyte - 32 Gbyte	64	32K
≥ 32 Gbyte	128	64K

Eine NTFS-Partition baut sich folgendermaßen auf:

...

Abbildung 3.14: NTFS-Partition

[NTFS-Aufbau](#)  
[NTFS-Internals](#)

ATTRIBUT-TYP	BEDEUTEUNG
Standard-Informationen	Zugriffsrechte (Nur-Lesen, Lesen/Schreiben, ect.), Zeitstempel (Dateierzeugung, Dateimodifikation), Link-Zähler
Attributliste	ergänzt den MFT-Eintrag, wenn dieser nicht genug Platz hat
Filename	eine Datei muß einen oder mehrere Namen besitzen
Sicherheits-Deskriptor	spezifiziert, wem die Datei gehört und wer zugriffsberechtigt ist
Daten(-typ)	Inhalt der Datei. Jede Datei besitzt ein unbenanntes Standard-Datenattribut und darf zusätzliche benannte Attribute besitzen
Index root	für Ordner
Index allocation	für Ordner
Volumen-Informationen	Name und Version des Volumes, ...
Bitmap	Bitmap der belegten MFT-Einträge

Tabelle 3.6: NTFS File- und Verzeichnis-Attribut-Typen

### 3.9.1 NTFS Schlüssel-Designkonzepte

NTFS ist ein flexibles leistungsstarkes Dateisystem, das auf einem eleganten einfachen Modell basiert. Die beachtenswerten Eigenschaften von NTFS sind unter anderem:

- **Recoverability/Wiederherstellbarkeit:** Ganz oben auf der Liste der Anforderungen an das neue W2K Dateisystem stand die Fähigkeit der Wiederherstellbarkeit nach Systemabstürzen und Festplattenfehlern. Im Falle solcher Fehler ist NTFS in der Lage, Plattenvolumina wieder in einen konsistenten Zustand zurückzusetzen.  
NTFS verwendet ein Transaktions-Modell für Änderungen im Dateisystem: jede signifikante Änderung wird als eine atomare Handlung behandelt, die entweder gänzlich oder überhaupt nicht durchgeführt wird. Jede Transaktion, die zur Zeit eines Fehlers im Gange war, wird beim Recover ungeschehen gemacht oder zur Vollendung gebracht.  
Außerdem verwendet NTFS eine redundante Speicherung für kritische Dateisystem-Daten, so dass ein defekter Plattensektor nicht zum Verlust von Struktur- oder Statusdaten des Dateisystems führen kann.
- **Security/Sicherheit:** NTFS verwendet zur Erzwingung von Sicherheit das W2K Objekt-Modell. Eine offene Datei wird als ein Exemplar eines File-Objekts mit einem Sicherheitsdeskriptor angesehen, der ihre Sicherheitsattribute definiert.
- **Große Platten und große Dateien:** NTFS unterstützt sehr große Festplatten und sehr große Dateien besser und effizienter als die meisten anderen Dateisysteme, FAT eingeschlossen.
- **Multiple data streams/mehrere Datenströme:** Der eigentliche Inhalt einer Datei wird als ein Strom von Bytes behandelt. In NTFS ist es möglich, mehrere Datenströme für eine einzige Datei zu definieren. Ein Beispiel für die Nützlichkeit dieser Eigenschaft ist es, dass NTFS W2K erlaubt, durch entfernte Macintosh-Systeme verwendet zu werden, um Dateien zu speichern und zu laden. Auf einem Macintosh-System hat jede Datei zwei Bestandteile: die Dateidaten und eine Ressourcenbeschreibung, die Information über die Datei enthält. NTFS behandelt diese beiden Komponenten als zwei getrennte Datenströme.

- **General indexing facility/allgemeine Indizierungsmöglichkeit:** NTFS verbindet mit jeder Datei eine Ansammlung von Attributen. Die Menge der Dateibeschreibungen im Dateiverwaltungssystem wird als eine relationale Datenbank organisiert, so dass Dateien nach jedem Attribut indiziert werden können.

