Arbeiten mit Unix

Alois Schütte

13. April 2015

Diese Ausarbeitung ist als Vorbereitung zum ersten Praktikumstermin von allen Teilnehmern durchzuarbeiten, damit das erste Praktikum im Labor testiert werden kann.

Die Anleitung beschränkt sich auf eine praktische Sicht, die hinter den gezeigten Kommandos sich verbergenden Konzepte werden in der Vorlesung behandelt.

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACULTY OF COMPUTER SCIENCE

Inhaltsverzeichnis

1 Einloggen

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

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

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

Man erkennt die erfolgreiche Anmeldung daran, dass die Eingabeaufforderung des Systems, der sog. Prompt erscheint:

```
$
```

Die Benutzernamen und Initialpassworte werden im ersten Praktikum bekannt gegeben.

2 Das Unix-Dateienkonzept

Unter Unix (wie auch unter DOS) liegen die Daten der Benutzer nicht einfach alle ungeordnet auf einem Datenträger, sondern sie sind in Dateien und Verzeichnisse organisiert. Eine Datei ist ein Bereich auf einem Datenträger, der logisch zusammenhängende Daten enthält und der mit einem Namen versehen ist. Dateinamen dürfen bis zu 256 Zeichen lang sein und aus Ziffern, Buchstaben, Bindestrichen, Unterstrichen und Punkten bestehen. Im Gegensatz zu DOS ist mehr als ein Punkt in einem Dateinamen erlaubt. Auch andere Sonderzeichen sind zulässig, aber man sollte damit vorsichtig umgehen, weil viele der übrigen Zeichen eine Sonderbedeutung innerhalb des Systems haben und daher zu Problemen führen können, wenn sie ohne besondere Vorkehrungen als Teile von Dateinamen verwendet werden.

Verzeichnisse sind Ordnungsstrukturen, quasi Behälter für Dateien. Man kann sich Verzeichnisse als Ordner vorstellen, in welche die Dateien eingeordnet sind. Diese Ordner können ineinander geschachtelt sein, d.h. ein Ordner 'Briefe' kann zum Beispiel zwei weitere Ordner 'Liebesbriefe' und 'Mahnungen' enthalten. In jedem dieser Ordner können wiederum weitere Ordner oder aber einzelne Dokumente (Briefe) enthalten sein. Verzeichnisnamen können, wie Dateinamen auch, frei gewählt und vergeben werden. Es gelten dabei dieselben Benennungsregeln wie für Dateinamen. Schematisch stellt man Verzeichnisstrukturen üblicherweise als 'Bäume' dar.

Da Unix ein Multiuser-System ist, muss der Zugriff auf einzelne Dateien und Verzeichnisse vom System so geregelt werden, dass kein Benutzer die Daten der anderen Benutzer manipulieren oder vertrauliche Daten unbefugterweise einsehen kann. Das wird dadurch gewährleistet, dass jede Datei und jedes Verzeichnis einen Besitzer (Owner) hat, und dieser Besitzer mit Hilfe bestimmter Befehle festlegen kann, welcher der anderen Benutzer auf welche Weise auf seine Dateien zugreifen darf. Die meisten Dateien und Verzeichnisse in einem Unix-System gehören naturgemäß dem Systemverwalter (der den Usernamen 'root' trägt) und jeder Benutzer kann auf diese Dateien lesend zugreifen, sie aber nicht verändern (zu dieser Art von Dateien gehören zum Beispiel alle Anwendungsprogramme, die das System zur Verfügung stellt). Darüber hinaus bekommt jedoch jeder Benutzer einen privaten Bereich auf der Festplatte zugeteilt, den er beliebig beschreiben

und verändern kann. Dieses Verzeichnis, das dem einzelnen Benutzer gehört, wird das HOME-Verzeichnis des entsprechenden Benutzers genannt und unterhalb dieses HOME-Verzeichnisses kann der einzelne Benutzer seine eigene Verzeichnisstruktur anlegen und pflegen.

Nach dem Einloggen befindet man sich automatisch im eigenen HOME-Verzeichnis. Man kann das aktuelle Verzeichnis jederzeit mit einem Unix-Befehl 'cd' wechseln und damit Dateien in anderen Verzeichnissen bearbeiten. Das aktuelle Verzeichnis lässt sich (ganz analog) jederzeit wechseln, aber ein Verzeichnis (und nur eines) ist zu einem bestimmten Zeitpunkt immer das aktuelle und alle unsere Kommandos beziehen sich dann auf dieses eine, aktuelle Verzeichnis.

