1 Unix Einführung

1.1 Was ist Unix?

- Unix ist wie DOS, OS/2 oder Windows ein Betriebssystem
- Es verwaltet Zugriffe die Hardware in Rechnern
- im Gegensatz zu anderen Betriebssystemen ist Unix ein vollwertiges 32 Bit Betriebssystem, was auf verschiedensten Architekturen verfügbar ist
- auf PC's und auf hochleistungs-Servern im Einsatz
- hervorragendes Netzwerk/Serverbetriebssystem
- Multiuser (Terminalunterstützung) und Multitaskingfähig
- viele Unix Derivate

1.2 Geschichte von Unix

- 1965-69: Bell Laboratories, General Electric und Projekt MAC (Massechussets Institute of Technology) versuchen das Betriebssystem Multics mit folgenden Eigenschaften zu entwickeln:
  - grosse Gruppe von Benutzern soll gleichzeitig an einem Rechner arbeiten koennen (Terminals)
  - allen soll der Zugriff auf die Rechenleistung und den Speicher gewährleistet werden
  - Benutzer sollen auf gemeinsame Dateien zugreifen können
- 1969: Projekt Multics wurden den Anforderungen nicht gerecht und wurde eingestellt
- 1969-71: die Mitarbeiter des Projektes entwickelten es weiter, es entstand ein erstes Filesystem für Unix
- 1971-73: erste Version von Unix wurde als Textprozessor eingesetzt (16 kb System, 8 kb Programme, 512 kb Festplatte, 64 kb maximale Dateigröße)
- zur Weiterentwicklung wurde die Programmiersprache B verbessert, es entstand C
- 1973-77: Unix wurde in C komplett neu geschrieben
- freie verfügbarkeit für Universitäten
- 1977-82: Unix wurde populärer, immer mehr Hersteller entwickelten eigene Versionen
- Unix wurde als C-Quellcode verkauft, so war sichergestellt, dass Organisationen es nicht nur benutzen sondern auch ändern können
- dadurch kam es zu immer neuen Verbesserungen, Unix wurde ein dynamisches Betriebssystem
- dadurch entstanden mehrere Unix Familien (neben der ursprünglichen Version Unix System V von AT&T entstanden z.B. die Berkeley-Standard-Distribution (BSD), Xenix, Sinix, Ultrix und auch Linux)

1.2.1 Entstehung von Linux

- anfang der 90er Jahre wurde Linux von einem finnischen Studenten Linus Torvalds ins Leben gerufen
- Zielstellung war es ein Unix System für den damals sehr verbreiteten 386 PC Prozessor zu schreiben
- AT&T hatte damals jegliche öffentliche Verwendung des Unix Quellcodes verboten, somit war auch eine private Anwendung kaum möglich
- hierbei orientierte sich Linux am System V Standard von AT&T und am allgemeingültigen POSIX Standard
- der Quellcode wurde frei verfügbar gemacht
- Linux wurde per ftp angeboten und bald weltweit weiterentwickelt
- nach einer rasanten Entwicklung kann man sagen, dass Linux zu den effizientesten Unix Versionen zählt

1.3 Merkmale von Unix

- lassen sich am besten am Vergleich Unix-DOS erklären
- Unix ist ein 32 Bit Betriebssystem un nutzt die Leistungsfähigkeit moderner 32 Bit CPU's voll aus (DOS ist 16 Bit)
- Aufgabe des BS ist es das Zusammenspiel von Software und Hardware zu steuern
- diese Organisation übernimmt der Kernel (der Kernel ist das eigentliche Betriebssystem)
- die Hardware wird von Treibern angesteuert, die dem Kernel ein einheitliches Interface zur Verfügung stellen
- jedoch sind DOS uns Unix Treiber nicht kompatibel (andere Architektur)
- Unix ist voll Multitaskingfähig (mehrere Prozesse laufen gleichzeitig auf nur einem Prozessor)
- im Kernel managt der Scheduler die Zuteilung des Prozessors zu den Programmen (jedes Programm darf ihn eine gewisse Zeit nutzen)
- jedem Prozess wird sein eigener Speicherbereich simuliert, dadurch kann man nicht auf Systemspeicher zugreifen, und Systemabstürze erzeugen
- zusätzlich kann durch Paging mehr Speicher als tatsächlich vorhandener RAM zur Verfügung gestellt werden (vgl. Swapping, Virtual Memory bei Windows)
- Unix ist Multiuserfähig (mehrere Benutzer arbeiten gleichzeitig an einem System)