### 3.9.2 Die NTFS Volumen- und Dateistruktur

NTFS verwendet zur Speicherung von Dateien das folgende Konzept:

1. **Sektoren:** Die kleinste physikalische Speichereinheit auf der Platte. Die Datengröße ist eine Potenz von 2 und ist fast immer **512 Bytes**
2. **Cluster:** Ein oder mehr aneinander grenzende (nebeneinander in derselben Spur) Sektoren. Die Cluster-Größe in Sektoren ist eine Potenz von 2.
3. **Volumen:** Eine logische Partition einer Festplatte, die aus einem oder mehreren Clustern besteht und durch ein Dateisystem dazu verwendet wird, Plattenplatz zu reservieren. Ein Volumen besteht zu jeder Zeit aus Dateisystem-Informationen, einer Ansammlung von Dateien und zusätzlichem unzugeteilten Plattenplatz, der Dateien zugeteilt werden kann. Ein Volumen kann auf einem Teil einer Platte realisiert sein oder es kann sich über mehrere Platten erstrecken. Wenn ein Hardware- oder Software-RAID 5 verwendet wird, besteht ein Volumen aus Streifen (Stripes), die sich über mehrere Platten verteilen. Die **maximale** Volumen-Größe für NTFS ist  $2^{64}$  Bytes.

Der Cluster ist die fundamentale Zuteilungseinheit in NTFS.

### 3.9.3 Recoverability/Wiederherstellbarkeit

NTFS ermöglicht es, das Dateisystem nach einem System-Absturz in einen konsistenten Zustand zurückzuführen. Die Schlüsselemente, die eine Wiederherstellung gewährleisten sind die folgenden:

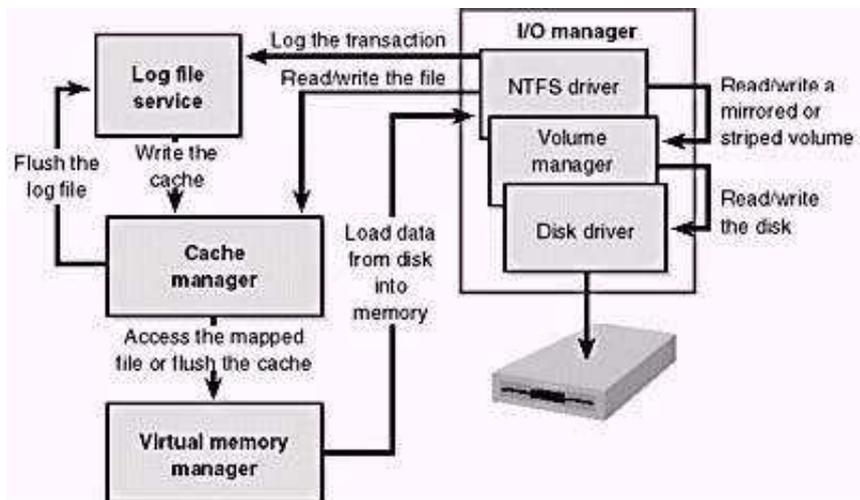


Abbildung 3.15: NTFS-Komponenten

- **I/O manager:** Enthält den NTFS Treiber, der die grundlegenden NTFS-Funktionen Öffnen, Schließen, Lesen, Schreiben von Dateien bereit stellt. Außerdem kann das Software-RAID-Modul FTDISK für die Benutzung konfiguriert werden.

- **Log file service:** Führt ein Log der Platten-Schreiboperationen. Die Log-Datei wird dazu verwendet, ein NTFS-formatiertes Volumen im Falle eines System-Fehlers wiederherzustellen.
- **Cache manager:** Verantwortlich dafür, Datei-Lese- und -Schreib-Operationen zu puffern, um die Leistung zu erhöhen. Der Cache-Manager optimiert das Platten-I/O, indem er eine lazy-write und lazy-commit Technik verwendet.
- **Virtual memory manager:** NTFS greift auf gepufferte Dateien zu, indem es Dateiverweise auf virtuellen Speicher abbildet und virtuellen Speicher liest beziehungsweise schreibt.