2.1 Verzeichnisinhalte anzeigen: ls

Der Befehl 'ls' (list) listet den Inhalt des aktuellen Verzeichnisses auf:

```
$ ls
Vorlesungen bin tmp
$
```

Die Ausgabe, die hier zu sehen ist, zeigt verschiedene Dateien in einem Verzeichnis. Die Namen der Dateien sind frei gewählt und haben hier keine besondere Bedeutung. Jeder Eintrag ist ein Dateiname. 'ls' entspricht dem 'DIR'-Befehl unter DOS, genauer: dem Befehl 'DIR /w'.

Mehr Informationen bekommt man, wenn man 'ls' mit den Optionen '-l' und '-a' aufruft. Optionen sind Angaben, die das Verhalten eines Kommandos beeinflussen. Sie werden immer durch einen Bindestrich eingeleitet (unter DOS verwendet man dafür den Schrägstrich: '/'). Die Option '-l' erzwingt die 'lange' Ausgabe, in der mehr Informationen zu jeder einzelnen Datei angezeigt werden, und die Option '-a' listet jede Datei auf. Normalerweise werden nämlich nur solche Dateien angezeigt, deren Namen nicht mit einem Punkt beginnt. Dateien, deren Name mit einem Punkt beginnt, werden als 'versteckte' Dateien angesehen. Ein Punkt als erstes Zeichen eines Dateinamens entspricht dem 'hidden'-Attribut unter DOS.

```
$ ls -al
total 2272
drwx------@   5 as    staff      5270 31 Jan 11:29 .
drwxr-xr-x   10 root  admin       340 28 Okt 11:26 ..
-rw-------    1 as    staff     95541  1 Feb 18:49 .bash_history
-rw-------    1 as    staff        26  1 Feb  2005 .bash_logout
-rwxr--r--    1 as    staff      1306 11 Sep 11:35 .bash_profile
-rw-------    1 as    staff       975 17 Jul  2013 .bashrc
drwx------    9 as    staff       306 12 Nov  2007 src
drwx------   11 as    staff       374 30 Jan 15:36 tmp
drwxr-xr-x    4 as    staff       136 26 Sep 18:34 var
$
```

Betrachten wir diese Ausgabe von rechts nach links. Ganz rechts in jeder Zeile steht der Dateiname, jede Zeile ist der Eintrag für eine Datei. Links davon steht in drei Feldern Datum und Uhrzeit der letzten Modifikation der Datei. Im fünften Feld (links vom Monat) steht die Größe der Datei in Bytes. Die weiteren zwei Felder links von der Dateigröße geben den Besitzer der Datei und dessen Gruppe an. Alle Dateien in diesem Verzeichnis gehören dem Benutzer 'as', der Mitglied der Gruppe 'staff' ist. Und schließlich stehen ganz links (in der ersten Spalte) die Zugriffsrechte, die ich als Besitzer für die einzelnen Dateien vergeben habe. Das erste Zeichen dieser ersten Spalte zeigt an, ob es sich beim entsprechenden Eintrag um eine normale Datei ('-'), um ein Verzeichnis ('d') oder um einen Link ('l') handelt. Eine besondere Erwähnung verdienen die beiden Einträge '.' und '..'. Das sind Stellvertreter für Verzeichnisnamen, die man statt der realen Namen (abkürzend) benutzen kann. Und zwar bezeichnet '.' das jeweils aktuelle Verzeichnis, und '..' das dem aktuellen Verzeichnis übergeordnete Verzeichnis, also denjenigen 'Ordner', in welchem der aktuelle 'Ordner' enthalten ist. Man kann dem Befehl 'ls' auch einen Parameter übergeben, d.h. man kann ihm sagen, auf welche Verzeichnisse man ihn anwenden möchte.

2.2 Verzeichnisse wechseln: cd und pwd

Das aktuelle Verzeichnis kann mit dem Kommando 'cd' (change directory) gewechselt werden. Bevor wir jedoch das Kommando 'cd' erfolgreich einsetzen können, müssen wir uns noch kurz darüber unterhalten, wie Verzeichnisnamen unter Unix gebildet werden.