1.4 Starten des Systems

- nach dem Einschalten erscheint der Bootmanager, hier kann man einen Linux-Kernel oder ein anderes zu bootendes Betriebssystem wählen
- der Runlevel, in den gebootet wird kann dem Kernel übergeben werden (im LILO z.B. LILO: linux 1 - bootet in Runlevel 1, sehr nützlich bei Netzwerkproblemen)
- das System bootet nun in das eingestellte Runlevel
- Beschreibung der verfügbaren Runlevels (Linux):
  - Runlevel 0: System anhalten
  - Runlevel 1: Single User Mode: Nur ein Benutzer kann arbeiten, meistens root. Es sollten nur die wichtigsten Dienste gestartet sein
  - Runlevel 2: Mulituser, no network: Es können mehrere Benutzer arbeiten, ohne Netzwerk-Exports (NFS) (multiuser with no network services exported)
  - Runlevel 3: Normal, Multiuser: Normaler Modus
  - Runlevel 4: Reserviert: normale Benutzung, Multiuser
  - Runlevel 5: Multiuser mit X-Anmeldung: Es erscheint der X-Server zur Benutzer-Anmeldung
  - Runlevel 6: Reboot: Rechner wird neugestartet
- mittels "/sbin/runlevel" kann das letzte und das aktuelle runlevel ermittelt werden
- wenn der Bootvorgang erfolgreich abgeschlossen ist erscheint die Aufforderung zur Anmeldung am System
- Login mit Benutzername und Passwort
- shell wird gestartet (in diesem Fall die bash)
- shell = Mensch-Maschiene Schnittstelle

1.5 Allgemeine Grundlagen

1.5.1 X-Windows

X-Windows ist eine grafische Oberfläche für Unix, im entferntesten vergleichbar mit Windows. Dient dazu, auch grafische Inhalte (Bilder, Videos ...) anzeigen zu können.
Die Oberfläche wird mittels des Kommandos "startx" gestartet (falls sie nicht defaultmäßig aktiviert ist).

1.5.2 Einführung in die Shell am Beispiel der Bash

Eingabe/Ausgabe

- Eingabe erfolgt über die Standardeingabe (STDIN - Standard Input), meist die Tastatur (aber auch ueber scripte oder pipes)
- Ausgabe teilt sich in zwei Ströme (Streams), die Standardausgabe (STDOUT - Standard Output) und die Standardfehlerausgabe (STDERR - Standard Error)
- dadurch wird eine getrennte Bahendlung von Nutzdaten und Fehlern möglich (bei vielen Daten sehr sinnvoll!)
- STDOUT wird oft durch "-" abgekürzt

Starten und Beenden von Programmen

- Starten von Programmen durch eingabe ihres Pfades (oder eines Kommandos)
- Programm in dem Hintergrund starten: <kommando> & (z.B. “less /etc/passwd &“)
- Programme im Hintergrund anzeigen: “jobs“
- Programm in den Vordergrund holen: “fg [jobid]“
- laufendes Programm stoppen und in den Hintergrund schicken: Strg-Z + “bg“
- laufendes Programm beenden: Strg-C

Aliase - Abkürzungen auf der Bash
Lange oft verwendete Kommandos kann man mittels aliasen abkürzen. Dies spart Tipparbeit. Mit dem Kommando “alias“ kann man alle gültigen aliase abfragen, und neue definieren z.B. “alias l=“ls -la““. Mit dem Kommando unalias kann man einen bestehenden alias löschen.