Es ist wichtig zu wissen, dass die von NTFS verwendeten Wiederherstellungsverfahren dafür bestimmt sind, Dateisystem-Metadaten wiederherzustellen, nicht Datei-Inhalte. So sollte der Benutzer nie ein Volumen oder die Verzeichniss-/Datei-Struktur einer Applikation wegen eines Absturzes verlieren.

Die folgenden Schritte garantieren die Dateisystem-Wiederherstellung:

1. Zunächst veranlaßt NTFS das Logdateisystem, in der Logdatei im Cache alle Transaktionen zu registrieren, bei der die Volumen-Struktur modifiziert wird.
2. NTFS modifiziert das Volumen (im Cache).
3. Der Cache-Manager weist das Logdateisystem an, die Logdatei auf die Platte zu schreiben.
4. Ist die Logdatei sicher auf der Festplatte gespeichert, schreibt der Cache-Manager die Volumen-Änderungen auf die Platte.

## 3.10 Weitere UNIX- und LINUX-Dateisysteme

### 3.10.1 System V FS (S5FS) (1972)

- Erstes UNIX Dateisystem, auch als **FS** bekannt
- Festplatten-Layout besteht aus: **Bootblock**, **Superblock**, **Inodes**, **Datenblöcke**
- **512, 1024, ...** Bytes Blockgröße, **keine Fragmente**
- Sehr einfach und sehr langsam

### 3.10.2 Berkley Fast File System (FFS, dann UFS/UFS2) (1983, 1994)

- Metadaten werden in **Cylindergruppen** redundant auf der Festplatte verteilt
- Blockgröße ist min. 4 KB, die Fragmentgröße ist 1 KB
- physische Platten-Parameter werden beim Layout berücksichtigt
- Viel bessere Performance
- 256 Byte Dateinamen, Datei-Locks, Symlinks, rename() und Quotas

#### Verbesserungen:

- 1991: Verbesserungen der Block-Zuteilung und der vorzeitigen Lesepolitik (Larry McVoy und Greg Ganger)
- 1999: Soft updates (Kirk McKusick und Greg Ganger)
- 1999: UFS logging (Neil Perrin)
- 2004: Multi-Terabyte UFS (Lori Alt und Jeff Bonwick)

### 3.10.3 Log-Structured File System (LFS) (1993)

- Alle **on-disk Daten** inkrementell in Form von aufeinander folgenden Einträgen im zirkularen Log
- Crash-/Fehler-/Störungsbeseitigung vorwärts vom letzten Checkpoint/Kontrollpunkt
- 70% vom gesamten Festplattenplatz effektiv genutzt, aber FFS kann dies ebenso
- Mendel Rosenblum und Jhon K. Ousterhoud, [The Design and Implementation of a Log-Structured File System](#), 13th ACM SOSP

### 3.10.4 ext2 und ext3 (1992, 1999)

- ähnlich wie FFS, keine Fragmente
- ext3 zusätzlich: Journaling System: nach einem Absturz/Crash wird **fsck** nicht benötigt; H-Tree-Organisation der Verzeichnis-Indices, Dateitypen im Verzeichnis, gelöschte inode-Adressen werden mit Nullen überschrieben
- Primäre Eigenschaften sind **Einfachheit**, **Performanz** und **Wiederherstellbarkeit**
- vergleiche zur Performanz: [http://www.sun.com/software/whitepapers/solaris10/fs\\_performance.pdf](http://www.sun.com/software/whitepapers/solaris10/fs_performance.pdf)
- ext4-Entwicklung:  
<http://www.heise.de/open/Neuigkeiten-rund-um-die-Linux-Dateisysteme-Reiser4-btrfs-ext3-und-ext4-/news/meldung/101944>