2.2.1 Absolute und relative Pfadangaben

Jede Datei (und jedes Verzeichnis) in einem Unix-System hat einen eindeutigen, vollen Namen, der sie von jeder anderen Datei auf dem System unterscheidet. Dieser absolute Pfadname wird gebildet, indem man vom obersten Verzeichnis ausgeht, dem Wurzelverzeichnis, und dann den 'Pfad' durch die Verzeichnisstruktur bis hin zur anzusprechenden Datei angibt. Das Wurzeverzeichnis hat immer den Namen '/' (Slash), und derselbe Schrägstrich dient auch als Trennzeichen zwischen den einzelnen Verzeichnisnamen eines Pfades. Absolute Pfadnamen verhindern Verwechslungen bei Dateien, die den gleichen Dateinamen haben.

2.2.2 Das aktuelle Verzeichnis

Ein Verzeichnis ist immer das aktuelle Verzeichnis (oder Arbeitsverzeichnis). Das Arbeitsverzeichnis ist deshalb von Bedeutung, weil alle Pfadangaben, die nicht mit einem Slash ('/') beginnen, als Angaben relativ zu diesem Verzeichnis interpretiert werden. Relative Pfadangaben gehen also immer vom aktuellen Verzeichnis aus.

2.2.3 Das HOME-Verzeichnis

Das sogenannte HOME-Verzeichnis ist das Verzeichnis, das vom Systemverwalter dem einzelnen Benutzer zugeteilt wird. Jeder Benutzer hat ein eigenes HOME-Verzeichnis und kann darin nach Belieben Dateien anlegen, kopieren, löschen, weitere Unterverzeichnisse anlegen. In der Regel kann ein Benutzer die meisten Verzeichnisse des Systems einsehen (d.h. 'lesend' darauf zugreifen), verändern jedoch kann er nur Daten, die sich in seinem HOME-Verzeichnis oder den darunterliegenden (von ihm selber angelegten) Verzeichnissen befinden. Nach dem Einloggen befindet man sich automatisch im eigenen HOME-Verzeichnis. Um jederzeit wieder in das HOME-Verzeichnis zurückzukehren genügt es, einfach das Kommando 'cd' ohne jeden Parameter aufzurufen. Das HOME-Verzeichnis kann auch mit einem speziellen Namen angesprochen werden: '~' (Tilde). Diese Tilde kann überall eingesetzt werden, wo wir sonst den Namen des eigenen HOME- Verzeichnisses schreiben würden.

2.3 Verzeichnisse anlegen: mkdir

Ein neues Verzeichnis kann man mit Hilfe des Befehls 'mkdir' (make directory) anlegen. Im folgenden Beispiel wird ein Verzeichnis mit dem Namen 'Verzeichnis' unterhalb des Verzeichnisses '~/tmp' angelegt. Anschließend wird mit 'cd' in das neu angelegte Verzeichnis gewechselt:

```
$ pwd
/home/as/tmp
$ mkdir Verzeichnis
$ cd Verzeichnis
$ pwd
/home/as/tmp/Verzeichnis
$
```

2.4 Verzeichnisse löschen: rmdir

Mit 'rmdir' (remove directory) kann man ein leeres Verzeichnis (eines, das keine Dateien oder Unterverzeichnisse enthält) entfernen. Das zu entfernende Verzeichnis darf nicht das aktuelle Verzeichnis sein, d.h. vor dem entfernen eines Verzeichnisses muß man uns mit 'cd' in ein anderes (normalerweise das darüberliegende) Verzeichnis bewegen:

```
$
$ ls -l
total 0
drwx------  2 as  staff  68  3 Feb 19:05 Verzeichnis
-rw-------  1 as  staff   0  3 Feb 19:04 m.cpp
$ rmdir Verzeichnis
$ ls -l
total 0
-rw-------  1 as  501  0  3 Feb 19:04 m.cpp
$
```

3 Benutzerrechte

3.1 Passwort ändern: passwd

Man kann als Benutzer sein eigenes Passwort ändern. Bitte seien Sie besonders sorgfältig, wenn Sie mit diesem Befehl umgehen. Falls Sie Ihr Passwort ändern, aber sich beim ändern vertippen oder sich nicht mehr an das neue Passwort erinnern können, dann muss der Systemverwalter Ihnen ein neues Passwort zuteilen. Das ist mit zusätzlicher (und unnötiger) Arbeit für den Laboringenieur verbunden und Sie machen sich dadurch nicht besonders beliebt.