Metazeichen - Meta Characters
... sind Zeichen, die auf der Bash besondere Funktionen auslösen und nicht ohne weiteres verwendbar sind (jedoch meist mit “\“ zu escapen (z.B. “\&“)

Zeichen	Bedeutung
|	unnamed Pipe
&	im Hintergrund starten
;	Befehl abschliessen
()	Funktionsaufruf
< >	Ausgabeumleitung - Output Redirection
<Leerzeichen>	Platzhalter für Parameter

Output Redirection
Grundlegend gibt es die drei genannten Standard Streams: stdin, stdout, stderr (std=standard). Zur besseren Handhabung der Daten kann man die Ausgabestreams umleiten (entweder in Dateien oder andere Streams).
Weiterleitungsmöglichkeiten:

stdout nach datei z.B. ls > ls.txt

stderr nach datei z.B. ls 2> ls-err.txt

stdout nach stderr z.B. ls 1>&2

stderr nach stdout z.B. ls 2>&1

stderr und stdout nach datei z.B. ls &> ls-all.txt

stdout an datei anhängen z.B. ls >> ls-all.txt

Verkettung - pipes
Mittels pipes kann man die Ausgaben eines Programmes als Eingaben einens zweiten Programmes nutzen (beliebig oft wiederholbar).
unbenannte (unnamed) Pipes (Befehle werden einfach durch | verbunden):

z.B. ls −1 | grep test

benannte (named) Pipes (Pipe als Datei im Filesystem

z.B. mkfifo testfifo
ls −1 > testfifo &
grep Folien testfifo

Variablen
Bash Variablen entsprechen allgemeinen Variablen bekannt aus anderen Programmiersprachen, mit den Ausnahmen, dass sie nicht deklariert werden müssen und keinen Datentyp besitzen.
Verwendung:

Eingabe:
<Variablenname>=<Inhalt>
z.B. TEST="HALLO"
Ausgabe:
echo $<Variablenname>
z.B. echo $TEST

Mit Variablen kann man auch rechnen:

```
X=2
# X=2
echo $X
# 1. Variante:
let X=X+5
# X=7
echo $X
# kurze Variante:
X=$[$X+5]
# X=12
echo $X
```

und Werte einlesen:

```
read X
echo $X;
```

Programmrückgaben in Variablen schreiben
oft ist es sinnvoll, die Rückgabewerte eines Programms in eine Variable zu schreiben, dies geschieht folgendermassen:

DATE=$(date)

in DATE steht jetzt der Rückgabewert von Kommando DATE (wenn dieser mehrzeilig ist werden die Zeilenumbrüche entfernt!)

echo $DATE

Einfache Bash-Scripte
ein erstes Hello World Script - mehrere Bash-Befehle hintereinander ausführen = Script (vgl. DOS .bat Dateien)

#!/bin/sh

echo ”hello world!”

1.6 Informationen zum System

- Kommandos werden über die shell eingegeben
- Befehle:

Kommando	Erläuterung
id [Nutzer]	Zeigt die User-ID und die Gruppenzugehörigkeit des angegebenen Nutzers an, falls kein Nutzer angegeben werden die Informationen des aktuell angemeldeten Nutzers angezeigt
date	Zeigt die aktuelle Systemzeit an
man [Kommando]	Zeigt eine Hilfe zum Kommando an, wird mit “q“ beendet
cat [Datei]	Gibt den Inhalt der Datei auf der Konsole aus
tac [Datei]	Gibt den Inhalt der Datei (letzte Zeile zuerst) auf der Konsole aus
more [Datei] oder less [Datei]	Gibt den Inhalt der Datei seitenweise auf der Konsole aus (nächste Seite mit Leertaste, Beenden mit “q“)
who oder w	Zeigt die derzeit am System angemeldeten Nutzer an
tty	Gibt den Namen des derzeit aktiven Terminals aus
write user [tty]	Schreibt dem User (auf das Terminal “tty“) eine Nachricht (Beenden mit Ctrl-D)
mesg [y|n]	Schaltet den Nachrichtenempfang ein (y) oder aus (n), ohne Parameter gibt es den aktuellen Status aus
sync	Schreibt alle Daten die derzeit im RAM sind auf physische Medien
echo [Parameter]	Gibt Parameter auf der Konsole aus
sleep [Zeit]	Verzögert um eine bestimmte Zeit (in Sekunden)
su [Username]	wechselt den derzeit angemeldeten Benutzer (wenn kein User angegeben zum Superuser root)
uname -a	Zeigt die vollständige Version des laufenden Systems an
cal [Monat] [Jahr]	Gibt einen Monatskalender aus
bc	ein Taschenrechner für die Konsole (mit Strg-D beenden)