### 3.10.5 SGIs **XFS** (1994)

- FFS mit Extensions zur Performanz-Steigerung
- **B+-Bäume** (Tracks/Spuren) für freien Plattenplatz, Index-Verzeichnisse, lokalisierte Dateiblocks und Inodes; zusätzlich erstes genutztes 64-Bit Dateisystem
- Der Fokus liegt in der Skalierbarkeit und im I/O-Streaming
- Dynamische Inode-Zuteilung, Logging, Volumen-Manager, Multithreadfähig beim lesen/schreiben, ACLs
- online Resizing
- online Defragmentierung
- real-time E/A (Video-Streaming)

### 3.10.6 IBMs **JFS** (1990)

- journaling Dateisystem
- ACLs
- B+Tree Adressierung
- sparse and dense Files

### 3.10.7 **Reiserfs Version 1-4** (ca 2001 - 2005)

- Entworfen für kleine Dateisysteme und große Verzeichniss- Performanz
- online Resizing (groth only), Journaling
- Logging, **B+-Bäume**, tail-Packing
- für kleine Dateien ( $\leq 4K$ ) 10...15 mal schneller als ext3/ext3
- keine Defragmentierungsmöglichkeiten
- **Reiser4**: effizienteres Journaling, effizienteres I/O für kleine Dateien, atomic Dateisystem, keine ACLs, Pluggin-Unterstützung (Codierung, Kompression), **Entwicklungen im Jahre 2008**

### 3.10.8 Apple Macintosh's **HFS+** (1998)

- Journaling
- B-Trees
- Erweitertes jedoch trotzdem abwärtskompatibles HFS+ (z.Zt.: Dateinamen mit signifikanter Groß-/Kleinschreibung)
- Erweiterte Datei-Attribute (Dateityp, ...), Ressourcen-Fork
- **alternate data stream**-Konzept
- ... wird auch für die meisten Macintosh CDs verwendet

### 3.10.9 Zetabyte File System (ZFS) (2005)

- verfügbar erst in SOLARIS10-Update
- Eingebauter Volumen-Manager: äußerst vereinfachte Volume-Administration (self-Managing)
- 128-Bit Dateisystem
- parallele, O(1)-Verzeichniss-Operationen
- online Resizing
- stets konsistenter Plattenstatus
- Selbstheilende Daten, dynamisches Striping, unterschiedliche Blockgröße, unbegrenzte, zeitkonstante Lese-/Schreib-Snapshots

<http://en.wikipedia.org/wiki/Zfs>

<http://opensolaris.org/os/community/zfs/demos/zfsdemo.pdf>

Vertiefung: [http://www.cs.vu.nl/res/theses/oey\\_thesis.ps](http://www.cs.vu.nl/res/theses/oey_thesis.ps)

Das Filesystem-Howto

## 4 RAID — Redundant Arrays of Independent Disks

RAID-Plattensysteme (**Arrays of Inexpensive Disks**) stellen auf der Basis kleiner (billiger) Platten hochperformante und/oder hochsichere virtuelle Platten bereit.

Dem RAID steht das **JBOD** gegenüber, bei dem zwar mehrere Festplatten innerhalb eines Computers eingebaut werden, aber keine logische Anordnung (Array) entsteht.

### 4.1 RAID 0: Beschleunigung ohne Fehlertoleranz/Redundanz (engl. striping)

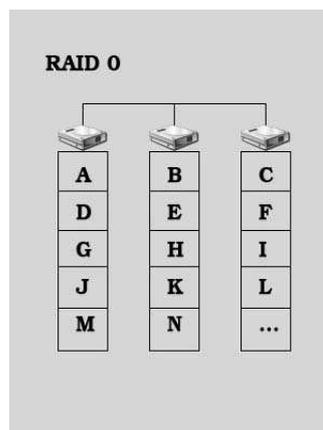


Abbildung 4.1: RAID 0

## 4.2 RAID 1: Spiegelung (mirroring/duplexing)

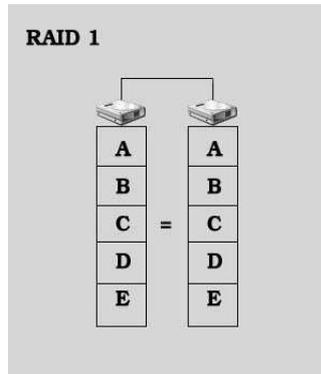


Abbildung 4.2: RAID 1

## 4.3 RAID 5: Performance + Parität

RAID 5 kombiniert Redundanz mit höherer Performanz:

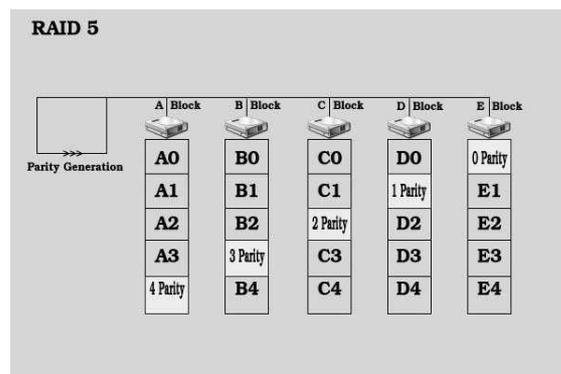


Abbildung 4.3: RAID 5

## 4.4 RAID 6

RAID 6 funktioniert ähnlich wie RAID 5, verkräftet aber den Ausfall von bis zu zwei Festplatten. Hier werden nicht ein, sondern zwei Fehlerkorrekturwerte berechnet und so über die Platten verteilt, so dass Daten und Paritätsblöcke auf unterschiedlichen Platten liegen.

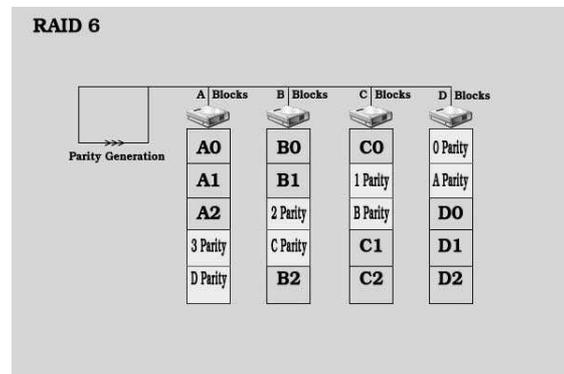


Abbildung 4.4: RAID 6

## 4.5 Kombinations-RAIDs (RAID 10, 01, ...)

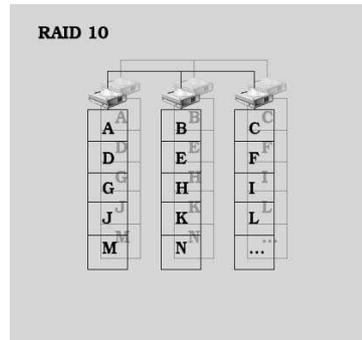


Abbildung 4.5: Kombinations-RAID 10

Weitere Vertiefung:

Fibre-Channel  
RAID 01  
Hardware-RAID  
SW-RAID  
Linux LVM

## 4.6 Metadevicees (Software-RAID) unter Solaris

In Solaris kann man Software-RAIDs als Metadevicees angeben:

d25: Spiegel

```
Untergeordneter Spiegel 0: d21
  Status: OK
Untergeordneter Spiegel 1: d22
  Status: OK
Kontrolllauf: 1
Leseoption: roundrobin (Standard)
Schreiboption: parallel (Standard)
Größe: 8388000 Blöcke (4,0 GB)
```

d21: Untergeordneter Spiegel von d25

```
Status: OK
Größe: 8388000 Blöcke (4,0 GB)
Stripe 0:
  Gerät      Startblock  Daten      Status Wiede Hot-Spare
  c0t8d0s7    0           Nein        OK     Ja
```

d22: Untergeordneter Spiegel von d25

```
Status: OK
Größe: 8415657 Blöcke (4,0 GB)
Stripe 0:
  Gerät      Startblock  Daten      Status Wiede Hot-Spare
  c0t9d0s7    0           Nein        OK     Ja
```

d23: Concat/Stripe

```
Größe: 8415657 Blöcke (4,0 GB)
Stripe 0:
  Gerät      Startblock  Daten Wiederzuw.
  c0t10d0s7  0           Nein     Ja
```