Ein Passwort sollte, um sicher zu sein, mindestens eine Ziffer und ein nicht-alphanumerisches Zeichen (zum Beispiel einen Punkt oder Bindestrich) enthalten. Nehmen Sie auf keinen Fall deutsche oder englische Wörter oder Namen von Leuten als Passwörter. Solche Passwörter lassen sich durch spezielle Programme, die frei verfügbar sind, in sehr kurzer Zeit herausfinden.

Sie ändern Ihr Passwort wie im folgenden Beispiel dargestellt. Das eingegebene Passwort ist während der Eingabe nicht sichtbar. Die Meldungen können geringfügig von den gezeigten abweichen:

```
$ passwd
Old Password:
New password:
Re-enter new password:
Password changed
$
```

Es kann eine gewisse Zeit dauen, bis ein geändertes Passwort auf allen Maschinen im Netz verfügbar ist. Warten Sie also ein paar Minuten, bevor Sie sich mit dem neuen Passwort auf einer anderen Maschine einzuloggen versuchen. Wenn die Änderung aus irgendeinem Grund fehlschlägt, dann werden Sie in einer Meldung (statt 'Passwort changed') darüber informiert. Es gilt dann weiterhin Ihr altes Passwort. Beobachten Sie also bitte genau die Meldungen des Systems, um später zu wissen, welches Passwort nun das gültige ist.

3.2 Zugriffsrechte ändern: chmod

Mit dem Befehl 'chmod' (chande mode) können Sie festlegen, wer auf welche Weise auf Ihre Dateien und Verzeichnisse zugreifen können soll. Jede Datei und jedes Verzeichnis ist unter Unix mit einer Reihe von Zugriffsrechten versehen. Diese Rechte können Sie mit dem Befehl 'ls' einsehen. 'ls -l' gibt für jede Datei eine Zeichenkette wie die folgende aus:

```
$ ls -l
-rwxr-xr-x 1 as staff 85 Feb 4 2000 t.ksh
$
```

Uns interessiert jetzt nur die kryptische Zeichenfolge ganz links. Sie besteht immer aus zehn Zeichen, davon steht das erste allein und die folgenden neun sind als drei Dreiergruppen aufzufassen:

```
- rwx r-x r-x
```

- Das erste, allein stehende Zeichen gibt den Typ der Datei an. Ein Bindestrich zeigt eine normale Datei an, 'd' steht für ein Verzeichnis (directory), 'l' für einen Link.
- Die drei Dreiergruppen bestehen aus den Zeichen 'r' für 'Lesezugriff erlaubt', 'w' für 'Schreibzugriff erlaubt' und 'x' für 'Ausführung erlaubt', immer in dieser Reihenfolge. Wenn eine dieser Zugriffsarten nicht erlaubt ist, dann steht ein Bindestrich statt des entsprechenden Buchstabens.
- Die erste Dreiergruppe gibt dabei die Rechte des Besitzers der Datei an (wer der Besitzer ist steht zwei Felder weiter rechts in der 'ls'-Ausgabe), die zweite Dreiergruppe gibt die Rechte der Benutzergruppe des Besitzers an, und die letzte die Rechte aller anderen Benutzer. Die Datei ãt.kshÒ im obigen Beispiel ist also für alle Benutzer lesund ausführbar, aber nur für den Besitzer der Datei (den Benutzer 'as') beschreibbar.

Noch ein Beispiel: Die Zugriffsrechte

```
-rwxr----
```

würden bedeuten, dass die entsprechende Datei für ihren Besitzer lesbar, beschreibbar und ausführbar ist, für die Gruppe, der der Besitzer angehört, nur lesbar und alle anderen Benutzer könnten diese Datei weder lesen, noch beschreiben, noch ausführen.

Etwas anders ist die Bedeutung mancher Zugriffsrechte, wenn sie sich auf Verzeichnisse (statt auf Dateien) beziehen. Ein Verzeichnis 'beschreiben' bedeutet, die darin enthaltenen Dateien löschen zu können. Ein Verzeichnis 'ausführen' - das Zugriffsrecht "x"bedeutet, dass man mit "cdïn das Verzeichnis wechseln kann. Und schließlich gibt es für Verzeichnisse noch ein Zugriffsrecht "t". Das bewirkt, wenn es gesetzt ist, daß zwar jeder das Verzeichnis beschreiben (also neue Dateien darin anlegen kann), dass aber jeder nur seine eigenen Dateien aus diesem Verzeichnis wieder löschen kann. Diese Einstellung wird für öffentlich beschreibbare Verzeichnisse, wie zum Beispiel '/tmp', benutzt. So können zwar alle Benutzer ins selbe Verzeichnis schreiben, aber sie können sich nicht gegenseitig Dateien weglöschen. Für die eigenen Dateien und Verzeichnisse kann man die Zugriffsrechte selber setzen. Dazu benutzt man den Befehl 'chmod':

```
chmod wer[+-=]recht Datei(en)
```