clear oder Strg-L	Löscht die aktuelle Konsole
df	Gibt Informationen zur Plattenbelegung des Systems aus
free	Gibt Informationen zur Speicherbelegung des Systems
last	Gibt die zuletzt angemeldeten Nutzer aus
top	Ermöglicht eine Echtzeit-Analyse des Systems (wird mit “q“ beendet)

1.6.1 Dateien und Programme / Filesystem

- Struktur des Filesystem entspricht weitestgehend dem DOS FS
- keine Laufwerke, nur ein root-Verzeichnis (/)
- Anordnung in Verzeichnisse, die wieder Verzeichnisse oder Dateien enthalten können -> Hierarchie
- Datei- oder Verzeichnisnamen können bis zu 255 Zeichen lang sein
- es wird zwischen Groß- und Kleinschreibung unterschieden
- Leerzeichen können verwendet werden, man muss dann jedoch den Pfad in Anführungszeichen schreiben (z.B. cd “old files“), oder per Backslash escapen (z.B. “cd old\ files“)
- Grundlegende Dateisystemstruktur:

Pfad	Beschreibung
/bin	(bin = Binaries) - Verzeichnis mit ausführbaren Programmen für alle Nutzer (Allgemeine Konsolentools)
/boot	Verzeichnis welches den/die Kernel und die Bootkonfigurationsdateien enthält
/dev	(dev = Devices) - Verzeichnis mit den Gerätedateien, jede Datei symbolisiert ein Gerät im Rechner
/etc	Verzeichnis mit allen Konfigurationsdateien
/home	Verzeichnis mit allen Home-Verzeichnissen der Nutzer
/lib	(lib = Libraries) - Verzeichnis mit Systembibliotheken
/lost+found	Hier landen verlorene Dateien nach einem Systemcrash
/mnt	Standard Mount-Points
/opt	(opt = optional) - Verzeichnis mit optionalen Softwarepaketen (z.B. Netscape)
/proc	Virtuelles Verzeichnis, um auf Kernelfunktionen zugreifen zu können
/root	Home-Verzeichnis des root-Nutzers
/sbin	(sbin = Superuser-Binaries) - Verzeichnis mit Programmen für den Superuser
/tmp	temporäres Verzeichnis für alle Nutzer
/usr	Verzeichnis mit weiteren Programmen, welches meist zentral auf einem Server abgelegt wird
/var	Verzeichnis mit variablen Bewegungsdaten (logfiles, mails ...)

- nach erfolgreichem Login ist jeder Nutzer in seinem Home-Verzeichnis (/home/<Username>)
- das Home einen Nutzers wird mit “∼“ abgekürzt
- Verzeichniswechsel durch Kommando “cd [Verzeichnisname]“ (wenn kein Verzeichnis angegeben, automatisch ins Home-verzeichnis)
- besondere Verzeichnisse:

  ∼ Home-Verzeichnis des aktuellen Nutzers

  / Root-Verzeichnis des Systems

- **.** Das aktuelle Verzeichnis (“cd .“ ändert nichts)
- **..** Das übergeordnete Verzeichnis (“cd ..“ steigt eine Ebene auf)