Die Arbeit mit Metadevicees ist jedoch sehr technisch und administrationsunfreundlich.  
Ein Beispiel:

- Lege spezielle Partitionen (Solaris-Slices) für die Aufnahme von Metadaten des SW-RAIDs an (Größe im MB-Bereich):

```
c0t12d0 <FUJITSU-MAT3073NC-0108 cyl 65533 alt 2 hd 2 sec 1095>
```

Total disk cylinders available: 65533 + 2 (reserved cylinders)

Part	Tag	Flag	Cylinders	Size	Blocks	
0	unassigned	wm	0	0	(0/0/0)	0
1	unassigned	wu	0	0	(0/0/0)	0
2	backup	wu	0 - 65532	68.43GB	(65533/0/0)	143517270
3	unassigned	wm	0 - 99	106.93MB	(100/0/0)	219000
4	unassigned	wm	0	0	(0/0/0)	0
5	unassigned	wm	0	0	(0/0/0)	0
6	unassigned	wm	0	0	(0/0/0)	0
7	unassigned	wm	100 - 65532	68.33GB	(65433/0/0)	143298270

- Erstelle mit Hilfe dieser Partitionen eine Metadatenbank mit genügend „Replica“ (Redundanz):

```
# metadb -a -f -c 3 c0t11d0s3
# metadb -a -c 3 c0t12d0s3
...
```

- Lege die UFS-Dateisysteme an und wandle sie zu Metadevices (stripes):

```
newfs /dev/rdisk/c0t11d0s7
newfs /dev/rdisk/c0t12d0s7

metainit -f d111 1 1 c0t11d0s7
metainit -f d112 1 1 c0t12d0s7
```

- Erzeuge eine Spiegel (als Metadevice):

```
metainit d125 -m d121
```

- Füge diesem den zweiten zugehörigen Stripe hinzu:

```
metattach d115 d112
```

- Überprüfe das Ergebnis:

```
metastat

d115: Spiegel
      Untergeordneter Spiegel 0: d111
      Status: OK
```

Untergeordneter Spiegel 1: d112  
 Status: OK  
 Kontrolllauf: 1  
 Leseoption: roundrobin (Standard)  
 Schreiboption: parallel (Standard)  
 Größe: 143298270 Blöcke (68 GB)

d111: Untergeordneter Spiegel von d115  
 Status: OK  
 Größe: 143298270 Blöcke (68 GB)  
 Stripe 0:

Gerät	Startblock	Daten	Status	Wiede	Hot-Spare
c0t11d0s7	0	Nein	OK	Ja	

d112: Untergeordneter Spiegel von d115  
 Status: OK  
 Größe: 143298270 Blöcke (68 GB)  
 Stripe 0:

Gerät	Startblock	Daten	Status	Wiede	Hot-Spare
c0t12d0s7	0	Nein	OK	Ja	

- Mounte das neue Device mit Hilfe der Datei /etc/vfstab:

/dev/md/dsk/d115	/dev/md/rdsk/d115	/data2	ufs	2	yes
------------------	-------------------	--------	-----	---	-----