- *wer* ist hierbei einer der Buchstaben u, g, o, a.
  Die Buchstaben bedeuten:
  - **u** der Besitzer selbst
  - **g** die Gruppe des Besitzers
  - **o** alle anderen Benutzer
  - **a** alle Benutzer, incl. Besitzer und Gruppe
- */+-=/* bedeutet, daß an dieser Stelle eines der Zeichen +, - oder = eingesetzt werden muß. Die Zeichen bedeuten:
  - **+** die genannten Zugriffsrechte werden zu den bestehenden hinzugefügt

- die genannten Zugriffsrechte werden von den bestehenden abgezogen

= die bestehenden Zugriffsrechte werden durch die genannten ersetzt

- *recht* ist eines der Zeichen r, w, x, t. Sie bedeuten:

r die genannten Benutzer haben das Leserecht für die genannten Dateien

w die genannten Benutzer haben das Schreibrecht für die genannten Dateien

x die genannten Benutzer dürfen die genannten Dateien (Programme) ausführen

t das genannte Verzeichnis soll nur dem Besitzer einer Datei das Löschen erlauben

So bewirkt z.B.

```
$ ls -l m.cpp
-rw-rw-rw-  1 as   staff      0B  3 Feb 19:04 m.cpp
$ chmod g-w,o-rw m.cpp
$ ls -l m.cpp
-rw-r-----  1 as   501      0B  3 Feb 19:04 m.cpp
$
```

dass jetzt nur der Besitzer lesen und schreiben darf und die Gruppe nur lesen, ansonsten hat niemand Rechte.

3.3 Benutzer auflisten: who

Mit dem Kommando "who"kann man sehen, wer gerade im System eingeloggt ist:

```
$ who
as       console  Feb  3 09:38
as       ttys000  Feb  4 11:15
root     ttys001  Feb  4 11:23
$
```

4 Dateien verwalten

4.1 Dateityp feststellen: file

Mit dem Kommando 'file' kann man den Typ einer Datei ermitteln. Das System gibt eine Schätzung darüber aus, was die genannte Datei enthalten könnte. Diese Angabe ist nicht immer zutreffend. Im Beispiel ist als Dateinamen ein Stern (*) angegeben, dadurch ermittelt 'file' den Typ aller Dateien im aktuellen Verzeichnis.

```
$ file *
Hoersaaluebung:   directory
access:           Mach-O 64-bit executable x86_64
access.c:         ASCII c program text
dat:              ASCII text
file.hole:        data
fork:             Mach-O 64-bit executable x86_64
fork.c:           ASCII c program text
$
```

4.2 Dateien kopieren: cp

Das Kommando 'cp' unter Unix entspricht etwa dem 'COPY /B'-Befehl unter DOS: es kopiert eine Datei. Die Wildcards ('*' und '?') funktionieren jedoch anders als unter DOS und sollten

deshalb mit gro§er Vorsicht eingesetzt werden. 'cp' bekommt zwei Parameter: der erste ist der Name der zu kopierenden Datei und der zweite ist der Name der Kopie, die erstellt werden soll (oder der Name eines Verzeichnisses, in welchem die Kopie dann stehen soll).

4.3 Dateien umbenennen: mv

Das Kommando 'mv' entspricht dem 'RENAME' unter DOS, kann jedoch, über das bloße Umbenennen hinaus, Dateien auch an einen anderen Ort im Verzeichnisbaum verschieben. Es bekommt zwei Parameter: erstens den Namen der umzubenennenden (oder zu verschiebenden) Datei; und zweitens den neuen Namen (oder das Zielverzeichnis).

4.4 Dateien löschen: rm

'rm' (remove) wird zum Löschen von Dateien verwendet. Dieser Befehl ist recht gefährlich, weil er vor dem Löschen nicht nachfragt, ob auch wirklich gelöscht werden soll, und weil es darüber hinaus unter Unix keine Möglichkeit gibt, gelöschte Dateien wiederherzustellen. 'rm' bekommt eine beliebige Anzahl von Parametern, das sind die Namen der zu löschenden Dateien. Wenn der aufrufende Benutzer nicht über die entsprechenden Rechte verfügt, um eine Datei löschen zu können, wird eine Meldung angezeigt.