- mit dem Kommando “ls“ kann man sich alle Dateien und Unterverzeichnisse des aktuellen Pfades anzeigen lassen
- Syntax des ls-Kommandos:

ls	zeigt alle normalen Dateien an (versteckte nicht)
ls -a	zeigt auch versteckte Dateien (beginnen mit .) an
ls -C	gibt Dateien spaltenweise aus
ls -1	gibt pro Zeile eine Datei aus
ls -l	langes Ausgabeformat (mehr Informationen zu den Dateien)
ls -F	Zeige Filetyp mit an (Filetyp ist am letzten Zeichen zu erkennen, z.B. Verzeichnisse: “/“, ausführbare Dateien: “*“, Pipes: “|“)
ls -R	Rekursive Ausgabe der Unterverzeichnisse

Dateitypen

Dateityp	Beschreibung	Erzeugung
Verzeichnis	vgl. Ordner oder Schubfach, enthält weitere Dateien oder Verzeichnisse	mkdir
normale Dateien	Dateien im Dateisystem, die beliebige Daten beinhalten	touch
ausführbare Dateien	sind normale Dateien, deren Rechte auf ausführbar gesetzt sind - z.B. Programme, Scripte ...	chmod
symbolische Links (symlinks)	sind Verknüpfungen auf andere Dateien	ln -s
feste Links (hardlinks)	echte Duplizierung ein und derselban Datei an zwei verschiedene Stellen/Namen	ln
fifo (named pipes)	First In First Out - virtuelle Dateien um Programmausgaben/-eingaben zu verknüpfen	mkfifo
character Devices	im /dev - Symbolisch für Systemkomponenten (z.B. /dev/console)	mknod c
block Devices	im /dev - Symbolisch für Systemkomponenten (z.B. /dev/hda)	mknod b

Berechtigungen
Jede Datei ist einem Besitzer und einer Gruppe zugeordnet. Für den Besitzer, die Gruppe und alle anderen Nutzer kann man Berechtigungen festlegen. Dies ist alles in der Ausgabe von “ls -l“ ersichtlich. Es gibt drei verschiedene Rechte, wobei jedem eine Zahl zugeordnet ist:

r read - Leserecht (4)

w write - Schreibrecht (2)

x execute - Ausführungsrecht (1)

In der Ausgabe von “ls -l“ werden die Recht der Reihe nach für Besitzer, Gruppe und andere Nutzer angegeben.
Man kann die Rechte mittels des Kommandos chmod ändern. Dabei werden sie als Zahl interpretiert, die sich aus dem Summierung der einzelnen Berechtigungen ergibt. Hierbei werden wieder 3 Ziffern aneinandergehangen. z.B. “chmod 750 test“ erzeugt dir Rechte rwxr-x— (Besitzer: lesen, schreiben, ausführen; Gruppe: lesen, ausführen; andere Nutzer: keine) für die Datei test. Verzeichnisse können nur betreten werden, wenn sie als ausführbar markiert sind!
Erweiterter Syntax von chmod:

chmod <rechte> <Datei>	setzt die Rechte (als Zahl für die Datei)
chmod u+r <Datei>	setzt das Leserecht des Besitzers auf die Datei
chmod u-r <Datei>	entzieht dem Besitzer das Leserecht auf die Datei
chmod <usergruppe>[+|-|=]<Berechtigung> <Datei>	allgemeine Form

verwendbare Usergruppen:

u user - Besitzer der Datei

g group - Gruppe der Datei

o others - andere Nutzer

a all - alle drei Gruppen

verwendbare Berechtigungen:

r read - lesen

w write - schreiben

x execute - auführen

Änderungen des Besitzers sind dem Administrator (meist root) vorbehalten. Dieser kann den Besitzer einer Datei mittels “chown <user>:<gruppe> <datei>“ den Besitzer und die Gruppe der Datei ändern.
Jeder Benutzer darf seine Dateien einer Gruppe, der er selbst angehört übereignen. Dies geschieht mit “chgrp <gruppe> <datei>“.