## 4.7 Als Alternative: Self-Managing mit Hilfe von zfs

(Auszug aus <http://opensolaris.org/os/community/zfs/demos/zfsdemo.pdf>.)

```
# format
...
format> par
partition> pr
Current partition table (original):
Total disk cylinders available: 24620 + 2 (reserved cylinders)

Part      Tag      Flag      Cylinders      Size      Blocks
  0 unassigned  wm        0 - 4101      5.65GB    (4102/0/0) 11850678
  1 unassigned  wm       4102 - 8203    5.65GB    (4102/0/0) 11850678
  2 backup      wm        0 - 24619    33.92GB   (24620/0/0) 71127180
  3 unassigned  wm         0              0          (0/0/0)      0
  4 unassigned  wm       8204 - 12305   5.65GB    (4102/0/0) 11850678
  5 unassigned  wm      12306 - 16407  5.65GB    (4102/0/0) 11850678
  6 unassigned  wm      16408 - 20509  5.65GB    (4102/0/0) 11850678
  7 unassigned  wm     20510 - 24611  5.65GB    (4102/0/0) 11850678

# zpool list
Keine Pools verf
# zpool status
Keine Pools verf
# zfs list
Keine Datengruppen verf

# zpool create mypool mirror c0t1d0s4 c0t1d0s5
# zpool list
NAME                SIZE    USED    AVAIL    CAP    HEALTH    ALTROOT
mypool              5,62G    90K    5,62G    0%    ONLINE    -
# zpool status
Pool: mypool
Status: ONLINE
scrub: Keine erforderlich
config:

NAME                STATE    READ    WRITE    CKSUM
mypool              ONLINE    0       0       0
  mirror            ONLINE    0       0       0
    c0t1d0s4        ONLINE    0       0       0
    c0t1d0s5        ONLINE    0       0       0
```

Fehler: Keine bekannten Datenfehler

# zfs list

NAME	USED	AVAIL	REFER	MOUNTPOINT
mypool	87K	5,54G	24,5K	/mypool

# zfs create mypool/myfs

# zfs list

NAME	USED	AVAIL	REFER	MOUNTPOINT
mypool	116K	5,54G	24,5K	/mypool
mypool/myfs	24,5K	5,54G	24,5K	/mypool/myfs

# zpool add mypool mirror c0t1d0s6 c0t1d0s7

# zpool list

NAME	SIZE	USED	AVAIL	CAP	HEALTH	ALTROOT
mypool	11,2G	123K	11,2G	0%	ONLINE	-

# zpool status

Pool: mypool

Status: ONLINE

scrub: Keine erforderlichlich

config:

NAME	STATE	READ	WRITE	CKSUM
mypool	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c0t1d0s4	ONLINE	0	0	0
c0t1d0s5	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c0t1d0s6	ONLINE	0	0	0
c0t1d0s7	ONLINE	0	0	0

Fehler: Keine bekannten Datenfehler

# zfs list

NAME	USED	AVAIL	REFER	MOUNTPOINT
mypool	120K	11,1G	26,5K	/mypool
mypool/myfs	24,5K	11,1G	24,5K	/mypool/myfs

# df -k

...

mypool	11612160	25	11612041	1%	/mypool
mypool/myfs	11612160	24	11612041	1%	/mypool/myfs

# ls -al /mypool

```
Gesamt 8
drwxr-xr-x  3 root    sys          3 Jul 10 12:04 .
drwxr-xr-x 40 root    root        1024 Jul 10 12:01 ..
drwxr-xr-x  2 root    sys          2 Jul 10 12:04 myfs
```

```
...
# zfs destroy mypool/myfs
# zfs list
NAME      USED  AVAIL  REFER  MOUNTPOINT
mypool    88K  11,1G  24,5K  /mypool
...
# zfs destroy mypool/myfs
# zfs list
NAME      USED  AVAIL  REFER  MOUNTPOINT
mypool    88K  11,1G  24,5K  /mypool
# zpool destroy mypool
# zpool list
Keine Pools verf
```

Die lästigen und vor allem fehleranfälligen Management-Operationen sind auf das minimal notwendige reduziert worden!