Das Kommando 'rm' kann auch mit der Option '-r' gestartet werden. Dann löscht es nicht nur die Dateien im angegebenen Verzeichnis, sondern 'rekursiv' auch alle darunter liegenden Dateien und Verzeichnisse. Diese Option ist demnach mit großer Zurückhaltung einzusetzen.

4.5 Links herstellen: ln

Ein Link ist ein Unix-typisches Konstrukt, zu dem es unter DOS keine Parallele gibt, in Windows ist ein link in etwa mit einem Alias vergleichbar. Normalerweise besteht eine Datei ja aus einem Namen und den Daten, die sie enthält. Ein Link ist nun ein purer Dateiname ohne die dazugehörigen Daten. Der Name (der Link) verweist dabei auf eine andere Datei, die die Daten enthält, die dem Dateinamen des Links zugeordnet sind.

Ähnlich wie die Verweise auf Abbildungen in einem Text, helfen Links zum einen Platz zu sparen: denn die (möglicherweise großen) Datenbestände befinden sich nur einmal im Dateisystem und die Links selber bestehen ja nur aus einem Dateinamen, der wenig Platz beansprucht. Zum anderen erleichtern Links die Pflege von Datenbeständen. Wenn eine änderung im Dateiinhalt (oder einer Abbildung) fällig wird, dann muß die änderung nur einmal gemacht werden, und alle Referenzen auf diese Datei (oder Abbildung) profitieren automatisch von der änderung. Hätte man den Datenbestand (die Abbildung) in mehreren Kopien vorliegen, dann müßte man eben mehrmals dieselbe änderung vornehmen (einmal für jede Kopie). Die tatsächlichen Anwendungsmöglichkeiten für Links werden sich für Sie natürlich erst in der Praxis ergeben.

Es gibt zwei Arten von Links: sogenannte symbolische und harte Links. Symbolische Links erzeugen Sie mit dem Kommando 'ln' und der Option '-s'. Der erste Parameter von 'ln' ist der Name der ursprünglichen Datei, auf die der Link verweisen soll, und der zweite Parameter ist der Name des Links (der Referenz). Also:

```
$ ln -s echte_datei linkname
$
```

Wird als *linkname* nur ein Verzeichnis angegeben (und kein konkreter Dateiname), dann entsteht im angegebenen Verzeichnis ein Link, der denselben Namen wie die ursprüngliche Datei trägt, auf die er verweist. Ein Link sieht in der ls-Ausgabe so aus:

```
lrwxrwxrwx 1 root root 9 Jul 23 1994 /home -> /MNT/home
```

In diesem Fall sehen wir ein Pseudoverzeichnis (einen Link) mit dem Namen /home, das auf ein (real existierendes) Verzeichnis /MNT/home verweist. Beachten Sie, daß ein Link immer alle Zugriffs- rechte gesetzt hat. Das bedeutet jedoch nur, daß er die Zugriffsrechte 'weiterleitet' an die Datei, die er referenziert. D.h. trotz des Aussehens der Zugriffsrechte ist es nicht so, daß jeder die- sen Link beschreiben könnte. Vielmehr gelten für den Link immer dieselben Rechte wie für die Datei auf die er zeigt!

4.6 Wildcard-Expansion

Im letzten Abschnitt haben wir bereits eine Anwendung von 'Wildcards' (Stellvertreterzeichen) gesehen. Wildcards sind besondere Zeichen, die andere Zeichen in Dateinamen ersetzen. Uns interessieren hier die zwei auch in DOS vorhandenen Wildcard-Zeichen '*' (Stern) und '?' (Fragezeichen). Der Stern ersetzt eine beliebige Anzahl beliebiger Zeichen, und das Fragezeichen ersetzt ein einzelnes (beliebiges) Zeichen.

Das Muster '*.txt' würde also zum Beispiel alle Dateien abdecken, die auf '.txt' enden, etwa 'meyer.txt', 'brief2.txt', 'test.old.xyz.txt' und so weiter. Das Muster 'a*' deckt alle Dateien ab, deren Name mit einem 'a' beginnt, also etwa 'andreas.txt', 'anfrage.html', 'ananas.gif'. Man kann diese Muster in allen Unix-Kommandos einsetzen, die einen Dateinamen erwarten, beim Kopieren, Umbenennen, Löschen usw.