Kommandos zum Verwalten von Verzeichnissen und Dateien

Kommando	Beschreibung
mkdir [Verzeichnisname]	legt ein neues Verzeichnis mit dem angegebenen Namen an
rmdir [Verzeichnisname]	löscht das angegebene Verzeichnis (muss leer sein)
du -sh [Verzeichnisname]	gibt die Größe des angegebenen Verzeichnisses an
touch [Dateiname]	legt die angegebene Datei leer an
file [Dateiname]	versucht zu ermitteln was die angegebene Datei für Daten enthält
rm [Dateiname]	löscht Datei
rm -r [Verzeichnis]	löscht Verzeichnis rekursiv (Vorsicht!)
find	gibt alle Dateien in allen tieferen Verzeichnissen aus
find -name [Name]	wie find, gibt jedoch nur Dateien mit dem angegebenen Namen aus
mc	ein Norton Commander Clone
head [Datei]	gibt die ersten Zeilen der Datei aus
tail [Datei]	gibt die letzten Zeilen der Datei aus
cp [Quelle] [Ziel]	Kopiert die Datei von der Quelle ans Ziel
ln -s [Quelle] [Ziel]	legt einen Link vom Ziel auf die Quelle
mv [Quelle] [Ziel]	verschiebt die Datei von der Quelle ans Ziel
gzip [Datei]	komprimiert die Datei
gzip -d [Datei]	dekomprimiert die Datei

1.7 der Editor vi

vi ist ein weit verbreiteter Unix-Editor und somit auf den meisten Unix Systemen verfügbar. Mittlerweile gibt es eine extrem verbesserte Version namens vim, die jedoch nur auf wenigen neueren Systemen zu finden ist.
Syntax: “vi [datei]“ - öffnet die Datei

im vi gibt es zwei Modi, den Befahlsmodus und den Einfügemodus. Im Befehlsmodus kann man sich im Text bewegen, löschen, und andere Formatierungen vornehmen. Im Einfügemodus kann man Texte eingeben.

1.7.1 Bewegungen

l oder Pfeil rechts	Cursor nach rechts bewegen
h oder Pfeil links	Cursor nach links bewegen
k oder Pfeil hoch	Cursor nach oben
j oder Pfeil runter	Cursor nach unten

1.7.2 vi beenden

:q!	beenden ohne speichern
:wq oder :x	beeden mit speichern

1.7.3 Textbearbeitung - Löschen

x	löscht das Zeichen unter Cursor

1.7.4 Textbearbeitung - Einfügen

i	aktiviert den Einfügemodus (jetzt kann man den Text eingeben)
r	aktiviert den Ersetzung-Modus (Text unter Cursor wird überschrieben)
ESC	kehrt zum Befehlsmodus zurück

1.7.5 Löschoperationen

dw	löscht das Wort rechts vom Cursor
d$	löscht vom Cursor bis zum Zeilenende
dd	löscht die aktuelle Zeile

Alle Kommandos können durch eine vorangestellte Zahl öfters wiederholt werden (z.B. “3dw“ löscht die nächsten drei Wörter)

1.7.6 Wiederherstellen/Wiederholung

u	macht die letzte Operation rückgängig (nur vim)
Strg-R	wiederholt die letzte zurückgenommene Operation wieder (nur vim)
.	wiederholt die letzte Aktion

1.7.7 Ausschneiden, Kopieren und Einfügen

Nach jeder Löschoperation ist der gelöschte Text in einem Puffer und kann mittels dem kommando “p“ eingefügt werden. Wenn z.B. eine Zeile Text nur kopiert nicht gelöscht werden soll kann man anstatt “dd“ “yy“ verwenden. Danach ebenfalls wieder mit “p“ einfügen.

1.7.8 Suchen

Man kann Text mittels "/<suchwort>" im vi suchen (z.B. "/hilfe"). Man kann die Unterscheidung von Gross und Kleinschreibung mittels ":set ignorecase" aus und mit ":set noignorecase" wieder einschalten.

1.7.9 Suchen und Ersetzen

Mittels "s/<wort1>/<wort2>/g" kann man alle auf der aktuellen Zeile vorkommenden <wort1>-Strings durch <wort2> ersetzen. Wenn man ein "%" voranstellt gilt die Ersetzung für das gesamte Dokument.