<http://sun.systemnews.com/articles/124/3/sw/20146>

[http://www.wizy.org/wiki/ZFS\\_on\\_FUSE](http://www.wizy.org/wiki/ZFS_on_FUSE)

<http://opensolaris.org/os/community/zfs/>

## 5 Fallstudie: Using Solaris containers for building a secure webshop

<http://www.sun.com/software/solaris/howtoguides/s10securityhowto.jsp>

Andere Anwendungsszenarios für virtuelle Computer



## 6 Secure global desktop/Virtual network computing/Thin clients

Im **virtual network computing** trennt man den Arbeitsplatz (Tastatur/Maus/...) über ein Netzwerk vom Computer. Ähnliche Methodiken:

- UNIX **X-Window** (UNIX (native), Windows: XWin32, ...)
- **Windows Terminal Services / RemoteDesktopProtocol** (UNIX: Rdesktop/Citrix, Windows: native/Citrix, ...)
- VNC (<http://www.realvnc.com/what.html>, „Secure Global Desktop“, ...)
- Nachfolger von X-Terminals: [http://en.wikipedia.org/wiki/Sun\\_Ray](http://en.wikipedia.org/wiki/Sun_Ray)
- XenDesktop: <http://www.citrix.de/produkte/schnellsuche/xendesktop/>

Siehe dazu auch: [http://en.wikipedia.org/wiki/Thin\\_client](http://en.wikipedia.org/wiki/Thin_client)



# 7 Kooperativ arbeitende Applikationen

## 7.1 Neue implizit parallelisierende Compiler/Programmiersprachen: Fortress

Implicit parallelism in Fortress:

<http://research.sun.com/sunlabsday/docs.2007/Talks/Track4/SteeleSunLabsOpenHouse2007Public.pdf>

<http://projectfortress.sun.com/Projects/Community/wiki/ImplicitParallelismInFortress>

<http://research.sun.com/projects/plrg/faq/index.html>

[http://de.wikipedia.org/wiki/Fortress\\_%28Programmiersprache%29](http://de.wikipedia.org/wiki/Fortress_%28Programmiersprache%29)

Parallelism is the default:

<http://research.sun.com/projects/plrg/PLDITutorialSlides9Jun2006.pdf>

## 7.2 OpenMP

<http://en.wikipedia.org/wiki/OpenMP>

<http://www.compunity.org/events/upcomingevents/iwomp2006/iwomp2006-tutorial-openmp-rvdp.pdf>

<http://msdn.microsoft.com/en-us/magazine/cc163717.aspx>

<http://bisqwit.iki.fi/story/howto/openmp/>

## 7.3 Das Klonen von Prozessen

Ein Prozess ist eine unterbrech- und wiederfortsetzbare Ablauf-Instanz eines Programms.

Mehrere Prozesse können „shared“ Codeteile gemeinsam nutzen.

Prozesse enthalten Informationen z.B. zu:

- Prozesspriorität
- Prozessstatus (blocked, running)

- Daten-Adressbereich
- Dateien in Nutzung
- ...

### Prozess-Attribute

Zwei Prozesse in einem Multitasking-Betriebssystem sind (absichtlich) so konzipiert, daß sie sich nicht gegenseitig stören. Sie können nur eingeschränkt durch Mittel der

### Inter-Prozess-Kommunikation

miteinander Daten austauschen.

Neue Prozesse (Kindprozesse) werden durch Klonen des Vaterprozesses erzeugt:

### YoLinux-Tutorial `fork()`

Änderung der Prozess-Segmentgröße:

<http://www.gnu.org/software/libtool/manual/libc/Resizing-the-Data-Segment.html>

Duplizieren von Filedeskriptoren im Kindprozess:

<http://www.pronix.de/pronix-63.html>

## 7.4 Posix Threads: Starten von (detachten) **Threads**

### YoLinux-Tutorial threads

### Multithread Programming

Literatur:

- D. R. Butenhof: Programming with POSIX Threads, Addison-Wesley, 1997
- B. Nichols u.a.: Pthreads Programming, O'Reilly, 1998

## 7.5 Client/Server-Prozesse

### Unix Daemon Server Programming

## 7.6 strace

Anwendungsszenarien von strace:

<http://en.wikipedia.org/wiki/Strace>

Unterbrechen blockierter System-Calls:

<http://answers.yahoo.com/question/index?qid=20080408161143AA2kZ6m>