Nehmen wir an, wir hätten im aktuellen Verzeichnis drei Dateien: a.txt, b.txt und c.txt. Unter DOS könnten wir jetzt ein Kommando zum Umbenennen aller drei Dateien in a.bak, b.bak und c.bak einsetzen: $C: ren * .txt * .bak$ Dieses Kommando würde den Stern für jede einzelne Datei in beiden Mustern (im alten und im neuen Namen) durch denselben Namen ersetzen. Dadurch hätten wir nach der Ausführung die- ses Befehls drei Dateien im aktuellen Verzeichnis, nämlich die gewünschten a.bak, b.bak und c.bak.

Unter Unix geht so etwas nicht. Um zu verstehen, warum das nicht klappt, müssen wir zunächst wissen, dass die Wildcards unter Unix nicht, wie unter DOS, von den einzelnen Kommandos abgearbeitet werden. Unix-Wildcards werden noch vor dem Aufruf des eigentlichen Befehls durch die Shell (das ist das Programm, das alle unsere Eingaben entgegennimmt und die einzelnen Kommandos aufruft) ersetzt. Erst nach der Ersetzung aller Wildcards wird das angegebene Kommando aufgerufen. Wie sieht also der Aufruf eines Kommandos, sagen wir 'ls *.txt' unter Unix wirklich aus?

Wir gehen, wie o.a. von folgender Situation aus:

```
$ ls
a.txt b.txt c.txt
$
```

Wenn nun das Kommando 'ls *.txt' ausgeführt wird, wird wie folgt vorgegangen: Zuerst expandiert die Shell die Wildcards, d.h. sie ersetzt sie durch alle Dateinamen, die auf das Muster passen. Aus der obigen Zeile wird also (unsichtbar für den Benutzer): 'ls a.txt b.txt c.txt' und dieses Kommando wird dann ausgeführt.

Das ist keine Besonderheit von 'ls'. Genau so funktionieren Wildcards immer unter Unix. Was passiert nun, wenn wir ein Muster angeben, das im aktuellen Verzeichnis zu nichts expandieren kann?

```
$ ls *.xyz
ls: *.xyz: Datei oder Verzeichnis nicht gefunden
$
```

Das Muster wird also unverändert an das 'ls'-Kommando weitergegeben. Da wir im aktuellen Verzeichnis keine Datei haben, die '*.xyz' heißt, gibt 'ls' eine Fehlermeldung aus. Immer dann, wenn ein Muster nicht durch einzelne Dateinamen ersetzt werden kann, wird es also unverändert gelassen und an das aufgerufene Kommando weitergereicht.

5 Dateiinhalte bearbeiten

Nachdem wir uns bisher mit der Verwaltung von Dateien 'von außen' beschäftigt haben, wollen wir jetzt sehen, wie man die Inhalte von Dateien bearbeiten kann.

5.1 Dateiinhalte anzeigen: cat, more

Mt Hilfe von 'cat' und 'more' kann man den Inhalt einer Textdatei auf den Bildschirm bringen. 'cat' entspricht dem DOS-Befehl 'TYPE' und ist, im Gegensatz zu letzterem, eines der vielseitigsten Unix-Kommandos. Es macht i.d.R. keinen Sinn, den Inhalt von Binärdateien (d.h. Programmen, Bild-Dateien usw.) mit 'cat' anzuzeigen, da sich in solchen Dateien keine für den Benutzer lesbaren Daten befinden.

Die allgemeine Form von 'cat' ist: 'cat dateiname(n)' wobei *dateiname* der Name einer Textdatei ist.

Die allgemeine Form von 'more' ist: 'more dateiname(n)' wobei *dateiname* der Name einer Textdatei ist.

'cat' zeigt den Inhalt der Dateien auf dem Bildschirm ohne Pause an, bei der Verwendung von 'more' wird der Benutzer am Bildschirm zur Eingabe aufgefordert, wenn die Ausgabe eine Bildschirmseite gefüllt hat.