1.7.10 erweitertes Speichern/Öffnen

Mit ":w <datei>" kann man die aktuelle Datei unter einem anderen Namen speichern. Mit ":e <datei>" kann man eine neue Datei öffnen.

1.8 montierte Laufwerke im Filesystem

Im Unix existieren keine Laufwerksbuchstaben wie im DOS, sondern die Massenspeicher werden ins virtuelle Dateisystem montiert (gemountet). Die Massenspeicher werden durch Einträge in /dev symbolisiert.

- /dev/hda = erste Festplatte/CD-ROM
- /dev/hdb = zweite Festplatte/CD-ROM ...
- /dev/hda1 = erste Partition auf erster Festplatte
- /dev/hda2 = zweite Partition auf zweiter Festplatte ...
- /dev/fd0 = erstes Diskettenlaufwerk
- /dev/fd1 = zweites Diskettenlaufwerk ...

Diese Laufwerke können an bestimmte vordefinierte Stellen (mount-Points) im Dateisystem montiert werden (ist in /etc/fstab festgelegt). Neue mount-Points darf nur der Administrator anlegen. Manche Laufwerke kann auch ein normaler User mounten (z.B. CD-ROM, Floppy), diese sind in der /etc/fstab durch das Attribut "user" gekennzeichnet.
Vorhandene montierte Laufwerke können mit dem Befehl "mount" angezeigt werden. Mittels "mount <Mountpoint>" oder "mount <Device>" kann man neue Laufwerke mounten (z.B. "mount /dev/fd0").
weitere Infos: es gibt auch Netzlaufwerke, die von anderen Rechnern aus gemountet werden, sie tauchen auch in der mount-Liste auf (z.B. perikles:/mnt/hdb on /mnt/perikles type nfs (rw)).

1.9 Sicherung von Daten

1.9.1 das tar Kommando

Unix "tar" ist vergleichbar mit dem DOS "backup". Es dient dazu mehrere Dateien zu einer zusammenzufassen. Die Funktionsweise soll an praktischen Beispielen erläutert werden.
Erzeugen eines neuen Archives archiv.tar mit dem Inhalt test (Verzeichnis) und file (Datei):
(1) "tar cf archiv.tar test file"
Anzeigen des Inhaltes von archiv.tar
(2) "tar tf archiv.tar"
Auspacken des Archives archiv.tar
(3) "tar xf archiv.tar"

Komprimieren von Archiv.tar (4) “gzip archiv.tar“ (es entsteht ein file archiv.tar.gz)
Dekomprimieren eines Archives
(5) “gzip -d archiv.tar.gz“ (es entsteht ein file archiv.tar)
Erzeugen eines komprimierten Archives archiv.tar.gz (1) und (4)
(6) “tar czf archiv.tar.gz test file“ (geht nicht auf jedem System)
Entpacken eines komprimierten Archives (5) und (3)
(7) “tar xzf archiv.tar.gz“ (geht nicht auf jedem System)
Entpacken einer speziellen Datei aus einem archiv
(8) “tar xf archiv.tar [datei]“

man kann auch direkt auf ein Laufwerk “tarren“, z.B. eine Diskette: “tar czf /dev/fd0 datei“ schreibt direkt auf eine Diskette, entpacken kann man es dann mit “tar xzf /dev/fd0“.

1.9.2 mtools, Zugriff auf ein msdos-Filesystem (Disketten)

Mit dem mtools kann man den Dateiaustauch zwischen Windows und Unix mittels Disketten realisieren. Mit “mformat a:“ formatiert man eine Diskette für das FAT Dateisystem (Achtung: alle alten Daten werden gelöscht!). Mit “mcopy <datei> a:“ kopiert man eine Datei auf die Diskette. Mit “mdir a:“ kann man den Inhalt anzeigen.
Übersicht über mtools:

Kommando	Funktion
mbadblocks a:	prüft eine Diskette und markiert die fehlerhaften Blöcke
mcat a: > <datei>	kopiert ein Diskettenimage in <datei>
mmd <verzeichnis>	legt ein neues Verzeichnis auf der Diskette an
mcd <verzeichnis>	wechselt das Verzeichnis auf der Diskette
mdir	zeigt den Inhalt des aktuellen Verzeichnisses der Diskette an
mcopy <datei> a:	kopiert <datei> auf die Diskette
mdel <datei>	löscht <datei> auf Diskette
mdeltree <verzeichnis>	löscht <verzeichnis> auf Diskette rekursiv
mrd <verzeichnis>	löscht <verzeichnis> auf Diskette (muss leer sein)
mren <quelle> <ziel>	benennt <quelle> in <ziel> um

1.9.3 raw copy mit dd

mit dem Kommando “dd <quelle> <ziel>“ kann man Rohdaten kopieren. So kann man z.B. ein erzeugtes tar-archiv auf Diskette kopieren, um es später mit “tar xzf /dev/fd0“ auszupacken. Dazu einfach “dd archiv.tar.gz /dev/fd0“ verwenden.

1.10 Erweiterte Bash-Funktionen (Scripting)

1.10.1 Bedingungen

Mittels Bedingungen kann man entscheiden, eine Aktion auszuführen oder nicht.
Wird mittels eines if then else Konstruktes realisiert.
Der Syntax lautet:

```
if [bedingung];
then
  <code>
else
  <code>
fi ;
```

z.B.

```
A="bla"
B="blub"

if [ "$A" = "$B" ];
then
   echo "$A = $B"
else
   echo "$A != $B"
fi;
```

1.10.2 Schleifen

for-Schleifen
Die for-Schleife unterscheidet sich stark von anderen Programmiersprachen, sie entspricht eher der foreach Schleife. Mit ihr kann man eine Reihe von Wörtern durchgehen. Syntax:

```
for i in a b c d e;
do
   echo $i;
done;
```

ebenso kann man wieder den Output eines Programmes nehmen, dabei gelten Leerzeichen und Zeilenumbrüche als „Trennzeichen".
z.B. alle Dateien in einem Verzeichnis ausgeben.

```
for i in $(ls -1);
do
   echo $i;
done;
```

oder for wie in C:

```
for i in $(seq 1 10);
do
   echo $i;
done;
```

while-Schleifen
Schleife wiederholen solange eine Bedingung erfüllt ist.
z.B.

```
i=0;
while [ $i -lt 10 ];
do
   echo $i;
   let i+=1;
done;
```

until-Schleifen
Schleife abbrechen, wenn bedingung erfüllt.
z.B.

```
i=0;
until [ $i -gt 10 ];
do
    echo $i;
    let i+=1;
done;
```

1.10.3 Funktionen

Funktionen wie aus anderen Programmiersprachen bekannt.

```
function hello()
{
    echo "hello";
}

# Funktion mit Parametern vgl. Perl
function say()
{
    echo $1;
}

hello;
say "test";
```

1.10.4 einfache Userinterfaces

Mittels des select-Befehls.

```
OPT="Hello Quit"
select opt in $OPT;
do
    if [ "$opt" = "Quit" ];
    then
        echo "quit"
    elif    [ "$opt" = "Hello" ];
    then
        echo "hello"
    else
        echo "bad option"
    fi
done
```

1.10.5 Sinnvolle Tools

sed Unix Stream Editor - zum Bearbeiten von Zeilen eines Streams

awk Änderungen an Dateien / Streams

tput Gibt Terminal-Infos aus

cat liest eine Datei und gibt sie nach stdout

tac das gleiche wie cat, nur von hinten beginnend

less Anzeige eines Files Streams Seitenweise

more siehe less

grep zeigt nur die Zeilen, die einem Muster entsprechen

sort Sortiert Zeilen eines Files

wc zählt Zeichen, Wörter und Zeilen einer Datei / eines Streams

... und viele mehr.

1.10.6 bash debugging

Falls mal etwas zu debuggen ist, einfach im Header

#!/bin/bash −x

eintragen.