5.2 Der Editor nano

Als Editor bezeichnet man ein einfaches Textverarbeitungsprogramm, das in der Regel keine komplexen Darstellungen (wie zum Beispiel Grafiken und Tabellen im Text) erlaubt. Verwenden Sie am Anfang nano. 'nano' ist weder der beste noch der leistungsfähigste aller Editoren, aber es ist einer, der leicht zu bedienen ist und für den Anfang ausreicht. Wenn Sie vorhaben das Satzsystem TeX zu benutzen, unter Unix zu programmieren oder eine umfangreiche wissenschaftliche Arbeit zu schreiben, dann sollten Sie auf jeden Fall noch einen leistungsfähigeren Editor erlernen, wie zum Beispiel den vi, der allerdings wegen seiner Komplexität einen eigenen Kurs erfordern würde.

5.3 Dateien drucken: lp

Dateien können auf dem Netzwerkdrucker mittels des Kommandos 'lp Dateiname' gedruckt werden.

6 Prozesse

Der Prozessbegriff ist zentral für ein Multitasking-System wie Unix. Ein Prozess ist einfach ein gerade laufendes Programm, dass sich im Hauptspeicher des Rechners befindet.

Es gibt unter Unix, im Gegensatz zu DOS, immer eine Vielzahl von Prozessen, also gleichzeitig laufenden Programmen. Viele dieser Prozesse werden nicht vom Benutzer gestartet, sondern vom Betriebssystem selber, und der Benutzer bekommt sie meist überhaupt nicht zu Gesicht (nur mittelbar, indem sie die Maschine, auf der er arbeitet, in Gang halten). So gehören zum Beispiel mehrere Prozesse zum grafischen Fenstersystem X, jedes einzelne Fenster wird von einem eigenen Prozess bedient, jede Eingabezeile ist ein eigener Prozess usw. Andere, unsichtbare Prozesse steuern das Verteilen von elektronischer Post, bedienen den Drucker und andere Ein-/Ausgabegeräte, realisieren die Netzwerkfunktionen des Systems und verwalten die Vergabe von Plattenplatz an die einzelnen Benutzer.

Jeder Prozess in einem Unix-System hat (ähnlich wie es auch bei Dateien ist) einen Besitzer, das ist in der Regel der Benutzer, der den Prozess gestartet hat (aber es gibt auch Prozesse, die festen Benutzern zugeordnet sind, egal, wer sie aufgerufen hat). Außerdem hat jeder Prozess eine einmalige Nummer, die ihn eindeutig kennzeichnet. Diese Nummer wird die Process ID (PID) des Prozesses genannt. Wir werden später sehen, wozu diese Nummer gebraucht werden kann.

6.1 Prozesse im Hintergrund starten: '&'

Bisher haben wir in unseren Übungen Prozesse immer nur nacheinander gestartet: wir haben zum Beispiel ein 'cp'-Kommando abgesetzt (auch das ist natürlich ein Prozess), haben gewartet, bis die Ausführung dieses Kommandos abgeschlossen war, und erst dann haben wir das nächste Kommando eingegeben. Soll die Verarbeitung im Hintergrund ausgeführt werden, so ist dem Kommando ein '&'-Zeichen am Ende mitzugeben. Die Shell gibt als Antwort die PID zurück.

Das Kommando 'ps' zeigt die laufenden Prozesse (ohne Option nur die des Benutzers).

```
$ date
Di  4 Feb 2014 12:24:28 CET
$ sleep 10
$ date
Di  4 Feb 2014 12:24:43 CET
$ sleep 10 &
[1] 17134
$ ps
  PID TTY           TIME CMD
15278 ttys000    0:00.13 -bash
17134 ttys000    0:00.00 sleep 10
15374 ttys001    0:00.03 -bash
$ date
Di  4 Feb 2014 12:24:46 CET
$
[1]+  Done                    sleep 10
$
$ date
Di  4 Feb 2014 12:25:35 CET
$
```

6.2 Prozesse abbrechen: kill

Ein Prozess kann durch das Kommando 'kill' abgebrochen werden. Dabei ist als Option eine Signalnummer anzugeben, die dem Prozess gesendet wird.

```
$ sleep 100 &
[1] 17159
$ ps
  PID TTY           TIME CMD
15278 ttys000    0:00.14 -bash
17159 ttys000    0:00.00 sleep 100
15374 ttys001    0:00.03 -bash
$
$ kill -2 17159
```

```
$
[1]+  Interrupt: 2            sleep 100
$ ps
  PID TTY           TIME CMD
15278 ttys000    0:00.14 -bash
15374 ttys001    0:00.03 -bash
$
```